사용자 가이드

AWS CloudHSM

AWS CloudHSM: 사용자 가이드

Table of Contents

무엇입니까 AWS CloudHSM?

AWS CloudHSM AWS 클라우드의 이점과 하드웨어 보안 모듈 (HSM) 의 보안을 결합합니다. 하드웨어 보안 모듈(HSM)은 암호화 작업을 처리하고 암호화 키에 보안 스토리지를 제공하는 컴퓨팅 장치입니다. 를 사용하면 AWS 클라우드에 있는 AWS CloudHSM고가용성 HSM을 완벽하게 제어하고 액세스 지연 시간이 짧으며 HSM 관리 (백업, 프로비저닝, 구성 및 유지 관리 포함) 를 자동화하는 안전한 신뢰 루트를 확보할 수 있습니다.

AWS CloudHSM 고객에게 다양한 혜택을 제공합니다.

FIPS 및 비 FIPS 클러스터에 액세스

AWS CloudHSM FIPS 모드와 비 FIPS의 두 가지 모드로 클러스터를 제공합니다. FIPS 모드에서는 연방 정보 처리 표준 (FIPS) 에서 승인한 키와 알고리즘만 사용할 수 있습니다. 비 FIPS 모드는 FIPS 승인과 상관없이 지원되는 모든 키와 알고리즘을 제공합니다. AWS CloudHSM자세한 정보는 AWS CloudHSM 클러스터 모드 및 HSM 유형을 참조하세요.

HSM은 범용 단일 테넌트이며 FIPS 모드의 클러스터에 대해 검증된 FIPS 140-2 레벨 3입니다.

AWS CloudHSM 는 애플리케이션에 대해 미리 정해진 알고리즘과 키 길이가 있는 완전 관리형 AWS 서비스와 비교할 때 더 많은 유연성을 제공하는 범용 HSM을 사용합니다. 당사는 표준을 준수하는 싱글 테넌트이며 FIPS 모드의 클러스터에 대해 FIPS 140-2 레벨 3 인증을 받은 HSM을 제공합니다. FIPS 140-2 레벨-3 검증 제한을 벗어나는 사용 사례가 있는 고객의 경우 비 FIPS 모드의 클러스터도 제공합니다. AWS CloudHSM 자세한 정보는 AWS CloudHSM 클러스터을 참조하세요.

AWS에서는 E2E 암호화가 표시되지 않음

데이터 플레인은 end-to-end (E2E) 암호화되어 AWS에서 볼 수 없으므로 IAM 역할 외부에서 사용자 관리를 제어할 수 있습니다. 이 제어의 단점은 관리형 AWS 서비스를 사용할 때보다 더 많은 책임이 있다는 것입니다.

키, 알고리즘 및 애플리케이션 개발을 완전히 제어할 수 있습니다.

AWS CloudHSM 사용하는 알고리즘과 키를 완전히 제어할 수 있습니다. 암호화 키(세션 키, 토큰 키, 대칭 키 및 비대칭 키 페어 포함)를 생성, 저장, 가져오기, 내보내기, 관리 및 사용할 수 있습니다. 또한 AWS CloudHSM SDK를 사용하면 애플리케이션 개발, 애플리케이션 언어, 스레딩 및 애플리케이션의 물리적 위치를 완벽하게 제어할 수 있습니다.

암호화 워크로드를 클라우드로 마이그레이션

공개 키 암호화 표준 #11 (PKCS #11), Java 암호화 확장 (JCE), 암호화 API: 차세대 (CNG) 또는 KSP (키 저장소 공급자) 를 사용하는 공개 키 인프라를 마이그레이션하는 고객은 애플리케이션을 변경하지 않고도 마이그레이션할 수 있습니다. AWS CloudHSM

수행할 수 있는 작업에 대해 자세히 알아보려면 다음 항목을 참조하십시오. AWS CloudHSM시작할 준비가 되면 을 참조하십시오시작하기. AWS CloudHSM

> **Note**
>
> 암호화 키를 생성하고 제어하기 위한 관리형 서비스를 원하지만 자체 HSM은 원하지 않거나 운영할 필요가 없는 경우 AWS Key Management Service 사용을 고려해 보십시오.
> 클라우드에서 결제 처리 애플리케이션을 위한 결제 HSM과 키를 관리하는 탄력적인 서비스를 찾고 있다면 AWS Payment Cryptography를 사용하는 것을 고려해 보십시오.

내용

AWS CloudHSM 사용 사례

AWS CloudHSM 다양한 목표를 달성하는 데 사용할 수 있습니다. 이 항목의 내용은 수행할 수 있는 작업에 대한 개요를 제공합니다. AWS CloudHSM

규제 준수 달성

기업 보안 표준을 준수해야 하는 기업은 고도의 기밀 데이터를 보호하는 개인 키를 관리하는 AWS CloudHSM 데 사용할 수 있습니다. 에서 제공하는 AWS CloudHSM HSM은 FIPS 140-2 레벨 3 인증을 받았으며 PCI DSS를 준수합니다. 또한 PCI PIN을 AWS CloudHSM 준수하고 PCI-3DS 규정을 준수합니다. 자세한 정보는 규정 준수을 참조하세요.

암호화 및 해독

고도의 기밀 데이터, 전송 중 암호화 및 저장 시 암호화를 보호하는 개인 키를 관리하는 AWS CloudHSM 데 사용합니다. 또한 여러 암호화 AWS CloudHSM SDK와의 표준 준수 통합을 제공합니다.

프라이빗 및 퍼블릭 키로 문서에 서명하고 확인합니다.

암호화에서는 프라이빗 키를 사용하여 문서에 서명하면 수신자는 퍼블릭 키를 사용하여 다른 사람이 아닌 사용자가 실제로 문서를 보냈는지 확인할 수 있습니다. AWS CloudHSM 이 목적을 위해 특별히 설계된 비대칭 공개 및 개인 키 쌍을 만드는 데 사용합니다.

HMAC 및 CMAC를 사용하여 메시지 인증

암호화에서는 암호 메시지 인증 코드(CMAC) 및 해시 기반 메시지 인증 코드(HMAC)를 사용하여 안전하지 않은 네트워크를 통해 전송되는 메시지를 인증하고 무결성을 보장합니다. 를 사용하면 HMAC 및 CMAC를 지원하는 대칭 키를 안전하게 생성하고 관리할 수 있습니다. AWS CloudHSM

및 의 이점을 활용하십시오. AWS CloudHSM AWS Key Management Service

고객은 FIPS 140-2 Level 3 인증을 받은 단일 테넌트 환경에서 주요 자료를 결합하여 AWS CloudHSM 저장하는 동시에 키 관리, 확장 및 클라우드 통합의 이점을 얻을 수 있습니다. AWS KMS AWS KMS이를 수행하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS CloudHSM 키 스토어를 참조하십시오.

웹 서버용 SSL/TLS 처리 오프로드

인터넷을 통해 데이터를 안전하게 전송하기 위해 웹 서버는 퍼블릭-프라이빗 키 페어와 SSL/TLS 퍼블릭 키 인증서를 사용하여 HTTPS 세션을 설정합니다. 이 프로세스에는 웹 서버의 많은 계산이 필요하지만 이 중 일부를 클러스터로 오프로드하여 보안을 강화하는 동시에 계산 부담을 줄일 수 있습니다. AWS CloudHSM 를 사용하여 SSL/TLS 오프로드를 설정하는 방법에 대한 자세한 내용은 을 참조하십시오. AWS CloudHSMSSL/TLS 오프로드

TDE(Transparent Data Encryption) 활성화

TDE(Transparent Data Encryption)는 데이터베이스 파일을 암호화하는 데 사용됩니다. TDE를 사용하면 데이터베이스 소프트웨어는 데이터를 디스크에 저장하기 전에 암호화합니다. TDE 마스터

암호화 키를 AWS CloudHSM의 HSM에 저장하여 보안을 강화할 수 있습니다. Oracle TDE 설정에 대한 자세한 내용은 을 참조하십시오. AWS CloudHSMOracle Database 암호화

발급 인증 기관(CA)의 프라이빗 키 관리

인증 기관(CA)은 퍼블릭 키를 ID(개인 또는 조직)에 바인딩하는 디지털 인증서를 발급하는 신뢰할 수 있는 엔터티입니다. CA를 운영하려면 CA에서 발급한 인증서에 서명하는 프라이빗 키를 보호하여 신뢰성을 유지해야 합니다. 이러한 개인 키를 AWS CloudHSM 클러스터에 저장한 다음 HSM을 사용하여 암호화 서명 작업을 수행할 수 있습니다.

무작위 숫자 생성

무작위 숫자를 생성하여 암호화 키를 생성하는 것은 온라인 보안의 핵심입니다. AWS CloudHSM 사용자가 제어하는 HSM에서 사용자만 볼 수 있는 난수를 안전하게 생성하는 데 사용할 수 있습니다.

AWS CloudHSM 작동 방식

이 항목에서는 HSM에서 데이터를 안전하게 암호화하고 암호화 작업을 수행하는 데 사용하는 기본 개념 및 아키텍처에 대한 개요를 제공합니다. AWS CloudHSM 자체 아마존 가상 사설 클라우드 (VPC) 에서 운영됩니다. 사용할 AWS CloudHSM수 있으려면 먼저 클러스터를 생성하고 HSM을 추가하고 사용자와 키를 생성한 다음 클라이언트 SDK를 사용하여 HSM을 애플리케이션과 통합해야 합니다. 이 작업이 완료되면 클라이언트 SDK 로그 AWS CloudTrail, 감사 로그 및 CloudWatch Amazon을 사용하여 AWS CloudHSM모니터링합니다.

기본 개념과 이러한 개념이 함께 작동하여 데이터를 보호하는 방법을 알아보십시오 AWS CloudHSM.

주제

- AWS CloudHSM 클러스터
- HSM 사용자
- HSM 키
- 클라이언트 SDK
- AWS CloudHSM 클러스터 백업
- 리전

AWS CloudHSM 클러스터

개별 HSM이 동기화되고 중복되며 가용성이 높은 방식으로 함께 작동하도록 만드는 것은 어려울 수 있지만 AWS CloudHSM , HSM (하드웨어 보안 모듈) 을 클러스터로 제공하면 번거로운 작업이 될 수 있습니다. 클러스터는 동기화 상태를 유지하는 개별 HSM의 모음입니다. AWS CloudHSM 클러스터에서 하나의 HSM에 대해 과업 또는 작업을 수행하면 해당 클러스터의 다른 HSM이 자동으로 최신 상태로 유지됩니다.

AWS CloudHSM FIPS 모드와 비 FIPS의 두 가지 모드로 클러스터를 제공합니다. FIPS 모드에서는 연방 정보 처리 표준 (FIPS) 에서 승인한 키와 알고리즘만 사용할 수 있습니다. 비 FIPS 모드는 FIPS 승인과 상관없이 지원되는 모든 키와 알고리즘을 제공합니다. AWS CloudHSM AWS CloudHSM 또한 hsm1.medium과 hsm2m.medium이라는 두 가지 유형의 HSM을 제공합니다. 각 HSM 유형과 클러스터 모드 간의 차이점에 대한 자세한 내용은 을 참조하십시오. AWS CloudHSM 클러스터 모드 및 HSM 유형

가용성, 내구성 및 확장성 목표를 충족하려면 여러 가용 영역에 걸쳐 클러스터의 HSM 수를 설정합니다. 1~28개의 HSM이 있는 클러스터를 생성할 수 있습니다 (기본 제한은 지역당 AWS 계정당AWS HSM 6개). 지역 내 여러 가용 영역에 HSM을 배치할 수 있습니다. AWS 클러스터에 더 많은 HSM을 추가하면 성능이 향상됩니다. 가용 영역에 클러스터를 분산하면 중복성 및 고가용성이 제공됩니다.

클러스터에 대한 자세한 내용은 AWS CloudHSM 클러스터 관리 섹션을 참조하십시오.

클러스터를 생성하려면 시작하기 섹션을 참조하십시오.

HSM 사용자

대부분의 AWS 서비스 및 리소스와는 달리 클러스터 내 리소스에 액세스할 때는 AWS Identity and Access Management (IAM) 사용자나 IAM 정책을 사용하지 않습니다. 대신 클러스터의 HSM에서 직접 HSM 사용자를 사용할 수 있습니다. AWS CloudHSM

HSM 사용자는 IAM 사용자와 다릅니다. 올바른 자격 증명을 보유한 IAM 사용자는 AWS API를 통해 리소스와 상호 작용하여 HSM을 생성할 수 있습니다. E2E 암호화는 AWS에서 볼 수 없으므로 자격 증명은 HSM에서 직접 수행되기 때문에 HSM 사용자 자격 증명을 사용하여 HSM에서 작업을 인증해야 합니다. HSM은 사용자가 정의하고 관리하는 자격 증명을 사용하여 각 HSM 사용자를 인증합니다. 각 HSM 사용자에게는 사용자가 HSM에서 수행할 수 있는 작업을 결정하는 유형이 있습니다. 각 HSM은 CloudHSM CLI를 사용하여 정의한 자격 증명을 통해 각 HSM 사용자를 인증합니다.

이전 SDK 버전 시리즈를 사용하는 경우 CloudHSM 관리 유틸리티(CMU)를 사용하게 됩니다.

HSM 키

AWS CloudHSM을 사용하면 AWS CloudHSM 클러스터에 있는 단일 테넌트 HSM에서 암호화 키를 안전하게 생성, 저장 및 관리할 수 있습니다. 키는 대칭 또는 비대칭일 수 있으며, 단일 세션용 세션 키(임시 키), 장기간 사용을 위한 토큰 키(영구 키)일 수 있고, AWS CloudHSM에서 내보내거나 AWS CloudHSM으로 가져올 수 있습니다. 키를 사용하여 일반적인 암호화 작업 및 기능을 완료할 수도 있습니다.

- 대칭 및 비대칭 암호화 알고리즘을 모두 사용하여 암호화 데이터 서명 및 서명 검증을 수행합니다.
- 해시 함수를 사용하여 컴퓨팅 메시지 다이제스트 및 해시 기반 메시지 인증 코드(HMAC)를 계산합니다.
- 다른 키를 래핑하고 보호합니다.
- 암호로 임의 보안 데이터에 액세스합니다.

클러스터가 가질 수 있는 최대 키는 클러스터에 있는 HSM의 유형에 따라 다릅니다. 예를 들어 hsm2m.medium은 hsm1, medium보다 더 많은 키를 저장합니다. 비교에 대한 내용은 을 참조하십시오. AWS CloudHSM 할당량

또한 키 사용 및 관리에 대한 몇 가지 기본 원칙을 AWS CloudHSM 따릅니다.

선택할 수 있는 다양한 키 유형 및 알고리즘

솔루션을 사용자 정의할 수 있도록 선택할 수 있는 다양한 키 유형과 알고리즘을 AWS CloudHSM 제공합니다. 알고리즘은 다양한 키 크기를 지원합니다. 자세한 내용은 각 AWS CloudHSM 클라이언트 SDK 속성 및 메커니즘 페이지를 참조하십시오.

키 관리 방법

AWS CloudHSM 키는 SDK 및 명령줄 도구를 통해 관리됩니다. 이러한 도구를 사용하여 키를 관리하는 방법에 대한 자세한 내용은 에서 키 관리 AWS CloudHSM 및 에 대한 모범 사례 AWS CloudHSM를 참조하십시오.

키 소유자

AWS CloudHSM에서는 키를 생성한 암호화 사용자 (CU) 가 키를 소유합니다. 소유자는 key share 및 key unshare 명령을 사용하여 다른 CU와 키를 공유하고 공유를 해제할 수 있습니다. 자세한 내용은 CloudHSM CLI를 사용하여 키 공유 및 공유 해제 섹션을 참조하십시오.

속성 기반 암호화로 액세스 및 사용을 제어할 수 있습니다.

> AWS CloudHSM 키 속성을 사용하여 정책에 따라 데이터를 복호화할 수 있는 사용자를 제어할 수 있는 암호화의 한 형태인 속성 기반 암호화를 사용할 수 있습니다.

클라이언트 SDK

를 사용할 AWS CloudHSM때는 AWS CloudHSM 클라이언트 소프트웨어 개발 키트 (SDK) 를 사용하여 암호화 작업을 수행합니다. AWS CloudHSM 클라이언트 SDK에는 다음이 포함됩니다.

- 퍼블릭 키 암호화 표준 #11(PKCS #11)
- JCE 공급자
- OpenSSL Dynamic Engine
- 암호화 API: Microsoft Windows용 차세대(CNG) 및 KSP(Key Storage Provider)

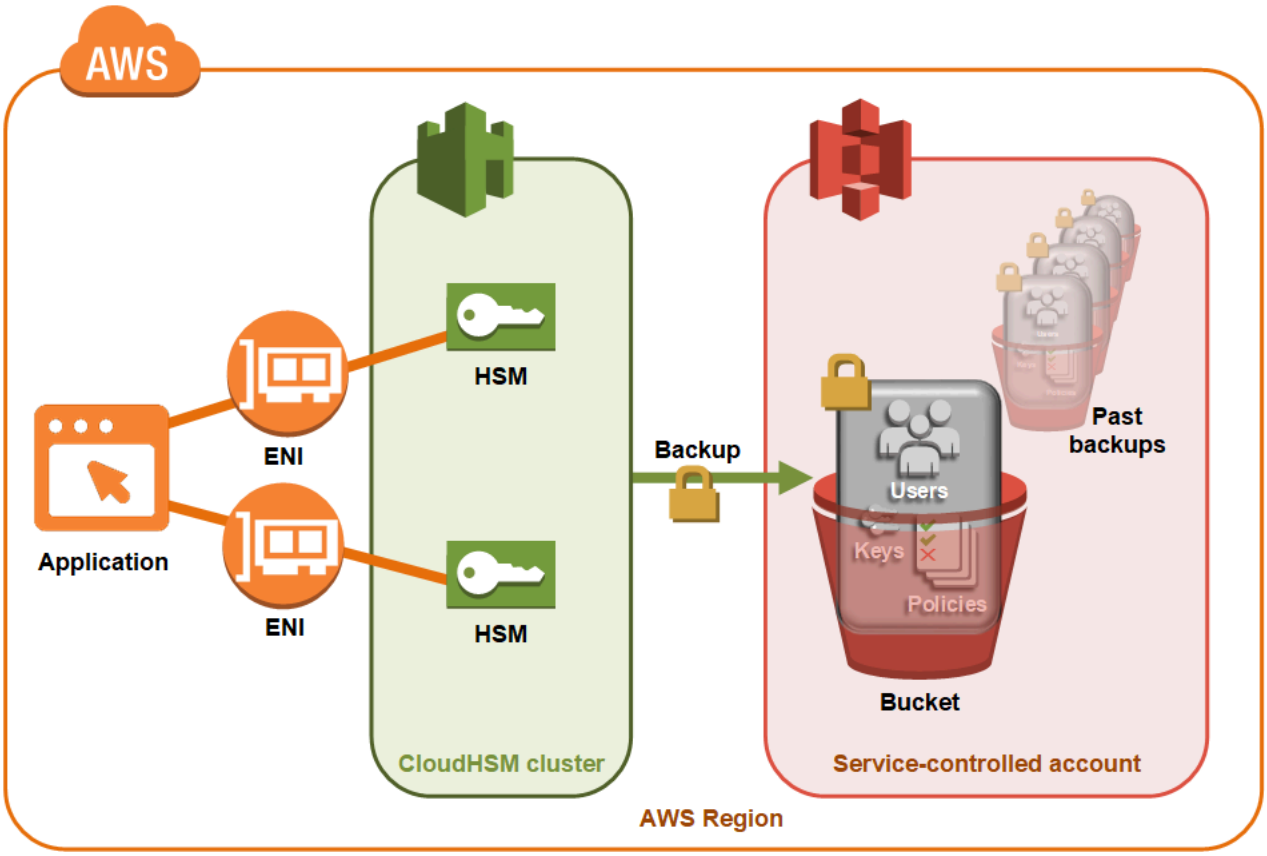
클러스터에서 이러한 SDK 중 일부 또는 전체를 사용할 수 있습니다. AWS CloudHSM 이러한 SDK를 사용하여 HSM에서 암호화 작업을 수행하도록 애플리케이션 코드를 작성합니다. 각 SDK를 지원하는 플랫폼 및 HSM 유형을 확인하려면 다음을 참조하십시오. Client SDK 5 지원 플랫폼

유틸리티와 명령줄 도구는 SDK를 사용할 뿐만 아니라 애플리케이션의 자격 증명, 정책 및 설정 구성에도 필요합니다. 자세한 정보는 AWS CloudHSM 명령줄 도구 섹션을 참조하십시오.

클라이언트 SDK 설치 및 사용 또는 클라이언트 연결 보안에 대한 자세한 내용은 클라이언트 SDK 및 End-to-end 암호화를 참조하십시오.

AWS CloudHSM 클러스터 백업

AWS CloudHSM 클러스터의 사용자, 키 및 정책을 정기적으로 백업합니다. 백업은 안전하고 내구성이 뛰어나며 예측 가능한 일정에 따라 업데이트됩니다. 다음 그림에서는 백업과 클러스터의 관계를 보여줍니다.



백업 작업에 대한 자세한 내용은 백업 관리을 참조하십시오.

보안

HSM에서 백업을 AWS CloudHSM 만들면 HSM은 데이터를 보내기 전에 모든 데이터를 암호화합니다. AWS CloudHSM해당 데이터는 HSM을 일반 텍스트 형식으로 두지 않습니다. 또한 백업은 백업을 해독하는 데 사용된 키에 액세스할 수 AWS 없으므로 AWS 복호화할 수 없습니다. 자세한 내용은 클러스터 백업의 보안단원을 참조하세요.

내구성

AWS CloudHSM 클러스터와 동일한 리전에 있는 서비스가 제어되는 Amazon Simple Storage Service (Amazon S3) 버킷에 백업을 저장합니다. 백업의 내구성 수준은 99.999999999%이며, 이는 Amazon S3에 저장된 모든 객체와 동일합니다.

리전

지원되는 지역에 대한 AWS CloudHSM자세한 내용은 또는 지역 표의AWS CloudHSM 지역 및 엔드포인트를 참조하십시오. AWS 일반 참조

AWS CloudHSM 특정 지역의 모든 가용 영역에서 사용 가능하지 않을 수 있습니다. 하지만 클러스터의 모든 HSM에서 AWS CloudHSM 자동으로 부하가 분산되므로 성능에는 영향을 미치지 않아야 합니다.

대부분의 AWS 리소스와 마찬가지로 클러스터와 HSM은 지역 리소스입니다. 리전을 교차하여 클러스터를 재사용하거나 확장할 수 없습니다. 새 리전에서 클러스터를 생성하려면 시작하기 AWS CloudHSM에 나열된 모든 필수 단계를 수행해야 합니다.

재해 복구를 위해 AWS CloudHSM 클러스터 백업을 한 지역에서 다른 지역으로 복사할 수 있습니다. AWS CloudHSM 자세한 내용은 AWS CloudHSM 클러스터 백업을(를) 참조하세요.

요금

를 사용하면 장기 약정이나 선결제 없이 시간당 요금을 지불할 수 있습니다. AWS CloudHSM자세한 내용은 웹 사이트의 AWS CloudHSMAWS 요금을 참조하십시오.

시작하기 AWS CloudHSM

다음 항목은 클러스터를 생성, 초기화 및 활성화하는 AWS CloudHSM 데 도움이 됩니다. 이러한 절차를 완료하면 사용자를 관리하고 클러스터를 관리하며 포함된 소프트웨어 라이브러리를 사용하여 암호화 작업을 수행할 수 있습니다.

내용

IAM 관리 그룹 생성

다음을 포함하여 사용자 인터페이스를 사용하여 상호 AWS작용하지 않는 AWS 계정 루트 사용자 것이 좋습니다 AWS CloudHSM. 대신 AWS Identity and Access Management (IAM) 을 사용하여 IAM 사용자, IAM 역할 또는 연동 사용자를 생성하십시오. IAM 사용자 및 관리자 그룹 생성섹션의 단계에 따라 관리자 그룹을 생성하고 정책을 연결합니다. AdministratorAccess 새로운 관리자 사용자를 생성하고 사용자를 그룹에 추가하십시오. 필요에 따라 그룹에 사용자를 더 추가합니다. 추가하는 각 사용자는 그룹의 AdministratorAccess정책을 상속합니다.

또 다른 모범 사례는 실행에 필요한 권한만 있는 AWS CloudHSM 관리자 그룹을 만드는 것입니다. AWS CloudHSM필요에 따라 개별 사용자를 이 그룹에 추가하십시오. 각 사용자는 전체 AWS 액세스가 아니라 그룹에 연결된 제한된 권한을 상속합니다. 다음 고객 관리형 정책은 다음과 같습니다. AWS CloudHSM 섹션에는 AWS CloudHSM 관리자 그룹에 연결해야 하는 정책이 포함되어 있습니다.

AWS CloudHSM 계정의 서비스 연결 역할을 정의합니다. AWS 서비스 연결 역할은 현재 사용자 계정에서 이벤트를 기록할 수 있는 권한을 정의합니다. AWS CloudHSM 역할은 사용자가 자동으로 AWS CloudHSM 생성하거나 수동으로 생성할 수 있습니다. 역할을 편집할 수 없지만 삭제할 수는 있습니다. 자세한 내용은 서비스 연결 역할은 다음과 같습니다. AWS CloudHSM 섹션을 참조하십시오.

IAM 사용자 및 관리자 그룹 생성

먼저 IAM 사용자와 함께 해당 사용자의 관리자 그룹을 생성합니다.

등록하십시오. AWS 계정

계정이 없는 경우 다음 단계를 완료하여 계정을 만드세요. AWS 계정

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 여세요.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중에는 전화를 받고 키패드로 인증 코드를 입력하는 과정이 있습니다.

   에 AWS 계정가입하면 AWS 계정 루트 사용자a가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스 액세스 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 가서 내 계정(My Account)을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리자 액세스 권한이 있는 사용자 생성

등록한 AWS 계정후에는 일상적인 작업에 루트 사용자를 사용하지 않도록 관리 사용자를 보호하고 AWS IAM Identity Center활성화하고 생성하십시오 AWS 계정 루트 사용자.

보안을 유지하세요 AWS 계정 루트 사용자

1. Root user를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 AWS Management Console소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.

2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자 (콘솔) 에 대한 가상 MFA 디바이스 활성화를 참조하십시오.

관리자 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 ID 소스로 사용하는 방법에 대한 자습서는 사용 설명서의 기본값으로 IAM Identity Center 디렉터리사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리 AWS IAM Identity Center

관리 액세스 권한이 있는 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM IDentity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털에 로그인을 참조하십시오.AWS 로그인

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Create a permission set를 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은AWS IAM Identity Center 사용 설명서의 Add groups를 참조하세요.

IAM 사용자 그룹에 연결할 수 AWS CloudHSM 있는 정책의 예는 을 참조하십시오. 자격 증명 및 액세스 관리: AWS CloudHSM

Virtual Private Cloud(VPC) 생성

Virtual Private Cloud(VPC)가 아직 없는 경우 이 주제의 단계에 따라 VPC를 생성하십시오.

> **Note**
>
> 이 단계를 따르면 퍼블릭 서브넷과 프라이빗 서브넷이 생성됩니다.

VPC를 생성하려면

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 여세요.
2. 탐색 표시줄에서 지역 선택기를 사용하여 현재 지원되는AWS 지역 AWS CloudHSM 중 하나를 선택합니다.
3. VPC 생성 버튼을 선택하십시오.
4. 생성할 리소스에서 VPC 등을 선택합니다.
5. 이름 태그 자동생성의 경우 **`CloudHSM`**과 같이 식별 가능한 이름을 입력하세요.
6. 다른 모든 옵션을 기본값으로 둡니다.
7. VPC 생성을 선택합니다.
8. VPC가 생성된 후 VPC 보기를 선택하여 방금 생성한 VPC를 확인할 수 있습니다.

클러스터 생성

클러스터는 개별 HSM의 모음입니다. AWS CloudHSM 각 클러스터의 HSM을 동기화하여 논리적 단위로 작동하도록 합니다. AWS CloudHSM hsm1.medium과 hsm2m.medium이라는 두 가지 유형의 HSM을 제공합니다. 클러스터를 생성할 때 두 클러스터 중 어느 것을 클러스터에 포함할지 선택합니다. 각 HSM 유형과 클러스터 모드 간의 차이점에 대한 자세한 내용은 을 참조하십시오AWS CloudHSM 클러스터 모드 및 HSM 유형.

클러스터를 생성할 때 사용자 대신 클러스터의 보안 그룹을 AWS CloudHSM 생성합니다. 이 보안 그룹은 클러스터의 HSM에 대한 네트워크 액세스를 제어합니다. 보안 그룹에 있는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스에서만 인바운드 연결을 허용합니다. 기본적으로 해당 보안 그룹은 어떠한 인스턴스도 포함하지 않습니다. 나중에 클라이언트 인스턴스를 시작하고 클러스터의 보안 그룹을 구성하여 HSM과의 통신 및 연결을 허용합니다.

> **Important**
>
> 클러스터를 생성하면 라는 서비스 연결 역할이 AWS CloudHSM 생성됩니다. AWSServiceRoleForCloudHSM 역할을 생성할 AWS CloudHSM 수 없거나 역할이 아직 없는 경우 클러스터를 생성하지 못할 수 있습니다. 자세한 정보는 클러스터 생성 실패 해결을 참조

하세요. 서비스 연결 역할에 대한 자세한 내용은 서비스 연결 역할은 다음과 같습니다. AWS CloudHSM 단원을 참조하십시오.

AWS CloudHSM 콘솔, AWS CLI(AWS Command Line Interface) 또는 AWS CloudHSM API에서 클러스터를 생성할 수 있습니다.

> **Note**
>
> 클러스터 인수 및 API에 대한 자세한 내용은 AWS CLI 명령 참조를 참조하십시오 create-cluster.

클러스터(콘솔) 생성

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 탐색 표시줄에서 지역 선택기를 사용하여 현재 지원되는AWS 지역 AWS CloudHSM 중 하나를 선택합니다.
3. 클러스터 생성을 선택합니다.
4. 클러스터 구성 단원에서 다음을 수행합니다.
   a. VPC의 경우, Virtual Private Cloud(VPC) 생성에서 앞서 생성한 VPC를 선택합니다.
   b. 가용 영역의 각 가용 영역 옆에서 생성된 프라이빗 서브넷을 선택합니다.

      > **Note**
      >
      > 특정 가용 영역에서 AWS CloudHSM 지원되지 않더라도 클러스터의 모든 HSM에서 AWS CloudHSM 자동으로 로드 밸런싱되므로 성능에 영향을 미치지 않아야 합니다. 가용 AWS CloudHSM 영역 지원을 AWS 일반 참조보려면 의 지역 및 엔드포인트를 참조하십시오. AWS CloudHSM

   c. HSM 유형의 경우 클러스터에서 생성할 수 있는 HSM 유형과 원하는 클러스터 모드를 선택합니다. 각 지역에서 지원되는 HSM 유형을 확인하려면 AWS CloudHSM 요금 계산기 를 참조하십시오.

> **Important**
>
> 클러스터를 생성한 후에는 HSM 유형과 클러스터 모드를 변경할 수 없습니다. 사용 사례에 적합한 유형 및 모드에 대한 자세한 내용은 을 참조하십시오AWS CloudHSM 클러스터 모드 및 HSM 유형.

d. 클러스터 소스의 경우 새 클러스터를 생성할지 기존 백업에서 복원할지를 지정합니다.

- 비 FIPS 모드의 클러스터 백업은 비 FIPS 모드에 있는 클러스터를 복원하는 데만 사용할 수 있습니다.
- FIPS 모드의 클러스터 백업은 FIPS 모드에 있는 클러스터를 복원하는 데만 사용할 수 있습니다.

5. 다음을 선택합니다.
6. 서비스에서 백업을 보존해야 하는 기간을 지정하십시오.

> **Note**
>
> 기본 보존 기간인 90일을 그대로 사용하거나 7일에서 379일 사이의 새 값을 입력합니다. 이 서비스는 여기에 지정한 값보다 오래된 이 클러스터의 백업을 자동으로 삭제합니다. 나중에 변경할 수 있습니다. 자세한 내용은 백업 보존 정책 구성 단원을 참조하십시오.

7. 다음을 선택합니다.
8. (선택 사항) 태그 키와 태그 값(선택)을 입력합니다. 클러스터에 두 개 이상의 태그를 추가하려면 태그 추가를 선택합니다.
9. 검토를 선택합니다.
10. 클러스터 구성을 검토한 다음 클러스터 생성을 선택합니다.

클러스터 생성 방법(AWS CLI)

- 명령 프롬프트에서 create-cluster 명령을 실행합니다. HSM 인스턴스 유형, 백업 보존 기간 및 HSM을 생성할 서브넷의 서브넷 ID를 지정합니다. 생성된 프라이빗 서브넷의 서브넷 ID를 사용합니다. 가용 영역당 하나의 서브넷만 지정합니다.

```
$ aws cloudhsmv2 create-cluster --hsm-type hsm1.medium \
      --backup-retention-policy Type=DAYS,Value=<number of days> \
      --subnet-ids <subnet ID>
```

```
{
    "Cluster": {
        "BackupPolicy": "DEFAULT",
        "BackupRetentionPolicy": {
            "Type": "DAYS",
            "Value": 90
         },
        "VpcId": "vpc-50ae0636",
        "SubnetMapping": {
            "us-west-2b": "subnet-49a1bc00",
            "us-west-2c": "subnet-6f950334",
            "us-west-2a": "subnet-fd54af9b"
        },
        "SecurityGroup": "sg-6cb2c216",
        "HsmType": "hsm1.medium",
        "Certificates": {},
        "State": "CREATE_IN_PROGRESS",
        "Hsms": [],
        "ClusterId": "cluster-igklspoyj5v",
        "ClusterMode": "FIPS",
        "CreateTimestamp": 1502423370.069
    }
}
```

> **Note**
>
> `ClusterMode`지정하지 않을 경우 FIPS 모드가 기본값입니다. 비 FIPS 클러스터를 생성하려면 다음 파라미터를 포함해야 합니다. `--mode`
>
> ```
> $ aws cloudhsmv2 create-cluster --hsm-type hsm2m.medium \
>       --backup-retention-policy Type=DAYS,Value=<number of days> \
>       --subnet-ids <subnet ID> \
>     --mode NON_FIPS
> ```

클러스터 (AWS CloudHSM API) 를 만들려면

- CreateCluster 요청을 보냅니다. HSM 인스턴스 유형, 백업 보존 정책, HSM을 생성할 서브넷의 서브넷 ID를 지정합니다. 생성된 프라이빗 서브넷의 서브넷 ID를 사용합니다. 가용 영역당 하나의 서브넷만 지정합니다.

클러스터 생성 시도가 실패할 경우 AWS CloudHSM 서비스 연결 역할에 문제가 있는 것일 수 있습니다. 실패 해결에 도움을 받으려면 클러스터 생성 실패 해결 단원을 참조하십시오.

클러스터 보안 그룹 검토하기

클러스터를 생성할 때 라는 이름을 가진 보안 그룹을 AWS CloudHSM 생성합니다`cloudhsm-cluster-`*`clusterID`*`-sg`. 이 보안 그룹에는 포트 2223-2225를 통해 클러스터 보안 그룹 내에서 인바운드 및 아웃바운드 통신을 허용하는 사전 구성 TCP 규칙이 있습니다. 이 SG를 사용하면 EC2 인스턴스가 VPC를 사용하여 클러스터의 HSM과 통신할 수 있습니다.

> **Warning**
>
> - 클러스터 보안 그룹에 채워진 사전 구성 TCP 규칙을 삭제하거나 수정하지 마십시오. 이 규칙으로 HSM 무단 액세스 및 연결 문제를 방지할 수 있습니다.
> - 클러스터 보안 그룹은 HSM에 대한 무단 액세스를 방지합니다. 보안 그룹의 인스턴스에 액세스할 수 있는 사용자는 누구나 HSM에 액세스할 수 있습니다. 대부분의 작업에서 사용자가 HSM에 로그인해야 합니다. 하지만 인증 없이 HSM을 초기화하면 키 자료, 인증서 및 기타 데이터가 삭제될 수 있습니다. 이 경우 가장 최근에 백업한 후에 생성되거나 수정된 데이터는 손실되며 복구할 수 없습니다. 무단 액세스를 방지하려면 신뢰할 수 있는 관리자만 기본 보안 그룹에 있는 인스턴스를 수정하거나 액세스할 수 있게 하십시오.

다음 단계에서는 Amazon EC2 인스턴스를 시작하고 클러스터 보안 그룹을 연결하여 이 인스턴스를 HSM에 연결할 수 있습니다.

Amazon EC2 클라이언트 인스턴스 시작하기

AWS CloudHSM 클러스터 및 HSM 인스턴스와 상호 작용하고 관리하려면 HSM의 엘라스틱 네트워크 인터페이스와 통신할 수 있어야 합니다. 가장 쉬운 방법은 동일한 VPC의 EC2 인스턴스를 클러스터로 사용하는 것입니다. 또한 다음 AWS 리소스를 사용하여 클러스터에 연결할 수 있습니다.

- Amazon VPC 피어링
- AWS Direct Connect
- VPN 연결

> **Note**
>
> 이 안내서는 EC2 인스턴스를 클러스터에 연결하는 방법에 대한 간단한 예를 제공합니다. AWS CloudHSM 보안 네트워크 구성과 관련된 모범 사례는 을 참조하십시오. 클러스터에 대한 보안 액세스

AWS CloudHSM 설명서에서는 일반적으로 클러스터를 생성한 VPC 및 가용 영역 (AZ) 과 동일한 VPC 에서 EC2 인스턴스를 사용하고 있다고 가정합니다.

EC2 인스턴스를 생성하려면

1. https://console.aws.amazon.com/ec2/에서 EC2 대시보드를 엽니다.
2. 인스턴스 시작을 선택합니다. 드롭다운 메뉴에서 인스턴스 실행을 선택합니다.
3. 이름 필드에 EC2 인스턴스 이름을 입력합니다.
4. 애플리케이션 및 OS 이미지(Amazon 머신 이미지) 섹션에서 CloudHSM이 지원하는 플랫폼에 해당하는 Amazon 머신 이미지(AMI)를 선택합니다. 자세한 내용은 Client SDK 5 지원 플랫폼 섹션을 참조하십시오.
5. 인스턴스 유형 섹션에서 인스턴스 유형을 선택합니다.
6. 키 페어 섹션에서 기존 키 페어를 사용하거나 새 키 페어 생성을 선택하고 다음 단계를 완료합니다.

   a. 키 페어 이름에 키 페어의 이름을 입력합니다.
   b. 키 페어 유형에서 키 페어 유형을 선택합니다.
   c. 프라이빗 키 파일 형식에서 프라이빗 키 파일 형식을 선택합니다.
   d. 키 페어 생성을 선택합니다.
   e. 프라이빗 키 파일을 다운로드하고 저장합니다.

   > **Important**
   >
   > 이때가 사용자가 프라이빗 키 파일을 저장할 수 있는 유일한 기회입니다. 파일을 다운로드하고 안전한 장소에 저장합니다. 인스턴스를 시작할 때 키 페어의 이름을 제공해야 합니다. 또한 인스턴스에 연결할 때마다 해당 프라이빗 키를 제공하고 설정 시 생성한 키 페어를 선택해야 합니다.

7. 네트워크 설정에서 편집을 선택합니다.

8. VPC에서 클러스터에 대해 이전에 생성한 VPC를 선택합니다.
9. 서브넷에서, VPC에 대해 생성한 퍼블릭 서브넷을 선택합니다.
10. 퍼블릭 IP 자동 할당(Auto-assign Public IP)의 경우 활성화(Enable)를 선택합니다.
11. 기존 보안 그룹 선택을 선택합니다.
12. 공통 보안 그룹의 드롭다운 메뉴에서 기본 보안 그룹을 선택합니다.
13. 스토리지 구성에서 드롭다운 메뉴를 사용하여 스토리지 구성을 선택합니다.
14. 요약 창에서 인스턴스 실행을 선택합니다.

> **Note**
>
> 이 단계를 완료하면 EC2 인스턴스를 만드는 프로세스가 시작됩니다.

Linux Amazon EC2 클라이언트 생성에 대한 자세한 내용은 Amazon EC2 Linux 인스턴스 시작하기를 참조하십시오. 실행 중인 클라이언트에 연결하는 방법에 대한 자세한 내용은 다음 주제를 참조하십시오.

- SSH를 사용하여 Linux 인스턴스에 연결
- PuTTY를 사용하여 Windows에서 Linux 인스턴스에 연결

Amazon EC2 사용 설명서에 Amazon EC2 인스턴스 설정 및 사용에 대한 지침이 자세히 나와 있습니다. 다음 목록은 Linux 및 Windows Amazon EC2 클라이언트에 사용할 수 있는 설명서를 개괄적으로 보여줍니다.

- Linux Amazon EC2 클라이언트를 생성하려면 Amazon EC2 Linux 인스턴스 시작하기 섹션을 참조하십시오.

  실행 중인 클라이언트에 연결하는 방법에 대한 자세한 내용은 다음 주제를 참조하십시오.

  - SSH를 사용하여 Linux 인스턴스에 연결
  - PuTTY를 사용하여 Windows에서 Linux 인스턴스에 연결
- Windows Amazon EC2 클라이언트를 생성하려면 Amazon EC2 Windows 인스턴스 시작하기 섹션을 참조하십시오. Windows 클라이언트 연결에 대한 자세한 내용은 Windows 인스턴스에 연결 섹션을 참조하십시오.

> **Note**
>
> EC2 인스턴스는 이 가이드에 포함된 모든 AWS CLI 명령을 실행할 수 있습니다. AWS CLI가 설치되어 있지 않은 경우 AWS Command Line Interface에서 다운로드할 수 있습니다. Windows를 사용하는 경우 64비트 또는 32비트 Windows 설치 관리자를 다운로드하면 됩니다. Linux 또는 macOS를 사용하는 경우 pip을 사용하여 CLI를 설치할 수 있습니다.

클라이언트 Amazon EC2 인스턴스 보안 그룹 구성

Amazon EC2 인스턴스를 시작할 때 기본 Amazon VPC 보안 그룹에 이 인스턴스를 연결했습니다. 이 주제에서는 클러스터 보안 그룹을 EC2 인스턴스와 연결하는 방법을 설명합니다. 이 연결을 통해 EC2 인스턴스에서 실행되는 AWS CloudHSM 클라이언트가 HSM과 통신할 수 있습니다. EC2 인스턴스를 AWS CloudHSM 클러스터에 연결하려면 VPC 기본 보안 그룹을 적절히 구성하고 클러스터 보안 그룹을 인스턴스에 연결해야 합니다.

기본 보안 그룹 수정

클라이언트 소프트웨어를 다운로드 및 설치하고 HSM과 상호 작용할 수 있도록 SSH 또는 RDP 연결을 허용하려면 기본 보안 그룹을 수정해야 합니다.

기본 보안 그룹을 수정하려면

1. https://console.aws.amazon.com/ec2/에서 EC2 대시보드를 엽니다.
2. 인스턴스 (실행 중) 를 선택한 다음 클라이언트를 설치하려는 EC2 인스턴스 옆의 확인란을 선택합니다. AWS CloudHSM
3. 보안 탭에서 이름이 기본값인 보안 그룹을 선택합니다.
4. 페이지 위쪽에서 작업을 선택한 후 인바운드 규칙 편집을 선택합니다.
5. 규칙 추가를 선택합니다.
6. 유형에서 다음 중 하나를 수행합니다.
   - Windows Server Amazon EC2 인스턴스에는 RDP를 선택합니다. 포트 3389가 자동으로 채워집니다.
   - Linux Amazon EC2 인스턴스에는 SSH를 선택합니다. 포트 범위 22가 자동으로 채워집니다.
7. 어느 옵션이든 소스를 내 IP로 설정하면 Amazon EC2 인스턴스와 통신할 수 있습니다.

> **Important**
>
> 아무나 인스턴스에 액세스할 수 없게 하려면 0.0.0.0/0을 CIDR 범위로 지정하지 마십시오.

8. 저장을 선택합니다.

Amazon EC2 인스턴스를 클러스터에 연결합니다. AWS CloudHSM

EC2 인스턴스가 클러스터의 HSM과 통신하려면 EC2 인스턴스에 클러스터 보안 그룹을 연결해야 합니다. 클러스터 보안 그룹에는 포트 2223-2225를 통한 인바운드 통신을 허용하는 사전 구성 규칙이 있습니다.

EC2 인스턴스를 클러스터에 연결하려면 AWS CloudHSM

1. https://console.aws.amazon.com/ec2/에서 EC2 대시보드를 엽니다.
2. 인스턴스 (실행 중) 를 선택한 다음 클라이언트를 설치할 EC2 인스턴스의 확인란을 선택합니다. AWS CloudHSM
3. 페이지 위쪽에서 작업, 보안, 보안 그룹 변경을 차례로 선택합니다.
4. 클러스터 ID와 일치하는 그룹 이름을 가진 보안 그룹을 선택합니다(예: `cloudhsm-cluster-`*`clusterID`*`-sg`).
5. 보안 그룹 추가를 선택합니다.
6. 저장을 선택합니다.

> **Note**
>
> 보안 그룹을 5개까지 Amazon EC2 인스턴스에 할당할 수 있습니다. 최대 한도에 도달한 경우 Amazon EC2 인스턴스의 기본 보안 그룹과 클러스터 보안 그룹을 수정해야 합니다.
>
> 기본 보안 그룹에서 다음을 수행합니다.
>
> - 클러스터 보안 그룹에서 포트 2223-2225를 통해 TCP 프로토콜을 사용하는 트래픽을 허용하도록 인바운드 규칙을 추가합니다.
>
> 클러스터 보안 그룹에서 다음을 수행합니다.

- 기본 보안 그룹에서 포트 2223-2225를 통해 TCP 프로토콜을 사용하는 트래픽을 허용하도록 인바운드 규칙을 추가합니다.

HSM 생성

클러스터를 생성한 후 HSM을 생성할 수 있습니다. 하지만 클러스터에 HSM을 생성하기 전에 해당 클러스터가 초기화되지 않은 상태여야 합니다. 클러스터의 상태를 확인하려면 AWS CloudHSM 콘솔에서 클러스터 페이지를 보거나 를 사용하여 describe-clusters 명령을 실행하거나 AWS CloudHSM API에서 DescribeClusters요청을 보내십시오. AWS CLI AWS CloudHSM 콘솔, AWS CLI 또는 AWS CloudHSM API에서 HSM을 생성할 수 있습니다.

HSM(콘솔)을 생성하려면

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. HSM을 생성할 클러스터의 ID 옆의 라디오 버튼을 선택합니다.
3. 작업을 선택합니다. 드롭다운 메뉴에서 초기화를 선택합니다.
4. 생성 중인 HSM에 대한 가용 영역(AZ)을 선택합니다.
5. 생성을 선택합니다.

HSM 생성 방법(AWS CLI)

- 명령 프롬프트에서 create-hsm 명령을 실행합니다. 이전에 생성한 클러스터의 클러스터 ID와 HSM의 가용 영역을 지정합니다. `us-west-2a`, `us-west-2b` 등의 형태로 가용 영역을 지정합니다.

```
$ aws cloudhsmv2 create-hsm --cluster-id <cluster ID> --availability-
zone <Availability Zone>

{
    "Hsm": {
        "HsmId": "hsm-ted36yp5b2x",
        "EniIp": "10.0.1.12",
        "AvailabilityZone": "us-west-2a",
        "ClusterId": "cluster-igklspoyj5v",
        "EniId": "eni-5d7ade72",
```

```
        "SubnetId": "subnet-fd54af9b",
        "State": "CREATE_IN_PROGRESS"
    }
}
```

HSM (AWS CloudHSM API) 을 만들려면

- CreateHsm 요청을 보냅니다. 이전에 생성한 클러스터의 클러스터 ID와 HSM의 가용 영역을 지정합니다.

클러스터와 HSM을 생성한 후에는 선택에 따라 HSM의 ID를 확인하거나 클러스터 초기화로 넘어갑니다.

클러스터 HSM의 자격 증명 및 신뢰성 확인(선택 사항)

클러스터를 초기화하려면 클러스터의 첫 번째 HSM이 생성한 인증서 서명 요청(CSR)에 서명합니다. 이에 앞서 HSM의 ID와 신뢰성을 확인하고 싶을 수 있습니다.

> **Note**
>
> 이 프로세스는 선택 사항입니다. 하지만 이 프로세스는 클러스터가 초기화될 때까지만 유효합니다. 클러스터가 초기화된 후에는 이 프로세스를 사용하여 인증서를 가져오거나 HSM을 확인할 수 없습니다.

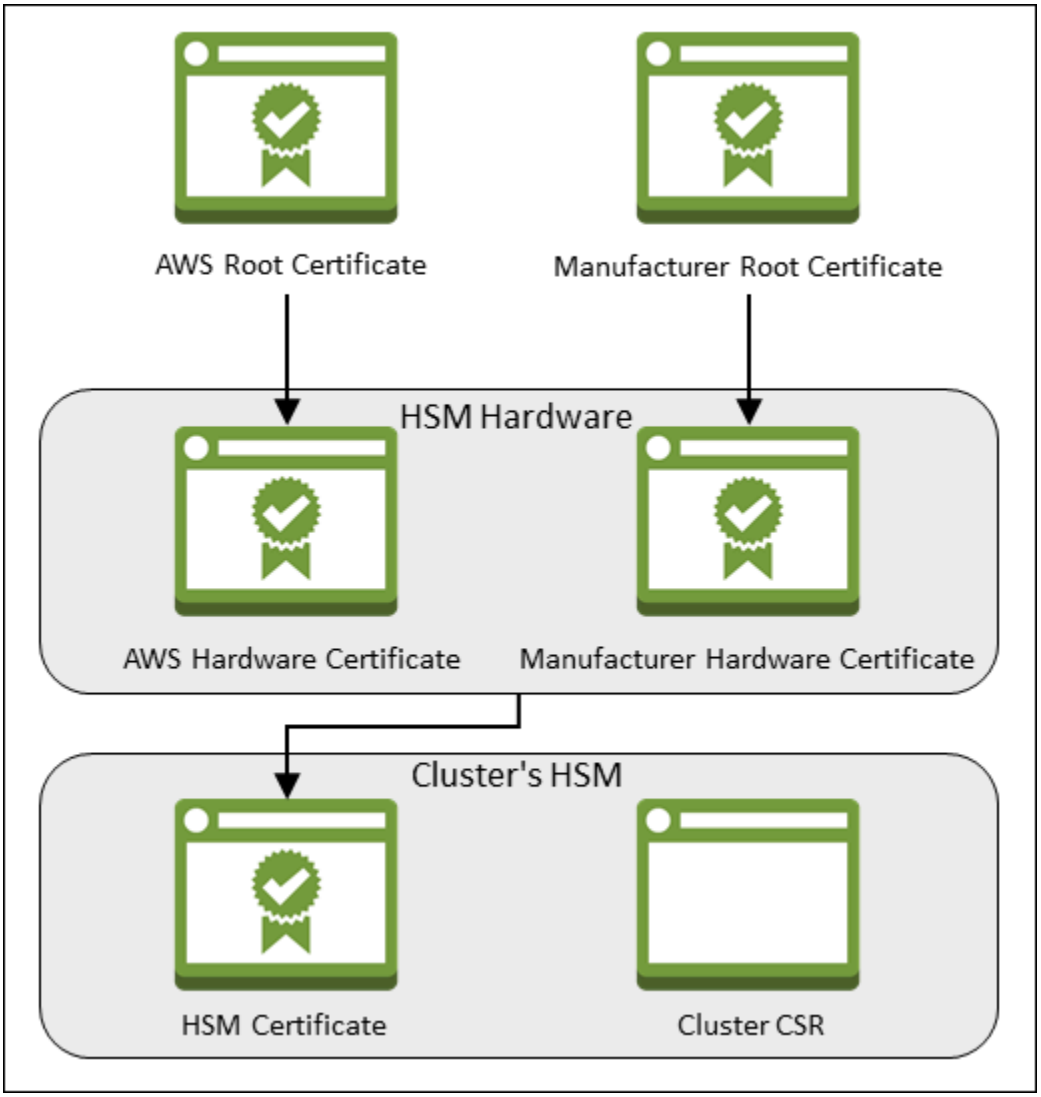
주제

개요

다음 단계를 완료하여 클러스터의 첫 번째 HSM의 ID를 확인할 수 있습니다.

1. 인증서 및 CSR 가져오기 – 이 단계에서는 HSM으로부터 세 개의 인증서와 CSR을 가져옵니다. 또한 HSM 하드웨어 제조업체에서 발급한 루트 인증서와 HSM 하드웨어 제조업체에서 AWS CloudHSM 발급한 루트 인증서를 각각 두 개 받을 수 있습니다.
2. 인증서 체인 확인 - 이 단계에서는 루트 인증서와 제조업체 AWS CloudHSM 루트 인증서에 각각 하나씩 총 두 개의 인증서 체인을 구성합니다. 그런 다음 이러한 인증서 체인으로 HSM 인증서를 확인하여 HSM의 ID 및 신뢰성을 보증하는지, AWS CloudHSM 그리고 하드웨어 제조업체가 모두 HSM의 ID 및 신뢰성을 증명하는지 확인합니다.
3. 퍼블릭 키 비교 – 이 단계에서는 HSM 인증서 및 클러스터 CSR에서 퍼블릭 키를 추출 및 비교하여 둘이 동일한지 확인합니다. 이렇게 하면 CSR가 신뢰할 수 있는 인증된 HSM에 의해 생성되었음에 안심할 수 있습니다.

다음 다이어그램은 CSR, 인증서 및 이들의 상호 관계를 보여줍니다. 다음에 나오는 목록은 각 인증서를 정의합니다.



AWS 루트 인증서

루트 AWS CloudHSM인증서입니다.

제조업체 루트 인증서

이것은 하드웨어 제조업체의 루트 인증서입니다.

AWS 하드웨어 인증서

AWS CloudHSM HSM 하드웨어가 플릿에 추가될 때 이 인증서를 생성했습니다. 이 인증서는 하드웨어를 AWS CloudHSM 소유하고 있다고 주장합니다.

제조업체 하드웨어 인증서

HSM 하드웨어 제조업체가 HSM 하드웨어를 생산할 때 이 인증서를 만들었습니다. 이 인증서는 제조업체가 하드웨어를 만들었음을 어셜션합니다.

HSM 인증서

HSM 인증서는 클러스터에서 처음 HSM을 생성할 때 FIPS 검증 하드웨어에 의해 생성됩니다. 이 인증서는 HSM 하드웨어가 HSM을 생성했음을 어셜션합니다.

클러스터 CSR

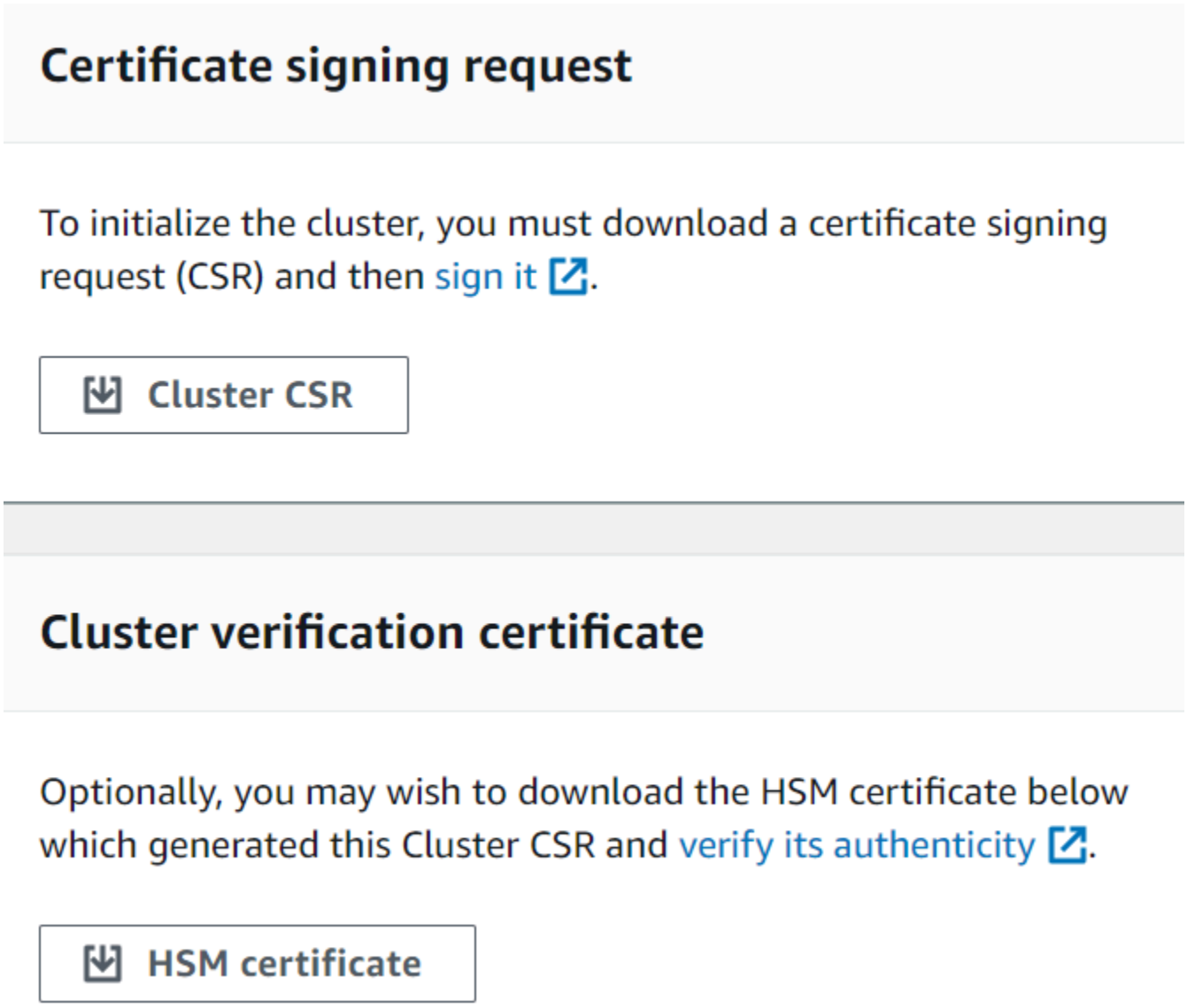
첫 번째 HSM이 클러스터 CSR을 생성합니다. 클러스터 CSR에 서명할 때 클러스터를 클레임합니다. 그런 다음 서명된 CSR을 사용하여 클러스터를 초기화할 수 있습니다.

HSM에서 인증서 가져오기

HSM의 ID와 신뢰성을 확인하려면 CSR과 5개의 인증서를 얻는 것부터 시작합니다. HSM에서 인증서 세 개를 받을 수 있으며, AWS CloudHSM 콘솔, AWS Command Line Interface (AWS CLI) 또는 API로 이 작업을 수행할 수 있습니다. AWS CloudHSM

CSR 및 HSM 인증서를 가져오려면(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 확인할 HSM의 클러스터 자격 증명 옆의 라디오 버튼을 선택합니다.
3. 작업을 선택합니다. 드롭다운 메뉴에서 초기화를 선택합니다.
4. HSM을 생성하기 위한 이전 단계를 완료하지 않은 경우, 생성 중인 HSM의 가용 영역을 선택합니다. 그런 다음 생성을 선택합니다.
5. 인증서 및 CSR가 이미 있는 경우에는 이들을 다운로드할 수 있는 링크가 나타납니다.



6. 각 링크를 선택하여 CSR 및 인증서를 다운로드하고 저장합니다. 후속 단계를 간소화하려면 모든 파일을 같은 디렉터리에 저장하고 기본 파일 이름을 사용합니다.

CSR 및 HSM 인증서를 가져오는 방법(AWS CLI)

- 명령 프롬프트에서 describe-clusters 명령을 네 번 실행하여 매번 CSR 및 다른 인증서를 추출하고 파일에 저장합니다.

  a. 다음 명령을 실행하여 클러스터 CSR을 추출합니다. *<cluster ID>*를 이전에 생성한 클러스터의 ID로 바꿉니다.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
```

```
                                    --output text \
                                    --query 'Clusters[].Certificates.ClusterCsr'
 \
                                    > <cluster ID>_ClusterCsr.csr
```

b. 다음 명령을 실행하여 HSM 인증서를 추출합니다. *<cluster ID>*를 이전에 생성한 클러스터의 ID로 바꿉니다.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                    --output text \
                                    --query
 'Clusters[].Certificates.HsmCertificate' \
                                    > <cluster ID>_HsmCertificate.crt
```

c. 다음 명령을 실행하여 AWS 하드웨어 인증서를 추출합니다. *<cluster ID>*를 이전에 생성한 클러스터의 ID로 바꿉니다.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                    --output text \
                                    --query
 'Clusters[].Certificates.AwsHardwareCertificate' \
                                    > <cluster ID>_AwsHardwareCertificate.crt
```

d. 다음 명령을 실행하여 제조업체 하드웨어 인증서를 추출합니다. *<cluster ID>*를 이전에 생성한 클러스터의 ID로 바꿉니다.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                    --output text \
                                    --query
 'Clusters[].Certificates.ManufacturerHardwareCertificate' \
                                    > <cluster
 ID>_ManufacturerHardwareCertificate.crt
```

CSR 및 HSM 인증서 (AWS CloudHSM API) 를 가져오려면

- DescribeClusters 요청을 전송한 다음, 응답에서 CSR 및 인증서를 추출하여 저장합니다.

루트 인증서 가져오기

다음 단계에 따라 제조업체의 루트 인증서를 받으십시오 AWS CloudHSM . 루트 인증서 파일을 CSR 및 HSM 인증서 파일이 들어 있는 디렉터리에 저장합니다.

AWS CloudHSM 및 제조업체 루트 인증서를 받으려면

1. AWS CloudHSM 루트 인증서 다운로드: AWS_CloudHSM_Root-G1.zip
2. HSM 유형에 적합한 제조업체 루트 인증서를 다운로드하십시오.

   - hsm1.medium 제조업체 루트 인증서: liquid_security_certificate.zip
   - hsm2m.medium 제조업체 루트 인증서: liquid_security_certificate.zip

   > **Note**
   >
   > 랜딩 페이지에서 각 인증서를 다운로드하려면 다음 링크를 사용하십시오.
   >
   > - hsm1.medium의 제조업체 루트 인증서 랜딩 페이지
   > - hsm2m.medium의 제조업체 루트 인증서 랜딩 페이지
   >
   > [Download Certificate] 링크를 마우스 오른쪽 버튼으로 클릭한 후 [Save Link As...]를 선택해서 인증서 파일을 저장해야 할 수 있습니다.

3. 파일을 다운로드한 후 압축을 풉니다.

인증서 체인 확인

이 단계에서는 두 개의 인증서 체인을 구성합니다. 하나는 루트 인증서에, 다른 하나는 제조업체 AWS CloudHSM 루트 인증서에 사용됩니다. 그런 다음 OpenSSL을 사용하여 각 인증서 체인으로 HSM 인증서를 확인합니다.

인증서 체인을 생성하려면 Linux 셸을 엽니다. OpenSSL(대부분의 Linux 셸에서 제공)이 필요하고, 다운로드한 루트 인증서 및 HSM 인증서 파일이 필요합니다. 하지만 이 단계에는 이 AWS CLI 필수가 없으며 셸을 AWS 계정과 연결할 필요도 없습니다.

AWS CloudHSM 루트 인증서로 HSM 인증서를 확인하려면

1. 다운로드한 루트 인증서 및 HSM 인증서 파일을 저장한 디렉터리로 이동합니다. 다음 명령은 모든 인증서가 현재 디렉터리에 위치하고 기본 파일 이름을 사용한다고 가정합니다.

   다음 명령을 사용하여 AWS 하드웨어 인증서와 AWS CloudHSM 루트 인증서를 순서대로 포함하는 인증서 체인을 생성합니다. *<cluster ID>*를 이전에 생성한 클러스터의 ID로 바꿉니다.

   ```
   $ cat <cluster ID>_AwsHardwareCertificate.crt \
         AWS_CloudHSM_Root-G1.crt \
         > <cluster ID>_AWS_chain.crt
   ```

2. 다음 OpenSSL 명령을 사용하여 AWS 인증서 체인으로 HSM 인증서를 확인합니다. *<cluster ID>*를 이전에 생성한 클러스터의 ID로 바꿉니다.

   ```
   $ openssl verify -CAfile <cluster ID>_AWS_chain.crt <cluster ID>_HsmCertificate.crt
   <cluster ID>_HsmCertificate.crt: OK
   ```

제조업체 루트 인증서로 HSM 인증서를 확인하려면

1. 다음 명령을 사용하여 제조업체 하드웨어 인증서와 제조업체 루트 인증서를 순서대로 포함시켜 인증서 체인을 생성합니다. *<cluster ID>*를 이전에 생성한 클러스터의 ID로 바꿉니다.

   ```
   $ cat <cluster ID>_ManufacturerHardwareCertificate.crt \
         liquid_security_certificate.crt \
         > <cluster ID>_manufacturer_chain.crt
   ```

2. 다음 OpenSSL 명령을 사용하여 제조업체 인증서 체인으로 HSM 인증서를 확인합니다. *<cluster ID>*를 이전에 생성한 클러스터의 ID로 바꿉니다.

   ```
   $ openssl verify -CAfile <cluster ID>_manufacturer_chain.crt <cluster
    ID>_HsmCertificate.crt
   <cluster ID>_HsmCertificate.crt: OK
   ```

퍼블릭 키 추출 및 비교

OpenSSL을 사용하여 HSM 인증서 및 클러스터 CSR에서 퍼블릭 키를 추출 및 비교하여 둘이 동일한지 확인합니다.

퍼블릭 키를 비교하려면 Linux 셸을 사용합니다. 대부분의 Linux 셸에서 사용할 수 있는 OpenSSL이 필요하지만 이 AWS CLI 단계에는 필요하지 않습니다. 셸을 AWS 계정과 연결할 필요는 없습니다.

퍼블릭 키를 추출 및 비교하려면

1. 다음 명령을 사용하여 HSM 인증서에서 퍼블릭 키를 추출합니다.

```
$ openssl x509 -in <cluster ID>_HsmCertificate.crt -pubkey -noout > <cluster
 ID>_HsmCertificate.pub
```

2. 다음 명령을 사용하여 클러스터 CSR에서 퍼블릭 키를 추출합니다.

```
$ openssl req -in <cluster ID>_ClusterCsr.csr -pubkey -noout > <cluster
 ID>_ClusterCsr.pub
```

3. 다음 명령을 사용하여 퍼블릭 키를 비교합니다. 퍼블릭 키들이 동일한 경우 다음 명령을 실행해도 출력이 생성되지 않습니다.

```
$ diff <cluster ID>_HsmCertificate.pub <cluster ID>_ClusterCsr.pub
```

HSM의 ID 및 신뢰성을 확인한 후에는 클러스터 초기화로 넘어갑니다.

클러스터 초기화

다음 항목의 단계를 완료하여 클러스터를 초기화하십시오. AWS CloudHSM

> **Note**
>
> 클러스터를 초기화하기 전에 HSM의 자격 증명 및 신뢰성을 확인할 수 있는 프로세스를 검토합니다. 이 프로세스는 선택 사항이며, 클러스터가 초기화될 때까지만 유효합니다. 클러스터가 초기화된 후에는 이 프로세스를 사용하여 인증서를 가져오거나 HSM을 확인할 수 없습니다.

주제

- 클러스터 CSR 가져오기
- CSR에 서명
- 클러스터 초기화

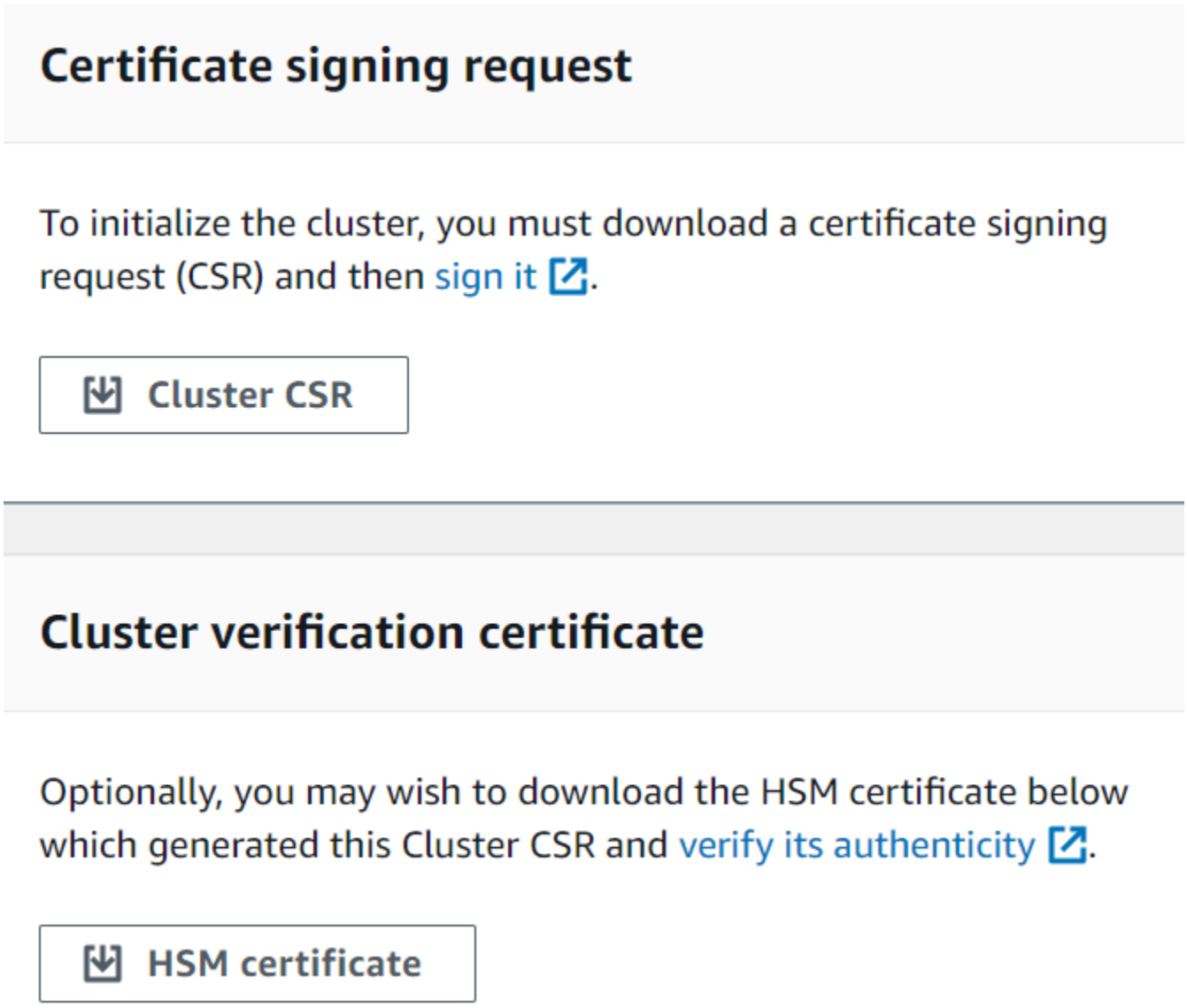
클러스터 CSR 가져오기

클러스터를 초기화하려면 먼저, 클러스터의 첫 번째 HSM에서 생성된 인증서를 다운로드하여 서명 요청(CSR)에 서명해야 합니다. 단계에 따라 클러스터의 HSM 자격 증명을 확인했다면 이미 CSR이 있는 것이므로 CSR에 서명할 수 있습니다. 그렇지 않으면 AWS CloudHSM 콘솔, AWS Command Line Interface (AWS CLI) 또는 API를 사용하여 지금 CSR을 가져오십시오. AWS CloudHSM

> **Important**
>
> 클러스터를 초기화하려면 트러스트 앵커가 RFC 5280을 준수하고 다음 요구 사항을 충족해야 합니다.
>
> - X509v3 확장을 사용하는 경우 X509v3 기본 제약 조건 확장이 있어야 합니다.
> - 트러스트 앵커는 자체 서명된 인증서여야 합니다.
> - 확장 값은 서로 충돌하지 않아야 합니다.

CSR을 가져오려면(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 확인할 HSM의 클러스터 자격 증명 옆의 라디오 버튼을 선택합니다.
3. 작업을 선택합니다. 드롭다운 메뉴에서 초기화를 선택합니다.
4. HSM을 생성하기 위한 이전 단계를 완료하지 않은 경우, 생성 중인 HSM의 가용 영역을 선택합니다. 그런 다음 생성을 선택합니다.
5. CSR이 이미 있는 경우에는 이를 다운로드할 수 있는 링크가 나타납니다.



6. [Cluster CSR]을 선택하여 CSR을 다운로드하고 저장합니다.

CSR을 가져오는 방법(AWS CLI)

- 명령 프롬프트에서 다음 describe-clusters 명령을 실행하여 CSR을 추출하고 파일에 저장합니다. *<cluster ID>*를 이전에 생성한 클러스터의 ID로 바꿉니다.

```
$ aws cloudhsmv2 describe-clusters --filters clusterIds=<cluster ID> \
                                   --output text \
                                   --query 'Clusters[].Certificates.ClusterCsr' \
                                   > <cluster ID>_ClusterCsr.csr
```

CSR (AWS CloudHSM API) 을 받으려면

1. DescribeClusters 요청을 보냅니다.
2. 응답에서 CSR을 추출하고 저장합니다.

CSR에 서명

현재는 자체 서명된 서명 인증서를 만들고 클러스터의 CSR에 서명하는 데 사용해야 합니다. 이 단계에는 가 AWS CLI 필요하지 않으며 셸을 AWS 계정과 연결할 필요도 없습니다. CSR에 서명하려면 다음을 수행해야 합니다.

1. 이전 섹션을 완료합니다 (클러스터 CSR 가져오기 참조).
2. 프라이빗 키를 만듭니다.
3. 프라이빗 키를 사용하여 서명 인증서를 만듭니다.
4. 클러스터 CSR에 서명합니다.

프라이빗 키를 만듭니다

> **Note**
>
> 프로덕션 클러스터의 경우 신뢰할 수 있는 임의 소스를 사용하여 안전한 방식으로 키를 생성해야 합니다. 안전한 오프사이트 및 오프라인 HSM이나 이에 상응하는 디바이스를 사용하는 것이 좋습니다. 키를 안전하게 저장하십시오. 키는 클러스터의 ID와 클러스터에 포함된 HSM에 대한 사용자의 단독 제어권을 설정합니다.
> 개발 및 테스트 중에 원하는 편리한 도구(예: OpenSSL)를 사용하여 클러스터 인증서를 만들고 서명할 수 있습니다. 다음 예에서는 키를 생성하는 방법을 보여줍니다. 키를 사용하여 자체 서명된 인증서를 생성한 후(아래 참조), 안전한 방식으로 저장해야 합니다. AWS CloudHSM 인스턴스에 로그인하려면 인증서가 있어야 하지만 프라이빗 키는 없습니다.

다음 명령을 사용하여 퍼블릭 키를 생성합니다. AWS CloudHSM 클러스터를 초기화할 때는 RSA 2048 인증서 또는 RSA 4096 인증서를 사용해야 합니다.

```
$ openssl genrsa -aes256 -out customerCA.key 2048
Generating RSA private key, 2048 bit long modulus
........+++
```

```
.............+++
e is 65537 (0x10001)
Enter pass phrase for customerCA.key:
Verifying - Enter pass phrase for customerCA.key:
```

프라이빗 키를 사용하여 자체 서명된 인증서 생성

프로덕션 클러스터의 프라이빗 키를 생성하는 데 사용하는 신뢰할 수 있는 하드웨어는 해당 키를 통해 자체 서명된 인증서를 생성하기 위한 소프트웨어 도구도 제공해야 합니다. 다음 예에서는 OpenSSL과 이전 단계에서 만든 프라이빗 키를 사용하여 서명 인증서를 생성합니다. 인증서는 10년(3,652일) 동안 유효합니다. 화면의 지침을 읽고 프롬프트의 메시지를 따릅니다.

```
$ openssl req -new -x509 -days 3652 -key customerCA.key -out customerCA.crt
Enter pass phrase for customerCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

이 명령은 customerCA.crt 인증서 파일을 생성합니다. 클러스터에 연결할 모든 호스트에 이 인증서를 설치하십시오. AWS CloudHSM 파일에 다른 이름을 지정하거나 호스트의 루트가 아닌 다른 경로에 파일을 저장하는 경우, 그에 따라 클라이언트 구성 파일을 편집해야 합니다. 방금 생성한 인증서와 프라이빗 키를 사용하여 다음 단계에서 클러스터 CSR(인증서 서명 요청)에 서명합니다.

클러스터 CSR에 서명

프로덕션 클러스터의 프라이빗 키를 생성하는 데 사용하는 신뢰할 수 있는 하드웨어는 해당 키를 통해 CSR에 서명하기 위한 도구도 제공해야 합니다. 다음 예에서는 OpenSSL을 사용하여 클러스터의 CSR에 서명합니다. 이 예에서는 이전 단계에서 생성한 프라이빗 키와 자체 서명된 인증서를 사용합니다.

```
$ openssl x509 -req -days 3652 -in <cluster ID>_ClusterCsr.csr \
                               -CA customerCA.crt \
                               -CAkey customerCA.key \
                               -CAcreateserial \
                               -out <cluster ID>_CustomerHsmCertificate.crt
Signature ok
subject=/C=US/ST=CA/O=Cavium/OU=N3FIPS/L=SanJose/CN=HSM:<HSM
 identifer>:PARTN:<partition number>, for FIPS mode
Getting CA Private Key
Enter pass phrase for customerCA.key:
```

이 명령은 *<cluster ID>*_CustomerHsmCertificate.crt 파일을 생성합니다. 클러스터를 초기화할 때 이 파일을 서명된 인증서로 사용합니다.

클러스터 초기화

서명한 HSM 인증서와 서명 인증서를 사용하여 클러스터를 초기화하십시오. AWS CloudHSM 콘솔 AWS CLI, 또는 AWS CloudHSM API를 사용할 수 있습니다.

클러스터를 초기화하려면(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 확인할 HSM의 클러스터 자격 증명 옆의 라디오 버튼을 선택합니다.
3. 작업을 선택합니다. 드롭다운 메뉴에서 초기화를 선택합니다.
4. HSM을 생성하기 위한 이전 단계를 완료하지 않은 경우, 생성 중인 HSM의 가용 영역을 선택합니다. 그런 다음 생성을 선택합니다.
5. [Download certificate signing request] 페이지에서 [Next]를 선택합니다. [Next]가 비활성화 되어 있는 경우에는 먼저 CSR 또는 인증서 링크 중 하나를 선택합니다. 다음을 선택합니다.
6. [Sign certificate signing request (CSR)] 페이지에서 [Next]를 선택합니다.
7. [Upload the certificates] 페이지에서 다음 작업을 수행하십시오.

    a. 클러스터 인증서 옆의 파일 업로드를 선택합니다. 그러면 이전에 서명한 HSM 인증서 위치를 확인하고 선택합니다. 이전 섹션에 나온 단계들을 완료했다면 *<cluster ID>*_CustomerHsmCertificate.crt라는 파일을 선택합니다.

    b. Issuing certificate(인증서 발급하기) 옆의 파일 업로드를 선택합니다. 그런 다음 서명 인증서를 선택합니다. 이전 섹션에 나온 단계들을 완료했다면 customerCA.crt라는 파일을 선택합니다.

c. [Upload and initialize]를 선택합니다.

클러스터 초기화 방법(AWS CLI)

- 명령 프롬프트에서 initialize-cluster 명령을 실행합니다. 다음을 제공합니다.
  - 이전에 생성한 클러스터의 ID입니다.
  - 이전에 서명한 HSM 인증서입니다. 이전 섹션에 나온 단계들을 완료하면 *<cluster ID>*_CustomerHsmCertificate.crt라는 파일에 저장이 됩니다.
  - 서명 인증서입니다. 이전 섹션에 나온 단계를 완료하면 서명 인증서가 customerCA.crt라는 파일에 저장됩니다.

```
$ aws cloudhsmv2 initialize-cluster --cluster-id <cluster ID> \
                                    --signed-cert file://<cluster
 ID>_CustomerHsmCertificate.crt \
                                    --trust-anchor file://customerCA.crt
{
    "State": "INITIALIZE_IN_PROGRESS",
    "StateMessage": "Cluster is initializing. State will change to INITIALIZED upon
 completion."
}
```

클러스터 (AWS CloudHSM API) 를 초기화하려면

- 다음과 함께 InitializeCluster 요청을 전송합니다.
  - 이전에 생성한 클러스터의 ID입니다.
  - 이전에 서명한 HSM 인증서입니다.
  - 서명 인증서입니다.

CloudHSM CLI 설치 및 구성

AWS CloudHSM 클러스터의 HSM과 상호 작용하려면 CloudHSM CLI가 필요합니다.

Tasks

- 명령줄 도구를 설치합니다. AWS CloudHSM

명령줄 도구를 설치합니다. AWS CloudHSM

클라이언트 인스턴스에 연결하고 다음 명령을 실행하여 AWS CloudHSM 명령줄 도구를 다운로드하고 설치합니다. 자세한 정보는 Amazon EC2 클라이언트 인스턴스 시작하기을 참조하세요.

다음 명령을 사용하여 CloudHSM CLI를 다운로드하고 설치합니다.

Amazon Linux 2

x86_64 아키텍처의 Amazon Linux 2:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

ARM64 아키텍처의 Amazon Linux 2:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.aarch64.rpm
```

Amazon Linux 2023

x86_64 아키텍처의 아마존 리눅스 2023:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

ARM64 아키텍처의 아마존 리눅스 2023:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

CentOS 7 (7.8+)

x86_64 아키텍처 기반 CentOS 7:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

RHEL 7 (7.8+)

x86_64 아키텍처 기반 RHEL 7:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

x86_64 아키텍처의 RHEL 8:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.x86_64.rpm
```

RHEL 9 (9.2+)

x86_64 아키텍처의 RHEL 9:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.x86_64.rpm
```

ARM64 아키텍처의 RHEL 9:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.aarch64.rpm
```

Ubuntu 20.04 LTS

x86_64 아키텍처의 Ubuntu 20.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-cli_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u20.04_amd64.deb
```

Ubuntu 22.04 LTS

x86_64 아키텍처의 Ubuntu 22.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-cli_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_amd64.deb
```

ARM64 아키텍처의 우분투 22.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-cli_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_arm64.deb
```

Windows Server 2016

x86_64 아키텍처의 Windows Server 2016의 경우 관리자 PowerShell 권한으로 열고 다음 명령을 실행합니다.

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2019

x86_64 아키텍처의 Windows Server 2019의 경우 관리자 PowerShell 권한으로 열고 다음 명령을 실행합니다.

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/
AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

다음 명령을 사용하여 CloudHSM CLI를 구성합니다.

Client SDK 5용 Linux EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM(s)의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IP addresses of the HSMs>
```

Client SDK 5용 Windows EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM(s)의 IP 주소를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI IP addresses
 of the HSMs>
```

클러스터 활성화

클러스터를 활성화하면 AWS CloudHSM 클러스터의 상태가 초기화에서 활성으로 바뀝니다. 그러면 하드웨어 보안 모듈(HSM) 사용자를 관리하고 HSM을 사용할 수 있습니다.

Important

클러스터를 활성화하려면 먼저 클러스터에 연결되는 각 EC2 인스턴스의 플랫폼 기본 위치에 발급 인증서를 복사해야 합니다(클러스터를 초기화할 때 발급 인증서를 생성합니다).

Linux

```
/opt/cloudhsm/etc/customerCA.crt
```

Windows

```
C:\ProgramData\Amazon\CloudHSM\customerCA.crt
```

발급 인증서를 배치한 후 CloudHSM CLI를 설치하고 첫 번째 HSM에서 cluster activate 명령을 실행합니다. 클러스터의 첫 번째 HSM 관리자 계정에 비활성화된 관리자 역할이 있는 것을 확인할 수 있습니다. 이 역할은 클러스터를 활성화하기 전에만 존재하는 임시 역할입니다. 클러스터를 활성화하면 활성화되지 않은 관리자 역할이 관리자로 변경됩니다.

클러스터 활성화

1. 이전에 시작한 클라이언트 인스턴스에 연결합니다. 자세한 정보는 Amazon EC2 클라이언트 인스턴스 시작하기을 참조하세요. Linux 인스턴스나 Windows Server를 실행할 수 있습니다.
2. CloudHSM CLI를 대화형 모드에서 실행합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

3. (선택 사항) user list 명령을 사용하여 기존 사용자를 표시합니다.

   ```
   aws-cloudhsm > user list
   {
     "error_code": 0,
     "data": {
       "users": [
   ```

```
      {
        "username": "admin",
        "role": "unactivated-admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

4. cluster activate 명령을 사용하여 초기 관리자 암호를 설정합니다.

```
aws-cloudhsm > cluster activate
Enter
 password:<NewPassword>
Confirm password:<NewPassword>
{
  "error_code": 0,
  "data": "Cluster activation successful"
}
```

암호 워크시트에 새 암호를 적어 두는 것이 좋습니다. 워크시트를 분실하지 마십시오. 암호 워크시트의 복사본을 인쇄하여 중요 HSM 암호를 기록한 다음 안전한 장소에 보관하는 것이 좋습니다. 또한, 안전한 외부 장소에 있는 스토리지에 이 워크시트의 복사본을 보관해두는 것이 좋습니다.

5. (선택 사항) user list 명령을 사용하여 사용자 유형이 관리자/CO로 변경되었는지 확인합니다.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
```

```
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
       {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

6. quit 명령을 사용하여 CloudHSM CLI 도구를 중지합니다.

```
aws-cloudhsm > quit
```

CloudHSM CLI 또는 CMU와 작업하는 방법에 대한 자세한 내용은 HSM 사용자 이해 및 CMU의 HSM 사용자 관리 이해를 참조하십시오.

새 인증서 및 프라이빗 키로 SSL 재구성(선택 사항)

AWS CloudHSM SSL 인증서를 사용하여 HSM에 대한 연결을 설정합니다. 기본 키와 SSL 인증서는 클라이언트를 설치할 때 포함됩니다. 하지만 직접 만들어서 사용할 수도 있습니다. 클러스터를 초기화할 때 만든 자체 서명 인증서(*customerCA.crt*)가 필요합니다.

상위 레벨에서는 다음과 같은 2단계 프로세스입니다.

1. 먼저 프라이빗 키를 생성한 다음 해당 키를 사용하여 인증서 서명 요청(CSR)을 생성합니다. 클러스터를 초기화할 때 생성한 인증서인 발급 인증서를 사용하여 CSR에 서명합니다.
2. 다음으로 구성 도구를 사용하여 키와 인증서를 적절한 디렉터리에 복사합니다.

키와 CSR을 생성한 다음 CSR에 서명합니다.

Client SDK 3 또는 Client SDK 5의 단계는 동일합니다.

새 인증서와 프라이빗 키로 SSL 재구성하려면

1. 다음 OpenSSL 명령을 사용하여 프라이빗 키를 만듭니다.

```
openssl genrsa -out ssl-client.key 2048
Generating RSA private key, 2048 bit long modulus
........+++
............+++
e is 65537 (0x10001)
```

2. 다음 OpenSSL 명령을 사용하여 인증서 서명 요청(CSR)을 생성합니다. 인증서에 대해 일련의 질문을 받게 됩니다.

```
openssl req -new -sha256 -key ssl-client.key -out ssl-client.csr
Enter pass phrase for ssl-client.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:
State or Province Name (full name) []:
Locality Name (eg, city) [Default City]:
Organization Name (eg, company) [Default Company Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (eg, your name or your server's hostname) []:
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

3. 클러스터를 초기화할 때 만든 *customerCA.crt* 인증서를 사용하여 CSR에 서명합니다.

```
openssl x509 -req -days 3652 -in ssl-client.csr \
        -CA customerCA.crt \
        -CAkey customerCA.key \
        -CAcreateserial \
```

```
        -out ssl-client.crt
Signature ok
subject=/C=US/ST=WA/L=Seattle/O=Example Company/OU=sales
Getting CA Private Key
```

사용자 지정 SSL을 활성화하는 경우 AWS CloudHSM

Client SDK 3 또는 Client SDK 5의 단계는 동일합니다. 구성 명령줄 도구 작업에 대한 자세한 내용은 ??? 단원을 참조하십시오.

주제

- Client SDK 3용 사용자 정의 SSL
- Client SDK 5용 사용자 지정 SSL

Client SDK 3용 사용자 정의 SSL

사용자 지정 SSL을 활성화하려면 Client SDK 3용 구성 도구를 사용하세요. Client SDK 3의 구성 도구에 대한 자세한 내용은 ??? 단원을 참조하십시오.

Linux용 Client SDK 3을 사용한 TLS 클라이언트-서버 상호 인증에 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. 구성 도구를 사용하여 `ssl-client.crt` 및 `ssl-client.key`을 지정합니다.

```
sudo /opt/cloudhsm/bin/configure --ssl \
 --pkey /opt/cloudhsm/etc/ssl-client.key \
 --cert /opt/cloudhsm/etc/ssl-client.crt
```

3. `customerCA.crt` 인증서를 트러스트 스토어에 추가합니다. 인증서 주체 이름 해시를 생성합니다. 이 해시는 해당 이름으로 인증서를 찾을 수 있는 인덱스를 생성합니다.

```
openssl x509 -in /opt/cloudhsm/etc/customerCA.crt -hash | head -n 1
1234abcd
```

디렉터리를 생성합니다.

```
mkdir /opt/cloudhsm/etc/certs
```

해당 해시 이름의 인증서를 포함하는 파일을 만듭니다.

```
sudo cp /opt/cloudhsm/etc/customerCA.crt /opt/cloudhsm/etc/certs/1234abcd.0
```

Client SDK 5용 사용자 지정 SSL

사용자 지정 SSL을 활성화하려면 Client SDK 5용 구성 도구를 사용하세요. Client SDK 5의 구성 도구에 대한 자세한 내용은 ??? 단원을 참조하십시오.

PKCS #11 library

Linux용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증을 위한 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. 구성 도구를 사용하여 `ssl-client.crt` 및 `ssl-client.key`을 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Windows용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증에 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. PowerShell 인터프리터를 사용하는 경우 구성 도구를 사용하여 `ssl-client.crt` 및 `ssl-client.key` 을 지정합니다.

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

OpenSSL Dynamic Engine

Linux용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증을 위한 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. 구성 도구를 사용하여 `ssl-client.crt` 및 `ssl-client.key`을 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-dyn \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

JCE provider

Linux용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증을 위한 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. 구성 도구를 사용하여 `ssl-client.crt` 및 `ssl-client.key`을 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-jce \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
```

```
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Windows용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증에 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. PowerShell 인터프리터의 경우 구성 도구를 사용하여 및 을 지정합니다`ssl-client.crt`. `ssl-client.key`

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

CloudHSM CLI

Linux용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증을 위한 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. 구성 도구를 사용하여 `ssl-client.crt` 및 `ssl-client.key`을 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Windows용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증에 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. PowerShell 인터프리터의 경우 구성 도구를 사용하여 및 을 지정합니다`ssl-client.crt`. `ssl-client.key`

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

애플리케이션 빌드하기

를 사용하여 애플리케이션을 빌드하고 키로 AWS CloudHSM작업하세요.

새 클러스터에서 키를 생성하고 사용하려면 먼저 CloudHSM 관리 유틸리티(CMU)를 사용하여 하드웨어 보안 모듈(HSM) 사용자를 생성해야 합니다. 자세한 내용은 HSM 사용자 관리 태스크 이해, AWS CloudHSM 명령줄 인터페이스 (CLI) 시작하기 및 HSM 사용자 관리 방법을 참조하십시오.

> **Note**
>
> 클라이언트 SDK 3을 사용하는 경우 CloudHSM CLI 대신 CloudHSM 관리 유틸리티(CMU)를 사용하십시오.

HSM 사용자가 제자리에 있으면 HSM에 로그인하여 다음 옵션 중 하나로 키를 생성하고 사용할 수 있습니다.

- 명령줄 도구인 키 관리 유틸리티 사용하기
- PKCS #11 라이브러리를 사용하여 C 애플리케이션 구축하기
- JCE 공급자를 사용하여 Java 애플리케이션을 빌드하기
- 명령줄에서 직접 OpenSSL Dynamic Engine 사용하기

- NGINX 및 Apache 웹 서버와 함께 TLS 오프로드에 OpenSSL Dynamic Engine 사용하기
- CNG 및 KSP 공급자를 사용하여 AWS CloudHSM Microsoft Windows 서버 인증 기관 (CA) 과 함께 사용할 수 있습니다.
- CNG 및 KSP 공급자를 사용하여 AWS CloudHSM Microsoft 서명 도구와 함께 사용할 수 있습니다.
- 인터넷 정보 서버(IIS) 웹 서버와 함께 TLS 오프로드에 CNG 및 KSP 공급자 사용하기

에 대한 모범 사례 AWS CloudHSM

AWS CloudHSM을 효과적으로 사용하려면 이 주제의 모범 사례를 수행하십시오.

내용

클러스터 관리

AWS CloudHSM 클러스터를 생성, 액세스 및 관리할 때는 이 섹션의 모범 사례를 따르십시오.

피크 트래픽을 처리할 수 있도록 클러스터 확장

클라이언트 인스턴스 크기, 클러스터 크기, 네트워크 지형, 사용 사례에 필요한 암호화 작업 등 여러 요인이 클러스터가 처리할 수 있는 최대 처리량에 영향을 미칠 수 있습니다.

시작점으로 일반적인 클러스터 크기 및 구성에 대한 성능 추정치는 주제 AWS CloudHSM 성능 섹션을 참조하십시오. 예상되는 최대 부하로 클러스터를 부하 테스트하여 현재 아키텍처가 복원력이 있고 적절한 규모인지 확인하는 것이 좋습니다.

고가용성을 위한 클러스터 설계

유지 관리를 위해 이중화 추가: 예정된 유지 관리 또는 문제가 감지된 경우 HSM을 교체할 AWS 수 있습니다. 일반적으로 클러스터 크기에는 +1 이상의 중복성이 있어야 합니다. 예를 들어, 피크 타임에 서비스를 운영하기 위해 2개의 HSM이 필요한 경우 이상적인 클러스터 크기는 3입니다. 가용성과 관련된 모범 사례를 따른다면 이러한 HSM 교체가 서비스에 영향을 미치지 않을 것입니다. 그러나 교체된 HSM에서 진행 중인 작업은 실패할 수 있으므로 다시 시도해야 합니다.

HSM을 여러 가용 영역으로 분산: 가용 영역 장애 발생 시 서비스를 어떻게 운영할 수 있을지 생각해 보십시오. AWS 에서는 가능한 한 많은 가용 영역에 HSM을 분산할 것을 권장합니다. HSM이 3개인 클러스터의 경우 HSM을 3개의 가용 영역에 분산해야 합니다. 시스템에 따라 추가 중복성이 필요할 수 있습니다.

새로 생성된 키의 내구성을 보장하려면 HSM이 3개 이상 있어야 합니다.

새로 생성된 키의 내구성을 필요로 하는 애플리케이션의 경우 한 리전에 있는 다양한 가용 영역에 3개 이상의 HSM을 분산시키는 것이 좋습니다.

클러스터에 대한 보안 액세스

프라이빗 서브넷을 사용하여 인스턴스에 대한 액세스 제한: VPC의 프라이빗 서브넷에서 HSM과 클라이언트 인스턴스를 시작하십시오. 이렇게 하면 외부에서 HSM에 액세스하는 것이 제한됩니다.

VPC 엔드포인트를 사용하여 API에 액세스: AWS CloudHSM 데이터 플레인은 인터넷 또는 AWS API에 액세스할 필요 없이 작동하도록 설계되었습니다. 클라이언트 인스턴스에서 AWS CloudHSM API에 액세스해야 하는 경우 클라이언트 인스턴스에서 인터넷에 액세스할 필요 없이 VPC 엔드포인트를 사용하여 API에 액세스할 수 있습니다. 자세한 정보는 AWS CloudHSM 및 VPC 엔드포인트을 참조하세요.

클라이언트-서버 통신을 보호하도록 SSL 재구성: TLS를 AWS CloudHSM 사용하여 HSM에 대한 연결을 설정합니다. 클러스터를 초기화한 후 외부 TLS 연결을 설정하는 데 사용되는 기본 TLS 인증서와 키를 교체할 수 있습니다. 자세한 내용은 SSL/TLS 오프로드를 통해 웹 서버 보안을 개선하세요. AWS CloudHSM 섹션을 참조하십시오.

필요에 맞게 확장하여 비용 절감

AWS CloudHSM사용에 따른 선결제 비용은 없습니다. HSM을 종료할 때까지 시작한 각 HSM에 대해 시간당 요금을 지불합니다. 서비스를 계속 사용할 필요가 없는 경우 필요하지 않을 때 HSM을 0으로 축소 (삭제) 하여 비용을 절감할 수 있습니다. AWS CloudHSM HSM이 다시 필요한 경우 백업에서 HSM을 복원할 수 있습니다. 예를 들어 한 달에 한 번, 특히 말일에 코드에 서명해야 하는 워크로드가 있는 경우 먼저 클러스터를 확장하고, 작업이 완료된 후 HSM을 삭제하여 축소하고, 다음 달 말에 서명 작업을 다시 수행하도록 클러스터를 복원할 수 있습니다.

AWS CloudHSM 클러스터에 있는 HSM을 정기적으로 자동으로 백업합니다. 나중에 새 HSM을 추가하면 최신 백업이 새 HSM에 복원되므로 이전 위치에서 다시 사용할 수 있습니다. AWS CloudHSM AWS CloudHSM 아키텍처 비용을 계산하려면 요금을 참조하십시오AWS CloudHSM .

관련 리소스:

- 백업의 일반 개요
- 백업 보관 정책

- AWS 지역 간 백업 복사

HSM 사용자 관리

AWS CloudHSM 클러스터의 사용자를 효과적으로 관리하려면 이 섹션의 모범 사례를 따르십시오. HSM 사용자는 IAM 사용자와 다릅니다. 적절한 권한이 있는 ID 기반 정책을 보유한 IAM 사용자 및 엔터티는 API를 통해 리소스와 상호 작용하여 HSM을 생성할 수 있습니다. HSM을 생성한 후에는 HSM 사용자 보안 인증 정보를 사용하여 HSM에서의 작업을 인증해야 합니다. HSM 사용자에 대한 자세한 안내는 HSM 사용자 관리 AWS CloudHSM 섹션을 참조하십시오.

HSM 사용자의 보안 인증 정보 보호

HSM 사용자는 HSM에서 암호화 및 관리 작업을 수행하고 액세스할 수 있는 엔터티이므로 HSM 사용자의 보안 인증 정보를 안전하게 보호하는 것이 필수적입니다. AWS CloudHSM 는 HSM 사용자 보안 인증 정보에 액세스할 수 없으며 액세스 권한을 상실할 경우 지원해 드릴 수 없습니다.

관리자를 두 명 이상 두고 계정 잠금 방지

클러스터에서 잠기지 않도록 하려면 관리자 암호 하나를 분실할 경우에 대비하여 최소 두 명의 관리자를 두는 것이 좋습니다. 이 경우 다른 관리자를 통해 암호를 재설정할 수 있습니다.

> **Note**
>
> 클라이언트 SDK 5의 관리자는 Client SDK 3의 CO(Crypto Officer)와 동의어입니다.

모든 사용자 관리 작업에 쿼럼 활성화

쿼럼을 사용하면 작업을 수행하기 전에 사용자 관리 작업을 승인해야 하는 최소 관리자 수를 설정할 수 있습니다. 관리자가 가질 수 있는 권한 때문에 모든 사용자 관리 작업에 대해 쿼럼을 활성화하는 것이 좋습니다. 이렇게 하면 관리자 암호 중 하나가 손상될 경우 미칠 수 있는 영향을 제한할 수 있습니다. 자세한 내용은 쿼럼 관리를 참조하십시오.

각각 권한이 제한된 암호 사용자를 여러 명 생성

암호 사용자의 책임을 분리함으로써 어느 사용자도 전체 시스템을 완전히 통제할 수 없게 됩니다. 따라서 여러 암호 사용자를 만들고 각 사용자의 권한을 제한하는 것이 좋습니다. 일반적으로 이 작업은 암

호 사용자마다 수행하는 책임과 작업이 확연히 다릅니다(예: 한 명의 암호 사용자가 키를 생성하고 다른 암호 사용자와 공유한 다음 이를 애플리케이션에서 사용하는 경우).

관련 리소스:

- 키 공유
- 키 공유 취소

HSM 키 관리

AWS CloudHSM의 키를 관리할 때는 이 섹션의 모범 사례를 따르십시오.

적합한 키 유형 선택

세션 키를 사용할 경우 초당 트랜잭션(TPS)은 키가 있는 HSM 1개로 제한됩니다. 클러스터의 추가 HSM은 해당 키에 대한 요청 처리량을 증가시키지 않습니다. 동일한 애플리케이션에 토큰 키를 사용하는 경우 요청이 클러스터에서 사용 가능한 모든 HSM에 걸쳐 로드 밸런싱됩니다. 자세한 내용은 의 키 동기화 및 내구성 설정 AWS CloudHSM 섹션을 참조하십시오.

키 스토리지 한도 관리

HSM에는 한 번에 HSM에 저장할 수 있는 최대 토큰 및 세션 키 수에 제한이 있습니다. 키 스토리지 한도에 대한 자세한 내용은 AWS CloudHSM 할당량 섹션을 참조하십시오. 애플리케이션에 한도 이상이 필요한 경우 다음 전략 중 하나 이상을 사용하여 키를 효과적으로 관리할 수 있습니다.

신뢰할 수 있는 래핑을 사용하여 외부 데이터 스토리지에 키 저장: 신뢰할 수 있는 키 래핑을 사용하면 모든 키를 외부 데이터 스토리지에 래핑하여 저장함으로써 키 스토리지 한도를 극복할 수 있습니다. 이 키를 사용해야 하는 경우 키를 HSM에서 세션 키로 해제하고 필요한 작업에 키를 사용한 다음 세션 키를 폐기할 수 있습니다. 원본 키 데이터는 필요할 때 언제든지 사용할 수 있도록 데이터 스토어에 안전하게 보관됩니다. 신뢰할 수 있는 키를 사용하면 보호를 극대화할 수 있습니다.

클러스터 간 키 배포: 키 스토리지 한도를 극복하기 위한 또 다른 전략은 키를 여러 클러스터에 저장하는 것입니다. 이 접근 방식에서는 각 클러스터에 저장된 키의 매핑을 유지 관리합니다. 이 매핑을 사용하여 필요한 키를 사용하여 클라이언트 요청을 클러스터로 라우팅할 수 있습니다. 동일한 클라이언트 애플리케이션에서 여러 클러스터에 연결하는 방법에 대한 내용은 다음 주제를 참조하십시오.

- JCE 공급자를 통해 여러 클러스터에 연결
- PKCS #11를 사용하여 여러 슬롯에 연결

키 래핑 관리 및 보안

키는 EXTRACTABLE 속성을 통해 추출 가능 또는 추출 불가능으로 표시될 수 있습니다. 기본적으로 HSM 키는 추출 가능으로 표시됩니다.

추출 가능한 키는 키 래핑을 통해 HSM에서 내보낼 수 있는 키입니다. 래핑된 키는 암호화되어 있으며 동일한 래핑 키를 사용하여 래핑을 해제해야 사용할 수 있습니다. 추출할 수 없는 키는 어떤 상황에서도 HSM에서 내보낼 수 없습니다. 추출할 수 없는 키를 추출 가능하게 만드는 방법은 없습니다. 따라서 키를 추출할 수 있어야 하는지 여부를 고려하고 그에 따라 해당 키 속성을 설정하는 것이 중요합니다.

애플리케이션에 키 래핑이 필요한 경우 신뢰할 수 있는 키 래핑을 활용하여 관리자가 신뢰할 수 있는 것으로 명시적으로 표시한 키만 HSM 사용자가 래핑/언래핑할 수 있도록 제한해야 합니다. 자세한 내용은 에서 키 관리 AWS CloudHSM의 신뢰할 수 있는 키 랩핑에 대한 항목을 참조하십시오.

관련 리소스

- 랩 및 언랩 해제 기능
- JCE의 암호 함수
- 지원되는 Java 키 속성
- CloudHSM CLI의 키 속성

애플리케이션 통합

이 섹션의 모범 사례에 따라 애플리케이션이 AWS CloudHSM 클러스터와 통합되는 방식을 최적화하십시오.

클라이언트 SDK 부트스트랩

클라이언트 SDK를 클러스터에 연결하려면 먼저 부트스트랩해야 합니다. 클러스터에 IP 주소를 부트스트래핑할 때 가능한 경우 `--cluster-id` 파라미터를 사용하는 것이 좋습니다. 이 방법을 사용하면 개별 주소를 추적할 필요 없이 클러스터의 모든 HSM IP 주소로 구성을 채울 수 있습니다. 이렇게 하면 HSM이 유지 관리 중이거나 가용 영역이 중단되는 경우 애플리케이션 초기화의 복원력이 향상됩니다. 자세한 내용은 Client SDK 부트스트랩합니다. 섹션을 참조하십시오.

작업 수행을 위한 인증

AWS CloudHSM에서는 암호화 작업과 같은 대부분의 작업을 수행하려면 먼저 클러스터에 인증을 받아야 합니다.

CloudHSM CLI를 사용하여 인증: CloudHSM CLI를 사용하면 단일 명령 모드 또는 대화형 모드를 사용하여 인증할 수 있습니다. login 명령을 사용하여 대화형 모드에서 인증할 수 있습니다. 단일 명령 모드에서 인증하려면 환경 변수(CLOUDHSM_ROLE 및 CLOUDHSM_PIN)를 설정해야 합니다. 이 작업에 대한 자세한 내용은 단일 명령 모드 섹션을 참조하십시오. AWS CloudHSM 는 애플리케이션에서 사용하지 않을 때는 HSM 보안 인증 정보를 안전하게 저장하는 것을 권장합니다.

PKCS #11 인증: PKCS #11 에서는 C_를 사용하여 세션을 연 후 C_Login API를 사용하여 로그인합니다. OpenSession 슬롯(클러스터)당 하나의 C_Login만 수행하면 됩니다. 로그인에 성공하면 추가 로그인 작업을 수행할 필요 없이 OpenSession C_를 사용하여 추가 세션을 열 수 있습니다. PKCS #11 인증에 대한 예는 PKCS #11 라이브러리의 코드 샘플 섹션을 참조하십시오.

JCE 인증: AWS CloudHSM JCE 제공자는 암시적 로그인과 명시적 로그인을 모두 지원합니다. 사용 사례에 따라 효과가 있는 방법이 다릅니다. 애플리케이션이 클러스터에서 연결이 끊어져 재인증이 필요한 경우 SDK가 자동으로 인증을 처리하므로 가능하면 암시적 로그인을 사용하는 것이 좋습니다. 애플리케이션 코드를 제어할 수 없는 통합을 사용하는 경우에도 암시적 로그인을 사용하면 애플리케이션에 보안 인증 정보를 제공할 수 있습니다. 로그인 방법에 대한 자세한 내용은 JCE 공급자에게 자격 증명을 제공하십시오. 섹션을 참조하십시오.

OpenSSL로 인증: OpenSSL 동적 엔진을 사용하면 환경 변수를 통해 보안 인증 정보를 제공합니다. AWS CloudHSM 는 애플리케이션에서 사용하지 않을 때 HSM 보안 인증 정보를 안전하게 저장할 것을 권장합니다. 가능하면 수동 입력 없이 이러한 환경 변수를 체계적으로 검색하고 설정하도록 환경을 구성해야 합니다. OpenSSL 인증에 대한 자세한 내용은 OpenSSL Dynamic Engine 설치 섹션을 참조하십시오.

애플리케이션의 키를 효과적으로 관리

키 속성을 사용하여 키가 수행할 수 있는 작업 제어: 키를 생성할 때 키 속성을 사용하여 해당 키에 대한 특정 유형의 작업을 허용하거나 거부하는 권한 집합을 정의하십시오. 작업을 완료하는 데 필요한 속성을 최소화하여 키를 생성하는 것이 좋습니다. 예를 들어, 암호화에 사용되는 AES 키는 HSM에서 키를 래핑하는 것도 허용해서는 안 됩니다. 자세한 내용은 다음 클라이언트 SDK의 속성 페이지를 참조하십시오.

- PKCS #11 키 속성
- JCE 키 속성

가능하면 키 객체를 캐시하여 지연 시간 최소화: 키 찾기 작업은 클러스터의 모든 HSM을 쿼리합니다. 이 작업은 비용이 많이 들고 클러스터의 HSM 수에 따라 확장되지 않습니다.

- PKCS #11에서는 C_FindObjects API를 사용하여 키를 찾을 수 있습니다.
- JCE에서는 를 사용하여 키를 찾을 수 있습니다. KeyStore

최적의 성능을 위해 응용 프로그램 시작 중에 키 찾기 명령 (예: findKey 및키 목록) 을 한 번만 사용하고 반환된 키 객체를 응용 프로그램 메모리에 캐시하는 것이 좋습니다. AWS 나중에 이 키 객체가 필요한 경우 각 작업마다 이 객체를 쿼리하여 상당한 성능 오버헤드를 추가하는 대신 캐시에서 객체를 검색해야 합니다.

멀티스레딩 사용

AWS CloudHSM 다중 스레드 응용 프로그램을 지원하지만 다중 스레드 응용 프로그램에서는 몇 가지 유의해야 할 사항이 있습니다.

PKCS #11에서는 PKCS #11 라이브러리(C_Initialize 호출)를 한 번만 초기화해야 합니다. 각 스레드에는 자체 세션(C_OpenSession)을 할당해야 합니다. 다중 스레드에서 동일한 세션을 사용하는 것은 권장되지 않습니다.

JCE에서는 제공자를 한 번만 초기화해야 합니다. AWS CloudHSM 스레드 간에 SPI 객체 인스턴스를 공유하지 마십시오. 예를 들어 암호, 서명, 다이제스트, Mac KeyFactory 또는 KeyGenerator 객체는 해당 스레드의 컨텍스트에서만 사용해야 합니다.

제한 오류 처리

다음과 같은 상황에서 HSM 제한 오류가 발생할 수 있습니다.

- 클러스터가 피크 트래픽을 관리할 수 있도록 적절하게 확장되지 않았습니다.
- 유지 관리 이벤트 중에 클러스터 크기가 +1 중복으로 조정되지 않았습니다.
- 가용 영역이 중단되면 클러스터에서 사용 가능한 HSM 수가 줄어듭니다.

이 시나리오를 가장 잘 처리하는 방법에 대한 내용은 HSM 스로틀링 섹션을 참조하십시오.

클러스터 크기가 적절하고 병목 현상이 발생하지 않도록 하려면 최대 트래픽이 예상되는 환경에서 AWS 부하 테스트를 수행하는 것이 좋습니다.

클러스터 작업에 재시도 통합

AWS 운영 또는 유지 관리상의 이유로 HSM을 교체할 수 있습니다. 이러한 상황에서 애플리케이션을 복원할 수 있도록 하려면 클러스터로 라우팅되는 모든 작업에 대해 클라이언트측 재시도 로직을 구현할 AWS 것을 권장합니다. 교체로 인해 실패한 작업에 대한 후속 재시도는 성공할 것으로 예상됩니다.

재해 복구 전략 구현

이벤트에 대응하여 전체 클러스터 또는 리전에서 트래픽을 이동해야 할 수도 있습니다. 다음 섹션에서는 이를 수행하기 위한 여러 전략에 대해 설명합니다.

VPC 피어링을 사용하여 다른 계정이나 지역에서 클러스터에 액세스: VPC 피어링을 사용하여 다른 계정이나 지역에서 AWS CloudHSM 클러스터에 액세스할 수 있습니다. 이를 설정하는 방법에 대한 자세한 내용은 VPC 피어링 가이드의 VPC 피어링이란?을 참조하십시오. 피어링 연결을 설정하고 보안 그룹을 적절하게 구성한 후에는 평소와 같은 방식으로 HSM IP 주소와 통신할 수 있습니다.

동일한 애플리케이션에서 여러 클러스터에 연결: 클라이언트 SDK 5의 JCE 공급자, PKCS #11 라이브러리 및 CloudHSM CLI는 동일한 애플리케이션에서 여러 클러스터에 연결할 수 있도록 지원합니다. 예를 들어, 각각 다른 리전에 두 개의 활성 클러스터를 둘 수 있으며, 애플리케이션은 두 가지 모두에 동시에 연결하여 일반 작업의 일부로 둘 사이의 로드 밸런싱을 수행할 수 있습니다. 애플리케이션에서 Client SDK 5(최신 SDK)를 사용하지 않는 경우 동일한 애플리케이션에서 여러 클러스터에 연결할 수 없습니다. 또는 다른 클러스터를 계속 실행하고, 리전 중단이 발생하는 경우 트래픽을 다른 클러스터로 이동하여 가동 중지를 최소화할 수 있습니다. 자세한 내용은 각 페이지를 참조하십시오.

- PKCS #11를 사용하여 여러 슬롯에 연결
- JCE 공급자를 통해 여러 클러스터에 연결
- CloudHSM CLI를 사용하여 여러 클러스터에 연결

백업에서 클러스터 복원: 기존 클러스터의 백업에서 새 클러스터를 생성할 수 있습니다. 자세한 내용은 AWS CloudHSM 백업 관리 섹션을 참조하십시오.

모니터링

이 섹션에서는 클러스터와 애플리케이션을 모니터링하는 데 사용할 수 있는 여러 메커니즘에 대해 설명합니다. 모니터링에 대한 자세한 내용은 모니터링 AWS CloudHSM 섹션을 참조하십시오.

클라이언트 로그 모니터링

모든 클라이언트 SDK는 모니터링할 수 있는 로그를 작성합니다. 클라이언트 로깅에 대한 자세한 내용은 클라이언트 SDK 로그로 작업 섹션을 참조하십시오.

Amazon ECS와 AWS Lambda같이 일시용으로 설계된 플랫폼에서는 파일에서 클라이언트 로그를 수집하기가 어려울 수 있습니다. 이러한 상황에서는 콘솔에 로그를 기록하도록 클라이언트 SDK 로깅을

구성하는 것이 좋습니다. 대부분의 서비스는 이 출력을 자동으로 수집하여 Amazon CloudWatch 로그에 게시하여 사용자가 보관하고 볼 수 있도록 합니다.

AWS CloudHSM 클라이언트 SDK에서 타사 통합을 사용하는 경우 출력을 콘솔에도 기록하도록 해당 소프트웨어 패키지를 구성해야 합니다. AWS CloudHSM 클라이언트 SDK의 출력은 이 패키지에서 캡처되고 그렇지 않으면 자체 로그 파일에 기록될 수 있습니다.

애플리케이션에서 로깅 옵션을 구성하는 방법에 대한 자세한 내용은 클라이언트 SDK 5 구성 도구 섹션을 참조하십시오.

감사 로그 모니터링

AWS CloudHSM Amazon CloudWatch 계정에 감사 로그를 게시합니다. 감사 로그는 HSM에서 가져오며 감사 목적으로 특정 작업을 추적합니다.

감사 로그를 사용하여 HSM에서 호출되는 모든 관리 명령을 추적할 수 있습니다. 예를 들어, 예상치 못한 관리 작업이 수행되고 있는 것을 발견하면 경보를 트리거할 수 있습니다.

자세한 내용은 HSM 감사 로깅 작동 방식 섹션을 참조하십시오.

모니터 AWS CloudTrail

AWS CloudHSM 에서 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업의 기록을 제공하는 AWS 서비스와 통합됩니다 AWS CloudHSM. AWS CloudTrail 모든 API 호출을 AWS CloudHSM 이벤트로 캡처합니다. 캡처된 호출에는 AWS CloudHSM 콘솔에서의 호출 및 AWS CloudHSM API 작업에 대한 코드 호출이 포함됩니다.

AWS CloudTrail 를 사용하면 AWS CloudHSM 컨트롤 플레인에 대한 모든 API 호출을 감사하여 계정에서 원치 않는 활동이 발생하지 않도록 할 수 있습니다.

세부 정보는 AWS CloudTrail 와 함께 일하기 AWS CloudHSM를 참조하세요.

아마존 CloudWatch 지표 모니터링

Amazon CloudWatch 메트릭을 사용하여 AWS CloudHSM 클러스터를 실시간으로 모니터링할 수 있습니다. 지표는 리전, 클러스터 ID 또는 HSM ID 및 클러스터 ID별로 그룹화할 수 있습니다.

Amazon CloudWatch 지표를 사용하면 서비스에 영향을 미칠 수 있는 잠재적 문제를 경고하도록 Amazon CloudWatch 경보를 구성할 수 있습니다. 다음을 모니터링하도록 경보를 구성하는 것이 좋습니다.

- HSM의 키 한도에 접근하기
- HSM의 HSM 세션 수 한도에 접근하기
- HSM의 HSM 사용자 수 한도에 접근하기
- 동기화 문제를 식별하기 위한 HSM 사용자 또는 키 수의 차이
- 비정상 HSM은 문제를 해결할 AWS CloudHSM 수 있을 때까지 클러스터를 확장합니다.

자세한 내용은 Amazon CloudWatch 로그 및 AWS CloudHSM 감사 로그 사용 섹션을 참조하십시오.

AWS CloudHSM 클러스터 관리

AWS CloudHSM 콘솔이나AWS SDK 또는 명령줄 도구 중 하나에서 AWS CloudHSM 클러스터를 관리할 수 있습니다. 자세한 내용은 다음 항목을 참조하십시오.

클러스터를 생성하려면 시작하기 단원을 참조하십시오.

클러스터 아키텍처

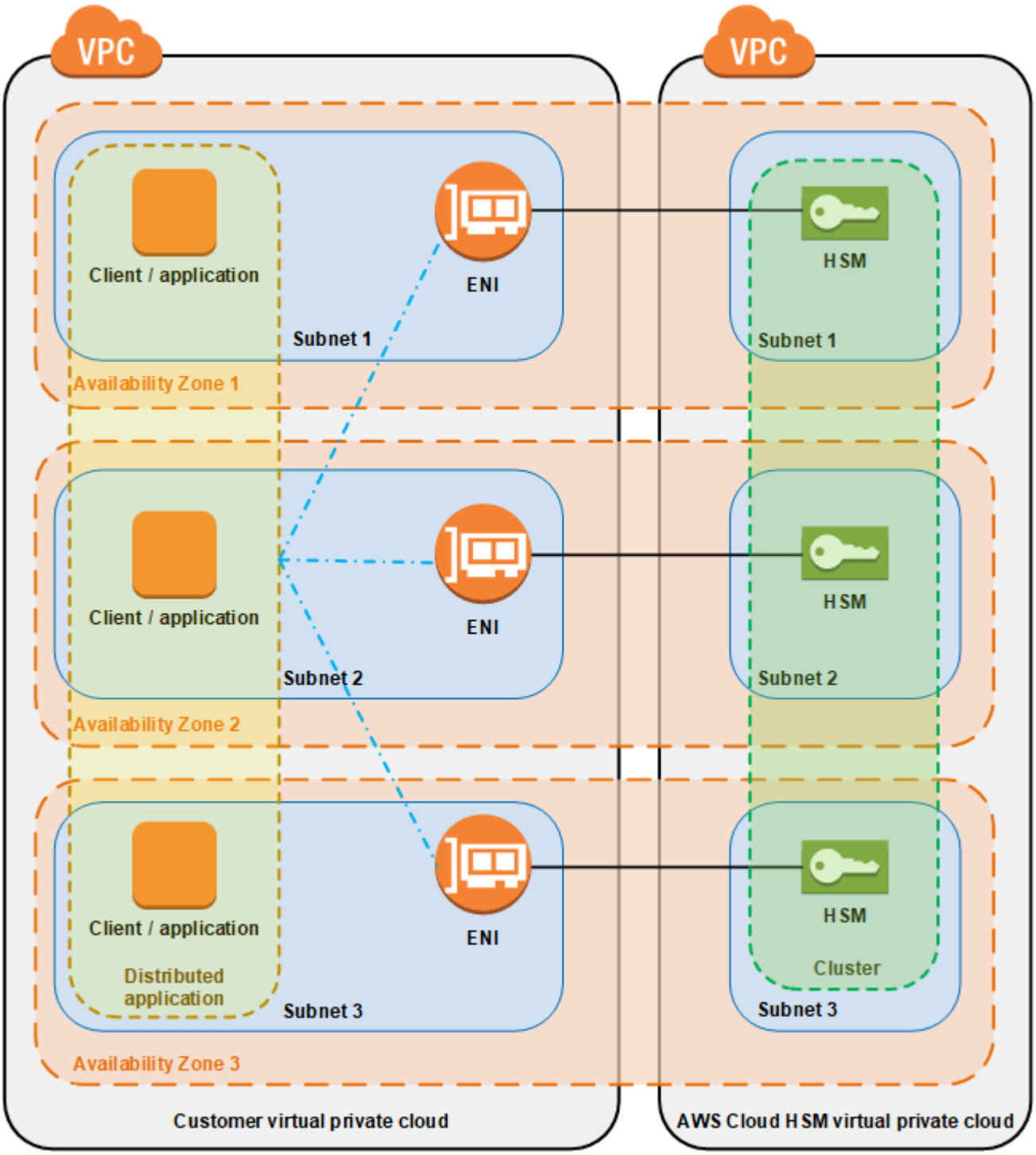
클러스터를 생성할 때 AWS 계정에 Amazon VPC (Virtual Private Cloud) 를 지정하고 해당 VPC에 하나 이상의 서브넷을 지정합니다. 선택한 지역의 각 가용 영역 (AZ) 에 서브넷을 하나씩 생성하는 것이 좋습니다. AWS VPC를 생성할 때 프라이빗 서브넷을 생성할 수 있습니다. 자세한 내용은 Virtual Private Cloud(VPC) 생성 단원을 참조하십시오.

HSM을 생성할 때마다 HSM에 대한 클러스터 및 가용 영역을 지정합니다. 서로 다른 가용 영역에 HSM을 배치하면 하나의 가용 영역을 사용할 수 없는 경우 중복성과 고가용성을 얻을 수 있습니다.

HSM을 생성할 때 계정의 지정된 서브넷에 ENI (Elastic Network Interface) 를 AWS CloudHSM 배치합니다. AWS 탄력적 네트워크 인터페이스는 HSM과의 상호 작용을 위한 인터페이스입니다. HSM은 소유한 AWS 계정의 별도 VPC에 있습니다. AWS CloudHSM HSM 및 해당 네트워크 인터페이스는 동일한 가용 영역에 있습니다.

클러스터의 HSM과 상호 작용하려면 클라이언트 소프트웨어가 필요합니다. AWS CloudHSM 일반적으로 클라이언트는 Amazon EC2 인스턴스에 설치합니다. 이 인스턴스는 다음 그림에 나와 있는 것처럼 HSM ENI와 동일한 VPC에 있는 클라이언트 인스턴스입니다. 기술적으로 반드시 여기에 설치해야 하는 것은 아니며, HSM ENI에 연결할 수 있다면 다른 호환 컴퓨터에 클라이언트를 설치해도 됩니다. 클라이언트는 클러스터의 개별 HSM과 해당 ENI를 통해 통신합니다.

다음 그림은 각각 VPC의 서로 다른 가용 영역에 있는 세 개의 HSM이 있는 AWS CloudHSM 클러스터를 나타냅니다.



클러스터 동기화

AWS CloudHSM 클러스터에서는 개별 HSM의 키를 동기화된 AWS CloudHSM 상태로 유지합니다. HSM의 키를 동기화하기 위해 아무 작업도 수행할 필요 없습니다. 각 HSM의 사용자와 정책을 동기화

된 상태로 유지하려면 HSM 사용자를 관리하기 전에 AWS CloudHSM 클라이언트 구성 파일을 업데이트하십시오. 자세한 정보는 HSM 사용자의 동기화 유지을 참조하세요.

클러스터에 새 HSM을 추가하면 기존 HSM의 모든 키, 사용자, 정책을 백업합니다. AWS CloudHSM 그런 다음 새 HSM에서 백업을 복원합니다. 이렇게 하면 두 HSM이 동기화 상태로 유지됩니다.

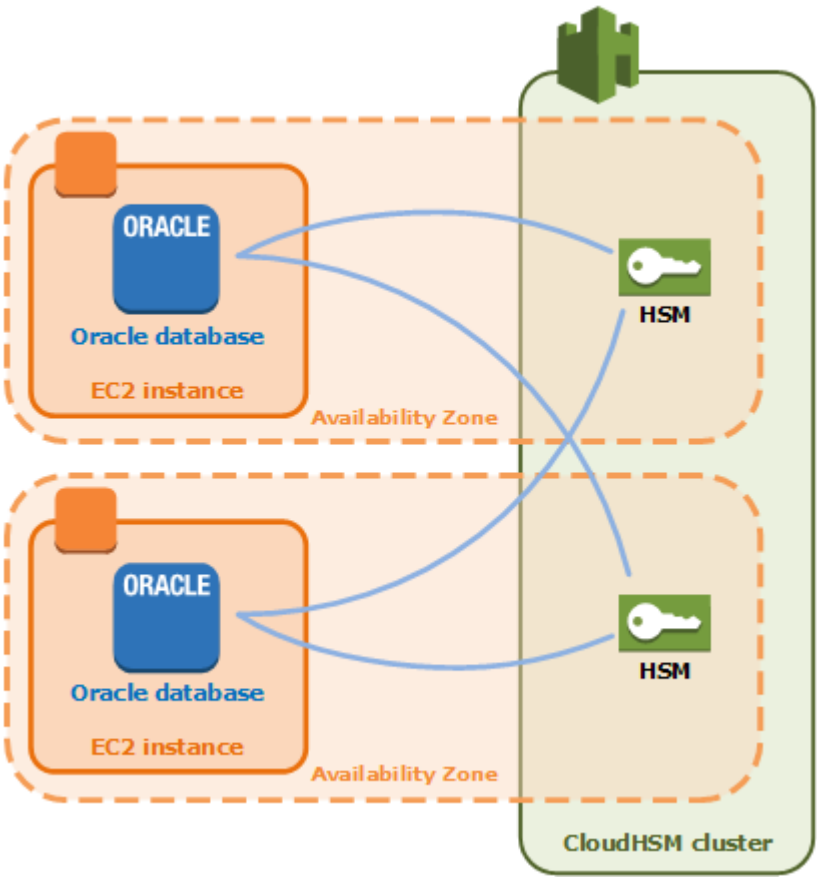
클러스터의 HSM이 동기화되지 않는 경우 HSM을 AWS CloudHSM 자동으로 재동기화합니다. 이를 활성화하려면 어플라이언스 사용자의 자격 증명을 AWS CloudHSM 사용합니다. 이 사용자는 에서 제공하는 모든 HSM에 AWS CloudHSM 존재하며 제한된 권한을 가집니다. HSM에서 객체의 해시를 가져올 수 있으며, 마스킹 처리된(암호화된) 객체를 추출하고 삽입할 수 있습니다. AWS 는 사용자 또는 키를 보거나 수정할 수 없으며 해당 키를 사용하여 암호화 작업을 수행할 수 없습니다.

클러스터 고가용성 및 로드 밸런싱

두 개 이상의 HSM으로 AWS CloudHSM 클러스터를 생성하면 자동으로 부하 분산이 이루어집니다. 로드 밸런싱은 AWS CloudHSM 클라이언트가 추가 처리를 위해 각 HSM의 용량을 기준으로 클러스터의 모든 HSM에 대해 암호화 작업을 배포한다는 것을 의미합니다.

서로 다른 가용 영역에 HSM을 생성하면 자동으로 AWS 고가용성이 확보됩니다. 고가용성은 개별 HSM이 단일 장애 지점이 아니기 때문에 더 높은 안정성을 얻는다는 것을 의미합니다. 각 클러스터에는 최소 두 개의 HSM을 설치하고 각 HSM은 지역 내 서로 다른 가용 영역에 두는 것이 좋습니다. AWS

예를 들어 다음 그림은 2개 가용 영역에 분산된 Oracle 데이터베이스 애플리케이션을 보여 줍니다. 데이터베이스 인스턴스는 각 가용 영역에 HSM이 포함된 클러스터에 마스터 키를 저장합니다. AWS CloudHSM 키를 두 HSM에 자동으로 동기화하여 두 HSM에 즉시 액세스하고 중복되도록 합니다.



AWS CloudHSM 클러스터 모드 및 HSM 유형

AWS CloudHSM FIPS와 비 FIPS라는 두 가지 클러스터 모드를 제공합니다. AWS CloudHSM 또한 hsm1.medium과 hsm2m.medium이라는 두 가지 HSM 유형을 제공합니다. 요구 사항에 적합한 클러스터 모드 및 HSM 유형을 결정하기 전에 이 페이지의 세부 정보를 검토하십시오.

> **Note**
>
> 2024년 6월 10일 이전에 생성된 모든 클러스터는 FIPS 모드이며 HSM 유형은 hsm1.medium입니다.

클러스터의 모드와 HSM 유형을 보려면 describe-cluster 명령을 사용하십시오.

클러스터 모드

AWS CloudHSM FIPS 모드와 비 FIPS의 두 가지 모드로 클러스터를 제공합니다. FIPS 모드에서는 연방 정보 처리 표준 (FIPS) 에서 승인한 키와 알고리즘만 사용할 수 있습니다. 비 FIPS 모드는 FIPS 승인과 상관없이 지원되는 모든 키와 알고리즘을 제공합니다. AWS CloudHSM

다음 표에는 각 클러스터 모드 간의 주요 차이점이 나와 있습니다.

차별화 특성	FIPS 모드	비 FIPS 모드
HSM 유형 호환성	hsm1.medium과 함께 사용할 수 있습니다.	hsm2m.medium과 함께 사용할 수 있습니다.
백업 호환성	FIPS 모드에서 클러스터를 백업하고 복원할 때만 사용할 수 있습니다.	비 FIPS 모드에서 복원 클러스터를 백업하는 데만 사용할 수 있습니다.
키 선택	FIPS 승인을 [1]받은 AWS CloudHSM 키를 지원합니다.	FIPS 승인을 받은 AWS CloudHSM 키와 FIPS 승인을 받지 않은 키를 모두 지원합니다.
알고리즘	FIPS AWS CloudHSM 승인을 받은 알고리즘을 지원합니다. [1]	FIPS 승인을 받은 AWS CloudHSM 알고리즘과 FIPS 승인을 받지 않은 알고리즘을 모두 지원합니다.
인증	FIPS 140-2, PCI PIN 및 PCI-3DS 규정을 준수합니다.	

[1] 자세한 내용은 지원 중단 알림을 참조하십시오.

클러스터 모드를 선택하기 전에 클러스터 모드 (FIPS 또는 비 FIPS) 는 생성 후에는 변경할 수 없으므로 필요에 맞는 모드를 선택해야 합니다.

HSM 유형

클러스터 모드 외에도 hsm1.medium과 AWS CloudHSM hsm2m.medium이라는 두 가지 HSM 유형을 제공합니다. 각 HSM 유형은 서로 다른 하드웨어를 사용하며 각 클러스터에는 한 가지 유형의 HSM만 포함할 수 있습니다. 다음 표에는 다음 둘 사이의 주요 차이점이 나열되어 있습니다.

차별화 특성	hsm1.medium	hsm2m.medium
클러스터 모드 호환성	FIPS 모드의 클러스터에 사용할 수 있습니다.	현재 비 FIPS 모드의 클러스터에서 사용할 수 있습니다.
백업 호환성	hsm1.medium 클러스터로의 백업 복원에만 사용할 수 있습니다.	hsm2m.medium 클러스터를 백업 및 복원하는 데만 사용할 수 있습니다.
키 용량	클러스터당 3,300개	총 키 16,666개, 비대칭 키는 클러스터당 최대 3,333개입니다.
클라이언트 SDK	모든 클라이언트 SDK를 지원합니다.	CNG 및 KSP 공급자를 제외한 모든 클라이언트 SDK를 지원합니다.
클라이언트 SDK 버전	SDK 버전 3.1.0 이상과 호환됩니다.	클라이언트 SDK 버전 5.12.0 이상과 호환됩니다.
리전 가용성	CloudHSM을 사용할 수 있는 모든 지역에서 사용할 수 있습니다.	제한된 수의 지역에서만 사용할 수 있으며 곧 추가 지원 지역이 제공될 예정입니다. 이 HSM 유형을 사용할 수 있는 지역을 확인하려면 AWS CloudHSM 가격 계산기를 참조하십시오.
성능	각 HSM 유형의 성능을 보려면 AWS CloudHSM 성능을 참조하십시오.	
인증	FIPS 140-2, PCI DSS, PCI PIN, SOC2 및 PCI-3DS 규정을 준수합니다.	PCI DSS 규격을 준수합니다.

[1] 자세한 내용은 지원 중단 알림을 참조하십시오.

클라이언트 SDK를 클러스터에 AWS CloudHSM 연결

Client SDK 5 또는 Client SDK 3을 사용하여 클러스터에 연결하기 위해 먼저 다음 두 가지 작업을 수행해야 합니다.

- EC2 인스턴스에 발급 인증서가 있습니다
- Client SDK를 클러스터로 부트스트랩합니다.

발급 인증서를 각 EC2 인스턴스에 배치합니다.

클러스터를 초기화할 때 발급 인증서를 생성합니다. 클러스터에 연결하는 각 EC2 인스턴스의 플랫폼 기본 위치에 발급 인증서를 복사합니다.

Linux

```
/opt/cloudhsm/etc/customerCA.crt
```

Windows

```
C:\ProgramData\Amazon\CloudHSM\customerCA.crt
```

인증서의 위치를 지정합니다.

Client SDK 5에서는 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

PKCS #11 library

Linux용 Client SDK 5에 발급 인증서를 배치하려면

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --hsm-ca-cert <customerCA certificate
 file>
```

Windows용 Client SDK 5에 발급 인증서 배치

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\configure-pkcs11.exe" --hsm-ca-
cert <customerCA certificate file>
```

OpenSSL Dynamic Engine

Linux용 Client SDK 5에 발급 인증서를 배치하려면

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --hsm-ca-cert <customerCA certificate
 file>
```

JCE provider

Linux용 Client SDK 5에 발급 인증서를 배치하려면

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-jce --hsm-ca-cert <customerCA certificate
 file>
```

Windows용 Client SDK 5에 발급 인증서 배치

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\configure-jce.exe" --hsm-ca-cert <customerCA
 certificate file>
```

CloudHSM CLI

Linux용 Client SDK 5에 발급 인증서를 배치하려면

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli --hsm-ca-cert <customerCA certificate
 file>
```

Windows용 Client SDK 5에 발급 인증서 배치

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\configure-cli.exe" --hsm-ca-cert <customerCA
 certificate file>
```

자세한 내용은 구성 도구를 참조하십시오.

클러스터 초기화 또는 인증서 생성 및 서명에 대한 자세한 내용은 클러스터 초기화를 참조하십시오.

Client SDK 부트스트랩합니다.

부트스트랩 프로세스는 사용 중인 Client SDK 버전에 따라 다르지만 클러스터에 있는 하드웨어 보안 모듈(HSM) 중 하나의 IP 주소가 있어야 합니다. 클러스터에 연결된 모든 HSM의 IP 주소를 사용할 수 있습니다. Client SDK가 연결되면 추가 HSM의 IP 주소를 찾아내고 로드 밸런싱 및 클라이언트측 키 동기화 작업을 수행합니다.

클러스터의 IP 주소 가져오기

HSM의 IP 주소를 가져오려면 (콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. AWS 리전을 변경하려면 페이지의 오른쪽 상단에 있는 리전 선택기를 사용합니다.

3. 클러스터 세부 정보 페이지를 열려면 클러스터 테이블에서 클러스터 ID를 선택합니다.
4. IP 주소를 가져오려면 HSM 탭에서 ENI IP 주소 아래에 나열된 IP 주소 중 하나를 선택합니다.

HSM의 IP 주소를 가져오려면 ()AWS CLI

- AWS CLI의 describe-clusters 명령을 사용하여 HSM의 IP 주소를 가져옵니다. 명령의 출력에서 HSM의 IP 주소는 `EniIp`의 값입니다.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

부트스트래핑에 대한 자세한 내용은 구성 도구를 참조하십시오.

Client SDK 5 부트스트랩

PKCS #11 library

Client SDK 5용 Linux EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 -a <HSM IP addresses>
```

Client SDK 5용 Windows EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" -a <HSM IP
 addresses>
```

OpenSSL Dynamic Engine

Client SDK 5용 Linux EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-dyn -a <HSM IP addresses>
```

JCE provider

Client SDK 5용 Linux EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-jce -a <HSM IP addresses>
```

Client SDK 5용 Windows EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" -a <HSM IP addresses>
```

CloudHSM CLI

Client SDK 5용 Linux EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM(s)의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IP addresses of the HSMs>
```

Client SDK 5용 Windows EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM(s)의 IP 주소를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI IP
 addresses of the HSMs>
```

> **Note**
>
> `-a <HSM_IP_ADDRESSES>` 대신 `--cluster-id` 파라미터를 사용할 수 있습니다. `--cluster-id` 사용 요구 사항을 보려면 클라이언트 SDK 5 구성 도구 단원을 참조하십시오.

Client SDK 3 부트스트랩

Client SDK 3용 Linux EC2 인스턴스 부트스트랩

- configure클러스터에 있는 HSM의 IP 주소를 지정하는 데 사용합니다.

```
sudo /opt/cloudhsm/bin/configure -a <IP address>
```

Client SDK 3용 Windows EC2 인스턴스 부트스트랩

- configure클러스터에 있는 HSM의 IP 주소를 지정하는 데 사용합니다.

```
C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe -a <HSM IP address>
```

구성에 대한 자세한 내용은 ??? 단원을 참조하십시오.

클러스터에서 HSM 추가 또는 제거 AWS CloudHSM

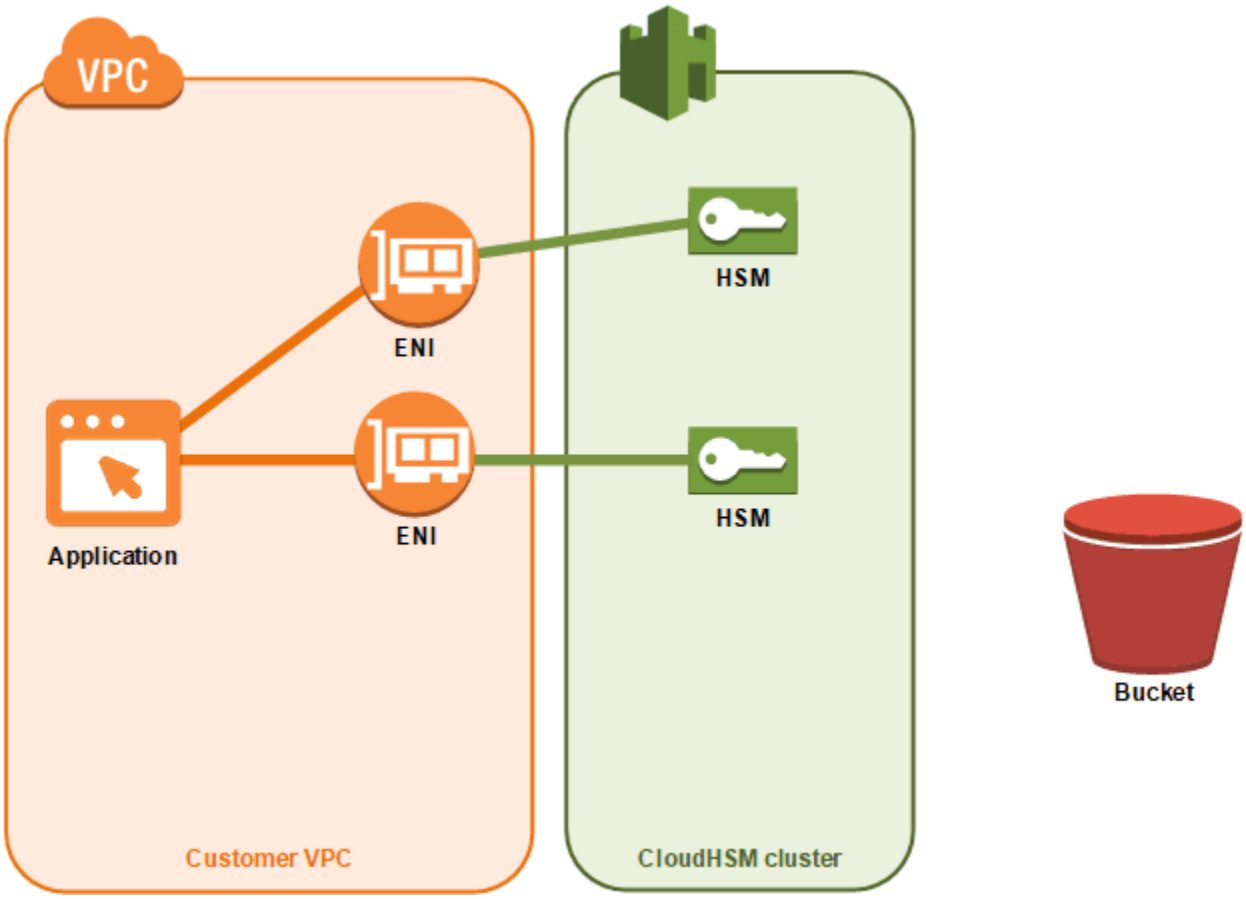
AWS CloudHSM 클러스터를 확장하거나 축소하려면 AWS CloudHSM 콘솔이나AWS SDK 또는 명령줄 도구 중 하나를 사용하여 HSM을 추가하거나 제거합니다. 클러스터의 부하 테스트를 통해 예상되는 최대 부하를 확인한 다음 HSM을 하나 더 추가하여 고가용성을 보장하는 것이 좋습니다.

주제

- HSM 추가
- HSM 제거

HSM 추가

다음 그림은 클러스터에 HSM을 추가할 때 발생하는 이벤트를 보여 줍니다.



1. 클러스터에 새 HSM을 추가합니다. 다음 절차에서는 AWS CloudHSM 콘솔, AWS Command Line Interface (AWS CLI) 및 AWS CloudHSM API에서 이렇게 하는 방법을 설명합니다.

   이것이 유일한 작업입니다. 나머지 이벤트는 자동으로 발생합니다.
2. AWS CloudHSM 클러스터에 있는 기존 HSM의 백업 복사본을 만듭니다. 자세한 정보는 백업을 참조하세요.
3. AWS CloudHSM 백업을 새 HSM에 복원합니다. 이렇게 하면 HSM이 클러스터에 있는 다른 HSM과 동기화됩니다.
4. 클러스터의 기존 HSM은 클러스터에 새 HSM이 있음을 AWS CloudHSM 클라이언트에 알립니다.
5. 클라이언트는 새 HSM과의 연결을 설정합니다.

HSM을 추가하려면(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 추가할 HSM의 클러스터를 선택합니다.
3. [HSMs] 탭에서 [Create HSM]을 선택합니다.

4. 생성 중인 HSM에 대한 가용 영역(AZ)을 선택합니다. 그런 다음 생성을 선택합니다.

HSM 추가 방법(AWS CLI)

- 명령 프롬프트에서 create-hsm 명령을 실행하고 생성할 HSM의 클러스터 ID와 가용 영역을 지정합니다. 원하는 클러스터의 클러스터 ID를 모르면, describe-clusters 명령을 실행합니다. `us-east-2a`, `us-east-2b` 등의 형태로 가용 영역을 지정합니다.

```
$ aws cloudhsmv2 create-hsm --cluster-id <cluster ID> --availability-
zone <Availability Zone>
{
    "Hsm": {
        "State": "CREATE_IN_PROGRESS",
        "ClusterId": "cluster-5a73d5qzrdh",
        "HsmId": "hsm-lgavqitns2a",
        "SubnetId": "subnet-0e358c43",
        "AvailabilityZone": "us-east-2c",
        "EniId": "eni-bab18892",
        "EniIp": "10.0.3.10"
    }
}
```

HSM (AWS CloudHSM API) 을 추가하려면

- CreateHsm 요청을 전송하고 생성할 HSM의 클러스터 ID와 가용 영역을 지정합니다.

HSM 제거

AWS CloudHSM 콘솔 AWS CLI, 또는 API를 사용하여 HSM을 제거할 수 있습니다. AWS CloudHSM

HSM을 제거하려면(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 제거할 HSM이 포함된 클러스터를 선택합니다.
3. [HSMs] 탭에서 제거할 HSM을 선택합니다. 그런 다음 [Delete HSM]을 선택합니다.
4. HSM을 삭제하려 한다는 것을 확인합니다. 그런 다음 삭제를 선택합니다.

HSM 제거 방법(AWS CLI)

- 명령 프롬프트에서 delete-hsm 명령을 실행합니다. 삭제할 HSM이 포함된 클러스터의 ID와 다음 HSM 식별자 중 하나를 전달합니다.
  - HSM ID(`--hsm-id`)
  - HSM IP 주소(`--eni-ip`)
  - HSM의 탄력적 네트워크 인터페이스 ID(`--eni-id`)

  이러한 식별자의 값을 모르면, describe-clusters 명령을 실행합니다.

```
$ aws cloudhsmv2 delete-hsm --cluster-id <cluster ID> --eni-ip <HSM IP address>
{
    "HsmId": "hsm-lgavqitns2a"
}
```

HSM (AWS CloudHSM API) 을 제거하려면

- DeleteHsm 요청을 전송하고 삭제할 HSM의 클러스터 ID와 식별자를 지정합니다.

AWS CloudHSM 클러스터 삭제

클러스터를 삭제하려면 먼저 클러스터에서 모든 HSM을 제거해야 합니다. 자세한 정보는 HSM 제거을 참조하세요.

HSM을 모두 제거한 후 AWS CloudHSM 콘솔, AWS Command Line Interface (AWS CLI) 또는 API를 사용하여 클러스터를 삭제할 수 있습니다. AWS CloudHSM

클러스터를 삭제하려면(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 삭제할 클러스터를 선택합니다. 그런 다음 [Delete cluster]를 선택합니다.
3. 클러스터를 정말로 삭제할 것인지 다시 한 번 묻는 메시지가 나오면 확인 후 [Delete]를 선택합니다.

클러스터(AWS CLI)를 삭제하는 방법

- 명령 프롬프트에서 delete-cluster 명령을 실행하여 삭제할 클러스터의 ID를 전달합니다. 클러스터 ID를 모르면 describe-clusters 명령을 실행합니다.

```
$ aws cloudhsmv2 delete-cluster --cluster-id <cluster ID>
{
    "Cluster": {
        "Certificates": {
            "ClusterCertificate": "<certificate string>"
        },
        "SourceBackupId": "backup-rtq2dwi2gq6",
        "SecurityGroup": "sg-40399d28",
        "CreateTimestamp": 1504903546.035,
        "SubnetMapping": {
            "us-east-2a": "subnet-f1d6e798",
            "us-east-2c": "subnet-0e358c43",
            "us-east-2b": "subnet-40ed9d3b"
        },
        "ClusterId": "cluster-kdmrayrc7gi",
        "VpcId": "vpc-641d3c0d",
        "State": "DELETE_IN_PROGRESS",
        "HsmType": "hsm1.medium",
        "StateMessage": "The cluster is being deleted.",
        "Hsms": [],
        "BackupPolicy": "DEFAULT"
    }
}
```

AWS CloudHSM 클러스터 삭제 방법(API)

- DeleteCluster 요청을 전송하고 삭제할 클러스터의 ID를 지정합니다.

백업에서 AWS CloudHSM 클러스터 생성

백업에서 AWS CloudHSM 클러스터를 복원하려면 이 항목의 단계를 따르십시오. 클러스터에는 백업에 있던 동일한 사용자, 키 자료, 인증서, 구성 및 정책이 포함됩니다. 백업 관리에 대한 자세한 내용은 백업 관리 단원을 참조하십시오.

백업에서 클러스터 생성하기 (콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 클러스터 생성을 선택합니다.
3. 클러스터 구성 단원에서 다음을 수행합니다.
   a. [VPC]에서 생성할 클러스터의 VPC를 선택합니다.
   b. [AZ(s)]에서 클러스터를 추가할 각 가용 영역의 프라이빗 서브넷을 선택합니다.
4. [Cluster source] 섹션에서 다음을 수행합니다.
   a. [Restore cluster from existing backup]을 선택합니다.
   b. 복제할 백업을 선택합니다.
5. 다음: 검토를 선택합니다.
6. 클러스터 구성을 검토한 다음 [Create cluster]를 선택합니다.
7. 서비스에서 백업을 보존해야 하는 기간을 지정하십시오.

   기본 보존 기간인 90일을 그대로 사용하거나 7일에서 379일 사이의 새 값을 입력합니다. 이 서비스는 여기에 지정한 값보다 오래된 이 클러스터의 백업을 자동으로 삭제합니다. 나중에 변경할 수 있습니다. 자세한 내용은 백업 보존 정책 구성 단원을 참조하십시오.
8. 다음을 선택합니다.
9. (선택 사항) 태그 키와 태그 값(선택)을 입력합니다. 클러스터에 두 개 이상의 태그를 추가하려면 태그 추가를 선택합니다.
10. 검토를 선택합니다.
11. 클러스터 구성을 검토한 다음 클러스터 생성을 선택합니다.

> **Tip**
>
> 복원한 백업에 있던 것과 동일한 사용자, 키 자료, 인증서, 구성 및 정책을 포함하는 HSM을 이 클러스터에 생성하려면 클러스터에 HSM을 추가하십시오.

백업에서 클러스터 생성하기 (AWS CLI)

백업 ID를 결정하려면 describe-backups 명령을 실행합니다.

- 명령 프롬프트에서 create-cluster 명령을 실행합니다. HSM 인스턴스 유형, HSM을 생성할 서브넷의 서브넷 ID, 복원할 백업의 백업 ID를 지정합니다.

```
$ aws cloudhsmv2 create-cluster --hsm-type hsm1.medium \
                                --subnet-ids <subnet ID 1> <subnet ID 2> <subnet ID
 N> \
                                --source-backup-id <backup ID>
{
    "Cluster": {
        "HsmType": "hsm1.medium",
        "VpcId": "vpc-641d3c0d",
        "Hsms": [],
        "State": "CREATE_IN_PROGRESS",
        "SourceBackupId": "backup-rtq2dwi2gq6",
        "BackupPolicy": "DEFAULT",
        "BackupRetentionPolicy": {
            "Type": "DAYS",
            "Value": 90
         },
        "SecurityGroup": "sg-640fab0c",
        "CreateTimestamp": 1504907311.112,
        "SubnetMapping": {
            "us-east-2c": "subnet-0e358c43",
            "us-east-2a": "subnet-f1d6e798",
            "us-east-2b": "subnet-40ed9d3b"
        },
        "Certificates": {
            "ClusterCertificate": "<certificate string>"
        },
        "ClusterId": "cluster-jxhlf7644ne"
    }
}
```

백업 (AWS CloudHSM API) 에서 클러스터 생성

API를 사용하여 백업에서 클러스터를 생성하는 방법을 알아보려면 다음 주제를 참조하십시오.

- CreateCluster

AWS CloudHSM 백업 관리

AWS CloudHSM 최소 24시간에 한 번씩 클러스터를 정기적으로 백업합니다. 각 백업에는 다음 데이터의 암호화된 복사본이 포함되어 있습니다:

- 사용자(CO, CU 및 AU)
- 키 구성 요소 및 인증서
- 하드웨어 시큐리티 모듈 (HSM) 구성 및 정책

서비스에 백업하도록 지시할 수는 없지만 서비스에서 백업을 생성하도록 강제하는 특정 작업을 수행할 수는 있습니다. 다음 작업 중 하나를 수행할 때 서비스가 백업을 생성합니다.

- 클러스터 활성화
- 활성 클러스터에 HSM 추가
- 클러스터에서 HSM 제거

AWS CloudHSM 클러스터를 생성할 때 설정한 백업 보존 정책에 따라 백업을 삭제합니다. 백업 보존 정책 관리에 대한 자세한 내용은 백업 보존 정책 구성 단원을 참조하십시오.

주제

백업 작업

이전에 하나 이상의 활성 HSM이 포함된 클러스터에 HSM을 추가하면, 서비스가 최신 백업을 새 HSM으로 복원합니다. 백업을 사용하여 자주 사용하지 않는 HSM을 관리합니다. HSM을 더 이상 사용할 필요가 없으면 이를 삭제하여 백업을 트리거할 수 있습니다. 나중에 HSM이 필요할 때 동일한 클러스터에 새 HSM을 생성하면 HSM 삭제 작업으로 이전에 생성한 백업이 복원됩니다.

만료된 키 또는 비활성 사용자 제거

만료된 키 또는 비활성 사용자와 같은 원치 않는 암호화 구성 요소를 사용자 환경에서 삭제할 수 있습니다. 다음 2단계 프로세스로 진행됩니다. 먼저 HSM에서 이러한 구성 요소를 삭제하십시오. 다음으로 기존 백업을 모두 삭제하십시오. 이 프로세스를 수행하면 백업에서 새 클러스터를 초기화할 때 삭제된 정보가 복원되지 않습니다. 자세한 내용은 the section called “백업 삭제 및 복원” 단원을 참조하십시오.

재난 복구 고려

백업에서 클러스터를 생성할 수 있습니다. 클러스터의 복구 지점을 설정하기 위해 이 작업을 수행하는 것이 좋습니다. 복구 지점에 저장하려는 모든 사용자, 키 자료, 인증서가 포함된 백업을 지정한 다음 해당 백업을 사용하여 새 클러스터를 생성합니다. 백업에서 클러스터를 생성에 대한 자세한 내용은 백업에서 클러스터 생성 을 참조하십시오.

클러스터의 백업을 다른 지역에 복사하여 새 클러스터를 원본의 복제본으로 만들 수도 있습니다. 재난 복구 프로세스의 간소화 등 몇 가지 이유에서 이 작업을 수행해야 할 때가 있습니다. 리전 간 백업 복사에 대한 자세한 내용은 리전 간 백업 복사 단원을 참조하십시오.

백업 삭제 및 복원

백업을 삭제한 후 서비스는 7일 동안 백업을 보류하며, 이 기간 동안 백업을 복원할 수 있습니다. 7일이 지나면 더 이상 백업을 복원할 수 없습니다. 백업 관리에 대한 자세한 내용은 백업 관리 단원을 참조하십시오.

백업 삭제 및 복원(콘솔)

백업 삭제(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. AWS 리전을 변경하려면 페이지의 오른쪽 상단에 있는 리전 선택기를 사용합니다.
3. 내비게이션 창에서 백업를 선택하십시오.
4. 삭제할 백업을 선택합니다.
5. 선택한 백업을 삭제하려면 작업, 삭제를 선택합니다.

   백업 삭제 대화 상자가 나타납니다.
6. 삭제를 선택합니다.

백업 상태가 로 변경됩니다PENDING_DELETE. 삭제 보류 중인 백업은 삭제 요청 후 최대 7일 동안 복원할 수 있습니다.

백업을 복원(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. AWS 리전을 변경하려면 페이지의 오른쪽 상단에 있는 리전 선택기를 사용합니다.
3. 내비게이션 창에서 백업을 선택합니다.
4. 복원할 PENDING_DELETE 상태의 백업을 선택합니다.
5. 선택한 백업을 복원하려면 작업, 복원을 선택합니다.

백업 삭제 및 복원(AWS CLI)

AWS CLI의 describe-backups 명령을 사용하여 백업 상태를 확인하거나 백업 ID를 찾으십시오.

백업 삭제(AWS CLI)

- 명령 프롬프트에서 delete-backup 명령을 실행하여 삭제할 백업의 ID를 전달합니다.

```
$ aws cloudhsmv2 delete-backup --backup-id <backup ID>
{
    "Backup": {
        "CreateTimestamp": 1534461854.64,
        "ClusterId": "cluster-dygnwhmscg5",
        "BackupId": "backup-ro5c4er4aac",
        "BackupState": "PENDING_DELETION",
        "DeleteTimestamp": 1536339805.522,
        "HsmType": "hsm1.medium",
        "Mode": "FIPS"
    }
}
```

백업 복원(AWS CLI)

- 백업을 복원하려면 restore-backup 명령을 실행하여 PENDING_DELETION 상태인 백업의 ID를 전달합니다.

```
$ aws cloudhsmv2 restore-backup --backup-id <backup ID>
{
    "Backup": {
        "ClusterId": "cluster-dygnwhmscg5",
        "CreateTimestamp": 1534461854.64,
        "BackupState": "READY",
        "BackupId": "backup-ro5c4er4aac"
    }
}
```

백업 나열(AWS CLI)

- PENDING_DELETION 상태인 모든 백업의 목록을 보고자 하는 경우 describe-backups 명령을 실행하고 states=PENDING_DELETION을 필터로 포함시킵니다.

```
$ aws cloudhsmv2 describe-backups --filters states=PENDING_DELETION
{
    "Backups": [
        {
            "BackupId": "backup-ro5c4er4aac",
            "BackupState": "PENDING_DELETION",
            "ClusterId": "cluster-dygnwhmscg5",
            "CreateTimestamp": 1534461854.64,
            "DeleteTimestap": 1536339805.522,
            "HsmType": "hsm2m.medium",
            "Mode": "NON_FIPS",
            "NeverExpires": false,
            "TagList": []
        }
}
```

백업 삭제 및 복원 (AWS CloudHSM API)

API를 사용하여 백업을 삭제하고 복원하는 방법을 알아보려면 다음 주제를 참조하십시오.

- DeleteBackup
- RestoreBackup

AWS CloudHSM 백업 보존 정책 구성

2020년 11월 18일 이전에 생성된 클러스터는 제외되며 클러스터의 기본 백업 보존 정책은 90일입니다. 이 기간은 7일에서 379일 사이로 설정할 수 있습니다. AWS CloudHSM 클러스터의 마지막 백업은 삭제하지 않습니다. 백업 관리에 대한 자세한 내용은 백업 관리 단원을 참조하십시오.

백업 보존 정책 파악

AWS CloudHSM 클러스터를 생성할 때 설정한 백업 보존 정책을 기반으로 백업을 제거합니다. 백업 보존 정책은 클러스터에 적용됩니다. 백업을 다른 지역으로 이동하면 해당 백업은 더 이상 클러스터와 연결되지 않으며 백업 보존 정책도 적용되지 않습니다. 클러스터와 연결되지 않은 백업은 수동으로 삭제해야 합니다. AWS CloudHSM 클러스터의 마지막 백업은 삭제하지 않습니다.

AWS CloudTrail은 삭제 표시된 백업을 보고합니다. 수동으로 삭제한 백업을 복원하는 것처럼 서비스가 제거한 백업을 복원할 수 있습니다. 경쟁 상태를 방지하려면 서비스에서 삭제한 백업을 복원하기 전에 클러스터의 백업 보존 정책을 변경해야 합니다. 보존 정책을 동일하게 유지하고 일부 백업을 보존하려면 서비스가 클러스터 백업 보존 정책에서 백업을 제외하도록 지정할 수 있습니다.

기존 클러스터 면제

AWS CloudHSM 2020년 11월 18일에 관리형 백업 보존을 시작했습니다. 2020년 11월 18일 이전에 생성된 클러스터에는 90일의 백업 보존 정책에 클러스터의 보존 기간을 더한 정책이 적용됩니다. 예를 들어, 2019년 11월 18일에 클러스터를 생성한 경우 서비스에서는 클러스터에 1년+90일(455일)의 백업 보존 정책을 할당합니다.

> **Note**
>
> 지원팀(https://aws.amazon.com/support)에 문의하여 백업 보존 관리를 완전히 취소할 수 있습니다.

백업 보존 구성(콘솔)

백업 보존 정책 구성(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. AWS 리전을 변경하려면 페이지의 오른쪽 상단에 있는 리전 선택기를 사용합니다.
3. 활성 상태인 클러스터의 클러스터 ID를 클릭하여 해당 클러스터의 백업 보존 정책을 관리합니다.
4. 백업 보존 정책을 변경하려면 작업, 백업 보존 기간 변경을 선택합니다.

백업 보존 기간 변경 대화 상자가 나타납니다.

5. 백업 보존 기간(일)에 7일에서 379일 사이의 값을 입력합니다.
6. 백업 보존 기간 변경을 선택합니다.

백업 보존 정책에서 백업 제외 또는 포함(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 백업을 보려면 탐색 창에서 백업을 선택합니다.
3. 제외하거나 포함하려면 준비 상태인 백업의 백업 ID를 클릭합니다.
4. 백업 세부 정보 페이지에서 다음 작업 중 하나를 수행합니다.
   - 만료 시간에 날짜가 있는 백업을 제외하려면 작업, 만료 비활성화를 선택합니다.
   - 만료되지 않는 백업을 포함하려면 작업, 클러스터 보존 정책 사용을 선택합니다.

백업 보존 구성(AWS CLI)

AWS CLI의 describe-backups 명령을 사용하여 백업 상태를 확인하거나 백업 ID를 찾으십시오.

백업 보존 정책 구성(AWS CLI)

- 명령 프롬프트에서 modify-cluster 명령을 실행합니다. 클러스터 ID와 백업 보존 정책을 지정합니다.

```
$ aws cloudhsmv2 modify-cluster --cluster-id <cluster ID> \
                                --backup-retention-policy  Type=DAYS,Value=<number
 of days to retain backups>
{
   "Cluster": {
      "BackupPolicy": "DEFAULT",
      "BackupRetentionPolicy": {
         "Type": "DAYS",
         "Value": 90
      },
      "Certificates": {},
      "ClusterId": "cluster-kdmrayrc7gi",
      "CreateTimestamp": 1504903546.035,
      "Hsms": [],
      "HsmType": "hsm1.medium",
```

```
      "SecurityGroup": "sg-40399d28",
      "State": "ACTIVE",
      "SubnetMapping": {
         "us-east-2a": "subnet-f1d6e798",
         "us-east-2c": "subnet-0e358c43",
         "us-east-2b": "subnet-40ed9d3b"
      },
      "TagList": [
         {
            "Key": "Cost Center",
            "Value": "12345"
         }
      ],
      "VpcId": "vpc-641d3c0d"
   }
}
```

백업 보존 정책에서 백업 제외(AWS CLI)

- 명령 프롬프트에서 modify-backup-attributes 명령을 실행합니다. 백업 ID를 지정하고 만료되지 않는 플래그를 설정하여 백업을 보존합니다.

```
$ aws cloudhsmv2 modify-backup-attributes --backup-id <backup ID> \
                                          --never-expires
{
   "Backup": {
      "BackupId": "backup-ro5c4er4aac",
      "BackupState": "READY",
      "ClusterId": "cluster-dygnwhmscg5",
      "NeverExpires": true
   }
}
```

백업 보존 정책에 백업 포함(AWS CLI)

- 명령 프롬프트에서 modify-backup-attributes 명령을 실행합니다. 백업 ID를 지정하고 백업 보존 정책에 백업을 포함하도록 no-never-expires 플래그를 설정합니다. 그러면 서비스가 결국 백업을 삭제하게 됩니다.

```
$ aws cloudhsmv2 modify-backup-attributes --backup-id <backup ID> \
```

```
                                                --no-never-expires
{
   "Backup": {
      "BackupId": "backup-ro5c4er4aac",
      "BackupState": "READY",
      "ClusterId": "cluster-dygnwhmscg5",
      "NeverExpires": false
   }
}
```

백업 보존 (AWS CloudHSM API) 구성

API를 사용하여 백업 보존을 관리하는 방법을 알아보려면 다음 항목을 참조하십시오.

- ModifyCluster
- ModifyBackupAttributes

AWS 지역 간 백업 복사

리전 간 복원력, 글로벌 워크로드 및 재난 복구 등 여러 가지 이유로 리전 간에 백업을 복사할 수 있습니다. 백업을 복사하면 대상 리전에 CREATE_IN_PROGRESS 상태와 함께 백업이 나타납니다. 성공적으로 복사가 완료될 경우 백업의 상태는 READY로 변합니다. 복사에 실패한 경우 백업의 상태는 DELETED로 변합니다. 입력 파라미터의 오류 여부를 확인하고 작업을 다시 실행하기 전에 지정된 소스 백업이 DELETED 상태가 아닌지 확인합니다. 백업 또는 백업에서 클러스터를 생성하는 방법에 대한 자세한 내용은 백업 관리 또는 백업에서 클러스터 생성를 참조하십시오.

유의할 사항:

- 클러스터 백업을 대상 리전에 복사하려면 계정에 적절한 IAM 정책 권한이 있어야 합니다. 백업을 다른 리전에 복사하려면 IAM 정책에서 백업이 있는 소스 리전에 대한 액세스를 허용해야 합니다. 리전 간에 복사하고 나면 IAM 정책에서는 복사된 백업과의 상호 작용을 위해 대상 리전에 대한 액세스를 허용해야 합니다. 이 액세스에는 CreateCluster 작업의 사용이 포함됩니다. 자세한 정보는 IAM 관리자 생성을 참조하세요.
- 대상 리전의 백업에서 빌드되었을 수 있는 클러스터와 원본 클러스터는 서로 연결되어 있지 않습니다. 이러한 각각의 클러스터를 독립적으로 관리해야 합니다. 자세한 정보는 클러스터 관리을 참조하세요.

- AWS 제한 지역과 표준 지역 간에는 백업을 복사할 수 없습니다. 백업은 AWS GovCloud (미국 동부) 및 AWS GovCloud (미국 서부) 지역 간에 복사할 수 있습니다.

백업을 다른 리전으로 복사(콘솔)

백업을 다른 리전으로 복사하려면(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. AWS 리전을 변경하려면 페이지의 오른쪽 상단에 있는 리전 선택기를 사용합니다.
3. 내비게이션 창에서 백업을 선택합니다.
4. 다른 리전에 복사할 백업을 선택합니다.
5. 선택한 백업을 복사하려면 작업, 백업을 다른 리전으로 복사를 선택합니다.

   백업을 다른 리전으로 복사 대화 상자가 나타납니다.
6. 대상 리전의 리전 선택에서 리전을 선택합니다.
7. (선택 사항) 태그 키와 태그 값(선택)을 입력합니다. 클러스터에 두 개 이상의 태그를 추가하려면 태그 추가를 선택합니다.
8. 백업 복사를 선택합니다.

백업을 다른 리전(AWS CLI)으로 복사

백업 ID를 결정하려면 describe-backups 명령을 실행합니다.

백업을 다른 리전(AWS CLI)에 복사하려면

- 명령 프롬프트에서 copy-backup-to-region 명령을 실행합니다. 원본 백업의 대상 리전과 백업 ID를 지정하십시오. 백업 ID를 지정하는 경우 연결된 백업이 복사됩니다.

```
$ aws cloudhsmv2 copy-backup-to-region --destination-region <destination region> \
                                       --backup-id <backup ID>
```

백업을 다른 지역 (AWS CloudHSM API) 으로 복사

API를 사용하여 백업을 다른 리전에 복사하는 방법을 알아보려면 다음 주제를 참조하십시오.

- CopyBackupToRegion

공유 백업 사용

CloudHSM은AWS RAM() AWS Resource Access Manager 와 통합되어 리소스 공유를 가능하게 합니다. AWS RAM 일부 CloudHSM 리소스를 다른 사람과 AWS 계정 공유하거나 이를 통해 공유할 수 있는 서비스입니다. AWS Organizations를 사용하면 리소스 공유를 생성하여 소유한 리소스를 공유할 수 있습니다. AWS RAM 리소스 공유는 공유할 리소스와 공유 대상 소비자를 지정합니다. 소비자에는 다음이 포함될 수 있습니다.

- 특정 조직 AWS 계정 내부 또는 외부 AWS Organizations
- 조직 내부의 조직 단위 AWS Organizations
- 조직 전체가 AWS Organizations

에 대한 AWS RAM자세한 내용은 AWS RAM 사용 설명서를 참조하십시오.

이 항목에서는 소유한 리소스를 공유하는 방법과 공유 리소스를 사용하는 방법을 설명합니다.

내용

- 백업 공유를 위한 사전 요구 사항
- 백업 공유
- 공유 백업 공유 취소
- 공유 백업 식별
- 공유 백업에 대한 권한
- 결제 및 측정

백업 공유를 위한 사전 요구 사항

- 백업을 공유하려면 백업을 소유해야 합니다. AWS 계정즉, 계정에서 리소스를 할당하거나 프로비저닝해야 합니다. 공유된 백업은 공유할 수 없습니다.
- 백업을 공유하려면 해당 백업이 READY 상태여야 합니다.
- 에서 AWS Organizations조직 또는 조직 구성 단위와 백업을 공유하려면 공유를 활성화해야 합니다 AWS Organizations. 자세한 내용은 AWS RAM 사용 설명서의 AWS Organizations과(와) 공유 활성화를 참조하세요.

백업 공유

백업을 다른 사람과 공유하는 AWS 계정경우 백업에 저장된 키와 사용자가 포함된 클러스터를 백업에서 복원할 수 있습니다.

백업을 공유하려면 리소스 공유에 백업을 추가해야 합니다. 리소스 공유는 AWS 계정전반에서 리소스를 공유할 수 있게 해주는 AWS RAM 리소스입니다. 리소스 공유는 공유할 리소스와 공유 대상 소비자를 지정합니다. CloudHSM 콘솔을 사용하여 백업을 공유하는 경우 기존 리소스 공유에 백업을 추가합니다. 새 리소스 공유에 백업을 추가하려면 먼저 콘솔을 사용하여 리소스 공유를 생성해야 합니다.AWS RAM

에서 조직에 속해 AWS Organizations 있고 조직 내 공유가 활성화되어 있는 경우 조직의 소비자에게 공유 백업에 대한 액세스 권한이 자동으로 부여됩니다. 그렇지 않으면 소비자는 리소스 공유에 참여하라는 초대를 받게 되며 초대를 수락한 후에 공유 백업에 대한 액세스 권한이 부여됩니다.

소유한 백업을 AWS RAM 콘솔이나 를 사용하여 공유할 수 AWS CLI있습니다.

AWS RAM 콘솔을 사용하여 소유한 백업을 공유하려면

AWS RAM 사용 설명서의 리소스 공유 생성을 참조하세요.

소유한 백업을 공유하려면 (AWS RAM 명령)

create-resource-share 명령을 사용합니다.

소유한 백업을 공유하려면 (CloudHSM 명령)

> **Important**
>
> PutResourcePolicy CloudHSM 작업을 사용하여 백업을 공유할 수 있지만 대신 () AWS Resource Access Manager 를 사용하는AWS RAM것이 좋습니다. 를 AWS RAM 사용하면 정책이 생성되므로 여러 가지 이점이 있고, 한 번에 여러 리소스를 공유할 수 있으며, 공유 리소스의 검색 가능성이 높아집니다. 공유한 백업을 PutResourcePolicy 사용하고 소비자가 설명할 수 있게 하려면 AWS RAM PromoteResourceShareCreatedFromPolicy API 작업을 사용하여 백업을 표준 AWS RAM 리소스 공유로 승격해야 합니다.

put-resource-policy 명령을 사용합니다.

1. 라는 `policy.json` 파일을 만들고 다음 정책을 파일에 복사하십시오.

```
{
  "Version":"2012-10-17",
  "Statement":[{
    "Effect":"Allow",
    "Principal":{
      "AWS":"<consumer-aws-account-id-or-user>"
    },
    "Action":[
      "cloudhsm:CreateCluster",
      "cloudhsm:DescribeBackups"],
    "Resource":"<arn-of-backup-to-share>"
  }]
}
```

2. 백업 ARN 및 공유할 식별자로 `policy.json` 업데이트하십시오. 다음 예시에서는 123456789012로 식별된 AWS 계정의 루트 사용자에게 읽기 전용 액세스 권한을 부여합니다.

```
{
  "Version":"2012-10-17",
  "Statement":[{
    "Effect":"Allow",
    "Principal":{
      "AWS": [
        "account-id"
      ]
    },
    "Action":[
      "cloudhsm:CreateCluster",
      "cloudhsm:DescribeBackups"],
    "Resource":"arn:aws:cloudhsm:us-west-2:123456789012:backup/backup-123"
  }]
}
```

> **Important**
>
> 계정 수준에서만 권한을 부여할 수 있습니다. DescribeBackups 다른 고객과 백업을 공유하는 경우 해당 계정에 대한 DescribeBackups 권한이 있는 모든 보안 주체는 백업에 대해 설명할 수 있습니다.

3. put-resource-policy 명령을 실행합니다.

```
$ aws cloudhsmv2 put-resource-policy --resource-arn <resource-arn> --policy file://
policy.json
```

> **Note**
>
> 이때 소비자는 백업을 사용할 수 있지만 공유 매개변수가 포함된 DescribeBackups 응답에는 표시되지 않습니다. 다음 단계에서는 백업이 응답에 포함되도록 AWS RAM 리소스 공유를 승격하는 방법을 설명합니다.

4. AWS RAM 리소스 공유 ARN을 가져옵니다.

```
$ aws ram list-resources --resource-owner SELF --resource-arns <backup-arn>
```

그러면 다음과 비슷한 응답이 반환됩니다.

```
{
  "resources": [
    {
      "arn": "<project-arn>",
      "type": "<type>",
      "resourceShareArn": "<resource-share-arn>",
      "creationTime": "<creation-time>",
      "lastUpdatedTime": "<last-update-time>"
    }
  ]
}
```

응답에서 다음 단계에서 사용할 *< resource-share-arn >* 값을 복사합니다.

5. AWS RAM promote-resource-share-created-from-policy 명령을 실행합니다.

```
$ aws ram promote-resource-share-created-from-policy --resource-share-
arn <resource-share-arn>
```

6. 명령을 실행하여 리소스 공유가 승격되었는지 확인할 수 있습니다. AWS RAM get-resource-shares

```
$ aws ram get-resource-shares --resource-owner SELF --resource-share-
arns <resource-share-arn>
```

정책이 승격되면 응답에 `featureSet` 다음과 같이 나열됩니다`STANDARD`. 즉, 정책의 새 계정이 백업을 설명할 수 있다는 의미이기도 합니다.

공유 백업 공유 취소

리소스 공유를 취소하면 소비자는 더 이상 해당 리소스를 사용하여 클러스터를 복원할 수 없습니다. 소비자는 공유 백업에서 복원한 모든 클러스터에 계속 액세스할 수 있습니다.

소유한 공유 백업을 공유 해제하려면 리소스 공유에서 해당 백업을 제거해야 합니다. AWS RAM 콘솔이나 AWS CLI를 사용하여 이 작업을 수행할 수 있습니다.

콘솔을 사용하여 소유한 공유 백업의 AWS RAM 공유를 취소하려면

AWS RAM 사용 설명서에서 리소스 공유 업데이트를 참조하세요.

소유한 공유 백업의 공유를 취소하려면 (명령)AWS RAM

disassociate-resource-share 명령을 사용합니다.

소유한 공유 백업의 공유를 취소하려면 (CloudHSM 명령)

delete-resource-policy 명령을 사용합니다.

```
$ aws cloudhsmv2 delete-resource-policy --resource-arn <resource-arn>
```

공유 백업 식별

소비자는 CloudHSM 콘솔 및 을 사용하여 공유된 백업을 식별할 수 있습니다. AWS CLI

CloudHSM 콘솔을 사용하여 사용자와 공유한 백업을 식별하려면

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 를 변경하려면 AWS 리전페이지 오른쪽 상단에 있는 지역 선택기를 사용하십시오.
3. 탐색 창에서 백업을 선택합니다.
4. 표에서 공유 백업 탭을 선택합니다.

를 사용하여 공유한 백업을 식별하려면 AWS CLI

describe-backups 명령어를 `--shared` 파라미터와 함께 사용하여 공유된 백업을 반환할 수 있습니다.

공유 백업에 대한 권한

소유자에 대한 권한

백업 소유자는 공유 백업을 설명 및 관리하고 클러스터를 복원하는 데 사용할 수 있습니다.

소비자에 대한 권한

백업 소비자는 공유 백업을 수정할 수는 없지만 공유 백업을 설명하고 클러스터를 복원하는 데 사용할 수는 있습니다.

결제 및 측정

백업 공유에는 추가 요금이 부과되지 않습니다.

리소스 태깅 AWS CloudHSM

태그는 AWS 리소스에 할당하는 레이블입니다. AWS CloudHSM 클러스터에 태그를 할당할 수 있습니다. 각 태그는 사용자가 정의하는 태그 키와 태그 값으로 구성됩니다. 예를 들어, 태그 키는 Cost Center이고 태그 값은 12345일 수 있습니다. 태그 키는 각 클러스터마다 고유해야 합니다.

다양한 용도로 태그를 사용할 수 있습니다. AWS 비용을 분류하고 추적하는 데 주로 사용됩니다. 비즈니스 범주를 나타내는 태그(예: 비용 센터, 애플리케이션 이름 또는 소유자)를 적용하여 여러 서비스에 대한 비용을 정리할 수 있습니다. AWS 리소스에 태그를 추가하면 사용량과 비용이 태그별로 집계된 비용 할당 보고서가 AWS 생성됩니다. 이 보고서를 사용하면 모든 AWS CloudHSM 비용을 단일 항목으로 보는 대신 프로젝트 또는 애플리케이션 측면에서 AWS CloudHSM 비용을 볼 수 있습니다.

비용 할당 태그 사용에 대한 자세한 내용은 AWS Billing 사용자 설명서의 비용 할당 태그 사용을 참조하세요.

AWS CloudHSM 콘솔 또는 AWS SDK나 명령줄 도구 중 하나를 사용하여 태그를 추가, 업데이트, 나열 및 제거할 수 있습니다.

주제

태그 추가 또는 업데이트

AWS CloudHSM 콘솔, AWS Command Line Interface (AWS CLI) 또는 AWS CloudHSM API에서 태그를 추가하거나 업데이트할 수 있습니다.

태그를 추가하거나 업데이트하려면(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 태그를 지정할 클러스터를 선택합니다.
3. [Tags]를 선택합니다.
4. 태그를 추가하려면 다음을 수행합니다.
    a. 태그 편집을 선택한 다음 태그 추가를 선택합니다.
    b. 키에 태그의 키를 입력합니다.

c. (선택 사항) 태그 값에 태그의 값을 입력합니다.

d. 저장을 선택합니다.

5. 태그를 업데이트하려면 다음을 수행합니다.

a. 태그 편집을 선택합니다.

> **Note**
>
> 기존 태그의 태그 키를 업데이트하는 경우, 콘솔은 기존 태그를 삭제하고 새 태그를 생성합니다.

b. 새 태그 값을 입력합니다.

c. 저장을 선택합니다.

태그를 추가하거나 업데이트하려면(AWS CLI)

1. 명령 프롬프트에서 tag-resource 명령을 실행하고 태그 및 태그를 지정할 클러스터의 ID를 지정합니다. 클러스터 ID를 모르면 describe-clusters 명령을 실행합니다.

```
$ aws cloudhsmv2 tag-resource --resource-id <cluster ID> \
                              --tag-list Key="<tag key>",Value="<tag value>"
```

2. 태그를 업데이트하려면 동일한 명령을 사용하되 기존 태그 키를 지정하십시오. 기존 키에 새 태그 값을 지정하면 새 값으로 태그를 덮어씁니다.

태그 추가 또는 업데이트하기 (AWS CloudHSM API)

- TagResource 요청을 보냅니다. 태그 및 태그를 지정할 클러스터의 ID를 지정합니다.

태그 나열

AWS CloudHSM 콘솔 AWS CLI, 또는 AWS CloudHSM API에서 클러스터의 태그를 나열할 수 있습니다.

태그를 나열하려면(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.

2. 태그를 나열할 클러스터를 선택합니다.
3. [Tags]를 선택합니다.

태그를 나열하려면(AWS CLI)

- 명령 프롬프트에서 list-tags 명령을 실행하고 태그를 나열할 클러스터의 ID를 지정합니다. 클러스터 ID를 모르면 describe-clusters 명령을 실행합니다.

```
$ aws cloudhsmv2 list-tags --resource-id <cluster ID>
{
    "TagList": [
        {
            "Key": "Cost Center",
            "Value": "12345"
        }
    ]
}
```

태그를 나열하려면 (AWS CloudHSM API)

- ListTags 요청을 전송하고 태그를 나열할 클러스터의 ID를 지정합니다.

태그 제거

AWS CloudHSM 콘솔, AWS CLI 또는 AWS CloudHSM API를 사용하여 클러스터에서 태그를 제거할 수 있습니다.

태그를 제거하려면(콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. 태그를 제거할 클러스터를 선택합니다.
3. [Tags]를 선택합니다.
4. 태그 편집을 선택한 다음 제거하려는 태그에 대한 태그 제거를 선택합니다.
5. 저장을 선택합니다.

태그를 제거하려면(AWS CLI)

- 명령 프롬프트에서 untag-resource 명령을 실행하고 제거할 태그의 태그 키와 태그를 제거할 클러스터의 ID를 지정합니다. 를 AWS CLI 사용하여 태그를 제거할 때는 태그 값이 아닌 태그 키만 지정하십시오.

```
$ aws cloudhsmv2 untag-resource --resource-id <cluster ID> \
                                --tag-key-list "<tag key>"
```

태그 삭제하기 (AWS CloudHSM API)

- 클러스터의 ID와 제거하려는 태그를 지정하여 AWS CloudHSM API로 UntagResource요청을 보냅니다.

HSM 사용자 및 키 관리 AWS CloudHSM

클러스터를 사용하여 암호화를 처리하려면 먼저 AWS CloudHSM 클러스터의 HSM에 사용자와 키를 생성해야 합니다. AWS CloudHSM에서 HSM 사용자 및 키를 관리하는 방법에 대한 자세한 내용은 다음 주제를 참조하십시오. 쿼럼 인증(N 중 M 액세스 제어라고도 함)을 사용하는 방법도 배울 수 있습니다.

주제

- HSM 사용자 관리 AWS CloudHSM
- 에서 키 관리 AWS CloudHSM
- 복제된 클러스터 관리

HSM 사용자 관리 AWS CloudHSM

AWS CloudHSM에서는 CloudHSM CLI 또는 CloudHSM 관리 유틸리티 (CMU) 명령줄 도구를 사용하여 HSM에서 사용자를 생성하고 관리해야 합니다. CloudHSM CLI는 최신 SDK 버전 시리즈와 함께 사용하도록 설계된 반면 CMU는 이전 SDK 버전 시리즈와 함께 사용하도록 설계되었습니다.

주제

- CloudHSM CLI를 사용한 HSM 사용자 관리
- CloudHSM 관리 유틸리티(CMU)를 통한 HSM 사용자 관리

CloudHSM CLI를 사용한 HSM 사용자 관리

CloudHSM CLI 명령줄 툴을 사용하면 최신 SDK로 HSM에서 사용자를 생성하고 관리할 수 있습니다.

주제

- HSM 사용자 이해
- HSM 사용자 권한 테이블
- CloudHSM CLI를 사용하여 사용자 관리
- CloudHSM CLI를 사용하여 MFA를 관리합니다
- CloudHSM CLI를 사용하여 쿼럼 인증 관리(M/N 액세스 제어)

HSM 사용자 이해

HSM에서 수행하는 대부분의 작업에는 HSM 사용자의 보안 인증이 필요합니다. HSM은 각 HSM 사용자를 인증하며, 각 HSM 사용자에게는 해당 사용자로서 HSM에서 수행할 수 있는 작업을 결정하는 유형이 있습니다.

> **Note**
>
> HSM 사용자는 IAM 사용자와 다릅니다. 올바른 자격 증명을 보유한 IAM 사용자는 AWS API를 통해 리소스와 상호 작용하여 HSM을 생성할 수 있습니다. HSM을 생성한 후에는 HSM 사용자 자격 증명을 사용하여 HSM에서의 작업을 인증해야 합니다.

사용자 유형

- 활성화되지 않은 관리자
- 관리자
- CU(Crypto User)
- AU(Appliance User)

활성화되지 않은 관리자

CloudHSM CLI에서 활성화되지 않은 관리자는 활성화되지 않은 AWS CloudHSM 클러스터의 첫 번째 HSM에만 존재하는 임시 사용자입니다. 클러스터를 활성화하려면 CloudHSM CLI에서 cluster activate 명령을 실행합니다. 이 명령을 실행하면 활성화되지 않은 관리자에게 암호를 변경하라는 메시지가 표시됩니다. 암호를 변경한 후 활성화되지 않은 관리자는 관리자가 됩니다.

관리자

CloudHSM CLI에서 관리자는 사용자 관리 작업을 수행할 수 있습니다. 예를 들어, 사용자를 생성 및 삭제하고 사용자 암호를 변경할 수 있습니다. 관리자에 대한 자세한 내용은 HSM 사용자 권한 테이블 단원을 참조하십시오.

CU(Crypto User)

CU(Crypto User)는 다음 키 관리 및 암호화 작업을 수행할 수 있습니다.

- 키 관리 – 암호화 키 생성, 삭제, 공유, 가져오기 및 내보내기
- 암호화 작업 – 암호화, 암호화 해제, 사인, 확인 등에 암호호화 키 사용

자세한 내용은 HSM 사용자 권한 테이블 단원을 참조하십시오.

AU(Appliance User)

어플라이언스 사용자 (AU) 는 클러스터의 HSM에서 복제 및 동기화 작업을 수행할 수 있습니다. AWS CloudHSM AU를 사용하여 클러스터의 HSM을 동기화합니다. AWS CloudHSM AU는 에서 제공하는 모든 HSM에 AWS CloudHSM존재하며 권한이 제한되어 있습니다. 자세한 내용은 HSM 사용자 권한 테이블을 참조하세요.

AWS HSM에서는 어떤 작업도 수행할 수 없습니다. AWS 사용자 또는 키를 보거나 수정할 수 없으며 해당 키를 사용하여 암호화 작업을 수행할 수 없습니다.

HSM 사용자 권한 테이블

다음 테이블에는 작업을 수행할 수 있는 HSM 사용자 또는 세션 유형별로 정렬된 HSM 작업이 나열되어 있습니다.

	관리자	CU(Crypto User)	AU(Appliance User)	인증된 세션
기본 클러스터 정보 가져오기	 예	 예	 예	 예
자체 암호 변경	 예	 예	 예	해당 사항 없음
사용자의 암호 변경	 예	 아니요	 아니요	 아니요
사용자 추가 및 제거	 예	 아니요	 아니요	 아니요

	관리자	CU(Crypto User)	AU(Appliance User)	인증된 세션
동기화 상태 가져오기	 예	 예	 예	 아니요
마스킹 처리된 객체 추출 및 삽입[3]	 예	 예	 예	 아니요
키 관리 기능[4]	 아니요	 예	 아니요	 아니요
암호화, 암호 해독	 아니요	 예	 아니요	 아니요
사인 및 확인	 아니요	 예	 아니요	 아니요
다이제스트 및 HMAC 생성	 아니요	 예	 아니요	 아니요

- [1] 기본 클러스터 정보에는 클러스터의 HSM 수와 각 HSM의 IP 주소, 모델, 일련 번호, 디바이스 ID, 펌웨어 ID 등이 포함됩니다.

- [2] 사용자는 HSM의 키에 해당하는 다이제스트(해시) 집합을 얻을 수 있습니다. 애플리케이션은 이러한 다이제스트 세트를 비교해서 클러스터에서 HSM의 동기화 상태를 파악할 수 있습니다.
- [3] 마스킹 처리된 객체는 HSM을 떠나기 전에 암호화가 되는 키입니다. HSM 밖에서는 암호를 해독할 수 없습니다. 키가 추출된 HSM과 같은 클러스터에 있는 HSM에 삽입된 후에만 암호가 해독됩니다. 애플리케이션은 마스킹 처리된 객체를 추출 및 삽입하여 클러스터에서 HSM을 동기화할 수 있습니다.
- [4] 키 관리 함수에는 키 속성 생성, 삭제, 래핑, 언래핑, 수정이 포함됩니다.

CloudHSM CLI를 사용하여 사용자 관리

이 항목에서는 CloudHSM CLI를 사용하여 하드웨어 보안 모듈 (HSM) 사용자를 관리하는 방법에 대한 step-by-step 지침을 제공합니다. CloudHSM CLI 또는 HSM 사용자에 대한 자세한 내용은 CloudHSM CLI 및 CloudHSM CLI 사용 단원을 참조하십시오.

Sections

- CloudHSM CLI를 사용한 HSM 사용자 관리에 대한 이해
- CloudHSM CLI 다운로드
- CloudHSM CLI를 사용하여 HSM 사용자를 관리하는 방법

CloudHSM CLI를 사용한 HSM 사용자 관리에 대한 이해

HSM 사용자를 관리하려면 관리자의 사용자 이름과 암호로 HSM에 로그인해야 합니다. 관리자만 사용자를 관리할 수 있습니다. HSM에는 admin이라는 기본 CO가 포함됩니다. 클러스터를 활성화할 때 admin에 대해 사용할 암호를 설정했습니다.

CloudHSM CLI를 사용하려면 구성 툴을 사용하여 로컬 구성을 업데이트해야 합니다. CloudHSM CLI를 사용하여 구성 툴을 실행하는 방법에 대한 지침은 CloudHSM 명령줄 인터페이스(CLI) 시작하기 단원을 참조하십시오. -a 파라미터를 사용하려면 클러스터에 HSM의 IP 주소를 추가해야 합니다. HSM이 여러 개 있는 경우 모든 IP 주소를 사용할 수 있습니다. 이렇게 하면 CloudHSM CLI가 전체 클러스터에서 변경한 내용을 전파할 수 있습니다. CloudHSM CLI는 로컬 파일을 사용하여 클러스터 정보를 추적한다는 점을 기억하십시오. 특정 호스트에서 CloudHSM CLI를 마지막으로 사용한 이후 클러스터가 변경된 경우 해당 호스트에 저장된 로컬 구성 파일에 해당 변경 사항을 추가해야 합니다. CloudHSM CLI를 사용하는 동안에는 HSM을 제거하지 마십시오.

HSM의 IP 주소를 가져오려면 (콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.

2. AWS 리전을 변경하려면 페이지의 오른쪽 상단에 있는 리전 선택기를 사용합니다.
3. 클러스터 세부 정보 페이지를 열려면 클러스터 테이블에서 클러스터 ID를 선택합니다.
4. IP 주소를 가져오려면 HSM 탭에서 ENI IP 주소 아래에 나열된 IP 주소 중 하나를 선택합니다.

HSM의 IP 주소를 가져오려면 ()AWS CLI

- AWS CLI의 describe-clusters 명령을 사용하여 HSM의 IP 주소를 가져옵니다. 명령의 출력에서 HSM의 IP 주소는 `EniIp`의 값입니다.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

CloudHSM CLI 다운로드

최신 버전의 CloudHSM CLI는 Client SDK 5의 HSM 사용자 관리 작업에 사용할 수 있습니다. CloudHSM CLI를 다운로드하고 설치하려면 CloudHSM CLI 설치 및 구성의 지침을 따르십시오.

CloudHSM CLI를 사용하여 HSM 사용자를 관리하는 방법

이 섹션에는 CloudHSM CLI를 사용하여 HSM 사용자를 관리하는 기본 명령이 포함되어 있습니다.

> **Note**
>
> 참고: CloudHSM CLI 사용자 명령은 CloudHSM CLI 사용자 명령 참조에 나열되어 있습니다

주제

관리자 생성

이러한 단계를 따라 관리자를 생성합니다.

1. 다음 명령을 사용하여 CloudHSM CLI 대화형 모드를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. login 명령을 사용하고 클러스터에 관리자로 로그인합니다.

   ```
   aws-cloudhsm > login --username <USERNAME> --role admin
   ```

3. 시스템이 암호를 묻는 메시지를 표시합니다. 암호를 입력하면 명령이 성공했다는 결과가 출력됩니다.

   ```
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "admin",
       "role": "admin"
     }
   }
   ```

4. 다음 명령을 입력하여 을 생성합니다:

```
aws-cloudhsm > user create --username <USERNAME> --role admin
```

5. 데이터베이스 사용자의 암호를 입력합니다.
6. 암호를 다시 입력하여 입력한 암호가 정확한지 확인합니다.

crypto user 생성

이러한 단계를 따라 사용자를 생성합니다.

1. 다음 명령을 사용하여 CloudHSM CLI 대화형 모드를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. login 명령을 사용하고 클러스터에 관리자로 로그인합니다.

   ```
   aws-cloudhsm > login --username <USERNAME> --role admin
   ```

3. 시스템이 암호를 묻는 메시지를 표시합니다. 암호를 입력하면 명령이 성공했다는 결과가 출력됩니다.

   ```
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "admin",
       "role": "admin"
     }
   }
   ```

4. 다음 명령을 입력하여 crypto user를 생성합니다:

   ```
   aws-cloudhsm > user create --username <USERNAME> --role crypto-user
   ```

5. 새 crypto user의 암호를 입력합니다.

6. 암호를 다시 입력하여 입력한 암호가 정확한지 확인합니다.

클러스터의 모든 HSM 사용자를 나열하려면

user list 명령을 사용하여 클러스터의 모든 사용자를 나열합니다. user list를 실행하기 위해 로그인할 필요는 없습니다. 모든 사용자 유형에서 사용자를 나열할 수 있습니다.

다음 단계에 따라 클러스터의 모든 사용자를 나열하십시오

1. 다음 명령을 사용하여 CloudHSM CLI 대화형 모드를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. 다음 명령을 입력하여 클러스터의 사용자를 나열합니다:

   ```
   aws-cloudhsm > user list
   ```

user list에 대한 자세한 내용은 사용자 목록을 참조하십시오.

HSM 사용자 암호 변경

user change-password 명령을 사용하여 암호를 변경합니다.

사용자 유형 및 암호는 대소문자를 구분하지만 사용자 이름은 대소문자를 구분하지 않습니다.

CU(Crypto User) 및 AU(appliance user)는 자체 암호만 변경할 수 있습니다. 다른 사용자의 암호를 변경하려면 관리자로 로그인해야 합니다. 현재 클라이언트 또는 에 로그인되어 있는 사용자의 암호를 변경할 수 없습니다.

암호 변경

1. 다음 명령을 사용하여 CloudHSM CLI 대화형 모드를 시작합니다.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
```

2. login 명령을 사용하고 암호를 변경할 사용자로서 로그인합니다.

```
aws-cloudhsm > login --username <USERNAME> --role <ROLE>
```

3. 사용자의 암호를 입력합니다.

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

4. user change-password 명령을 입력합니다.

```
aws-cloudhsm > user change-password --username <USERNAME> --role <ROLE>
```

5. 새 암호를 입력합니다.
6. 새 암호를 다시 입력합니다.

다른 사용자의 암호 변경

1. 다음 명령을 사용하여 CloudHSM CLI 대화형 모드를 시작합니다.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
```

2. login 명령을 사용하고 관리자로 로그인합니다.

```
aws-cloudhsm > login --username <USERNAME> --role admin
```

3. 관리자 암호를 입력합니다.

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

4. 암호를 변경할 사용자의 사용자 이름과 함께 user change-password 명령을 입력합니다.

```
aws-cloudhsm > user change-password --username <USERNAME> --role <ROLE>
```

5. 새 암호를 입력합니다.
6. 새 암호를 다시 입력합니다.

user change-password에 대한 자세한 정보는, 사용자 암호 변경을 참조하십시오.

HSM 사용자 삭제

user delete를 사용하여 사용자를 삭제합니다. 다른 사용자를 삭제하려면 관리자로 로그인해야 합니다.

> **Tip**
> 키를 소유한 CU(Crypto User)는 삭제할 수 없습니다.

사용자 삭제

1. 다음 명령을 사용하여 CloudHSM CLI 대화형 모드를 시작합니다.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
```

2. login 명령을 사용하고 클러스터에 관리자로 로그인합니다.

```
aws-cloudhsm > login --username <USERNAME> --role admin
```

3. 시스템이 암호를 묻는 메시지를 표시합니다. 암호를 입력하면 명령이 성공했다는 결과가 출력됩니다.

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

4. user delete 명령을 사용하여 사용자를 삭제합니다.

```
aws-cloudhsm > user delete --username <USERNAME> --role <ROLE>
```

user delete에 대한 자세한 내용은 deleteUser를 참조하십시오.

CloudHSM CLI를 사용하여 MFA를 관리합니다

보안 강화를 위해 멀티 팩터 인증(MFA)을 구성하여 리소스를 보호하는 것이 좋습니다. 자세한 내용은 아래 주제를 참조하십시오.

주제

- HSM 사용자를 위한 MFA 이해
- HSM 사용자를 위한 MFA 사용

HSM 사용자를 위한 MFA 이해

MFA 지원 HSM 사용자 계정으로 클러스터에 로그인하면 Cloud HSM CLI에 암호(알 수 있는 첫 번째 요소)를 제공하고 Cloud HSM CLI는 토큰을 제공하고 토큰에 사인하라는 메시지를 표시합니다.

가지고 있는 두 번째 요소를 제공하려면 이미 생성하여 HSM 사용자와 연결한 키 페어의 프라이빗 키로 토큰에 서명합니다. 클러스터에 액세스하려면 사인된 토큰을 CloudHSM CLI에 제공합니다.

사용자에 대한 MFA 설정에 대한 자세한 내용은 CloudHSM CLI용 MFA 설정 단원을 참조하십시오

쿼럼 인증 및 MFA

클러스터는 쿼럼 인증과 MFA에 동일한 키를 사용합니다. 즉, MFA가 활성화된 사용자는 MoFN 또는 쿼럼 액세스 제어에 효과적으로 등록됩니다. 동일한 HSM 사용자에 대해 MFA 및 쿼럼 인증을 성공적으로 사용하려면 다음 사항을 고려하십시오.

- 현재 사용자에 대해 쿼럼 인증을 사용하는 경우 쿼럼 사용자에 대해 생성한 것과 동일한 키 페어를 사용하여 해당 사용자에 대해 MFA를 활성화해야 합니다.
- 쿼럼 인증 사용자가 아닌 비 MFA 사용자에 대한 MFA 요구 사항을 추가하는 경우, 해당 사용자를 MFA 인증을 통해 쿼럼(MoFN) 등록 사용자로 등록합니다.
- 등록된 쿼럼 인증 사용자이기도 한 MFA 사용자의 MFA 요구 사항을 제거하거나 암호를 변경하면 해당 사용자의 쿼럼(MoFN) 사용자 등록도 제거됩니다.
- 쿼럼 인증 사용자이기도 한 MFA 사용자의 MFA 요구 사항을 제거하거나 암호를 변경하면서도 해당 사용자가 여전히 쿼럼 인증에 참여하도록 하려면 해당 사용자를 쿼럼(MoFn) 사용자로 다시 등록해야 합니다.

쿼럼 인증에 대한 자세한 내용은 쿼럼 관리(M/N) 단원을 참조하십시오.

HSM 사용자를 위한 MFA 사용

이 항목에서는 CloudHSM CLI를 사용하여 멀티팩터 인증(MFA)을 관리하기 위한 정보와 지침을 제공합니다. CloudHSM CLI에 대한 자세한 내용은 CloudHSM 명령줄 인터페이스(CLI) 단원을 참조하십시오.

주제

- MFA 키 페어 요구 사항
- CloudHSM CLI용 MFA 설정
- MFA가 활성화된 사용자 생성
- MFA가 활성화된 사용자 로그인

- MFA가 활성화된 사용자의 키 순환
- MFA 퍼블릭 키 등록 시 관리자 사용자의 MFA 퍼블릭 키 등록 취소
- 토큰 파일 참조

사용자 작업에 대한 자세한 내용은 CloudHSM 명령줄 인터페이스(CLI) 단원을 참조하십시오.

MFA 키 페어 요구 사항

HSM 사용자를 위한 MFA를 활성화하려면 새 키 페어를 생성하거나 다음 요구 사항을 충족하는 기존 키를 사용할 수 있습니다:

- 키 유형: 비대칭
- 키 사용: 사인 및 확인
- 키 사양: RSA_2048
- 사인 알고리즘에 포함: sha256WithRSAEncryption

> **Note**
>
> 쿼럼 인증을 사용하거나 쿼럼 인증을 사용하려는 경우 쿼럼 인증 및 MFA 단원을 참조하십시오

CloudHSM CLI와 키 페어를 사용하여 MFA가 활성화된 새 관리자 사용자를 생성할 수 있습니다.

CloudHSM CLI용 MFA 설정

CloudHSM CLI용 MFA를 설정하려면 다음 단계를 따르십시오.

1. 토큰 사인 전략을 사용하여 MFA를 설정하려면 먼저 2048비트 RSA 프라이빗 키와 관련 퍼블릭 키를 생성해야 합니다.

```
$ openssl genrsa -out officer1.key 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................+++++
.....................................................................+++++
e is 65537 (0x010001)

$ openssl rsa -in officer1.key -outform PEM -pubout -out officer1.pub
writing RSA key
```

2. CloudHSM CLI를 사용하여 사용자 계정에 로그인합니다.

```
$ cloudhsm-cli interactive
aws-cloudhsm > login --username admin --role admin --cluster-id <cluster ID>
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

3. 그런 다음 명령을 실행하여 MFA 전략을 변경합니다. 파라미터 `--token`을 제공해야 합니다. 이 파라미터 변수는 사인되지 않은 토큰이 기록되는 파일을 지정합니다.

```
aws-cloudhsm > user change-mfa token-sign --token unsigned-tokens.json --
username <USERNAME> --role crypto-user --change-quorum
Enter password:
Confirm password:
```

4. 이제 사인이 필요한 사인되지 않은 토큰이 들어 있는 파일이 생성됨: `unsigned-tokens.json` 이 파일의 토큰 수는 클러스터의 HSM 수에 따라 달라집니다. 각 토큰은 하나의 HSM을 나타냅니다. 이 파일은 JSON 형식이며 프라이빗 키가 있음을 증명하기 위해 사인이 필요한 토큰을 포함하고 있습니다.

```
$ cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
{
    {
      "unsigned": "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
```

```
    }
  ]
}
```

5. 다음 단계는 1단계에서 생성한 프라이빗 키를 사용하여 이러한 토큰에 사인하는 것입니다. 사인을 파일에 다시 배치합니다. 먼저 base64로 인코딩된 토큰을 추출하고 디코딩해야 합니다.

```
$ echo "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
$ base64 -d token3.b64 > token3.bin
```

6. 이제 1단계에서 만든 RSA 프라이빗 키를 사용하여 사인할 수 있는 바이너리 토큰이 생겼습니다.

```
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token1.bin \
      -out token1.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token2.bin \
      -out token2.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token3.bin \
      -out token3.sig.bin
```

7. 이제 토큰의 바이너리 사인이 생성되었습니다. base64를 사용하여 인코딩한 다음 토큰 파일에 다시 배치해야 합니다.

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
```

```
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

8. 마지막으로 base64 값을 복사하여 토큰 파일에 다시 붙여넣을 수 있습니다:

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```

9. 이제 토큰 파일에 필요한 모든 사인이 포함되었으므로 계속 진행할 수 있습니다. 사인된 토큰이 들어 있는 파일 이름을 입력하고 Enter 키를 누릅니다. 마지막으로 퍼블릭 키의 경로를 입력합니다.

```
Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:officer1.pub
{
  "error_code": 0,
```

```
  "data": {
    "username": "<USERNAME>",
    "role": "crypto-user"
  }
}
```

이제 사용자를 MFA로 설정했습니다.

```
{
    "username": "<USERNAME>",
    "role": "crypto-user",
    "locked": "false",
    "mfa": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
},
```

MFA가 활성화된 사용자 생성

다음 단계에 따라 MFA가 활성화된 사용자를 생성하십시오.

1. CloudHSM CLI를 사용하여 HSM에 관리자로 로그인합니다.
2. user create 명령을 사용하여 원하는 사용자를 생성합니다. 그런 다음 CloudHSM CLI용 MFA 설정의 단계에 따라 사용자를 위한 MFA를 설정합니다.

MFA가 활성화된 사용자 로그인

다음 단계에 따라 MFA를 활성화한 상태로 사용자를 로그인하십시오.

1. CloudHSM CLI의 login mfa-token-sign 명령을 사용하여 MFA를 활성화한 사용자에 대해 MFA로 로그인 프로세스를 시작할 수 있습니다.

```
aws-cloudhsm > login --username <USERNAME> --role <ROLE> mfa-token-sign --token
 unsigned-tokens.json
Enter password:
```

2. 암호를 입력합니다. 그러면 사인되지 않은/사인된 토큰 페어가 포함된 토큰 파일의 경로를 입력하라는 메시지가 표시됩니다. 여기서 사인된 토큰은 프라이빗 키를 사용하여 생성된 토큰입니다.

```
aws-cloudhsm > login --username <USERNAME> --role <ROLE> mfa-token-sign --token
 unsigned-tokens.json
Enter password:
Enter signed token file path (press enter if same as the unsigned token file):
```

3. 사인된 토큰 파일 경로를 입력하라는 메시지가 표시되면 별도의 터미널에서 사인되지 않은 토큰 파일을 검사할 수 있습니다. 사인이 필요한 사인되지 않은 토큰이 있는 파일 식별: unsigned-tokens.json 이 파일의 토큰 수는 클러스터의 HSM 수에 따라 달라집니다. 각 토큰은 하나의 HSM을 나타냅니다. 이 파일은 JSON 형식이며 프라이빗 키가 있음을 증명하기 위해 사인이 필요한 토큰을 포함하고 있습니다.

```
$ cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

4. 2단계에서 생성한 프라이빗 키로 사인되지 않은 토큰에 사인합니다. 먼저 base64로 인코딩된 토큰을 추출하고 디코딩해야 합니다.

```
$ echo "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
```

```
$ base64 -d token3.b64 > token3.bin
```

5. 이제 바이너리 토큰이 생겼습니다. MFA 설정 단계 1에서 이전에 생성한 RSA 프라이빗 키를 사용하여 사인합니다.

```
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token1.bin \
      -out token1.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token2.bin \
      -out token2.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token3.bin \
      -out token3.sig.bin
```

6. 이제 토큰의 바이너리 사인이 생겼습니다. base64를 사용하여 인코딩한 다음 토큰 파일에 다시 배치합니다.

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

7. 마지막으로 base64 값을 복사하여 토큰 파일에 다시 붙여넣습니다.

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
```

```
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```

8. 이제 토큰 파일에 필요한 모든 사인이 포함되었으므로 계속 진행할 수 있습니다. 사인된 토큰이 들어 있는 파일 이름을 입력하고 Enter 키를 누릅니다. 이제 성공적으로 로그인할 수 있습니다.

```
aws-cloudhsm > login --username <USERNAME> --role <ROLE> mfa-token-sign --token
 unsigned-tokens.json
Enter password:
Enter signed token file path (press enter if same as the unsigned token file):
{
  "error_code": 0,
  "data": {
    "username": "<USERNAME>",
    "role": "<ROLE>"
  }
}
```

MFA가 활성화된 사용자의 키 순환

다음 단계에 따라 MFA를 활성화한 사용자의 키를 순환하십시오.

<result>

생성된 JSON 형식 토큰 파일에 프라이빗 키로 사인하고 새 MFA 퍼블릭 키를 등록했습니다.
</result>

1. CloudHSM CLI를 사용하여 관리자 또는 MFA가 활성화된 특정 사용자로 HSM에 로그인합니다(자세한 내용은 MFA가 활성화된 상태로 사용자 로그인 참조).
2. 그런 다음 명령을 실행하여 MFA 전략을 변경합니다. 파라미터 `--token`을 제공해야 합니다. 이 파라미터 변수는 사인되지 않은 토큰이 기록되는 파일을 지정합니다.

```
aws-cloudhsm > user change-mfa token-sign --token unsigned-tokens.json --
username <USERNAME> --role crypto-user --change-quorum
Enter password:
Confirm password:
```

3. 사인이 필요한 사인되지 않은 토큰이 있는 파일 식별: `unsigned-tokens.json` 이 파일의 토큰 수는 클러스터의 HSM 수에 따라 달라집니다. 각 토큰은 하나의 HSM을 나타냅니다. 이 파일은 JSON 형식이며 프라이빗 키가 있음을 증명하기 위해 사인이 필요한 토큰을 포함하고 있습니다. 이 키는 현재 등록된 퍼블릭 키를 교체하는 데 사용하려는 새 RSA 공개/프라이빗 키 페어의 새 프라이빗 키입니다.

```
$cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

4. 설정 과정에서 이전에 생성한 프라이빗 키로 이 토큰에 사인하십시오. 먼저 base64로 인코딩된 토큰을 추출하고 디코딩해야 합니다.

```
$ echo "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=" > token1.b64
$ echo "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=" > token2.b64
$ echo "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=" > token3.b64
$ base64 -d token1.b64 > token1.bin
$ base64 -d token2.b64 > token2.bin
$ base64 -d token3.b64 > token3.bin
```

5. 이제 바이너리 토큰이 생겼습니다. 설정 중에 이전에 생성한 RSA 프라이빗 키를 사용하여 사인하십시오.

```
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token1.bin \
      -out token1.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token2.bin \
      -out token2.sig.bin
$ openssl pkeyutl -sign \
      -inkey officer1.key \
      -pkeyopt digest:sha256 \
      -keyform PEM \
      -in token3.bin \
      -out token3.sig.bin
```

6. 이제 토큰의 바이너리 사인이 생겼습니다. base64를 사용하여 인코딩한 다음 토큰 파일에 다시 배치합니다.

```
$ base64 -w0 token1.sig.bin > token1.sig.b64
$ base64 -w0 token2.sig.bin > token2.sig.b64
$ base64 -w0 token3.sig.bin > token3.sig.b64
```

7. 마지막으로 base64 값을 복사하여 토큰 파일에 다시 붙여넣습니다.

```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/
TK0PVaxLN42X+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/
mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQ
    }
  ]
}
```


8. 이제 토큰 파일에 필요한 모든 사인이 포함되었으므로 계속 진행할 수 있습니다. 사인된 토큰이 들어 있는 파일 이름을 입력하고 Enter 키를 누릅니다. 마지막으로 새 퍼블릭 키의 경로를 입력합니다. 이제 user list 출력의 일부로 다음이 표시됩니다.

```
Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:officer1.pub
{
  "error_code": 0,
  "data": {
    "username": "<USERNAME>",
    "role": "crypto-user"
```

```
  }
}
```

이제 사용자를 MFA로 설정했습니다.

```
{
    "username": "<USERNAME>",
    "role": "crypto-user",
    "locked": "false",
    "mfa": [
      {
        "strategy": "token-sign",
        "status": "enabled"
      }
    ],
    "cluster-coverage": "full"
},
```

MFA 퍼블릭 키 등록 시 관리자 사용자의 MFA 퍼블릭 키 등록 취소

MFA 퍼블릭 키가 등록된 경우 관리자 사용자의 MFA 퍼블릭 키 등록을 취소하려면 다음 단계를 따르십시오.

1. CloudHSM CLI를 사용하여 MFA가 활성화된 상태에서 관리자로 HSM에 로그인할 수 있습니다.
2. user change-mfa token-sign 명령을 사용하여 사용자용 MFA를 제거합니다.

```
aws-cloudhsm > user change-mfa token-sign --username <USERNAME> --role admin --
deregister --change-quorum
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "<USERNAME>",
    "role": "admin"
  }
}
```

토큰 파일 참조

MFA 퍼블릭 키를 등록하거나 MFA를 사용하여 로그인을 시도할 때 생성되는 토큰 파일은 다음과 같이 구성됩니다.

- 토큰: JSON 객체 리터럴 형태의 base64로 인코딩된 사인되지 않은/사인된 토큰 페어 배열.
- 미사인: base64로 인코딩되고 SHA256 해시 처리된 토큰입니다.
- 사인: RSA2048비트 프라이빗 키를 사용하여 사인되지 않은 토큰의 base64로 인코딩된 사인된 토큰(사인).


```
{
  "version": "2.0",
  "tokens": [
    {
      "unsigned": "1jqwxb9bJOUUQLiNb7mxXS1uBJsEXh0B9nj05BqnPsE=",
      "signed": "eiw3fZeCKIY50C4zPeg9Rt90M1Qlq3WlJh6Yw7xXm4nF6e9ETLE39+9M
+rUqDWMRZjaBfaMbg5d9yDkz5p13U7ch2tlF9LoYabsWutkT014KRq/rcYMvFsU9n/Ey/TK0PVaxLN42X
+pebV4juwMhN4mK4CzdFAJgM+UGBOj4yB9recpOBB9K8QFSpJZALSEdDgUc/mS1eDq3rU0int6+4NKuLQjpR
+LSEIWRZ6g6+MND2vXGskxHjadCQ09L7Tz8VcWjKDbxJcBiGKvkqyozl9zrGo8fA3WHBmwiAgS61Merx77ZGY4PFR37+j/
YMSC14prCN15DtMRv2xA1SGSb4w=="
    },
    {
      "unsigned": "LMMFc34ASPnvNPFzBbMbr9FProS/Zu2P8zF/xzk5hVQ=",
      "signed": "HBImKnHmw+6R2TpFEpfiAg4+hu2pFNwn43ClhKPkn2higbEhUD0JVi
+4MerSyvU/NN79iWVxDvJ9Ito+jpiRQjTfTGEoIteyuAr1v/Bzh+HjmrO53OQpZaJ/VXGIgApD0myuu/
ZGNKQTCSkkL7+V81FG7yR1Nm22jUeGa735zvm/E+cenvZdy0VVx6A7WeWrl3JEKKBweHbi+7BwbaW
+PTdCuIRd4Ug76Sy+cFhsvcG1k7cMwDh8MgXzIZ2m1f/hdy2j8qAxORTLlmwyUOYvPYOvUhc
+s83hx36QpGwGcD7RA0bPT5OrTx7PHd0N1CL+Wwy91We8yIOFBS6nxo1R7w=="
    },
    {
      "unsigned": "dzeHbwhiVXQqcUGj563z51/7sLUdxjL93SbOUyZRjH8=",
      "signed": "VgQPvrTsvGljVBFxHnswduq16x8ZrnxfcYVYGf/
N7gEzI4At3GDs2EVZWTRdvS0uGHdkFYp1apHgJZ7PDVmGcTkIXVD2lFYppcgNlSzkYlftr5EOjqS9ZjYEqgGuB4g//
MxaBaRbJai/6BlcE92NIdBusTtreIm3yTpjIXNAVoeRSnkfuw7wZcL96QoklNb1WUuSHw
+psUyeIVtIwFMHEfFoRC0t
+VhmnlnFnkjGPb9W3Aprw2dRRvFM3R2ZTDvMCiOYDzUCd43GftGq2LfxH3qSD51oFHglHQVOY0jyVzzlAvub5HQdtOQdErI
    }
  ]
}
```

CloudHSM CLI를 사용하여 쿼럼 인증 관리(M/N 액세스 제어)

AWS CloudHSM 클러스터의 HSM은 쿼럼 인증 (M of N 액세스 제어라고도 함) 을 지원합니다. 쿼럼 인증의 경우, HSM의 단일 사용자는 HSM에서 쿼럼 제어 작업을 수행할 수 없습니다. 이러한 작업을 하려면 최소 숫자의 HSM 사용자(최소 2명)가 협력해야 합니다. 쿼럼 인증의 경우, 2명 이상의 HSM 사용자의 승인을 요구함으로써 보호 계층을 추가할 수 있습니다.

쿼럼 인증은 다음 작업을 제어할 수 있습니다.

- 관리자의 HSM 사용자 관리 – HSM 사용자 생성 및 삭제, 다른 HSM 사용자의 암호 변경. 자세한 정보는 관리자를 위한 쿼럼 인증 사용을 참조하세요.

다음 주제에는 AWS CloudHSM의 쿼럼 인증에 대한 자세한 내용이 있습니다.

주제

토큰 서명 전략을 사용한 쿼럼 인증 개요

다음 단계는 쿼럼 인증 프로세스를 요약합니다. 구체적인 단계와 도구는 관리자를 위한 쿼럼 인증 사용을 참조하십시오.

1. 각각의 HSM 사용자는 서명을 위한 비대칭 키를 생성합니다. 사용자들은 HSM 외부에서 이를 수행하며 키를 적절히 보호하기 위해 주의합니다.
2. 각 HSM 사용자는 HSM에 로그인하고 자신의 서명 키의 퍼블릭 부분(퍼블릭 키)을 HSM에 등록합니다.
3. HSM 사용자가 쿼럼 제어 작업을 수행하려는 경우, 동일한 사용자가 HSM에 로그인하고 쿼럼 토큰을 가져옵니다.
4. HSM 사용자는 한 명 이상의 다른 HSM 사용자에게 쿼럼 토큰을 제공하고 승인을 요청합니다.
5. 다른 HSM 사용자는 자신의 키를 사용하여 암호화된 방식으로 쿼럼 토큰에 서명함으로써 승인합니다. 이는 HSM 외부에서 이루어집니다.

6. HSM 사용자가 필요한 수의 승인을 받으면 동일한 사용자가 HSM에 로그인하고 --approval 인수로 쿼럼 제어 작업을 실행하여 필요한 모든 승인(서명)이 포함된 서명된 쿼럼 토큰 파일을 제공합니다.
7. HSM은 각 서명자의 등록된 퍼블릭 키를 사용하여 서명을 확인합니다. 서명이 유효하면 HSM이 토큰을 승인하고 쿼럼 제어 작업이 수행됩니다.

쿼럼 인증에 대한 추가 세부 정보

AWS CloudHSM의 쿼럼 인증 사용에 대해 다음 내용을 추가로 참고하십시오.

- HSM 사용자는 자신의 쿼럼 토큰에 서명할 수 있습니다. 즉, 요청한 사용자는 쿼럼 인증에 필요한 승인 중 하나를 제공할 수 있습니다.
- 쿼럼 제어 작업을 위한 최소 숫자의 쿼럼 승인자를 선택합니다. 선택할 수 있는 최소 숫자는 2이고, 선택할 수 있는 최대 수는 8입니다.
- HSM은 최대 1024개의 쿼럼 토큰을 저장할 수 있습니다. 새 토큰을 생성하려고 할 때 HSM에 이미 1024개의 토큰이 있는 경우, HSM은 만료된 토큰 중 하나를 삭제합니다. 기본적으로 토큰은 생성 10분 후에 만료됩니다.
- MFA가 활성화된 경우 클러스터는 쿼럼 인증과 멀티 팩터 인증(MFA)에 동일한 키를 사용합니다. 쿼럼 인증 및 2FA 사용에 대한 자세한 내용은 CloudHSM CLI를 사용하여 MFA 관리를 참조하십시오.
- 각 HSM은 서비스당 한 번에 하나의 토큰만 포함할 수 있습니다.

쿼럼 인증을 지원하는 서비스 이름 및 유형

관리 서비스: 쿼럼 인증은 사용자 생성, 사용자 삭제, 사용자 비밀번호 변경, 쿼럼 값 설정, 쿼럼 및 MFA 기능 비활성화와 같은 관리자 권한 서비스에 사용됩니다.

각 서비스 유형은 수행할 수 있는 특정 쿼럼 지원 서비스 작업 세트를 포함하는 적격 서비스 이름으로 더 세분화됩니다.

서비스 이름	서비스 유형	서비스 작업
사용자	관리자	• 사용자 생성 • 사용자 삭제 • 사용자의 암호-변경 • 사용자 변경-mfa

서비스 이름	서비스 유형	서비스 작업
쿼럼	관리자	• 쿼럼 토큰 사인 set-quorum-value

관리자를 위한 쿼럼 인증 사용: 최초 설정

다음 주제에서는 관리자가 쿼럼 인증을 사용할 수 있도록 하드웨어 보안 모듈(HSM)을 구성하기 위해 완료해야 하는 단계를 설명합니다. 관리자에 대한 쿼럼 인증을 처음 구성할 때 이러한 단계를 한 번만 수행하면 됩니다. 이 단계들을 완료한 후 관리자를 위한 쿼럼 인증 사용 섹션을 참조하십시오.

주제

- 사전 조건
- 서명용 키 생성 및 등록
- HSM에서 쿼럼 최소값 설정

사전 조건

이 예를 이해하려면 CloudHSM CLI에 익숙해야 합니다. 이 예시에서는 명령의 다음 출력과 같이 AWS CloudHSM 클러스터에 각각 동일한 관리자가 있는 HSM이 두 개 있습니다. user list 사용자 생성에 대한 자세한 내용은 CloudHSM CLI 사용 단원을 참조하십시오.

```
aws-cloudhsm>user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin2",
        "role": "admin",
        "locked": "false",
```

```
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin3",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin4",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

서명용 키 생성 및 등록

쿼럼 인증을 사용하려면 각 관리자가 다음 단계를 모두 완료해야 합니다.

주제

- RSA 키 페어 생성
- 등록 토큰 생성 및 서명
- HSM에 퍼블릭 키 등록

RSA 키 페어 생성

키 쌍을 생성하고 보호하는 방법에는 여러 가지가 있습니다. 다음 예제에서는 OpenSSL을 사용한 작업 방법을 보여 줍니다.

Example — OpenSSL을 사용하여 개인 키를 생성하세요

다음 예제는 OpenSSL을 사용하여 암호로 보호되는 2048비트 RSA 키를 생성하는 방법을 보여 줍니다. 이 예를 사용하려면 *<admin.key>*를 키를 저장하려는 파일 이름으로 바꾸십시오.

```
$ openssl genrsa -out <admin.key> -aes256 2048
Generating RSA private key, 2048 bit long modulus
.....................................+++
.+++
e is 65537 (0x10001)
Enter pass phrase for admin.key:
Verifying - Enter pass phrase for admin.key:
```

다음으로 방금 생성한 프라이빗 키를 사용하여 퍼블릭 키를 생성합니다.

Example — OpenSSL로 퍼블릭 키 생성하기

다음 예에서는 OpenSSL을 사용하여 방금 생성한 프라이빗 키에서 퍼블릭 키를 생성하는 방법을 보여 줍니다.

```
$ openssl rsa -in admin.key -outform PEM -pubout -out admin1.pub
Enter pass phrase for admin.key:
writing RSA key
```

등록 토큰 생성 및 서명

토큰을 생성하고 이전 단계에서 방금 생성한 프라이빗 키로 서명합니다.

Example — 등록 토큰 생성

1. 다음 명령을 사용하여 CloudHSM CLI를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
```

2. 쿼럼 토큰 서명 생성 명령을 실행하여 등록 토큰을 생성합니다.

```
aws-cloudhsm > quorum token-sign generate --service registration --token /path/
tokenfile
{
  "error_code": 0,
  "data": {
    "path": "/path/tokenfile"
  }
}
```

3. 쿼럼 토큰 서명 생성 명령은 지정된 파일 경로에 등록 토큰을 생성합니다. 토큰 파일 검사:

```
$ cat /path/tokenfile{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": ""
    }
  ]
}
```

토큰 파일은 다음으로 구성됩니다.

- approval_data: 원시 데이터가 최대 245바이트를 초과하지 않는 base64로 인코딩된 무작위 데이터 토큰입니다.
- 서명되지 않음: approval_data의 base64로 인코딩되고 SHA256 해시 처리된 토큰입니다.
- 서명됨: 이전에 OpenSSL로 생성된 RSA 2048비트 개인 키를 사용하여 서명되지 않은 토큰의 base64로 인코딩된 서명된 토큰(서명)입니다.

개인 키로 서명되지 않은 토큰에 서명하여 개인 키에 액세스할 수 있음을 증명합니다. 관리자를 클러스터의 쿼럼 사용자로 등록하려면 서명과 공개 키로 완전히 채워진 등록 토큰 파일이 필요합니다. AWS CloudHSM

Example — 서명되지 않은 등록 토큰에 서명

1. base64로 인코딩된 서명되지 않은 토큰을 디코딩하여 바이너리 파일에 배치합니다:

```
$ echo -n '6BMUj6mUjjko6ZLCEdzGlWpR5sILhFJfqhW1ej3Oq1g=' | base64 -d > admin.bin
```

2. OpenSSL과 개인 키를 사용하여 현재 서명되지 않은 바이너리 등록 토큰에 서명하고 바이너리 서명 파일을 생성합니다.

```
$ openssl pkeyutl -sign \
-inkey admin.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in admin.bin \
-out admin.sig.bin
```

3. 바이너리 서명을 base64로 인코딩합니다.

```
$ base64 -w0 admin.sig.bin > admin.sig.b64
```

4. base64로 인코딩된 서명을 복사하여 토큰 파일에 붙여넣습니다.

```
{
  "version": "2.0",
  "tokens": [
    {
      "approval_data": <approval data in base64 encoding>,
      "unsigned": <unsigned token in base64 encoding>,
      "signed": <signed token in base64 encoding>
    }
  ]
}
```

HSM에 퍼블릭 키 등록

키를 생성한 후 관리자는 클러스터에 공개 키를 등록해야 합니다. AWS CloudHSM

퍼블릭 키를 HSM에 등록하려면

1. 다음 명령을 사용하여 CloudHSM CLI를 시작합니다.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
```

2. CloudHSM CLI를 사용하여 관리자로 로그인합니다.

```
aws-cloudhsm > login --username admin --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

3. 사용자 변경-쿼럼 토큰-사인 등록 명령을 사용하여 퍼블릭 키를 등록합니다. 자세한 내용은 다음 예제를 참조하거나 help user change-quorum token-sign register 명령을 사용하십시오.

Example — AWS CloudHSM 클러스터에 퍼블릭 키 등록

다음 예에서는 CloudHSM CLI의 user change-quorum token-sign register 명령을 사용하여 관리자의 퍼블릭 키를 HSM에 등록하는 방법을 보여줍니다. 이 명령을 사용하려면 관리자가 HSM에 로그인되어 있어야 합니다. 이 값들을 사용자의 값으로 대체합니다.

```
aws-cloudhsm > user change-quorum token-sign register --public-key </path/admin.pub> --
signed-token </path/tokenfile>
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

> **Note**
>
> /path/admin.pub: 퍼블릭 키 PEM 파일의 파일 경로
> 필수 항목 여부: 예
> /path/tokenfile: 사용자 개인 키로 서명된 토큰이 있는 파일 경로
> 필수 항목 여부: 예

모든 관리자가 퍼블릭 키를 등록하면 user list 명령 출력의 쿼럼 필드에 다음과 같이 활성화된 쿼럼 전략이 사용 중인 것으로 표시되어 다음과 같이 표시됩니다:

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin2",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
```

```
      {
        "username": "admin3",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin4",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

HSM에서 쿼럼 최소값 설정

쿼럼 인증을 사용하려면 관리자가 HSM에 로그인한 다음 쿼럼 최소값을 설정해야 합니다. 이는 HSM 사용자 관리 작업을 수행하는 데 필요한 최소 관리자 승인 수입니다. 서명을 위해 키를 등록하지 않은

관리자를 포함하여 HSM의 모든 관리자는 쿼럼 최소값을 설정할 수 있습니다. 쿼럼 최소값은 언제든지 변경할 수 있습니다. 자세한 내용은 최소값 변경 단원을 참조하십시오.

HSM에서 쿼럼 최소값을 설정하려면

1. 다음 명령을 사용하여 CloudHSM CLI를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. CloudHSM CLI를 사용하여 관리자로 로그인합니다.

   ```
   aws-cloudhsm > login --username admin --role admin
   Enter password:
   {
     "error_code": 0,
     "data": {
       "username": "admin",
       "role": "admin"
     }
   }
   ```

3. 쿼럼 토큰 사인 set-quorum-value 명령을 사용하여 쿼럼 최소값을 설정합니다. 자세한 내용은 다음 예제를 참조하거나 help quorum token-sign set-quorum-value 명령을 사용하십시오.

Example - HSM에서 쿼럼 최소값 설정

이 예제에서는 쿼럼 최소값 이(2)를 사용합니다. HSM의 총 관리자 수까지 이(2)~팔(8) 사이의 값을 선택할 수 있습니다. 이 예제에서 HSM에는 네(4) 명의 관리자가 있으므로 가능한 최대값은 넷(4)입니다.

다음 예제 명령을 사용하려면 마지막 숫자(*<2>*)를 기본 쿼럼 최소값으로 바꿉니다.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service user --value <2>
{
  "error_code": 0,
  "data": "Set quorum value successful"
```

```
}
```

이 예에서 서비스는 쿼럼 최소값을 설정 중인 HSM 서비스를 식별합니다. 이 쿼럼 토큰 사인 list-quorum-values 명령은 서비스에 포함된 HSM 서비스 유형, 이름 및 설명을 나열합니다.

관리 서비스: 쿼럼 인증은 사용자 생성, 사용자 삭제, 사용자 비밀번호 변경, 쿼럼 값 설정, 쿼럼 및 MFA 기능 비활성화와 같은 관리자 권한 서비스에 사용됩니다.

각 서비스 유형은 수행할 수 있는 특정 쿼럼 지원 서비스 작업 세트를 포함하는 적격 서비스 이름으로 더 세분화됩니다.

서비스 이름	서비스 유형	서비스 작업
사용자	관리자	• 사용자 생성 • 사용자 삭제 • 사용자의 암호-변경 • 사용자 변경-mfa
쿼럼	관리자	• 쿼럼 토큰 서명 set-quorum-value

quorum token-sign list-quorum-values 명령을 사용하여 서비스 쿼럼 최소값을 가져옵니다.

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 1
  }
}
```

이전 quorum token-sign list-quorum-values 명령의 출력은 사용자 관리 작업을 담당하는 HSM 사용자 서비스의 쿼럼 최소값이 이제 이(2) 임을 보여줍니다. 이 단계들을 완료한 후 (M/N) 쿼럼 사용 섹션을 참조하십시오.

관리자를 위한 쿼럼 인증 사용

HSM의 관리자는 클러스터의 다음 작업에 대해 쿼럼 인증을 구성할 수 있습니다. AWS CloudHSM

- 사용자 생성
- 사용자 삭제
- 사용자-암호 변경
- 사용자 변경-mfa

쿼럼 인증을 위해 AWS CloudHSM 클러스터를 구성한 후에는 관리자가 직접 HSM 사용자 관리 작업을 수행할 수 없습니다. 다음 예에서는 관리자가 HSM에서 새 사용자를 생성하려고 할 때의 출력을 보여 줍니다. 쿼럼 인증이 필요하다는 오류 메시지와 함께 명령이 실패합니다.

```
aws-cloudhsm > user create --username user1 --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}
```

HSM 사용자 관리 작업을 수행하려면 관리자가 다음 작업을 완료해야 합니다.

주제

- 쿼럼 토큰 가져오기
- 승인하는 관리자로부터 서명 받기
- AWS CloudHSM 클러스터에서 토큰을 승인하고 사용자 관리 작업을 실행합니다.

쿼럼 토큰 가져오기

먼저 관리자는 CloudHSM CLI를 사용하여 쿼럼 토큰을 요청해야 합니다.

쿼럼 토큰을 가져오려면

1. 다음 명령을 사용하여 CloudHSM CLI를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
```

2. login 명령을 사용하고 관리자로 클러스터에 로그인합니다.

```
aws-cloudhsm>login --username admin --role admin
```

3. quorum token-sign generate 명령을 사용하여 쿼럼 토큰을 생성합니다. 자세한 내용은 다음 예제를 참조하거나 help quorum token-sign generate 명령을 사용하십시오.

Example — 쿼럼 토큰 생성

이 예에서는 사용자 이름이 admin인 관리자에 대한 쿼럼 토큰을 가져오고 해당 토큰을 admin.token라는 파일에 저장합니다. 예제 명령을 사용하려면 이들 값을 본인의 것으로 바꿉니다.

- *<admin>* – 토큰을 받는 관리자의 이름. 이는 HSM에 로그인하여 이 명령을 실행하는 관리자와 동일해야 합니다.
- *<admin.token>* – 쿼럼 토큰을 저장하는 데 사용할 파일 이름입니다.

다음 명령에서 user는 생성 중인 토큰을 사용할 수 있는 서비스 이름을 식별합니다. 이 경우 토큰은 HSM 사용자 관리 작업 (user 서비스)을 위한 것입니다. .

```
aws-cloudhsm > login --username <ADMIN> --role <ADMIN> --password <PASSWORD>
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}

aws-cloudhsm > quorum token-sign generate --service user --token </path/admin.token>
{
  "error_code": 0,
  "data": {
    "path": "/home/tfile"
  }
}
```

quorum token-sign generate 명령은 지정된 파일 경로에 사용자 서비스 쿼럼 토큰을 생성합니다. 토큰 파일을 검사할 수 있습니다.

```
$cat </path/admin.token>
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW/
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAABAAMAAAAUA
+b23gAAAAAAAAAA",
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": []
}
```

토큰 파일은 다음으로 구성됩니다.

- approval_data: HSM에서 생성한 base64로 인코딩된 원시 데이터 토큰입니다.
- 토큰: Approval_data의 base64로 인코딩되고 SHA-256 해시된 토큰
- 서명: 승인자의 각 서명이 JSON 객체 리터럴 형식인 서명되지 않은 토큰의 base64로 인코딩된 서명된 토큰(서명) 배열입니다.

  ```
  {
        "username": "<APPROVER_USERNAME>",
        "signature": "<APPROVER_RSA2048_BIT_SIGNATURE>"
  }
  ```

  각 서명은 HSM에 공개 키가 등록된 해당 RSA 2048비트 개인 키를 사용하여 승인자의 결과로 생성됩니다.

생성된 사용자 서비스 쿼럼 토큰이 quorum token-sign list 명령을 실행하여 CloudHSM 클러스터에 존재하는지 확인할 수 있습니다.

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": [
      {
        "username": "admin",
        "service": "user",
```

```
        "approvals-required": {
          "value": 2
        },
        "number-of-approvals": {
          "value": 0
        },
        "token-timeout-seconds": {
          "value": 597
        },
        "cluster-coverage": "full"
      }
    ]
  }
}
```

`token-timeout-seconds` 시간은 생성된 토큰이 만료되기 전에 승인되는 데 걸리는 제한 시간 (초)을 나타냅니다.

승인하는 관리자로부터 서명 받기

쿼럼 토큰이 있는 관리자는 다른 관리자로부터 토큰 승인을 받아야 합니다. 승인을 위해 다른 관리자는 서명 키를 사용하여 암호화 방식으로 토큰에 서명합니다. 이 절차는 HSM 외부에서 이루어집니다.

토큰에 서명하는 방법은 다양합니다. 다음 예제에서는 OpenSSL을 사용한 작업 방법을 보여 줍니다. 다른 서명 도구를 사용하려면 해당 도구가 관리자의 개인 키(서명 키)를 사용하여 토큰의 SHA-256 다이제스트에 서명하는지 확인하십시오.

Example – 승인 관리자로부터 서명 받기

이 예에서 토큰(admin)이 있는 관리자는 최소 두(2) 개의 승인이 필요합니다. 다음 예제 명령은 두(2) 명의 관리자가 OpenSSL을 사용하여 암호화 방식으로 토큰에 서명할 수 있는 방법을 보여줍니다.

1. base64로 인코딩된 서명되지 않은 토큰을 디코딩하여 바이너리 파일에 배치합니다:

   ```
   $echo -n '0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=' | base64 -d > admin.bin
   ```

2. OpenSSL과 승인자 (admin3)의 개별 개인 키를 사용하여 사용자 서비스에 대한 현재 바이너리 쿼럼의 서명되지 않은 토큰에 서명하고 바이너리 서명 파일을 생성합니다.

   ```
   $openssl pkeyutl -sign \
   -inkey admin3.key \
   -pkeyopt digest:sha256 \
   ```

```
-keyform PEM \
-in admin.bin \
-out admin.sig.bin
```

3. 바이너리 서명을 base64로 인코딩합니다.

```
$base64 -w0 admin.sig.bin > admin.sig.b64
```

4. 마지막으로, 앞서 승인자 서명에 대해 지정한 JSON 개체 리터럴 형식에 따라 base64로 인코딩된 서명을 복사하여 토큰 파일에 붙여넣습니다.

```
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAABAAMAA
+b23gAAAAAAAAAA",
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": [
    {
      "username": "admin2",
      "signature": "O6qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+OsDtvYujhEU4aOFVLcrUFmyB/CX9OQmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1nOXwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWWdtDgfT7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyruGFLpXs1n0tJOEglGhx2qbYTs+omKWZdORl5WIWEXW3IXw/
Dg5vVObrNpvG0eZKO8nSMc27+cyPySc+ZbNw=="
    },
    {
      "username": "admin3",
      "signature": "O6qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+OsDtvYujhEU4aOFVLcrUFmyB/CX9OQmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1nOXwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWWdtDgfT7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyruGFLpXs1n0tJOEglGhx2qbYTs+omKWZdORl5WIWEXW3IXw/
Dg5vVObrNpvG0eZKO8nSMc27+cyPySc+ZbNw=="
    }
  ]
}
```

AWS CloudHSM 클러스터에서 토큰을 승인하고 사용자 관리 작업을 실행합니다.

이전 섹션에서 설명한 대로 관리자가 필요한 승인/서명을 받은 후 관리자는 다음 사용자 관리 작업 중 하나와 함께 해당 토큰을 AWS CloudHSM 클러스터에 제공할 수 있습니다.

- create
- delete
- 암호-변경
- user change-mfa

이러한 명령을 사용하는 방법은 CloudHSM CLI 사용 섹션을 참조하십시오.

트랜잭션 중에 토큰은 AWS CloudHSM 클러스터 내에서 승인되고 요청된 사용자 관리 작업을 실행합니다. 사용자 관리 작업의 성공 여부는 승인된 유효한 쿼럼 토큰과 유효한 사용자 관리 작업에 따라 결정됩니다.

관리자는 한 번의 작업에만 토큰을 사용할 수 있습니다. 해당 작업이 성공하면 토큰은 더 이상 유효하지 않습니다. 다른 HSM 사용자 관리 작업을 수행하려면 관리자가 위에 설명된 프로세스를 반복해야 합니다. 즉, 관리자는 새 쿼럼 토큰을 생성하고 승인자로부터 새 서명을 받은 다음 요청된 사용자 관리 작업을 통해 HSM에서 새 토큰을 승인하고 사용해야 합니다.

> **Note**
>
> 쿼럼 토큰은 현재 로그인 세션이 열려 있는 동안에만 유효합니다. CloudHSM CLI에서 로그아웃하거나 네트워크 연결이 끊기면 토큰은 더 이상 유효하지 않습니다. 마찬가지로 승인된 토큰은 CloudHSM CLI 내에서만 사용할 수 있습니다. 다른 애플리케이션에서 인증하는 데는 사용할 수 없습니다.

Example 관리자로 새 사용자 생성

다음 예에서는 로그인한 관리자가 HSM에 새 사용자를 생성합니다.

```
aws-cloudhsm > user create --username user1 --role crypto-user --approval /path/
admin.token
Enter password:
Confirm password:
{
  "error_code": 0,
```

```
  "data": {
    "username": "user1",
    "role": "crypto-user"
  }
}
```

그후에 관리자는 user list 명령을 입력하여 새 사용자 생성을 확인합니다.

```
aws-cloudhsm > user list{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin2",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin3",
        "role": "admin",
        "locked": "false",
        "mfa": [],
```

```
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "admin4",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "user1",
        "role": "crypto-user",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

관리자가 다른 HSM 사용자 관리 작업을 수행하려고 하면 쿼럼 인증 오류로 인해 실패합니다.

```
aws-cloudhsm > user delete --username user1 --role crypto-user
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}
```

아래와 같이 quorum token-sign list 명령을 실행하면 관리자에게 승인된 토큰이 없는 것으로 표시됩니다. 다른 HSM 사용자 관리 작업을 수행하려면 사용자 관리 작업을 실행하는 동안 승인되고 소비될 쿼럼 토큰을 제공하기 위해 관리자가 새 쿼럼 토큰을 생성하고 승인자로부터 새 서명을 받은 다음 --approval 인수와 함께 원하는 사용자 관리 작업을 실행해야 합니다.

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
  "data": {
    "tokens": []
  }
}
```

관리자의 쿼럼 최소값 변경

관리자가 쿼럼 인증을 사용할 수 있도록 쿼럼 최소값을 설정한 후 쿼럼 최소값을 변경할 수 있습니다. HSM에서는 승인자의 수가 현재 쿼럼 최소값 이상인 경우에만 쿼럼 최소값을 변경할 수 있습니다. 예를 들어 쿼럼 최소값이 2라면 적어도 2명의 관리자가 승인해야 쿼럼 최소값을 변경할 수 있습니다.

> **Note**
>
> 사용자 서비스의 쿼럼 값은 항상 쿼럼 서비스의 쿼럼 값보다 작아야 합니다. 쿼럼 서비스 및 사용자 서비스와 같은 서비스 이름에 대한 자세한 내용은 쿼럼 인증을 지원하는 서비스 이름 및 유형 섹션을 참조하십시오.

쿼럼 최소값을 변경하기 위한 쿼럼 승인을 받으려면 quorum token-sign set-quorum-value 명령을 사용할 quorum service에 대한 쿼럼 토큰이 필요합니다. quorum token-sign set-quorum-value 명령을 사용할 quorum service에 대한 쿼럼 토큰을 생성하려면 쿼럼 서비스가 1보다 높아야 합니다. 즉, 사용자 서비스의 쿼럼 최소값을 변경하려면 먼저 쿼럼 서비스의 쿼럼 최소값을 변경해야 할 수도 있습니다.

관리자의 쿼럼 최소값을 변경하려면

1. 다음 명령을 사용하여 CloudHSM CLI 대화형 모드를 시작합니다.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
```

2. login 명령을 사용하고 클러스터에 관리자로 로그인합니다.

```
aws-cloudhsm>login --username <admin> --role admin
```

3. quorum token-sign list-quorum-values 명령을 사용하여 모든 서비스 이름에 대한 쿼럼 최소값을 얻습니다. 자세한 내용은 아래 예제를 참조하십시오.
4. 쿼럼 서비스의 쿼럼 최소값이 사용자 서비스의 값보다 낮은 경우 quorum token-sign set-quorum-value 명령을 사용하여 쿼럼 서비스의 값을 변경합니다. 쿼럼 서비스 값을 사용자 서비스 값보다 높거나 같은 1로 변경합니다. 자세한 내용은 다음 예제를 참조하세요.
5. 쿼럼 토큰을 생성합니다. 토큰을 사용할 수 있는 서비스로 쿼럼 서비스를 지정하도록 주의하십시오.
6. 다른 관리자의 승인(서명)을 받습니다.
7. AWS CloudHSM 클러스터에서 토큰을 승인하고 사용자 관리 작업을 실행합니다. .
8. quorum token-sign set-quorum-value 명령을 사용하여 사용자 서비스의 쿼럼 최소값을 변경합니다.

Example – 쿼럼 최소값 가져오기 및 쿼럼 서비스 값 변경

다음 예제 명령은 사용자 서비스의 쿼럼 최소값이 현재 2임을 보여줍니다.

```
aws-cloudhsm > quorum token-sign list-quorum-values{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 1
  }
}
```

쿼럼 서비스의 쿼럼 최소값을 변경하려면 quorum token-sign set-quorum-value 명령을 사용하여 사용자 서비스 값보다 높거나 같은 값을 설정합니다. 다음 예에서는 쿼럼 서비스의 쿼럼 최소값을 사용자 서비스에 설정된 값과 동일한 2로 설정합니다.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2{
  "error_code": 0,
  "data": "Set quorum value successful"
}
```

다음 명령은 이제 사용자 서비스 및 쿼럼 서비스에 대한 쿼럼 최소값이 2임을 보여줍니다.

```
aws-cloudhsm > quorum token-sign list-quorum-values{
  "error_code": 0,
  "data": {
    "user": 2,
    "quorum": 2
  }
}
```

CloudHSM 관리 유틸리티(CMU)를 통한 HSM 사용자 관리

AWS CloudHSM에서는 CloudHSM CLI 또는 CloudHSM 관리 유틸리티 (CMU) 명령줄 도구를 사용하여 HSM에서 사용자를 생성하고 관리해야 합니다. CloudHSM CLI는 최신 SDK와 함께 사용하도록 설계되었으며 CMU는 이전 SDK와 함께 사용하도록 설계되었습니다.

주제

- HSM 사용자 이해
- HSM 사용자 권한 테이블
- CloudHSM 관리 유틸리티(CMU)를 사용하여 사용자 관리
- CloudHSM 관리 유틸리티(CMU)를 사용하여 CO(Crypto Officer)를 위한 2팩터 인증(2FA) 관리
- CloudHSM 관리 유틸리티(CMU)를 사용하여 쿼럼 인증 관리(M/N 액세스 제어)

HSM 사용자 이해

HSM에서 수행하는 대부분의 작업에는 HSM 사용자의 보안 인증이 필요합니다. HSM은 각 HSM 사용자를 인증하며, 각 HSM 사용자에게는 해당 사용자로서 HSM에서 수행할 수 있는 작업을 결정하는 유형이 있습니다.

> **Note**
>
> HSM 사용자는 IAM 사용자와 다릅니다. 올바른 자격 증명을 보유한 IAM 사용자는 AWS API를 통해 리소스와 상호 작용하여 HSM을 생성할 수 있습니다. HSM을 생성한 후에는 HSM 사용자 자격 증명을 사용하여 HSM에서의 작업을 인증해야 합니다.

사용자 유형

- PRESCO(Precrypto Officer)
- CO(Crypto Officer)
- CU(Crypto User)
- AU(Appliance User)

PRESCO(Precrypto Officer)

클라우드 관리 유틸리티 (CMU) 와 키 관리 유틸리티 (KMU) 모두에서 PRECO는 AWS CloudHSM 클러스터의 첫 번째 HSM에만 존재하는 임시 사용자입니다. 새 클러스터의 첫 번째 HSM에는 이 클러스터가 활성화된 적이 없음을 나타내는 PRECO 사용자가 포함되어 있습니다. 클러스터를 활성화하려면 cloudhsm-cli를 실행하고 cluster activate 명령을 실행합니다. HSM에 로그인하고 PRECO 암호를 변경합니다. 암호를 변경하면 이 사용자가 CO(Crypto Officer)가 됩니다.

CO(Crypto Officer)

클라우드 관리 유틸리티(CMU)와 키 관리 유틸리티(KMU) 모두에서 CO(Crypto Officer)는 사용자 관리 작업을 수행할 수 있습니다. 예를 들어, 사용자를 생성 및 삭제하고 사용자 암호를 변경할 수 있습니다. CO 사용자에 대한 자세한 내용은 HSM 사용자 권한 테이블 단원을 참조하십시오. 새 클러스터를 활성화하면 사용자가 PRECO(Precrypto Officer)에서 CO(Crypto Officer)로 바뀝니다.

CU(Crypto User)

CU(Crypto User)는 다음 키 관리 및 암호화 작업을 수행할 수 있습니다.

- 키 관리 – 암호화 키 생성, 삭제, 공유, 가져오기 및 내보내기
- 암호화 작업 – 암호화, 암호화 해제, 사인, 확인 등에 암호호화 키 사용

자세한 내용은 HSM 사용자 권한 테이블 단원을 참조하십시오.

AU(Appliance User)

어플라이언스 사용자 (AU) 는 클러스터의 HSM에서 복제 및 동기화 작업을 수행할 수 있습니다. AWS CloudHSM AU를 사용하여 클러스터의 HSM을 동기화합니다. AWS CloudHSM AU는 에서 제공하는 모든 HSM에 AWS CloudHSM존재하며 권한이 제한되어 있습니다. 자세한 내용은 HSM 사용자 권한 테이블을 참조하세요.

AWS HSM에서는 어떤 작업도 수행할 수 없습니다. AWS 사용자 또는 키를 보거나 수정할 수 없으며 해당 키를 사용하여 암호화 작업을 수행할 수 없습니다.

HSM 사용자 권한 테이블

다음 테이블에는 작업을 수행할 수 있는 HSM 사용자 또는 세션 유형별로 정렬된 HSM 작업이 나열되어 있습니다.

	CO(Crypto Officer)	CU(Crypto User)	AU(Appliance User)	인증된 세션
기본 클러스터 정보 가져오기	 예	 예	 예	 예
자체 암호 변경	 예	 예	 예	해당 사항 없음
사용자의 암호 변경	 예	 아니요	 아니요	 아니요
사용자 추가 및 제거	 예	 아니요	 아니요	 아니요

	CO(Crypto Officer)	CU(Crypto User)	AU(Appliance User)	인증된 세션
동기화 상태 가져오기	 예	 예	 예	 아니요
마스킹 처리된 객체 추출 및 삽입[3]	 예	 예	 예	 아니요
키 관리 기능[4]	 아니요	 예	 아니요	 아니요
암호화, 암호 해독	 아니요	 예	 아니요	 아니요
사인 및 확인	 아니요	 예	 아니요	 아니요
다이제스트 및 HMAC 생성	 아니요	 예	 아니요	 아니요

- [1] 기본 클러스터 정보에는 클러스터의 HSM 수와 각 HSM의 IP 주소, 모델, 일련 번호, 디바이스 ID, 펌웨어 ID 등이 포함됩니다.

- [2] 사용자는 HSM의 키에 해당하는 다이제스트(해시) 집합을 얻을 수 있습니다. 애플리케이션은 이러한 다이제스트 세트를 비교해서 클러스터에서 HSM의 동기화 상태를 파악할 수 있습니다.
- [3] 마스킹 처리된 객체는 HSM을 떠나기 전에 암호화가 되는 키입니다. HSM 밖에서는 암호를 해독할 수 없습니다. 키가 추출된 HSM과 같은 클러스터에 있는 HSM에 삽입된 후에만 암호가 해독됩니다. 애플리케이션은 마스킹 처리된 객체를 추출 및 삽입하여 클러스터에서 HSM을 동기화할 수 있습니다.
- [4] 키 관리 함수에는 키 속성 생성, 삭제, 래핑, 언래핑, 수정이 포함됩니다.

CloudHSM 관리 유틸리티(CMU)를 사용하여 사용자 관리

이 항목에서는 클라이언트 step-by-step SDK와 함께 제공되는 명령줄 도구인 CloudHSM 관리 유틸리티 (CMU) 를 사용하여 하드웨어 보안 모듈 (HSM) 사용자를 관리하는 방법에 대한 지침을 제공합니다. CMU 또는 HSM 사용자에 대한 자세한 정보는 CloudHSM 관리 유틸리티 및 HSM 사용자 이해 단원을 참조하십시오.

Sections

- CMU를 사용한 HSM 사용자 관리에 대한 이해
- CloudHSM 관리 유틸리티 다운로드
- CMU로 HSM 사용자를 관리하는 방법

CMU를 사용한 HSM 사용자 관리에 대한 이해

HSM 사용자를 관리하려면 CO(Cryptographic Officer)의 사용자 이름과 암호로 HSM에 로그인해야 합니다. CO만 사용자를 관리할 수 있습니다. HSM에는 admin이라는 기본 CO가 포함됩니다. 클러스터를 활성화할 때 admin에 대해 사용할 암호를 설정했습니다.

CMU를 사용하려면 구성 도구를 사용하여 로컬 구성을 업데이트해야 합니다. CMU는 클러스터에 대한 자체 연결을 생성하며 이 연결은 클러스터를 인식하지 않습니다. 클러스터 정보를 추적하기 위해 CMU는 로컬 구성 파일을 유지 관리합니다. 즉, CMU를 사용할 때마다 먼저 구성 명령줄 도구를 `--cmu` 파라미터와 함께 실행하여 구성 파일을 업데이트해야 합니다. Client SDK 3.2.1 이전 버전을 사용하는 경우 `--cmu`와 다른 파라미터를 사용해야 합니다. 자세한 내용은 the section called “CMU를 Client SDK 3.2.1 이하와 함께 사용” 단원을 참조하십시오.

`--cmu` 파라미터를 사용하려면 클러스터에 HSM의 IP 주소를 추가해야 합니다. HSM이 여러 개 있는 경우 모든 IP 주소를 사용할 수 있습니다. 이렇게 하면 CMU가 전체 클러스터에 변경 내용을 전파할 수 있습니다. CMU는 로컬 파일을 사용하여 클러스터 정보를 추적한다는 점을 기억하십시오. 특정 호스

트의 CMU를 마지막으로 사용한 이후 클러스터가 변경된 경우 해당 호스트에 저장된 로컬 구성 파일에 해당 변경 사항을 추가해야 합니다. CMU를 사용하는 동안에는 HSM을 추가하거나 제거하지 마십시오.

HSM의 IP 주소를 가져오려면 (콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. AWS 리전을 변경하려면 페이지의 오른쪽 상단에 있는 리전 선택기를 사용합니다.
3. 클러스터 세부 정보 페이지를 열려면 클러스터 테이블에서 클러스터 ID를 선택합니다.
4. IP 주소를 가져오려면 HSM 탭에서 ENI IP 주소 아래에 나열된 IP 주소 중 하나를 선택합니다.

HSM의 IP 주소를 가져오려면 ()AWS CLI

- AWS CLI의 describe-clusters 명령을 사용하여 HSM의 IP 주소를 가져옵니다. 명령의 출력에서 HSM의 IP 주소는 `EniIp`의 값입니다.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

CMU를 Client SDK 3.2.1 이하와 함께 사용

Client SDK 3.3.0에서는 `--cmu` 파라미터에 대한 지원이 AWS CloudHSM 추가되어 CMU의 구성 파일 업데이트 프로세스가 간소화되었습니다. Client SDK 3.2.1 이전 버전의 CMU를 사용하는 경우, `-a` 및 `-m` 파라미터를 계속 사용하여 구성 파일을 업데이트해야 합니다. 이러한 파라미터에 대한 자세한 내용은 도구 구성을 참조하십시오.

CloudHSM 관리 유틸리티 다운로드

Client SDK 5와 Client SDK 3을 사용하든 관계없이 HSM 사용자 관리 작업에 최신 버전의 CMU를 사용할 수 있습니다.

CMU 다운로드 및 설치

- CMU를 다운로드하여 설치합니다.

  Amazon Linux

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-mgmt-util-latest.el6.x86_64.rpm
  ```

  ```
  $ sudo yum install ./cloudhsm-mgmt-util-latest.el6.x86_64.rpm
  ```

  Amazon Linux 2

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  ```
  $ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  CentOS 7.8+

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  ```
  $ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
  ```

  CentOS 8.3+

  ```
  $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-mgmt-util-latest.el8.x86_64.rpm
  ```

  ```
  $ sudo yum install ./cloudhsm-mgmt-util-latest.el8.x86_64.rpm
  ```

RHEL 7 (7.8+)

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-mgmt-util-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-mgmt-util-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-mgmt-util-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-mgmt-util_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-mgmt-util_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-mgmt-util_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-mgmt-util_latest_u18.04_amd64.deb
```

Windows Server 2012

1. CloudHSM 관리 유틸리티를 다운로드합니다.
2. Windows 관리 권한으로 CMU 설치 프로그램 (AWSCloudHSMManagementUtil-latest.msi)을 실행합니다.

Windows Server 2012 R2

1. CloudHSM 관리 유틸리티를 다운로드합니다.
2. Windows 관리자 권한으로 CMU 설치 프로그램 (AWSCloudHSMManagementUtil-latest.msi) 을 실행합니다.

Windows Server 2016

1. CloudHSM 관리 유틸리티를 다운로드합니다.
2. Windows 관리자 권한으로 CMU 설치 프로그램 (AWSCloudHSMManagementUtil-latest.msi) 을 실행합니다.

CMU로 HSM 사용자를 관리하는 방법

이 섹션에는 CMU로 HSM 사용자를 관리하는 기본 명령이 포함되어 있습니다.

HSM 사용자 생성

createUser를 사용하여 HSM에서 새 사용자를 생성합니다. 사용자를 생성하려면 CO로 로그인해야 합니다.

새로운 CO 사용자 생성

1. 구성 도구를 사용하여 CMU 구성을 업데이트합니다.

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. CMU를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

Windows

```
C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. HSM에 CO 사용자로 로그인합니다.

```
aws-cloudhsm>loginHSM CO admin co12345
```

CMU 목록의 연결 수가 클러스터의 HSM 수와 일치하는지 확인합니다. 그렇지 않은 경우 로그아웃하고 다시 시작합니다.

4. createUser를 사용하여 암호가 **password1**인 **example_officer**라는 이름의 CO 사용자를 생성합니다.

```
aws-cloudhsm>createUser CO example_officer password1
```

CMU는 사용자 생성 작업에 대한 메시지를 표시합니다.

```
*************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?
```

5. **y**를 입력합니다.

새로운 사용자 생성

1. 구성 도구를 사용하여 CMU 구성을 업데이트합니다.

Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
```

2. CMU를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
   \CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. HSM에 CO 사용자로 로그인합니다.

   ```
   aws-cloudhsm>loginHSM CO admin co12345
   ```

   CMU 목록의 연결 수가 클러스터의 HSM 수와 일치하는지 확인합니다. 그렇지 않은 경우 로그아웃하고 다시 시작합니다.

4. createUser를 사용하여 암호가 **password1**인 **example_user**라는 이름의 CU 사용자를 생성합니다.

   ```
   aws-cloudhsm>createUser CU example_user password1
   ```

   CMU는 사용자 생성 작업에 대한 메시지를 표시합니다.

   ```
   
   ************************CAUTION*******************************
   This is a CRITICAL operation, should be done on all nodes in the
   cluster. AWS does NOT synchronize these changes automatically with the
   nodes on which this operation is not executed or failed, please
   ensure this operation is executed on all nodes in the cluster.
   **************************************************************
   
   Do you want to continue(y/n)?
   ```

5. **y**를 입력합니다.

createUser에 대한 자세한 정보는 createUser를 참조하십시오.

클러스터의 모든 HSM 사용자 나열

listUsers 명령을 사용하여 클러스터의 모든 사용자를 나열합니다. listUsers를 실행하기 위해 로그인할 필요가 없으며 모든 사용자 유형이 사용자를 나열할 수 있습니다.

클러스터의 모든 사용자 나열

1. 구성 도구를 사용하여 CMU 구성을 업데이트합니다.

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. CMU를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
   \CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. 클러스터의 모든 사용자를 나열하는 데 listUsers을 사용합니다.

   ```
   aws-cloudhsm>listUsers
   ```

   CMU는 클러스터의 모든 사용자를 나열합니다.

   ```
   Users on server 0(10.0.2.9):
   Number of users found:4
   ```

```
    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              AU              app_user                                NO
               0              NO
         2              CO              example_officer                         NO
               0              NO
         3              CU              example_user                            NO
               0              NO
Users on server 1(10.0.3.11):
Number of users found:4

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              AU              app_user                                NO
               0              NO
         2              CO              example_officer                         NO
               0              NO
         3              CU              example_user                            NO
               0              NO
Users on server 2(10.0.1.12):
Number of users found:4

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              AU              app_user                                NO
               0              NO
         2              CO              example_officer                         NO
               0              NO
         3              CU              example_user                            NO
               0              NO
```

listUsers에 대한 자세한 내용은 listUsers를 참조하십시오.

HSM 사용자 암호 변경

changePswd를 사용하여 암호를 변경합니다.

사용자 유형 및 암호는 대소문자를 구분하지만 사용자 이름은 대소문자를 구분하지 않습니다.

CU(Crypto User) 및 AU(Appliance User)는 자기 암호만 변경할 수 있습니다. 다른 사용자의 암호를 변경하려면 CO로 로그인해야 합니다. 현재 클라이언트 또는 에 로그인되어 있는 사용자의 암호를 변경할 수 없습니다.

암호 변경

1. 구성 도구를 사용하여 CMU 구성을 업데이트합니다.

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. CMU를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
   \CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. HSM에 로그인합니다.

   ```
   aws-cloudhsm>loginHSM CO admin co12345
   ```

   CMU 목록의 연결 수가 클러스터의 HSM 수와 일치하는지 확인합니다. 그렇지 않은 경우 로그아웃하고 다시 시작합니다.

4. changePswd를 사용하여 자신의 암호를 변경합니다.

   ```
   aws-cloudhsm>changePswd CO example_officer <new password>
   ```

   CMU는 암호 변경 작업에 대한 메시지를 표시합니다.

   ```
   
   ************************CAUTION*********************************
   This is a CRITICAL operation, should be done on all nodes in the
   cluster. AWS does NOT synchronize these changes automatically with the
   nodes on which this operation is not executed or failed, please
   ```

```
ensure this operation is executed on all nodes in the cluster.
*************************************************************

Do you want to continue(y/n)?
```

5. **y**를 입력합니다.

   CMU는 암호 변경 작업에 대한 메시지를 표시합니다.

```
Changing password for example_officer(CO) on 3 nodes
```

다른 사용자의 암호 변경

1. 구성 도구를 사용하여 CMU 구성을 업데이트합니다.

   Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

   Windows

```
C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
```

2. CMU를 시작합니다.

   Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

   Windows

```
C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

3. HSM에 CO 사용자로 로그인합니다.

```
aws-cloudhsm>loginHSM CO admin co12345
```

CMU 목록의 연결 수가 클러스터의 HSM 수와 일치하는지 확인합니다. 그렇지 않은 경우 로그아웃하고 다시 시작합니다.

4. changePswd를 사용하여 다른 사용자의 암호를 변경합니다.

```
aws-cloudhsm>changePswd CU example_user <new password>
```

CMU는 암호 변경 작업에 대한 메시지를 표시합니다.

```
**********************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**************************************************************

Do you want to continue(y/n)?
```

5. **y**를 입력합니다.

CMU는 암호 변경 작업에 대한 메시지를 표시합니다.

```
Changing password for example_user(CU) on 3 nodes
```

changePswd에 대한 자세한 내용은 changePswd를 참조하십시오.

HSM 사용자 삭제

deleteUser를 사용하여 사용자를 삭제합니다. 다른 사용자를 삭제하려면 CO로 로그인해야 합니다.

> **Tip**
>
> 키를 소유한 CU(Crypto User)는 삭제할 수 없습니다.

사용자 삭제

1. 구성 도구를 사용하여 CMU 구성을 업데이트합니다.

Linux

```
$ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
```

2. CMU를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
   \CloudHSM\data\cloudhsm_mgmt_util.cfg
   ```

3. HSM에 CO 사용자로 로그인합니다.

   ```
   aws-cloudhsm>loginHSM CO admin co12345
   ```

   CMU 목록의 연결 수가 클러스터의 HSM 수와 일치하는지 확인합니다. 그렇지 않은 경우 로그아웃하고 다시 시작합니다.

4. deleteUser를 사용하여 사용자를 삭제합니다.

   ```
   aws-cloudhsm>deleteUser CO example_officer
   ```

   CMU는 사용자를 삭제합니다.

   ```
   Deleting user example_officer(CO) on 3 nodes
   deleteUser success on server 0(10.0.2.9)
   deleteUser success on server 1(10.0.3.11)
   deleteUser success on server 2(10.0.1.12)
   ```

deleteUser에 대한 자세한 내용은 deleteUser를 참조하십시오.

CloudHSM 관리 유틸리티(CMU)를 사용하여 CO(Crypto Officer)를 위한 2팩터 인증(2FA) 관리

보안을 강화하기 위해, 2팩터 인증(2FA)을 구성하여 클러스터를 보호할 수 있습니다. CO(Crypto Officer)에 대해서만 2FA를 활성화할 수 있습니다.

> **Note**
>
> CU(Crypto User) 또는 애플리케이션에는 2FA를 활성화할 수 없습니다. 2팩터 인증(2FA)은 CO 사용자만 사용할 수 있습니다.

주제

HSM 사용자를 위한 2FA 이해

2FA 지원 하드웨어 서비스 모듈(HSM) 계정으로 클러스터에 로그인할 때 첫 번째 요소인 cloudhsm_mgmt_util(CMU)에 암호를 입력하면 CMU는 토큰을 제공하고 토큰에 서명하라는 메시지를 표시합니다. 가지고 있는 두 번째 요소를 제공하려면 이미 생성하여 HSM 사용자와 연결한 키 페어의 프라이빗 키로 토큰에 서명합니다. 클러스터에 액세스하려면 서명된 토큰을 CMU에 제공합니다.

쿼럼 인증 및 2FA

클러스터는 쿼럼 인증과 2FA에 동일한 키를 사용합니다. 즉, 2FA가 활성화된 사용자가 M-of-N 액세스 제어(MoFn)에 효과적으로 등록되었다는 것을 의미합니다. 동일한 HSM 사용자에 대해 2FA 및 쿼럼 인증을 성공적으로 사용하려면 다음 사항을 고려해야 합니다.

- 현재 사용자에 대해 쿼럼 인증을 사용하는 경우 쿼럼 사용자에 대해 생성한 것과 동일한 키 페어를 사용하여 사용자에 대해 2FA를 활성화해야 합니다.
- 쿼럼 인증 사용자가 아닌 비 2FA 사용자에 대한 2FA 요구 사항을 추가하면 2FA 인증을 통해 해당 사용자를 MofN 사용자로 등록합니다.
- 2FA 요구 사항을 제거하거나 쿼럼 인증 사용자이기도 한 2FA 사용자의 암호를 변경하면 해당 쿼럼 사용자의 MoFN 사용자 등록도 제거됩니다.

- 쿼럼 인증 사용자이기도 한 2FA 사용자의 2FA 요구 사항을 제거하거나 암호는 변경하지만 해당 사용자가 여전히 쿼럼 인증에 참여하도록 하려면 해당 사용자를 MoFN 사용자로 다시 등록해야 합니다.

쿼럼 인증에 대한 자세한 내용은 CMU를 사용하여 쿼럼 인증 관리 섹션을 참조하십시오.

HSM 사용자를 위한 2FA 사용

이 섹션에서는 2FA HSM 사용자 생성, 키 교체, 2FA 지원 사용자로 HSM에 로그인하기 등 HSM 사용자를 위한 2FA 사용 방법에 대해 설명합니다. HSM 사용자 작업에 대한 자세한 내용은 ???, ???, ???, ???, ??? 섹션을 참조하십시오.

2FA 사용자 생성

HSM 사용자에 대한 2FA를 활성화하려면 다음 요구 사항을 충족하는 키를 사용하십시오.

2FA 키 페어 요구 사항

새 키 페어를 만들거나 다음 요구 사항을 충족하는 기존 키를 사용할 수 있습니다.

- 키 유형: 비대칭
- 키 사용: 서명 및 확인
- 키 사양: RSA_2048
- 서명 알고리즘에는 다음이 포함됩니다.
  - `sha256WithRSAEncryption`

> **Note**
>
> 쿼럼 인증을 사용하거나 쿼럼 인증을 사용할 계획인 경우 the section called “쿼럼 인증 및 2FA” 섹션을 참조하십시오.

CMU와 키 페어를 사용하여 2FA가 활성화된 새 CO 사용자를 생성합니다.

2FA가 활성화된 CO 사용자를 생성하려면

1. 한 터미널에서 다음 단계를 수행합니다.

   a. HSM에 액세스하고 CloudHSM 관리 유틸리티에 로그인합니다.

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

b. CO로 로그인하고 다음 명령을 사용하여 2FA가 있는 새 사용자 MFA를 생성합니다.

```
aws-cloudhsm>createUser CO MFA <CO USER NAME> -2fa /home/ec2-user/authdata
**********CAUTION**********This is a
 CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
************************************************************

Do you want to continue(y/n)?

yCreating User exampleuser3(CO) on 1 nodesAuthentication data written to: "/
home/ec2-user/authdata"Generate Base64-encoded signatures for SHA256 digests in
 the authentication datafile.
To generate the signatures, use the RSA private key, which is the second factor
 ofauthentication for this user. Paste the signatures and the corresponding
 public keyinto the authentication data file and provide
the file path below.Leave this field blank to use the path initially
 provided.Enter filename:
```

c. 위 터미널을 이 상태로 둡니다. Enter 키를 누르거나 파일 이름을 입력하면 안 됩니다.

2. 다른 터미널에서 다음 단계를 수행합니다.

a. HSM에 액세스하고 CloudHSM 관리 유틸리티에 로그인합니다.

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

b. 다음 명령을 사용하여 퍼블릭-프라이빗 키 페어를 생성합니다.

```
openssl genpkey -algorithm RSA -out private_key.pem -pkeyopt
 rsa_keygen_bits:2048
```

```
openssl rsa -pubout -in private_key.pem -out public_key.pem
```

c. 다음 명령을 실행하여 authdata 파일에서 다이제스트를 추출하기 위한 json 쿼리 기능을 설치합니다.

```
sudo yum install jq
```

d. 다이제스트 값을 추출하려면 먼저 authdata 파일에서 다음 데이터를 찾습니다.

```
{
  "Version":"1.0",
  "PublicKey":"",
  "Data":[
    {
      "HsmId": <"HSM ID">,
      "Digest": <"DIGEST">,
      "Signature":""
    }
  ]
}
```

> **Note**
>
> 획득한 다이제스트는 base64로 인코딩되지만 다이제스트에 서명하려면 먼저 파일을 디코딩한 다음 서명해야 합니다. 다음 명령은 다이제스트를 디코딩하고 디코딩된 콘텐츠를 'digest1.bin'에 저장합니다.
>
> ```
> cat authdata | jq '.Data[0].Digest' | cut -c2- | rev | cut -c2- | rev |
>  base64 -d > digest1.bin
> ```

e. 다음과 같이 퍼블릭 키 콘텐츠를 변환하고 "\n"을 추가하며 공백을 제거합니다.

```
-----BEGIN PUBLIC KEY-----\n<PUBLIC KEY>\n-----END PUBLIC KEY-----
```

> **Important**
>
> 위 명령은 BEGIN PUBLIC KEY----- 바로 뒤에 "\n"이 추가되고, "\n"과 퍼블릭 키의 첫 번째 문자 사이의 공백이 제거되며, "\n"이 -----END PUBLIC KEY 앞에 추가되고, "\n"과 퍼블릭 키의 끝 사이의 공백이 제거되는 방식을 보여줍니다.

이는 authdata 파일에 허용되는 퍼블릭 키의 PEM 형식입니다.

f. authdata 파일의 퍼블릭 키 섹션에 퍼블릭 키 pem 형식 콘텐츠를 붙여 넣습니다.

```
vi authdata
```

```
{
  "Version":"1.0",
  "PublicKey":"-----BEGIN PUBLIC KEY-----\n<"PUBLIC KEY">\n-----END PUBLIC
 KEY-----",
  "Data":[
{
  "HsmId":<"HSM ID">,
  "Digest":<"DIGEST">,
  "Signature":””
}
]
}
```

g. 다음 명령을 사용하여 토큰 파일에 서명합니다.

```
openssl pkeyutl -sign -in digest1.bin -inkey private_key.pem -pkeyopt
 digest:sha256 | base64

Output Expected:
<"THE SIGNATURE">
```

> **Note**
>
> 위 명령에서 볼 수 있듯이, openssl dgst 서명 대신 openssl pkeyutl을 사용합니다.

h. Authdata 파일의 “서명” 필드에 서명된 다이제스트를 추가합니다.

```
vi authdata
```

```
{
    "Version": "1.0",
    "PublicKey": "-----BEGIN PUBLIC KEY----- ... -----END PUBLIC KEY-----",
    "Data": [
        {
            "HsmId": <"HSM ID">,
            "Digest": <"DIGEST">,
```

```
            "Signature": "Kkdl ... rkrvJ6Q=="
        },
        {
            "HsmId": <"HSM ID">,
            "Digest": <"DIGEST">,
            "Signature": "K1hxy ... Q261Q=="
        }
    ]
}
```

3. 첫 번째 터미널로 돌아가서 **Enter**을 누릅니다.

```
Generate Base64-encoded signatures for SHA256 digests in the authentication
 datafile. To generate the signatures, use the RSA private key,
which is the second factor ofauthentication for this user. Paste the signatures and
 the corresponding public keyinto the authentication data file and provide the file
 path below. Leave this field blank to use the path initially provided.
Enter filename: >>>>> Press Enter here

createUser success on server 0(10.0.1.11)
```

HSM 사용자의 2FA 관리

암호 변경을 사용하여 2FA 사용자의 암호를 변경하거나, 2FA를 활성화 또는 비활성화하거나, 2FA 키를 교체합니다. 2FA를 활성화할 때마다 2FA 로그인을 위한 퍼블릭 키를 제공해야 합니다.

암호 변경은 다음 시나리오 중 하나를 수행합니다.

- 2FA 사용자의 암호 변경
- 2FA가 아닌 사용자의 암호 변경
- 2FA가 아닌 사용자에게 2FA 추가
- 2FA 사용자의 2FA 삭제
- 2FA 사용자의 키 교체

작업을 결합할 수도 있습니다. 예를 들어, 사용자로부터 2FA를 제거하고 동시에 암호를 변경하거나 2FA 키를 교체하여 사용자 암호를 변경할 수도 있습니다.

2FA가 활성화된 CO 사용자의 암호를 변경하거나 키를 교체하려면

1. CMU를 사용하여 2FA가 활성화된 상태에서 CO로 HSM에 로그인합니다.

2. changePswd을 사용하여 2FA가 활성화된 CO 사용자의 암호를 변경하거나 키를 교체합니다. `-2fa` 파라미터를 사용하고 시스템이 `authdata` 파일을 쓸 수 있는 위치를 파일 시스템에 포함시킵니다. 이 파일에는 클러스터의 각 HSM에 대한 다이제스트가 포함됩니다.

```
aws-cloudhsm>changePswd CO example-user <new-password> -2fa /path/to/authdata
```

CMU는 프라이빗 키를 사용하여 `authdata` 파일의 다이제스트에 서명하고 퍼블릭 키와 함께 서명을 반환하라는 메시지를 표시합니다.

3. 프라이빗 키를 사용하여 `authdata` 파일의 다이제스트에 서명하고 서명과 퍼블릭 키를 JSON 형식의 `authdata` 파일에 추가한 다음 CMU에 `authdata` 파일 위치를 제공하십시오. 자세한 내용은 the section called “구성 참조” 섹션을 참조하십시오.

> **Note**
>
> 클러스터는 쿼럼 인증 및 2FA에 동일한 키를 사용합니다. 쿼럼 인증을 사용하거나 쿼럼 인증을 사용할 계획인 경우 the section called “쿼럼 인증 및 2FA” 섹션을 참조하십시오.

2FA가 활성화된 CO 사용자에 대해 2FA를 비활성화하려면

1. CMU를 사용하여 2FA가 활성화된 상태에서 CO로 HSM에 로그인합니다.
2. changePswd을 사용하여 2FA가 활성화된 CO 사용자로부터 2FA를 제거합니다.

```
aws-cloudhsm>changePswd CO example-user <new password>
```

CMU는 암호 변경 작업을 확인하라는 메시지를 표시합니다.

> **Note**
>
> 2FA 요구 사항을 제거하거나 쿼럼 인증 사용자이기도 한 2FA 사용자의 암호를 변경하면 해당 쿼럼 사용자의 MoFN 사용자 등록도 제거됩니다. 쿼럼 사용자 및 2FA에 대한 자세한 내용은 the section called “쿼럼 인증 및 2FA” 섹션을 참조하십시오.

3. **y**를 입력합니다.

CMU는 암호 변경 작업을 확인합니다.

구성 참조

다음은 CMU 생성 요청 및 응답 모두에 대한 authdata 파일의 2FA 속성의 예입니다.

```
{
    "Version": "1.0",
    "PublicKey": "-----BEGIN PUBLIC KEY----- ... -----END PUBLIC KEY-----",
    "Data": [
        {
            "HsmId": "hsm-lgavqitns2a",
            "Digest": "k5O1p3f6foQRVQH7S8Rrjcau6h3TYqsSdr16A54+qG8=",
            "Signature": "Kkdl ... rkrvJ6Q=="
        },
        {
            "HsmId": "hsm-lgavqitns2a",
            "Digest": "IyBcx4I5Vyx1jztwvXinCBQd9lDx8oQe7iRrWjBAi1w=",
            "Signature": "K1hxy ... Q261Q=="
        }
    ]
}
```

데이터

최상위 노드 클러스터의 각 HSM에 대한 하위 노드를 포함합니다. 모든 2FA 명령에 대한 요청 및 응답에 나타납니다.

다이제스트

두 번째 인증 요소를 제공하기 위해 서명해야 하는 사항입니다. CMU는 모든 2FA 명령에 대한 요청에서 생성됩니다.

HsmId

HSM의 ID입니다. 모든 2FA 명령에 대한 요청 및 응답에 나타납니다.

PublicKey

생성한 키 페어의 퍼블릭 키 부분이 PEM 형식의 문자열로 삽입되었습니다. createUser 및 changePswd에 대한 응답에 이를 입력합니다.

서명

Base 64로 인코딩된 서명된 다이제스트입니다. 모든 2FA 명령에 대한 응답에 이를 입력합니다.

버전

인증 데이터 JSON 형식 파일의 버전입니다. 모든 2FA 명령에 대한 요청 및 응답에 나타납니다.

CloudHSM 관리 유틸리티(CMU)를 사용하여 쿼럼 인증 관리(M/N 액세스 제어)

AWS CloudHSM 클러스터의 HSM은 쿼럼 인증 (M of N 액세스 제어라고도 함) 을 지원합니다. 쿼럼 인증의 경우, HSM의 단일 사용자는 HSM에서 쿼럼 제어 작업을 수행할 수 없습니다. 이러한 작업을 하려면 최소 숫자의 HSM 사용자(최소 2명)가 협력해야 합니다. 쿼럼 인증의 경우, 2명 이상의 HSM 사용자의 승인을 요구함으로써 보호 계층을 추가할 수 있습니다.

쿼럼 인증은 다음 작업을 제어할 수 있습니다.

- CO(Crypto Officer)의 HSM 사용자 관리 – HSM 사용자 생성 및 삭제, 다른 HSM 사용자의 암호 변경. 자세한 정보는 CO(Crypto Officer)의 쿼럼 인증 사용을 참조하세요.

다음 주제에는 AWS CloudHSM의 쿼럼 인증에 대한 자세한 내용이 있습니다.

주제

- 쿼럼 인증 개요
- 쿼럼 인증에 대한 추가 세부 정보
- CO(Crypto Officer)에 대한 쿼럼 인증 사용: 최초 설정
- CO(Crypto Officer)의 쿼럼 인증 사용
- CO(Crypto Officer)의 쿼럼 최소값 변경

쿼럼 인증 개요

다음 단계는 쿼럼 인증 프로세스를 요약합니다. 구체적인 단계와 도구는 CO(Crypto Officer)의 쿼럼 인증 사용을 참조하십시오.

1. 각각의 HSM 사용자는 서명을 위한 비대칭 키를 생성합니다. 이 사용자들은 HSM 외부에서 이를 수행하며 키를 적절히 보호하기 위해 주의합니다.
2. 각 HSM 사용자는 HSM에 로그인하고 자신의 서명 키의 퍼블릭 부분(퍼블릭 키)을 HSM에 등록합니다.
3. HSM 사용자가 쿼럼 제어 작업을 수행하려는 경우, 각 사용자는 HSM에 로그인하고 쿼럼 토큰을 가져옵니다.

4. HSM 사용자는 한 명 이상의 다른 HSM 사용자에게 쿼럼 토큰을 제공하고 승인을 요청합니다.
5. 다른 HSM 사용자는 자신의 키를 사용하여 암호화된 방식으로 쿼럼 토큰에 서명함으로써 승인합니다. 이는 HSM 외부에서 이루어집니다.
6. HSM 사용자가 필요한 수의 승인을 받으면 동일한 사용자가 HSM에 로그인하여 쿼럼 토큰과 승인(서명)을 HSM에 제공합니다.
7. HSM은 각 서명자의 등록된 퍼블릭 키를 사용하여 서명을 확인합니다. 서명이 유효하면 HSM이 토큰을 승인합니다.
8. HSM 사용자는 이제 쿼럼 제어 작업을 할 수 있습니다.

쿼럼 인증에 대한 추가 세부 정보

AWS CloudHSM의 쿼럼 인증 사용에 대해 다음 내용을 추가로 참고하십시오.

- HSM 사용자는 자신의 쿼럼 토큰에 서명할 수 있습니다. 즉, 요청한 사용자는 쿼럼 인증에 필요한 승인 중 하나를 제공할 수 있습니다.
- 쿼럼 제어 작업을 위한 최소 숫자의 쿼럼 승인자를 선택합니다. 선택할 수 있는 최소 숫자는 2이고, 선택할 수 있는 최대 수는 8입니다.
- HSM은 최대 1024개의 쿼럼 토큰을 저장할 수 있습니다. 새 토큰을 생성하려고 할 때 HSM에 이미 1024개의 토큰이 있는 경우, HSM은 만료된 토큰 중 하나를 삭제합니다. 기본적으로 토큰은 생성 10분 후에 만료됩니다.
- 클러스터는 쿼럼 인증과 2단계 인증(2FA)에 동일한 키를 사용합니다. 쿼럼 인증 및 2FA 사용에 대한 자세한 내용은 쿼럼 인증 및 2FA를 참조하십시오.

CO(Crypto Officer)에 대한 쿼럼 인증 사용: 최초 설정

다음 주제에서는 CO(Crypto Officer)가 쿼럼 인증을 사용할 수 있도록 하드웨어 보안 모듈(HSM)을 구성하기 위해 완료해야 하는 단계를 설명합니다. 이러한 단계는 처음으로 CO 쿼럼 인증을 구성할 때 한 번만 수행하면 됩니다. 이 단계들을 완료한 후 CO(Crypto Officer)의 쿼럼 인증 사용 섹션을 참조하십시오.

주제

- 필수 조건
- 서명용 키 생성 및 등록
- HSM에서 쿼럼 최소값 설정

필수 조건

이 예제 코드를 이해하려면 cloudhsm_mgmt_util명령줄 도구(CMU)에 익숙해야 합니다. 이 예시에서는 명령의 다음 출력과 같이 AWS CloudHSM 클러스터에 각각 동일한 CoS를 가진 HSM이 두 개 있습니다. listUsers 사용자 생성에 대한 자세한 내용은 HSM 사용자 관리 단원을 참조하십시오.

```
aws-cloudhsm>listUsers
Users on server 0(10.0.2.14):
Number of users found:7

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                                NO
           0               NO
         2              AU              app_user                             NO
           0               NO
         3              CO              officer1                             NO
           0               NO
         4              CO              officer2                             NO
           0               NO
         5              CO              officer3                             NO
           0               NO
         6              CO              officer4                             NO
           0               NO
         7              CO              officer5                             NO
           0               NO
Users on server 1(10.0.1.4):
Number of users found:7

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                                NO
           0               NO
         2              AU              app_user                             NO
           0               NO
         3              CO              officer1                             NO
           0               NO
         4              CO              officer2                             NO
           0               NO
         5              CO              officer3                             NO
           0               NO
         6              CO              officer4                             NO
           0               NO
```

```
        7             CO             officer5                                NO
          0               NO
```

서명용 키 생성 및 등록

쿼럼 인증을 사용하려면 각 CO는 다음 단계를 모두 수행해야 합니다.

주제

- RSA 키 페어 생성
- 등록 토큰 생성 및 서명
- HSM에 퍼블릭 키 등록

RSA 키 페어 생성

키 쌍을 생성하고 보호하는 방법에는 여러 가지가 있습니다. 다음 예제에서는 OpenSSL을 사용한 작업 방법을 보여 줍니다.

Example — OpenSSL을 사용하여 개인 키를 생성하세요

다음 예제는 OpenSSL을 사용하여 암호로 보호되는 2048비트 RSA 키를 생성하는 방법을 보여 줍니다. 이 예제를 사용하려면 키를 저장할 파일의 이름으로 *officer1.key*를 대체하십시오.

```
$ openssl genrsa -out officer1.key -aes256 2048
Generating RSA private key, 2048 bit long modulus
.....................................+++
.+++
e is 65537 (0x10001)
Enter pass phrase for officer1.key:
Verifying - Enter pass phrase for officer1.key:
```

다음으로 방금 생성한 프라이빗 키를 사용하여 퍼블릭 키를 생성합니다.

Example — OpenSSL로 퍼블릭 키 생성하기

다음 예에서는 OpenSSL을 사용하여 방금 생성한 프라이빗 키에서 퍼블릭 키를 생성하는 방법을 보여 줍니다.

```
$ openssl rsa -in officer1.key -outform PEM -pubout -out officer1.pub
Enter pass phrase for officer1.key:
```

```
writing RSA key
```

등록 토큰 생성 및 서명

토큰을 생성하고 이전 단계에서 방금 생성한 프라이빗 키로 서명합니다.

Example – 토큰 생성

등록 토큰은 최대 크기인 245바이트를 초과하지 않는 임의의 데이터가 포함된 파일입니다. 프라이빗 키로 토큰에 서명하여 프라이빗 키에 액세스할 수 있음을 입증합니다. 다음 명령은 에코를 사용하여 문자열을 파일로 리디렉션합니다.

```
$ echo "token to be signed" > officer1.token
```

토큰에 서명하고 서명 파일에 저장합니다. CO를 HSM에 MofN 사용자로 등록하려면 서명된 토큰, 서명되지 않은 토큰, 퍼블릭 키가 필요합니다.

Example – 토큰에 서명하기

OpenSSL과 프라이빗 키를 사용하여 등록 토큰에 서명하고 서명 파일을 생성합니다.

```
$ openssl dgst -sha256 \
    -sign officer1.key \
    -out officer1.token.sig officer1.token
```

HSM에 퍼블릭 키 등록

키를 생성한 후 CO는 키의 퍼블릭 부분(퍼블릭 키)을 HSM에 등록해야 합니다.

퍼블릭 키를 HSM에 등록하려면

1. 다음 명령을 사용하여 cloudhsm_mgmt_util 명령줄 도구를 시작합니다.

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

2. loginHSM 명령을 사용하여 HSM에 CO로 로그인합니다. 자세한 내용은 ??? 섹션을 참조하십시오.
3. registerQuorumPubKey 명령을 사용하여 퍼블릭 키를 등록합니다. 자세한 내용은 다음 예제를 참조하거나 help registerQuorumPubKey 명령을 사용하십시오.

Example – HSM에 퍼블릭 키 등록

다음 예제에서는 cloudhsm_mgmt_util 명령줄 도구의 registerQuorumPubKey 명령을 사용하여 HSM에 CO의 퍼블릭 키를 등록하는 방법을 보여 줍니다. 이 명령을 사용하려면 CO가 HSM에 로그인해야 합니다. 이 값들을 사용자의 값으로 대체합니다.

```
aws-cloudhsm> registerQuorumPubKey CO <officer1> <officer1.token> <officer1.token.sig>
 <officer1.pub>

*************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?y
registerQuorumPubKey success on server 0(10.0.2.14)
```

<officer1.token>

서명되지 않은 등록 토큰이 포함된 파일의 경로입니다. 최대 파일 크기가 245바이트인 임의의 데이터를 포함할 수 있습니다.

필수 여부: 예

<officer1.token.sig>

등록 토큰의 SHA256_PKCS 메커니즘 서명 해시가 포함된 파일의 경로입니다.

필수 여부: 예

<officer1.pub>

비대칭 RSA-2048 키 페어의 퍼블릭 키를 포함하는 파일의 경로입니다. 프라이빗 키를 사용하여 등록 토큰에 서명합니다.

필수 여부: 예

모든 CO가 퍼블릭 키를 등록한 후 다음 예제와 같이 listUsers 명령 출력의 MofnPubKey 열에 이 내용이 표시됩니다.

```
aws-cloudhsm>listUsers
```

```
Users on server 0(10.0.2.14):
Number of users found:7

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                             NO
           0               NO
         2              AU              app_user                          NO
           0               NO
         3              CO              officer1                          YES
           0               NO
         4              CO              officer2                          YES
           0               NO
         5              CO              officer3                          YES
           0               NO
         6              CO              officer4                          YES
           0               NO
         7              CO              officer5                          YES
           0               NO
Users on server 1(10.0.1.4):
Number of users found:7

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                             NO
           0               NO
         2              AU              app_user                          NO
           0               NO
         3              CO              officer1                          YES
           0               NO
         4              CO              officer2                          YES
           0               NO
         5              CO              officer3                          YES
           0               NO
         6              CO              officer4                          YES
           0               NO
         7              CO              officer5                          YES
           0               NO
```

HSM에서 쿼럼 최소값 설정

CO에 쿼럼 인증을 사용하려면 CO가 HSM에 로그인한 다음 쿼럼 최소값(m 값이라고도 함)을 설정해야 합니다. 이것은 HSM 사용자 관리 작업을 수행하는 데 필요한 최소 CO 승인 수입니다. 서명을 위해

키를 등록하지 않은 CO를 포함하여 HSM의 모든 CO는 쿼럼 최소값을 설정할 수 있습니다. 쿼럼 최소값은 언제든지 변경할 수 있습니다. 자세한 내용은 최소값 변경을 참조하십시오.

HSM에서 쿼럼 최소값을 설정하려면

1. 다음 명령을 사용하여 cloudhsm_mgmt_util 명령줄 도구를 시작합니다.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. loginHSM 명령을 사용하여 HSM에 CO로 로그인합니다. 자세한 내용은 ??? 섹션을 참조하십시오.
3. setMValue 명령을 사용하여 쿼럼 최소값을 설정합니다. 자세한 내용은 다음 예제를 참조하거나 help setMValue 명령을 사용하십시오.

Example - HSM에서 쿼럼 최소값 설정

이 예제에서는 쿼럼 최소값 2를 사용합니다. HSM의 총 CO 수까지 2~8 사이의 어떤 값도 선택할 수 있습니다. 이 예에서 HSM에는 6개의 CO가 있으므로 가능한 최대 값은 6입니다.

다음 예제 명령을 사용하려면 마지막 숫자(*2*)를 원하는 쿼럼 최소값으로 대체해야 합니다.

```
aws-cloudhsm>setMValue 3 2
*************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?y
Setting M Value(2) for 3 on 2 nodes
```

앞의 예제에서 첫 번째 숫자(3)는 쿼럼 최소값을 설정할 HSM 서비스를 식별합니다.

다음 표에는 HSM 서비스 식별자와 함께 식별자의 이름, 설명, 서비스에 포함된 명령이 나열되어 있습니다.

서비스 식별자	서비스 이름	서비스 설명	HSM 명령
3	USER_MGMT	HSM 사용자 관리	• createUser

서비스 식별자	서비스 이름	서비스 설명	HSM 명령
			• deleteUser • changePswd(다른 HSM 사용자의 암호를 변경할 때만 해당됨)
4	MISC_CO	기타 CO 서비스	• setMValue

서비스의 쿼럼 최소값을 가져오려면 다음 예제와 같이 getMValue 명령을 사용합니다.

```
aws-cloudhsm>getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

앞에 나온 getMValue 명령의 출력은 HSM 사용자 관리 작업(서비스 3)의 쿼럼 최소값이 이제 2라는 것을 보여 줍니다.

이 단계들을 완료한 후 CO(Crypto Officer)의 쿼럼 인증 사용 섹션을 참조하십시오.

CO(Crypto Officer)의 쿼럼 인증 사용

HSM의 CO(Crypto Officer)는 HSM에서 다음 작업에 대한 쿼럼 인증을 구성할 수 있습니다.

- HSM 사용자 생성
- HSM 사용자 삭제
- 다른 HSM 사용자의 암호 변경

HSM이 쿼럼 인증에 대해 구성된 후에는 CO가 단독으로 HSM 사용자 관리 작업을 수행할 수 없습니다. 다음 예제는 CO가 HSM에서 새 사용자를 생성하려 할 때의 출력을 보여 줍니다. 명령이 실패하고 RET_MXN_AUTH_FAILED 오류가 반환됩니다. 이 오류는 쿼럼 인증이 실패했음을 나타냅니다.

```
aws-cloudhsm>createUser CU user1 password
**************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
```

```
ensure this operation is executed on all nodes in the cluster.
*************************************************************

Do you want to continue(y/n)?y
Creating User user1(CU) on 2 nodes
createUser failed: RET_MXN_AUTH_FAILED
creating user on server 0(10.0.2.14) failed

Retry/Ignore/Abort?(R/I/A):A
```

HSM 사용자 관리 작업을 수행하려면 CO가 다음 작업을 완료해야 합니다.

1. 쿼럼 토큰을 가져옵니다.
2. 다른 CO의 승인(서명)을 받습니다.
3. HSM에 대한 토큰을 승인합니다.
4. HSM 사용자 관리 작업을 수행합니다.

아직 HSM에서 CO의 쿼럼 인증을 구성하지 않았다면 지금 구성합니다. 자세한 내용은 최초 설정 섹션을 참조하십시오.

쿼럼 토큰 가져오기

먼저 CO가 cloudhsm_mgmt_util 명령줄 도구를 사용하여 쿼럼 토큰을 요청해야 합니다.

쿼럼 토큰을 가져오려면

1. 다음 명령을 사용하여 cloudhsm_mgmt_util 명령줄 도구를 시작합니다.

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

2. loginHSM 명령을 사용하여 HSM에 CO로 로그인합니다. 자세한 내용은 ??? 섹션을 참조하십시오.
3. getToken 명령을 사용하여 쿼럼 토큰을 가져옵니다. 자세한 내용은 다음 예제를 참조하거나 help getToken 명령을 사용하십시오.

Example – 쿼럼 토큰 가져오기

이 예제는 사용자 이름이 officer1인 CO용 쿼럼 토큰을 가져와 이름이 `officer1.token`인 파일에 이 토큰을 저장합니다. 예제 명령을 사용하려면 이들 값을 본인의 것으로 바꿉니다.

- *officer1* – 토큰을 가져오는 CO의 이름입니다. 이것은 HSM에 로그인하여 이 명령을 실행하는 CO와 동일해야 합니다.
- *officer1.token* – 쿼럼 토큰을 저장하는 데 사용할 파일의 이름입니다.

다음 명령에서 3은 가져오는 토큰을 사용할 수 있는 서비스를 식별합니다. 이 사례에서, 토큰은 HSM 사용자 관리 작업(서비스 3)용입니다. 자세한 내용은 HSM에서 쿼럼 최소값 설정 섹션을 참조하십시오.

```
aws-cloudhsm>getToken 3 officer1 officer1.token
getToken success on server 0(10.0.2.14)
Token:
Id:1
Service:3
Node:1
Key Handle:0
User:officer1
getToken success on server 1(10.0.1.4)
Token:
Id:1
Service:3
Node:0
Key Handle:0
User:officer1
```

승인 CO로부터 서명 받기

쿼럼 토큰을 보유한 CO는 다른 CO로부터 토큰을 승인받아야 합니다. 승인을 위해 다른 CO는 서명 키를 사용하여 암호화 방식으로 토큰에 서명합니다. 이 절차는 HSM 외부에서 이루어집니다.

토큰에 서명하는 방법은 다양합니다. 다음 예제에서는 OpenSSL을 사용한 작업 방법을 보여 줍니다. 다른 서명 도구를 사용하려면 해당 도구가 CO의 프라이빗 키(서명 키)를 사용하여 토큰의 SHA-256 다이제스트에 서명하도록 해야 합니다.

Example – 승인 CO로부터 서명 받기

이 예제에서는 토큰을 보유한 CO(officer1)가 2개 이상의 승인을 필요로 합니다. 다음 예제 명령은 두 CO가 OpenSSL을 사용하여 암호화 방식으로 토큰에 서명하는 방법을 보여 줍니다.

첫 번째 명령에서, officer1이 자신의 토큰에 서명합니다. 다음 예제 명령을 사용하려면 다음 값을 본인의 것으로 바꿉니다.

- *officer1.key* 및 *officer2.key* – CO의 서명 키를 포함하는 파일의 이름입니다.
- *officer1.token.sig1* 및 *officer1.token.sig2* – 서명을 저장하는 데 사용할 파일의 이름입니다. 각 서명을 다른 파일에 저장해야 합니다.
- *officer1.token* – CO가 서명하는 토큰을 포함하는 파일의 이름입니다.

```
$ openssl dgst -sha256 -sign officer1.key -out officer1.token.sig1 officer1.token
Enter pass phrase for officer1.key:
```

다음 명령에서, officer2가 동일한 토큰에 서명합니다.

```
$ openssl dgst -sha256 -sign officer2.key -out officer1.token.sig2 officer1.token
Enter pass phrase for officer2.key:
```

HSM에서 서명된 토큰 승인

CO는 다른 CO로부터 최소 승인(서명)을 얻은 후 HSM에서 서명된 토큰을 승인해야 합니다.

HSM에서 서명된 토큰을 승인하려면

1. 토큰 승인 파일을 생성합니다. 자세한 내용은 다음 예제를 참조하세요.
2. 다음 명령을 사용하여 cloudhsm_mgmt_util 명령줄 도구를 시작합니다.

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

3. loginHSM 명령을 사용하여 HSM에 CO로 로그인합니다. 자세한 내용은 ??? 섹션을 참조하십시오.
4. approveToken 명령을 사용하여 서명된 토큰을 승인하고 토큰 승인 파일을 전달합니다. 자세한 내용은 다음 예제를 참조하세요.

Example – 토큰 승인 파일 생성 및 HSM에서 서명된 토큰 승인

토큰 승인 파일은 HSM이 요구하는 특정 형식의 텍스트 파일입니다. 이 파일에는 토큰, 승인자 및 승인자 서명에 관한 정보가 포함됩니다. 다음은 토큰 승인 파일의 예입니다.

```
# For "Multi Token File Path", type the path to the file that contains
# the token. You can type the same value for "Token File Path", but
# that's not required. The "Token File Path" line is required in any
# case, regardless of whether you type a value.
```

```
Multi Token File Path = officer1.token;
Token File Path = ;

# Total number of approvals
Number of Approvals = 2;

# Approver 1
# Type the approver's type, name, and the path to the file that
# contains the approver's signature.
Approver Type = 2; # 2 for CO, 1 for CU
Approver Name = officer1;
Approval File = officer1.token.sig1;

# Approver 2
# Type the approver's type, name, and the path to the file that
# contains the approver's signature.
Approver Type = 2; # 2 for CO, 1 for CU
Approver Name = officer2;
Approval File = officer1.token.sig2;
```

토큰 승인 파일을 생성한 후 CO는 cloudhsm_mgmt_util 명령줄 도구를 사용하여 HSM에 로그인하고 다음 예제와 같이 approveToken 명령을 사용하여 토큰을 승인합니다. *approval.txt*를 토큰 승인 파일의 이름으로 바꿉니다.

```
aws-cloudhsm>approveToken approval.txt
approveToken success on server 0(10.0.2.14)
approveToken success on server 1(10.0.1.4)
```

이 명령이 성공하면 HSM이 쿼럼 토큰을 승인한 것입니다. 토큰 상태를 확인하려면 다음 예제와 같이 listTokens 명령을 사용합니다. 명령의 출력은 토큰이 필요한 수의 승인을 획득한 것을 보여 줍니다.

토큰 유효 기간은 토큰이 HSM에서 얼마 동안 보관될지를 나타냅니다. 토큰 유효 기간이 경과한 후에도(0초) 토큰을 사용할 수 있습니다.

```
aws-cloudhsm>listTokens

======================
    Server 0(10.0.2.14)
======================
-------- Token - 0 ----------
Token:
Id:1
```

```
Service:3
Node:1
Key Handle:0
User:officer1
Token Validity: 506 sec
Required num of approvers : 2
Current num of approvals : 2
Approver-0: officer1
Approver-1: officer2
Num of tokens = 1

====================
    Server 1(10.0.1.4)
====================
-------- Token - 0 ----------
Token:
Id:1
Service:3
Node:0
Key Handle:0
User:officer1
Token Validity: 506 sec
Required num of approvers : 2
Current num of approvals : 2
Approver-0: officer1
Approver-1: officer2
Num of tokens = 1

listTokens success
```

사용자 관리 작업에 토큰 사용

이전 단원에서 설명한 대로 CO가 필요한 수의 승인을 획득한 후, CO는 다음의 HSM 사용자 관리 작업 중 하나를 수행할 수 있습니다.

- createUser 명령을 사용하여 HSM 사용자를 생성
- deleteUser 명령을 사용하여 HSM 사용자 삭제
- changePswd 명령을 사용하여 다른 HSM 사용자의 암호 변경

이러한 명령을 사용하는 방법은 HSM 사용자 관리 섹션을 참조하십시오.

CO는 한 작업에만 토큰을 사용할 수 있습니다. 해당 작업이 성공하면 토큰은 더 이상 유효하지 않습니다. 다른 HSM 사용자 관리 작업을 수행하려면 CO는 새로운 쿼럼 토큰을 가져오고, 승인자로부터 새로운 서명을 받고, HSM에서 새로운 토큰을 승인해야 합니다.

> **Note**
>
> MofN 토큰은 현재 로그인 세션이 열려 있는 동안에만 유효합니다. cloudhsm_mgmt_util에서 로그아웃하거나 네트워크 연결이 끊기면 토큰은 더 이상 유효하지 않습니다. 마찬가지로 승인된 토큰은 cloudhsm_mgmt_util 내에서만 사용할 수 있으며 다른 애플리케이션에서 인증하는 데는 사용할 수 없습니다.

다음 예제 명령에서는 CO가 HSM에서 새 사용자를 생성합니다.

```
aws-cloudhsm>createUser CU user1 password
*************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?y
Creating User user1(CU) on 2 nodes
```

이전 명령이 성공한 후 후속 listUsers 명령이 새 사용자를 보여 줍니다.

```
aws-cloudhsm>listUsers
Users on server 0(10.0.2.14):
Number of users found:8

    User Id             User Type       User Name                           MofnPubKey
  LoginFailureCnt         2FA
        1               PCO             admin                                 NO
          0               NO
        2               AU              app_user                              NO
          0               NO
        3               CO              officer1                             YES
          0               NO
        4               CO              officer2                             YES
          0               NO
```

```
         5              CO             officer3                                YES
           0               NO
         6              CO             officer4                                YES
           0               NO
         7              CO             officer5                                YES
           0               NO
         8              CU             user1                                    NO
           0               NO
Users on server 1(10.0.1.4):
Number of users found:8

    User Id             User Type      User Name                           MofnPubKey
  LoginFailureCnt         2FA
         1              PCO            admin                                    NO
           0               NO
         2              AU             app_user                                 NO
           0               NO
         3              CO             officer1                                YES
           0               NO
         4              CO             officer2                                YES
           0               NO
         5              CO             officer3                                YES
           0               NO
         6              CO             officer4                                YES
           0               NO
         7              CO             officer5                                YES
           0               NO
         8              CU             user1                                    NO
           0               NO
```

CO가 다른 HSM 사용자 관리 작업을 시도할 경우, 다음 예제와 같이 쿼럼 인증 오류가 발생하며 실패합니다.

```
aws-cloudhsm>deleteUser CU user1
Deleting user user1(CU) on 2 nodes
deleteUser failed: RET_MXN_AUTH_FAILED
deleteUser failed on server 0(10.0.2.14)

Retry/rollBack/Ignore?(R/B/I):I
deleteUser failed: RET_MXN_AUTH_FAILED
deleteUser failed on server 1(10.0.1.4)

Retry/rollBack/Ignore?(R/B/I):I
```

다음 예제와 같이 listTokens 명령은 승인된 토큰이 CO에게 없음을 보여 줍니다. 다른 HSM 사용자 관리 작업을 수행하려면 CO는 새로운 쿼럼 토큰을 가져오고, 승인자로부터 새로운 서명을 받고, HSM에서 새로운 토큰을 승인해야 합니다.

```
aws-cloudhsm>listTokens

====================
    Server 0(10.0.2.14)
====================
Num of tokens = 0

====================
    Server 1(10.0.1.4)
====================
Num of tokens = 0

listTokens success
```

CO(Crypto Officer)의 쿼럼 최소값 변경

CO(Crypto Officer)가 쿼럼 인증을 사용할 수 있도록 쿼럼 최소값을 설정한 후 쿼럼 최소값을 변경할 수 있습니다. HSM에서는 승인자의 수가 현재 쿼럼 최소값 이상인 경우에만 쿼럼 최소값을 변경할 수 있습니다. 예를 들어 쿼럼 최소값이 2라면 적어도 2명의 CO가 승인해야 쿼럼 최소값을 변경할 수 있습니다.

쿼럼 최소값을 변경하기 위해 쿼럼 승인을 받으려면 setMValue 명령(서비스 4)에 대한 쿼럼 토큰이 필요합니다. setMValue 명령(서비스 4)에 대한 쿼럼 토큰을 얻으려면 서비스 4의 쿼럼 최소값이 1보다 커야 합니다. 즉, CO(서비스 3)의 쿼럼 최소값을 변경하려면 그전에 서비스 4의 쿼럼 최소값을 변경해야 합니다.

다음 표에는 HSM 서비스 식별자와 함께 식별자의 이름, 설명, 서비스에 포함된 명령이 나열되어 있습니다.

서비스 식별자	서비스 이름	서비스 설명	HSM 명령
3	USER_MGMT	HSM 사용자 관리	• createUser • deleteUser • changePswd(다른 HSM 사용자의 암호

서비스 식별자	서비스 이름	서비스 설명	HSM 명령
			를 변경할 때만 해당됨)
4	MISC_CO	기타 CO 서비스	• setMValue

crypto officer의 쿼럼 최소값을 변경하려면

1. 다음 명령을 사용하여 cloudhsm_mgmt_util 명령줄 도구를 시작합니다.

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

2. loginHSM 명령을 사용하여 HSM에 CO로 로그인합니다. 자세한 내용은 ??? 섹션을 참조하십시오.
3. getMValue 명령을 사용하여 서비스 3의 쿼럼 최소값을 가져옵니다. 자세한 내용은 다음 예제를 참조하세요.
4. getMValue 명령을 사용하여 서비스 4의 쿼럼 최소값을 가져옵니다. 자세한 내용은 다음 예제를 참조하세요.
5. 서비스 4의 쿼럼 최소값이 서비스 3의 값보다 작다면 setMValue 명령을 사용하여 서비스 4의 값을 변경합니다. 서비스 3의 값보다 크거나 같은 값으로 서비스 4의 값을 변경합니다. 자세한 내용은 다음 예제를 참조하세요.
6. 쿼럼 토큰을 가져오고 토큰을 사용할 수 있는 서비스로 서비스 4를 지정하도록 주의하십시오.
7. 다른 CO의 승인(서명)을 받습니다.
8. HSM에 대한 토큰을 승인합니다.
9. setMValue 명령을 사용하여 서비스 3(CO가 수행하는 사용자 관리 작업)의 쿼럼 최소값을 변경합니다.

Example – 서비스 4의 쿼럼 최소값 가져오기 및 변경

다음 예제 명령은 서비스 3의 쿼럼 최소값이 현재 2임을 보여 줍니다.

```
aws-cloudhsm>getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

다음 예제 명령은 서비스 4의 쿼럼 최소값이 현재 1임을 보여 줍니다.

```
aws-cloudhsm>getMValue 4
MValue of service 4[MISC_CO] on server 0 : [1]
MValue of service 4[MISC_CO] on server 1 : [1]
```

서비스 4의 쿼럼 최소값을 변경하려면 setMValue 명령을 사용하여 서비스 3의 쿼럼 최소값보다 크거나 같은 값을 설정합니다. 다음 예제는 서비스 4의 쿼럼 최소값을 서비스 3에 대해 설정된 값과 같은 2로 설정합니다.

```
aws-cloudhsm>setMValue 4 2
************************CAUTION*******************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**************************************************************

Do you want to continue(y/n)?y
Setting M Value(2) for 4 on 2 nodes
```

다음 명령은 서비스 3과 서비스 4의 쿼럼 최소값이 이제 2임을 보여 줍니다.

```
aws-cloudhsm>getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

```
aws-cloudhsm>getMValue 4
MValue of service 4[MISC_CO] on server 0 : [2]
MValue of service 4[MISC_CO] on server 1 : [2]
```

에서 키 관리 AWS CloudHSM

AWS CloudHSM에서는 다음 중 하나를 사용하여 클러스터의 HSM에 있는 키를 관리하십시오.

- PKCS #11 라이브러리
- JCE 공급자
- CNG 및 KSP 공급자
- CloudHSM CLI

키를 관리하려면 CU(Crypto User)의 사용자 이름과 암호로 HSM에 로그인합니다. CU만 키를 생성할 수 있습니다. 키를 생성한 CU가 해당 키를 소유하고 관리합니다.

주제

의 키 동기화 및 내구성 설정 AWS CloudHSM

이 항목에서는 의 키 동기화 설정 AWS CloudHSM, 클러스터에서 키를 사용하는 고객이 직면하는 일반적인 문제, 키의 내구성을 높이기 위한 전략에 대해 설명합니다.

주제

개념

토큰 키

키 생성, 가져오기 또는 래핑 해제 작업 중에 생성하는 영구 키. AWS CloudHSM 클러스터 전체에서 토큰 키를 동기화합니다.

세션 키

클러스터의 한 하드웨어 보안 모듈 (HSM) 에만 존재하는 임시 키. AWS CloudHSM 클러스터 전체에서 세션 키를 동기화하지 않습니다.

클라이언트측 키 동기화

키 생성, 가져오기 또는 래핑 해제 작업 중에 생성한 토큰 키를 복제하는 클라이언트 측 프로세스입니다. 최소 두 개의 HSM이 있는 클러스터를 실행하여 토큰 키의 내구성을 높일 수 있습니다.

서버측 키 동기화

클러스터의 모든 HSM에 주기적으로 키를 복제합니다. 관리할 필요가 없습니다.

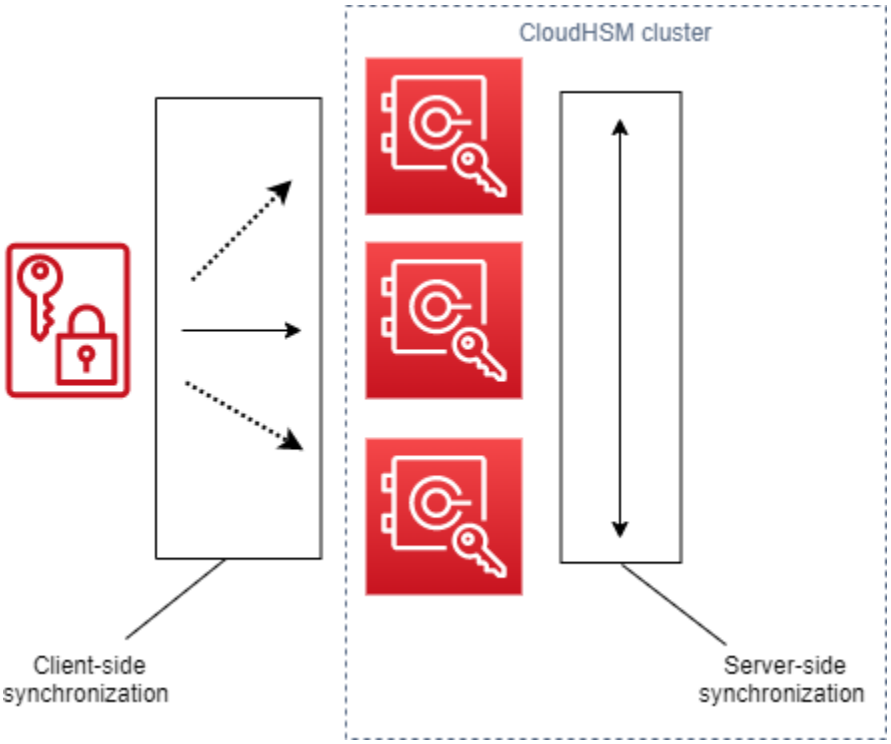
클라이언트 키 내구성 설정

클라이언트에서 구성한 설정으로 키 내구성에 영향을 줍니다. 이러한 설정은 Client SDK 5와 Client SDK 3에서 다르게 작동합니다.

- Client SDK 5에서는 이 설정을 사용하여 단일 HSM 클러스터를 실행합니다.
- Client SDK 3에서는 이 설정을 사용하여 키 생성 작업이 성공하는 데 필요한 HSM 수를 지정합니다.

키 동기화 이해

AWS CloudHSM 키 동기화를 사용하여 클러스터의 모든 HSM에서 토큰 키를 복제합니다. 키 생성, 가져오기 또는 래핑 해제 작업 중에 토큰 키를 영구 키로 생성합니다. 클러스터 전체에 이러한 키를 분산하기 위해 CloudHSM은 클라이언트 측 키 동기화 및 서버 측 키 동기화 모두를 제공합니다.



서버 측과 클라이언트 측 모두에서 키 동기화의 목표는 새 키를 생성한 후 가능한 한 빨리 클러스터 전체에 배포하는 것입니다. 이는 새 키를 사용하기 위해 거는 후속 호출이 클러스터의 사용 가능한 모든 HSM으로 라우팅될 수 있기 때문에 중요합니다. 키가 없는 HSM으로 호출을 라우팅하면 호출이 실패합니다. 키 생성 작업 후에 수행된 후속 호출을 응용 프로그램에서 재시도하도록 지정하여 이러한 유형

의 실패를 완화할 수 있습니다. 동기화에 필요한 시간은 클러스터의 워크로드 및 기타 무형 자산에 따라 달라질 수 있습니다. CloudWatch 메트릭을 사용하여 애플리케이션이 이러한 유형의 상황에서 사용해야 하는 타이밍을 결정하십시오. 자세한 내용은 CloudWatch 메트릭을 참조하십시오.

클라우드 환경에서 키 동기화의 문제는 키 내구성입니다. 단일 HSM에서 키를 생성하고 해당 키를 즉시 사용하기 시작하는 경우가 많습니다. 키가 클러스터의 다른 HSM에 복제되기 전에 키를 생성한 HSM에서 장애가 발생하면 키를 잃고 키로 암호화된 모든 항목에 액세스할 수 없게 됩니다. 이러한 위험을 완화하기 위해 클라이언트측 동기화를 제공합니다. 클라이언트측 동기화는 키 생성, 가져오기 또는 래핑 해제 작업 중에 생성한 키를 복제하는 클라이언트측 프로세스입니다. 키를 생성할 때 복제하면 키의 내구성이 향상됩니다. 물론 단일 HSM으로 클러스터의 키를 복제할 수는 없습니다. 키의 내구성을 높이려면 최소 두 개의 HSM을 사용하도록 클러스터를 구성하는 것도 좋습니다. 클라이언트측 동기화와 HSM이 두 개 있는 클러스터를 사용하면 클라우드 환경에서 키 내구성 문제를 해결할 수 있습니다.

클라이언트 키 내구성 설정 관련 작업

키 동기화는 대부분 자동 프로세스이지만 클라이언트측 키 내구성 설정은 관리할 수 있습니다. 클라이언트측 키 내구성 설정은 Client SDK 5와 Client SDK 3에서 다르게 작동합니다.

- Client SDK 5에서는 최소 2개의 HSM으로 클러스터를 실행해야 하는 키 가용성 쿼럼의 개념을 소개합니다. 클라이언트측 키 내구성 설정을 사용하여 두 HSM 요구 사항을 옵트아웃할 수 있습니다. 쿼럼에 대한 자세한 내용은 the section called “Client SDK 5 개념” 단원을 참조하십시오.
- Client SDK 3에서는 클라이언트측 키 내구성 설정을 사용하여 전체 작업이 성공으로 간주되려면 키 생성이 성공해야 하는 HSM 수를 지정합니다.

Client SDK 5 클라이언트 키 내구성 설정

Client SDK 5에서 키 동기화는 완전 자동 프로세스입니다. 키 가용성 쿼럼을 사용하여 새로 만든 키가 클러스터의 두 HSM에 있어야 애플리케이션에서 키를 사용할 수 있습니다. 키 가용성 쿼럼을 사용하려면 클러스터에 최소 두 개의 HSM이 있어야 합니다.

클러스터 구성이 키 내구성 요구 사항을 충족하지 않는 경우 토큰 키를 생성하거나 사용하려고 하면 로그에 다음 오류 메시지가 표시되면서 실패합니다.

```
Key <key handle> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

클라이언트 구성 설정을 사용하여 키 가용성 쿼럼에서 제외할 수 있습니다. 예를 들어, 단일 HSM으로 클러스터를 실행하지 않도록 옵트아웃할 수 있습니다.

Client SDK 5 개념

주요 가용성 쿼럼

AWS CloudHSM 애플리케이션이 키를 사용할 수 있으려면 먼저 키가 있어야 하는 클러스터의 HSM 수를 지정합니다. 최소 두 개의 HSM이 있는 클러스터가 필요합니다.

클라이언트 키 내구성 설정 관리

클라이언트 키 내구성 설정을 관리하려면 Client SDK 5용 구성 도구를 사용해야 합니다.

PKCS #11 library

Linux용 Client SDK 5의 클라이언트 키 내구성 비활성화

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Windows용 Client SDK 5의 클라이언트 키 내구성을 사용하지 않도록 설정

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-
availability-check
```

OpenSSL Dynamic Engine

Linux용 Client SDK 5의 클라이언트 키 내구성 비활성화

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

JCE provider

Linux용 Client SDK 5의 클라이언트 키 내구성 비활성화

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Windows용 Client SDK 5의 클라이언트 키 내구성을 사용하지 않도록 설정

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-
availability-check
```

CloudHSM CLI

Linux용 Client SDK 5의 클라이언트 키 내구성 비활성화

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Windows용 Client SDK 5의 클라이언트 키 내구성을 사용하지 않도록 설정

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-
availability-check
```

Client SDK 3 클라이언트 키 내구성 설정

Client SDK 3에서 키 동기화는 대부분 자동 프로세스이지만 클라이언트 키 내구성 설정을 사용하여 키의 내구성을 높일 수 있습니다. 전체 작업이 성공으로 간주되려면 키 생성이 성공해야 하는 HSM 수를 지정합니다. 클라이언트측 동기화는 어떤 설정을 선택하든 항상 클러스터의 모든 HSM에 키를 복제하기 위해 최선을 다합니다. 설정에 따라 지정한 HSM 수에 대한 키 생성이 적용됩니다. 값을 지정했는데 시스템에서 해당 수의 HSM에 키를 복제할 수 없는 경우 시스템에서 자동으로 불필요한 키 구성 요소를 정리하므로 다시 시도할 수 있습니다.

> **Important**
>
> 클라이언트 키 내구성 설정을 설정하지 않으면(또는 기본값 1을 사용하는 경우) 키가 손실되기 쉽습니다. 서버측 서비스가 해당 키를 다른 HSM에 복제하기 전에 현재 HSM에 장애가 발생하면 키 구성 요소를 잃게 됩니다.

키 내구성을 극대화하려면 클라이언트측 동기화에 HSM을 두 개 이상 지정하는 것이 좋습니다. HSM을 얼마나 많이 지정하든 클러스터의 워크로드는 동일하게 유지된다는 점을 기억하십시오. 클라이언트측 동기화는 항상 클러스터의 모든 HSM에 키를 복제하기 위해 최선을 다합니다.

권장 사항

- 최소: 클러스터당 HSM 2개
- 최대: 클러스터의 총 HSM 수보다 하나 적음

클라이언트측 동기화가 실패하는 경우 클라이언트 서비스는 생성되어 지금은 원치 않는 불필요한 키를 모두 정리합니다. 이 정리는 최선의 방법이지만 항상 효과가 있는 것은 아닙니다. 정리가 실패하면 불필요한 주요 자료를 삭제해야 할 수도 있습니다. 자세한 내용은 키 동기화 실패를 참조하십시오.

클라이언트 키 내구성을 위한 구성 파일 설정

클라이언트 키 내구성 설정을 지정하려면 `cloudhsm_client.cfg`를 편집해야 합니다.

클라이언트 구성 파일 편집

1. `cloudhsm_client.cfg`를 엽니다.

   Linux:

```
/opt/cloudhsm/etc/cloudhsm_client.cfg
```

Windows:

```
C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

2. 파일 `client` 노드에서 키 생성 작업이 성공하기 위해 성공적으로 키를 AWS CloudHSM 생성해야 하는 HSM의 최소 개수 값을 `create_object_minimum_nodes` 추가하고 지정합니다.

```
"create_object_minimum_nodes" : 2
```

> **Note**
>
> key_mgmt_util (KMU) 명령줄 도구에는 클라이언트 키 내구성을 위한 추가 설정이 있습니다. 자세한 정보는 the section called “KMU 및 클라이언트측 동기화” 단원을 참조하십시오.

구성 참조

클라이언트 측 동기화 속성은 다음 `cloudhsm_client.cfg`의 발췌문에 나와 있습니다:

```
{
    "client": {
        "create_object_minimum_nodes" : 2,
        ...
    },

    ...
}
```

create_object_minimum_nodes

키 생성, 키 가져오기 또는 키 래핑 해제 작업을 성공으로 간주하는 데 필요한 최소 HSM 수를 지정합니다. 기본값은 "1"로 설정되어 있습니다. 즉, 모든 키 생성 작업에 대해 클라이언트측 서비스는 클러스터의 모든 HSM에 키 생성을 시도하지만 성공하려면 클러스터의 HSM 하나에 단일 키만 생성하면 됩니다.

KMU 및 클라이언트측 동기화

key_mgmt_util(KMU) 명령줄 도구를 사용하여 키를 생성하는 경우 선택적 명령줄 파라미터(`-min_srv`)를 사용하여 키를 복제할 HSM 수를 제한합니다. 구성 파일에 명령줄 매개 변수와 값을 지정하는 경우 두 값 중 더 큰 값을 적용합니다. AWS CloudHSM

자세한 정보는 다음 주제를 참조하세요.

- GenDSA KeyPair
- 제네C KeyPair
- GenRSA KeyPair
- genSymKey
- importPrivateKey
- importPubKey
- imSymKey
- insertMaskedObject
- unWrapKey

복제된 클러스터 간 키 동기화

클라이언트측 및 서버측 동기화는 동일한 클러스터 내에서 키를 동기화하는 용도로만 사용됩니다. 클러스터의 백업을 다른 지역에 복사하는 경우 cloudhsm_mgmt_util(CMU)의 SyncKey 명령을 사용하여 클러스터 간에 키를 동기화할 수 있습니다. 지역 간 이중화를 위해 또는 재해 복구 프로세스를 단순화하기 위해 복제된 클러스터를 사용할 수 있습니다. 자세한 내용은 syncKey을 참조하십시오.

AES 키 랩핑 인 AWS CloudHSM

이 항목에서는 AES 키 래핑 옵션에 대해 설명합니다 AWS CloudHSM. AES 키 래핑은 AES 키(래핑 키)를 사용하여 모든 유형의 다른 키(대상 키)를 래핑합니다. 키 래핑을 사용하여 저장된 키를 보호하거나 안전하지 않은 네트워크를 통해 키를 전송할 수 있습니다.

주제

- 지원되는 알고리즘
- AES 키 랩인 사용 AWS CloudHSM

지원되는 알고리즘

AWS CloudHSM AES 키 래핑을 위한 세 가지 옵션을 제공합니다. 각 옵션은 래핑되기 전에 대상 키가 패딩되는 방식을 기반으로 합니다. 패딩은 키 래핑을 호출할 때 사용하는 알고리즘에 따라 자동으로 수행됩니다. 다음 표에는 애플리케이션에 적합한 래핑 메커니즘을 선택하는 데 도움이 되는 지원되는 알고리즘과 관련된 세부 정보가 나와 있습니다.

AES 키 래핑 알고리즘	사양	지원되는 대상 키 유형	패딩 체계	AWS CloudHSM 클라이언트 가용성
제로 패딩을 사용하는 AES 키 래핑	RFC 5649 및 SP 800 - 38F	모두	정렬 차단을 위해 필요한 경우 키 비트 뒤에 0 추가	SDK 3.1 이상
패딩이 없는 AES 키 래핑	RFC 3394 및 SP 800 - 38F	AES 및 3DES 등의 블록 정렬 키	None	SDK 3.1 이상
PKCS #5 패딩을 사용하는 AES 키 래핑	None	모두	정렬 차단을 위해 PKCS #5 패딩 체계에 따라 최소 8바이트가 추가됨	모두

애플리케이션에서 이전 표의 AES 키 래핑 알고리즘을 사용하는 방법은 AWS CloudHSM에서 AES 키 래핑 사용을 참조하십시오.

AES 키 래핑의 초기화 벡터 이해

래핑 전에 CloudHSM은 데이터 무결성을 위해 대상 키에 초기화 벡터(IV)를 추가합니다. 각 키 래핑 알고리즘에는 허용되는 IV 유형에 대한 특정 제한이 있습니다. IV를 AWS CloudHSM설정하려면 다음 두 가지 옵션이 있습니다.

- 암시적: IV를 NULL로 설정합니다. 그러면 CloudHSM에서 래핑 및 언래핑 작업의 해당 알고리즘에 대해 기본값을 사용합니다(권장).
- 명시적: 기본 IV 값을 키 래핑 함수로 전달하여 IV를 설정합니다.

> **Important**
>
> 애플리케이션에 어떤 IV를 사용하고 있는지 알고 있어야 합니다. 키를 언래핑하려면 키 래핑에 사용한 것과 동일한 IV를 제공해야 합니다. 암시적 IV를 사용하여 래핑할 경우 언래핑하려면 암시적 IV를 사용하십시오. 암시적 IV를 사용하면 CloudHSM이 기본값을 사용하여 언래핑합니다.

다음 표에서는 래핑 알고리즘이 지정하는 IV에 대해 허용되는 값을 설명합니다.

AES 키 래핑 알고리즘	암시적 IV	명시적 IV
제로 패딩을 사용하는 AES 키 래핑	필수 기본값: (사양에 따라 IV가 내부적으로 계산됨)	허용되지 않음
패딩이 없는 AES 키 래핑	허용됨(권장) 기본 값: `0xA6A6A6A6A6A6A6A6`	허용됨 다음 값만 허용됨: `0xA6A6A6A6A6A6A6A6`
PKCS #5 패딩을 사용하는 AES 키 래핑	허용됨(권장) 기본 값: `0xA6A6A6A6A6A6A6A6`	허용됨 다음 값만 허용됨: `0xA6A6A6A6A6A6A6A6`

AES 키 랩인 사용 AWS CloudHSM

다음과 같이 키를 래핑하고 언래핑합니다.

- PKCS #11 라이브러리에서 다음 표에 표시된 대로 `C_WrapKey` 및 `C_UnWrapKey` 함수에 대해 적절한 메커니즘을 선택합니다.
- JCE 공급자에서 적절한 알고리즘, 모드 및 패딩 조합을 선택하고 다음 표에 표시된 대로 암호 메서드 `Cipher.WRAP_MODE` 및 `Cipher.UNWRAP_MODE`를 구현합니다.
- CloudHSM CLI에서 다음 표와 같이 키 언랩 지원되는 알고리즘 및 알고리즘 목록에서 키 랩 적절한 알고리즘을 선택합니다.

- key_mgmt_util(KMU)에서 다음 표에 표시된 대로 적절한 m 값과 함께 wrapKey 및 unWrapKey 명령을 사용합니다.

AES 키 래핑 알고리즘	PKCS #11 메커니즘	Java 메서드	클라우드HSM CLI 하위 명령	키 관리 유틸리티(KMU) 인수
제로 패딩을 사용하는 AES 키 래핑	• CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD (공급업체 정의 메커니즘)	AESWrap/ECB/ZeroPadding	aes-zero-pad	m = 6
패딩이 없는 AES 키 래핑	• CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD (공급업체 정의 메커니즘)	AESWrap/ECB/NoPadding	aes-no-pad	m = 5
PKCS #5 패딩을 사용하는 AES 키 래핑	• CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD (공급업체 정의 메커니즘)	AESWrap/ECB/PKCS5Padding	aes-pkcs5-pad	m = 4

에서 신뢰할 수 있는 키 사용 AWS CloudHSM

AWS CloudHSM 신뢰할 수 있는 키 래핑을 지원하여 내부자 위협으로부터 데이터 키를 보호합니다. 이 주제에서는 신뢰할 수 있는 키를 생성하여 데이터를 보호하는 방법에 대해 설명합니다.

주제

- 신뢰할 수 있는 키 이해
- 신뢰할 수 있는 키 속성

- 신뢰할 수 있는 키를 사용하여 데이터 키를 래핑하는 방법
- 신뢰할 수 있는 키로 데이터 키의 래핑을 해제하는 방법

신뢰할 수 있는 키 이해

신뢰할 수 있는 키는 다른 키를 래핑하는 데 사용되는 키이며, 관리자와 CO(Crypto Officer)는 CKA_TRUSTED 속성을 사용하여 신뢰할 수 있는 것으로 구체적으로 식별합니다. 또한 관리자와 CO(Crypto Officer)는 관련 속성을 사용하여 CKA_UNWRAP_TEMPLATE 신뢰할 수 있는 키로 데이터 키가 래핑을 해제한 후 수행할 수 있는 작업을 지정합니다. 신뢰할 수 있는 키에 의해 래핑 해제된 데이터 키에는 래핑 해제 작업이 성공하기 위한 이러한 속성도 포함되어야 합니다. 이는 래핑 해제된 데이터 키가 의도한 용도로만 허용되도록 하는 데 도움이 됩니다.

CKA_WRAP_WITH_TRUSTED 속성을 사용하여 신뢰할 수 있는 키로 래핑하려는 모든 데이터 키를 식별할 수 있습니다. 이렇게 하면 데이터 키를 제한할 수 있어서 애플리케이션은 신뢰할 수 있는 키만 사용하여 래핑을 해제할 수 있습니다. 데이터 키에 이 속성을 설정하면 속성은 읽기 전용이 되며 변경할 수 없습니다. 이러한 속성을 적용하면 애플리케이션은 신뢰할 수 있는 키로만 데이터 키를 언래핑할 수 있으며, 래핑을 해제하면 항상 이러한 키 사용 방법을 제한하는 속성을 가진 데이터 키가 생성됩니다.

신뢰할 수 있는 키 속성

다음 속성을 사용하면 키를 신뢰할 수 있는 것으로 표시하고, 신뢰할 수 있는 키로만 데이터 키를 래핑 및 래핑 해제할 수 있도록 지정하고, 래핑이 해제된 후 데이터 키가 수행할 수 있는 작업을 제어할 수 있습니다.

- CKA_TRUSTED: 데이터 키를 래핑할 키에 이 속성(CKA_UNWRAP_TEMPLATE와 함께)을 적용하여 관리자 또는 CO(Crypto Officer)가 필요한 조사를 완료하고 이 키를 신뢰하도록 지정합니다. 관리자 또는 CO만 CKA_TRUSTED 설정할 수 있습니다. 암호화 사용자 (CU) 가 키를 소유하지만 CO만 해당 CKA_TRUSTED 속성을 설정할 수 있습니다.
- CKA_WRAP_WITH_TRUSTED: 이 속성을 내보낼 수 있는 데이터 키에 적용하여 이 키를 CKA_TRUSTED로 표시된 키로만 래핑할 수 있도록 지정합니다. CKA_WRAP_WITH_TRUSTED는 true로 설정하면 속성이 읽기 전용이 되며 속성을 변경하거나 제거할 수 없습니다.
- CKA_UNWRAP_TEMPLATE: 이 속성을 래핑 키 (CKA_TRUSTED와 같이) 에 적용하여 서비스가 래핑을 푸는 데이터 키에 자동으로 적용해야 하는 속성 이름과 값을 지정합니다. 애플리케이션이 언래핑을 위해 키를 제출할 때 애플리케이션은 자체 언래핑 템플릿을 제공할 수도 있습니다. 언랩 템플릿을 지정하고 애플리케이션이 자체 언랩 템플릿을 제공하는 경우 HSM은 두개의 템플릿을 모두 사용하여 속성 이름과 값을 키에 적용합니다. 그러나 CKA_UNWRAP_TEMPLATE 래핑 키에 대한 값이 래핑 해제 요청 중에 애플리케이션에서 제공한 속성과 충돌하는 경우 래핑 해제 요청이 실패합니다.

속성에 대한 자세한 내용은 다음 항목을 참조하세요:

- PKCS #11 키 속성
- JCE 키 속성
- CloudHSM CLI 키 속성

신뢰할 수 있는 키를 사용하여 데이터 키를 래핑하는 방법

신뢰할 수 있는 키를 사용하여 데이터 키를 래핑하려면 다음 세 가지 기본 단계를 완료해야 합니다.

1. 신뢰할 수 있는 키로 래핑하려는 데이터 키의 경우 해당 CKA_WRAP_WITH_TRUSTED 속성을 true로 설정합니다.
2. 데이터 키를 래핑하려는 신뢰할 수 있는 키의 경우 해당 CKA_TRUSTED 속성을 true로 설정합니다.
3. 신뢰할 수 있는 키를 사용하여 데이터 키를 래핑합니다.

단계 1: **CKA_WRAP_WITH_TRUSTED** 데이터 키를 true로 설정

래핑하려는 데이터 키의 경우 다음 옵션 중 하나를 선택하여 키의 CKA_WRAP_WITH_TRUSTED 속성을 true로 설정합니다. 이렇게 하면 데이터 키가 제한되므로 애플리케이션은 신뢰할 수 있는 키만 사용하여 데이터를 래핑할 수 있습니다.

옵션 1: 새 키를 생성하는 경우 **CKA_WRAP_WITH_TRUSTED**는 true로 설정합니다.

PKCS #11, JCE 또한 CloudHSM CLI 사용하여 키를 생성합니다. 자세한 내용은 다음 예를 참조하십시오.

PKCS #11

PKCS #11로 키를 생성하려면 키의 CKA_WRAP_WITH_TRUSTED 속성을 true로 설정해야 합니다. 다음 예제와 같이 이 속성을 CK_ATTRIBUTE template 키에 포함시킨 후 속성을 true로 설정하여 이 작업을 수행하십시오.

```
CK_BYTE_PTR label = "test_key";
CK_ATTRIBUTE template[] = {
        {CKA_WRAP_WITH_TRUSTED, &true_val,         sizeof(CK_BBOOL)},
        {CKA_LABEL,             label,             strlen(label)},
        ...
};
```

자세한 내용은 PKCS #11 기반 키 생성을 보여주는 퍼블릭 샘플을 참조하십시오.

JCE

JCE를 사용하여 키를 생성하려면 키 WRAP_WITH_TRUSTED 속성을 true로 설정해야 합니다. 다음 예제와 같이 이 속성을 KeyAttributesMap 키에 포함시킨 후 속성을 true로 설정하여 이 작업을 수행하십시오.

```
final String label = "test_key";
final KeyAttributesMap keySpec = new KeyAttributesMap();
keySpec.put(KeyAttribute.WRAP_WITH_TRUSTED, true);
keySpec.put(KeyAttribute.LABEL, label);
...
```

자세한 내용은 JCE를 사용한 키 생성을 보여주는 퍼블릭 샘플을 참조하십시오.

CloudHSM CLI

CloudHSM CLI를 사용하여 키를 생성하려면 wrap-with-trusted 키의 속성을 true로 설정해야 합니다. 키 생성 명령의 적절한 인수에 wrap-with-trusted=true를 포함시키면 됩니다.

- 대칭 키의 경우 attributes 인수에 wrap-with-trusted를 추가합니다.
- 퍼블릭 키의 경우 public-attributes 인수에 wrap-with-trusted를 추가합니다.
- 프라이빗 키의 경우 private-attributes 인수에 wrap-with-trusted를 추가합니다.

키 페어에 대한 자세한 내용은 키 generate-asymmetric-pair 단원을 참조하십시오.

대칭 키 생성에 대한 자세한 내용은 키 생성-대칭 단원을 참조하십시오.

옵션 2: 기존 키를 사용하는 경우 CloudHSM CLI를 사용하여 해당 **CKA_WRAP_WITH_TRUSTED** 키를 true로 설정합니다.

기존 키의 CKA_WRAP_WITH_TRUSTED 속성을 true로 설정하려면 다음 단계를 따르십시오.

1. login 명령을 사용하여 CU(Crypto User)로 로그인합니다.
2. 키 세트-속성 명령을 사용하여 키 wrap-with-trusted 속성을 true로 설정합니다.

```
aws-cloudhsm > key set-attribute --filter attr.label=test_key --name wrap-with-
trusted --value true
{
  "error_code": 0,
```

```
  "data": {
    "message": "Attribute set successfully"
  }
}
```

단계 2: 신뢰할 수 있는 키의 **CKA_TRUSTED**를 true로 설정합니다.

키를 신뢰할 수 있는 키로 만들려면 해당 CKA_TRUSTED 속성을 true로 설정해야 합니다. CloudHSM CLI 또는 CloudHSM 관리 유틸리티(CMU)를 사용하여 이 작업을 수행할 수 있습니다.

- CloudHSM CLI를 사용하여 키의 CKA_TRUSTED 속성을 설정하는 경우 CloudHSM CLI를 사용하여 키를 신뢰할 수 있는 것으로 표시하는 방법 단원을 참조하십시오.
- CMU를 사용하여 키의 CKA_TRUSTED 속성을 설정하는 경우 CMU에서 키를 신뢰할 수 있는 것으로 표시하는 방법 단원을 참조하십시오.

단계 3. 신뢰할 수 있는 키를 사용하여 데이터 키를 래핑합니다.

단계 1에서 참조한 데이터 키를 단계 2에서 설정한 신뢰할 수 있는 키로 래핑하려면 다음 링크의 코드 샘플을 참조하십시오. 각 그림은 키를 래핑하는 방법을 보여줍니다.

- AWS CloudHSM PKCS #11 예제
- AWS CloudHSM JCE 예제

신뢰할 수 있는 키로 데이터 키의 래핑을 해제하는 방법

데이터 키의 래핑을 해제하려면 CKA_UNWRAP가 있는 신뢰할 수 있는 키를 true로 설정해야 합니다. 이러한 키가 되려면 다음 기준도 충족해야 합니다.

- 키의 CKA_TRUSTED 속성은 true로 설정되어야 합니다.
- 키는 래핑이 해제된 후 데이터 키가 수행할 수 있는 작업을 지정하기 위해 CKA_UNWRAP_TEMPLATE 및 관련 속성을 사용해야 합니다. 예를 들어, 래핑이 해제된 키를 내보낼 수 없도록 하려면 CKA_UNWRAP_TEMPLATE의 일부로 CKA_EXPORTABLE = FALSE를 설정합니다.

> **Note**
>
> CKA_UNWRAP_TEMPLATE는 PKCS #11에서만 사용할 수 있습니다.

애플리케이션이 래핑을 해제할 키를 제출하면 애플리케이션은 자체 래핑 해제 템플릿을 제공할 수도 있습니다. 언랩 템플릿을 지정하고 애플리케이션이 자체 언랩 템플릿을 제공하는 경우 HSM은 두개의 템플릿을 모두 사용하여 속성 이름과 값을 키에 적용합니다. 하지만 래핑 해제 요청 중에 신뢰할 수 있는 키의 CKA_UNWRAP_TEMPLATE은 애플리케이션에서 제공한 속성과 충돌하는 경우 래핑 해제 요청이 실패합니다.

신뢰할 수 있는 키를 사용한 데이터 키 래핑 해제에 대한 예를 보려면 이 PKCS #11 예제를 참조하십시오.

CloudHSM CLI를 사용한 키 관리

최신 SDK 버전 시리즈를 사용하는 경우 CloudHSM CLI를 사용하여 클러스터의 키를 관리하십시오. AWS CloudHSM 자세한 내용은 아래 주제를 참조하십시오.

- 신뢰할 수 있는 키 사용에서는 CloudHSM CLI를 사용하여 데이터를 보호하는 신뢰할 수 있는 키를 생성하는 방법을 설명합니다.
- 키 생성에는 대칭 키, RSA 키 및 EC 키를 비롯한 키 생성 지침이 포함됩니다.
- 키 삭제는 키 소유자가 키를 삭제하는 방법을 설명합니다.
- 키 공유 및 공유 해제에는 키 소유자가 키를 공유하고 공유를 해제하는 방법에 대한 자세한 정보를 제공합니다.
- 키 필터링은 필터를 사용하여 키를 찾는 방법에 대한 지침을 제공합니다.

CloudHSM CLI를 사용하여 키 생성

키를 생성하려면 먼저 CloudHSM CLI를 시작하고 암호화 사용자(crypto user)로 로그인해야 합니다. HSM에서 키를 생성하려면 생성하려는 키 종류에 해당하는 명령을 사용합니다.

주제

- 대칭 키 생성하기
- 비대칭 키 생성하기
- 관련 주제

대칭 키 생성하기

키 생성-대칭 에 나열된 명령을 사용하여 대칭 키를 생성합니다. 사용 가능한 옵션을 모두 확인하려면 help key generate-symmetric 명령을 사용합니다.

AES 키 생성하기

key generate-symmetric aes 명령을 사용하여 AES 키를 생성합니다. 사용 가능한 옵션을 모두 확인하려면 help key generate-symmetric aes 명령을 사용합니다.

Example

다음 예시에서는 32바이트 AES 키를 생성합니다.

```
aws-cloudhsm > key generate-symmetric aes \
    --label aes-example \
    --key-length-bytes 32
```

인수

<LABEL>

AES 키에 대해 사용자 정의 레이블을 지정합니다.

필수 여부: 예

<KEY-LENGTH-BYTES>

키 크기를 바이트 단위로 지정합니다.

유효한 값:

- 16, 24 및 32

필수 여부: 예

<KEY_ATTRIBUTES>

생성된 AES 키에 대해 설정할 키 속성의 공백으로 구분된 목록을 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (예: `token=true`)의 형식으로 지정합니다.

지원되는 AWS CloudHSM 키 속성 목록은 을 참조하십시오CloudHSM CLI의 키 속성.

필수 여부: 아니요

<SESSION>

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다. 다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

세션 키를 영구(토큰) 키로 변경하려면 set-attribute 키를 사용합니다.

기본적으로 키는 생성될 때 지속/토큰 키입니다. <SESSION>을 사용하면 이를 변경하여 이 인수로 생성된 키가 세션/임시 키인지 확인합니다.

필수 여부: 아니요

일반 보안 키 생성하기

key generate-symmetric generic-secret 명령을 사용하여 일반 보안 키를 생성합니다. 사용 가능한 옵션을 모두 확인하려면 help key generate-symmetric generic-secret 명령을 사용합니다.

Example

다음 예시는 32바이트 일반 보안 키를 생성합니다.

```
aws-cloudhsm > key generate-symmetric generic-secret \
    --label generic-secret-example \
    --key-length-bytes 32
```

인수

<LABEL>

일반 보안 키에 대한 사용자 정의 레이블을 지정합니다.

필수 여부: 예

<KEY-LENGTH-BYTES>

키 크기를 바이트 단위로 지정합니다.

유효한 값:

- 1~800

필수 여부: 예

<KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE의 형식(예: token=true)으로 생성된 일반 보안 키에 대해 설정할 키 속성의 공백으로 구분된 목록을 지정합니다.

지원되는 AWS CloudHSM 키 속성 목록은 을 참조하십시오CloudHSM CLI의 키 속성.

필수 여부: 아니요

<SESSION>

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다. 다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

세션 키를 영구(토큰) 키로 변경하려면 set-attribute 키를 사용합니다.

기본적으로 키는 생성될 때 지속/토큰 키입니다. <SESSION>을 사용하면 이를 변경하여 이 인수로 생성된 키가 세션/임시 키인지 확인합니다.

필수 여부: 아니요

비대칭 키 생성하기

키 generate-asymmetric-pair 에 나열된 명령을 사용하여 비대칭 키 쌍을 생성합니다.

RSA 키 생성하기

key generate-asymmetric-pair rsa 명령을 사용하여 RSA 키 페어를 생성합니다. 사용 가능한 옵션을 모두 확인하려면 help key generate-asymmetric-pair rsa 명령을 사용합니다.

Example

다음은 RSA 2048비트 키 페어를 생성하는 예제입니다.

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
    --public-exponent 65537 \
    --modulus-size-bits 2048 \
    --public-label rsa-public-example \
    --private-label rsa-private-example
```

인수

<PUBLIC_LABEL>

퍼블릭 키에 대한 사용자 정의 레이블을 지정합니다.

필수 여부: 예

<PRIVATE_LABEL>

프라이빗 키에 대한 사용자 정의 레이블을 지정합니다.

필수 여부: 예

<MODULUS_SIZE_BITS>

모듈러스 길이를 비트 단위로 지정합니다. 최소값은 2048입니다.

필수 여부: 예

<PUBLIC_EXPONENT>

퍼블릭 지수를 지정합니다. 값은 65537 이상의 홀수여야 합니다.

필수 여부: 예

<PUBLIC_KEY_ATTRIBUTES>

`KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`의 형식(예: `token=true`)으로 생성된 RSA 퍼블릭 키에 대해 설정할 키 속성의 공백으로 구분된 목록을 지정합니다.

지원되는 AWS CloudHSM 키 속성 목록은 을 참조하십시오CloudHSM CLI의 키 속성.

필수 여부: 아니요

<SESSION>

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다. 다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

세션 키를 영구(토큰) 키로 변경하려면 set-attribute 키를 사용합니다.

기본적으로 키는 생성될 때 지속/토큰 키입니다. <SESSION>을 사용하면 이를 변경하여 이 인수로 생성된 키가 세션/임시 키인지 확인합니다.

필수 여부: 아니요

EC(타원 곡선 암호) 키 쌍 생성하기

key generate-asymmetric-pair ec 명령을 사용하여 EC 키 쌍을 생성합니다. 지원되는 타원 곡선의 목록을 포함하여 사용 가능한 옵션을 모두 확인하려면 help key generate-asymmetric-pair ec 명령을 사용합니다.

Example

다음 예시에서는 SecP384R1 타원 곡선을 사용하여 EC 키 쌍을 생성합니다.

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp384r1 \
    --public-label ec-public-example \
    --private-label ec-private-example
```

인수

<PUBLIC_LABEL>

퍼블릭 키에 대한 사용자 정의 레이블을 지정합니다. 클라이언트 SDK 5.11 이상의 경우 허용되는 최대 `label` 크기는 127자입니다. 클라이언트 SDK 5.10 이전 버전은 126자로 제한됩니다.

필수 여부: 예

<PRIVATE_LABEL>

프라이빗 키에 대한 사용자 정의 레이블을 지정합니다. 클라이언트 SDK 5.11 이상의 경우 허용되는 최대 `label` 크기는 127자입니다. 클라이언트 SDK 5.10 이전 버전은 126자로 제한됩니다.

필수 여부: 예

<CURVE>

타원 곡선의 식별자를 지정합니다.

유효한 값:

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1

필수 여부: 예

<PUBLIC_KEY_ATTRIBUTES>

`KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`의 형식(예: `token=true`)으로 생성된 EC 퍼블릭 키에 대해 설정할 키 속성의 공백으로 구분된 목록을 지정합니다.

지원되는 AWS CloudHSM 키 속성 목록은 을 참조하십시오. CloudHSM CLI의 키 속성

필수 여부: 아니요

<PRIVATE_KEY_ATTRIBUTES>

KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE의 형식(예: `token=true`)으로 생성된 EC 프라이빗 키에 대해 설정할 키 속성의 공백으로 구분된 목록을 지정합니다.

지원되는 AWS CloudHSM 키 속성 목록은 을 참조하십시오CloudHSM CLI의 키 속성.

필수 여부: 아니요

<SESSION>

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다. 다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

세션 키를 영구(토큰) 키로 변경하려면 set-attribute 키를 사용합니다.

기본적으로 생성되는 키는 영구(토큰) 키입니다. <SESSION>을 전달하면 이를 변경하여 이 인수로 생성된 키가 세션(임시) 키인지 확인합니다.

필수 여부: 아니요

관련 주제

- CloudHSM CLI의 키 속성
- 키 generate-asymmetric-pair
- 키 생성-대칭

CloudHSM CLI를 사용하여 키 삭제

이 항목의 예제를 사용하여 CloudHSM CLI로 키를 삭제합니다. 키 소유자만 키를 삭제할 수 있습니다.

주제

- 예: 키 삭제
- 관련 주제

예: 키 삭제

1. key list 명령을 실행하여 삭제할 키를 식별합니다.

```
aws-cloudhsm > key list --filter attr.label="my_key_to_delete" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "my_key_to_delete",
          "id": "",
          "check-value": "0x29bbd1",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1217,
          "public-exponent": "0x010001",
```

```
          "modulus":
 "0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
```

2. 키를 식별한 후 키의 고유 `label` 속성을 사용하여 key delete를 실행하여 키를 삭제합니다.

```
aws-cloudhsm > key delete --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

3. 키의 고유 `label` 속성을 사용하여 key list 명령을 실행하고 키가 삭제되었는지 확인합니다. 다음 예제에서 볼 수 있듯이 HSM 클러스터에는 `my_key_to_delete` 레이블이 있는 키가 없습니다.

```
aws-cloudhsm > key list --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "matched_keys": [],
    "total_key_count": 0,
    "returned_key_count": 0
  }
}
```

관련 주제

- CloudHSM CLI의 키 속성
- 키 삭제

CloudHSM CLI를 사용하여 키 공유 및 공유 해제

이 주제의 명령을 사용하여 CloudHSM CLI에서 키를 공유하고 공유를 해제할 수 있습니다. AWS CloudHSM에서는 키를 생성한 암호화폐 사용자 (CU) 가 키를 소유합니다. 소유자는 key share 및 key unshare 명령을 사용하여 다른 CU와 키를 공유하고 공유를 해제할 수 있습니다. 키를 공유하는 사용자는 암호화 작업에서 키를 사용할 수 있지만 키를 내보내고 키를 삭제하거나 다른 사용자와 공유할 수 없습니다.

키를 공유하려면 먼저 HSM에 키를 소유한 암호화 사용자(crypto user)로 로그인해야 합니다.

주제

예시: 키 공유 및 공유 해제

Example

다음 예는 암호 사용자(crypto user) `alice`과 키를 공유하고 공유를 해제하는 방법을 보여줍니다. key share 및 key unshare 명령과 함께, 공유 및 공유 해제 명령에는 CloudHSM CLI 키 필터를 사용하는 특정 키와 키를 공유하거나 공유 해제할 사용자의 특정 사용자 이름도 필요합니다.

1. 먼저 필터로 key list 명령을 실행하여 특정 키를 반환하고 키가 이미 누구와 공유되었는지 확인합니다.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
```

```
            "username": "cu2",
            "key-coverage": "full"
          },
          {
            "username": "cu1",
            "key-coverage": "full"
          },
          {
            "username": "cu4",
            "key-coverage": "full"
          },
          {
            "username": "cu5",
            "key-coverage": "full"
          },
          {
            "username": "cu6",
            "key-coverage": "full"
          },
          {
            "username": "cu7",
            "key-coverage": "full"
          },
        ],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa_key_to_share",
        "id": "",
        "check-value": "0xae8ff0",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
```

```
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

2. shared-users 출력을 보고 현재 키가 공유된 사람을 식별할 수 있습니다.
3. 이 키를 암호화 사용자(crypto user) alice과 공유하려면 다음 명령을 입력합니다.

```
aws-cloudhsm > key share --filter attr.label="rsa_key_to_share" attr.class=private-
key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key shared successfully"
  }
}
```

이 명령은 key share 명령과 함께 키의 고유 레이블과 키를 공유할 사용자의 이름을 사용합니다.

4. key list 명령을 실행하여 alice와 키가 공유되었는지 확인합니다.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
```

```
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
            },
            {
              "username": "alice",
              "key-coverage": "full"
            }
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
```

```
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

5. alice과 동일한 키의 공유를 해제하려면 다음 unshare 명령을 실행합니다.

```
aws-cloudhsm > key unshare --filter attr.label="rsa_key_to_share"
 attr.class=private-key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key unshared successfully"
  }
}
```

이 명령은 key unshare 명령과 함께 키의 고유 레이블과 키를 공유할 사용자의 이름을 사용합니다.

6. key list 명령을 다시 실행하고 키가 암호화 사용자 alice와 공유가 해제되었는지 확인합니다.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
```

```
            },
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

관련 주제

- CloudHSM CLI의 키 속성
- 키 공유
- 키 공유 취소
- CloudHSM CLI를 사용하여 키를 필터링

CloudHSM CLI를 사용하여 키를 필터링

다음 키 명령을 사용하여 CloudHSM CLI의 표준화된 키 필터링 메커니즘을 활용하십시오.

- key list
- key delete
- key share
- key unshare
- key set-attribute

CloudHSM CLI로 키를 선택 및/또는 필터링하기 위해 키 명령은 CloudHSM CLI의 키 속성을 기반으로 하는 표준화된 필터링 메커니즘을 활용합니다. 단일 키 또는 다중 키를 식별할 수 있는 하나 이상의 AWS CloudHSM 속성을 사용하여 키 명령에서 키 또는 키 세트를 지정할 수 있습니다. 키 필터링 메커니즘은 현재 로그인한 사용자가 소유하고 공유하는 키와 AWS CloudHSM 클러스터의 모든 공개 키에서만 작동합니다.

주제

- 요구 사항
- 단일 키를 찾기 위한 필터링
- 필터링 오류
- 관련 주제

요구 사항

키를 필터링하려면 암호화 사용자(crypto user)로 로그인해야 합니다.

단일 키를 찾기 위한 필터링

다음 예시에서는 필터로 사용되는 각 속성을 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`의 형식으로 기록해야 한다는 점에 유의하십시오. 예를 들어, 레이블 속성을 기준으로 필터링하려는 경우 `attr.label=my_label`을 기록합니다.

Example 단일 속성을 사용하여 단일 키를 찾을 수 있습니다.

이 예시에서는 단일 식별 속성만 사용하여 단일 고유 키로 필터링하는 방법을 보여줍니다.

```
aws-cloudhsm > key list --filter attr.label="my_unique_key_label" --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "alice",
              "key-coverage": "full"
            }
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "my_unique_key_label",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
```

```
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254c8f5
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

Example 여러 속성을 사용하여 단일 키를 찾을 수 있습니다.

다음 예시는 여러 키 속성을 사용하여 단일 키를 찾는 방법을 보여줍니다.

```
aws-cloudhsm > key list --filter attr.key-type=rsa attr.class=private-key attr.check-
value=0x29bbd1 --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
```

```
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            }
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "my_crypto_user",
          "id": "",
          "check-value": "0x29bbd1",
          "class": "my_test_key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1217,
          "public-exponent": "0x010001",
          "modulus":
"0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990c2a7
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
```

```
  }
}
```

Example 필터링하여 키 세트 찾기

다음 예시는 프라이빗 rsa 키 세트를 찾기 위해 필터링하는 방법을 보여줍니다.

```
aws-cloudhsm > key list --filter attr.key-type=rsa attr.class=private-key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
```

```
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
"0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254c8f5
          "modulus-size-bits": 2048
        }
      },
      {
        "key-reference": "0x0000000000540011",
        "key-info": {
          "key-owners": [
            {
              "username": "my_crypto_user",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            }
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "my_test_key",
          "id": "",
          "check-value": "0x29bbd1",
          "class": "private-key",
          "encrypt": false,
```

```
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1217,
          "public-exponent": "0x010001",
          "modulus":
 "0x8b3a7c20618e8be08220ed8ab2c8550b65fc1aad8d4cf04fbf2be685f97eeb78fcbbad9b02cd91a3b15e990c2a7
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 2,
    "returned_key_count": 2
  }
}
```

필터링 오류

특정 키 작업은 한 번에 단일 키에서만 수행할 수 있습니다. 이러한 작업의 경우 필터링 기준이 충분히 세분화되지 않고 여러 키가 기준과 매칭하는 경우 CloudHSM CLI에서 오류가 발생합니다. 이러한 예시 중 하나가 키 삭제와 함께 아래에 나와 있습니다.

Example 너무 많은 키를 매칭하는 경우 필터링 오류.

```
aws-cloudhsm > key delete --filter attr.key-type=rsa
{
  "error_code": 1,
  "data": "Key selection criteria matched 48 keys. Refine selection criteria to select
 a single key."
```

```
}
```

관련 주제

- CloudHSM CLI의 키 속성

CloudHSM CLI를 사용하여 키를 신뢰할 수 있는 것으로 표시하는 방법

이 섹션의 내용은 CloudHSM CLI를 사용하여 키를 신뢰할 수 있는 것으로 표시하는 방법에 대한 지침을 제공합니다.

1. CloudHSM CLI login 명령을 사용하여 암호화 사용자(crypto user)로 로그인합니다.
2. key list 명령을 사용하여 신뢰할 수 있는 것으로 표시하려는 키의 키 참조를 식별합니다. 다음 예시에서는 키를 레이블 `key_to_be_trusted`과 함께 나열합니다.

```
aws-cloudhsm > key list --filter attr.label=test_aes_trusted
        {
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000200333",
        "attributes": {
          "label": "test_aes_trusted"
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

3. 로그아웃 명령을 사용하여 암호화 사용자(crypto user)로 로그아웃합니다.
4. login 명령을 사용하여 관리자로 로그인합니다.
5. 2단계에서 식별한 키 참조와 함께 key set-attribute 명령을 사용하여 키의 신뢰할 수 있는 값을 true로 설정합니다.

```
aws-cloudhsm > key set-attribute --filter key-reference=<Key Reference> --name
 trusted --value true
{
```

```
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

KMU 및 CMU를 사용한 키 관리

최신 SDK 버전 시리즈를 사용하는 경우 CloudHSM CLI를 사용하여 클러스터의 키를 관리하십시오. AWS CloudHSM

이전 SDK 버전 시리즈를 사용하는 경우 key_mgmt_util 명령줄 도구를 사용하여 AWS CloudHSM 클러스터에 있는 HSM의 키를 관리할 수 있습니다. 키를 관리하려면 먼저 AWS CloudHSM 클라이언트를 시작하고 key_mgmt_util을 시작한 다음 HSM에 로그인해야 합니다. 자세한 내용은 key_mgmt_util 시작을 참조하십시오.

- 신뢰할 수 있는 키 사용에서는 PKCS #11 라이브러리 속성 및 CMU를 사용하여 신뢰할 수 있는 키를 생성하여 데이터를 보호하는 방법을 설명합니다.
- 키 생성에는 대칭 키, RSA 키 및 EC 키 등 키 생성에 대한 지침이 나와 있습니다.
- 키 가져오기는 키 소유자가 키를 가져오는 방법에 대한 자세한 정보를 제공합니다.
- 키 내보내기는 키 소유자가 키를 내보내는 방법에 대한 자세한 정보를 제공합니다.
- 키 삭제는 키 소유자가 키를 삭제하는 방법에 대한 자세한 정보를 제공합니다.
- 키 공유 및 공유 해제에는 키 소유자가 키를 공유하고 공유를 해제하는 방법에 대한 자세한 정보를 제공합니다.

키 생성

HSM에서 키를 생성하려면 생성하려는 키 종류에 해당하는 명령을 사용합니다.

주제

- 대칭 키 생성하기
- RSA 키 쌍 생성하기
- ECC(타원 곡선 암호) 키 쌍 생성하기

대칭 키 생성하기

genSymKey명령을 사용하여 AES 및 기타 유형의 대칭 키를 생성합니다. 사용 가능한 옵션을 모두 확인하려면 genSymKey -h 명령을 사용합니다.

다음은 256비트 AES 키를 생성하는 예제입니다.

```
Command: genSymKey -t 31 -s 32 -l aes256
CfmGenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created.  Key Handle: 524295

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

RSA 키 쌍 생성하기

RSA 키 쌍을 생성하려면 KeyPairGenRSA 명령을 사용합니다. 사용 가능한 옵션을 모두 확인하려면 genRSAKeyPair -h 명령을 사용합니다.

다음은 RSA 2048비트 키 페어를 생성하는 예제입니다.

```
Command: genRSAKeyPair -m 2048 -e 65537 -l rsa2048
CfmGenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

CfmGenerateKeyPair:    public key handle: 524294    private key handle: 524296

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

ECC(타원 곡선 암호) 키 쌍 생성하기

ECC 키 쌍을 생성하려면 KeyPairGeneCC 명령을 사용합니다. 지원되는 타원 곡선 목록을 포함하여 사용 가능한 옵션을 모두 확인하려면 genECCKeyPair -h 명령을 사용합니다.

다음 예제에서는 NIST FIPS 간행물 186-4에 정의된 P-384 타원 곡선을 사용하여 ECC 키 페어를 생성합니다.

```
Command: genECCKeyPair -i 14 -l ecc-p384
```

```
Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 524297    private key handle: 524298

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

키 가져오기

보안 키, 즉 대칭 키와 비대칭 프라이빗 키를 HSM으로 가져오려면 먼저 HSM에 래핑 키를 생성해야 합니다. 래핑 키 없이 퍼블릭 키를 바로 가져올 수 있습니다.

주제

- 보안 키 가져오기
- 퍼블릭 키 가져오기

보안 키 가져오기

다음 단계를 수행하여 보안 키를 가져옵니다. 보안 키를 가져오려면 파일에 키를 저장합니다. 대칭적 키를 원시 바이트로 저장하고 비대칭적 프라이빗 키를 PEM 형식으로 저장합니다.

이 예제는 평문 보안 키를 파일에서 HSM으로 가져오는 방법을 보여 줍니다. 파일에서 HSM으로 암호화된 키를 가져오려면 명령을 사용합니다. unWrapKey

보안 키를 가져오려면

1. genSymKey명령을 사용하여 래핑 키를 생성합니다. 다음 명령은 현재 세션에 대해서만 유효한 128비트 AES 래핑 키를 생성합니다. 세션 키 또는 영구 키를 래핑 키로 사용할 수 있습니다.

   ```
   Command: genSymKey -t 31 -s 16 -sess -l import-wrapping-key
   Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

   Symmetric Key Created.  Key Handle: 524299

   Cluster Error Status
   Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
   ```

2. 가져오는 보안 키의 유형에 따라 다음 명령 중 하나를 사용합니다.

- 대칭 키를 가져오려면 imSymKey명령을 사용합니다. 다음 명령은 이전 단계에서 생성한 래핑 키를 사용하여 aes256.key라는 파일의 AES 키를 가져옵니다. 사용 가능한 옵션을 모두 확인하려면 imSymKey -h 명령을 사용합니다.

```
Command: imSymKey -f aes256.key -t 31 -l aes256-imported -w 524299
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Unwrapped.  Key Handle: 524300

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

- 비대칭 개인 키를 가져오려면 명령을 사용합니다. importPrivateKey 다음 명령은 이전 단계에서 생성한 래핑 키를 사용하여 rsa2048.key라는 파일의 프라이빗 키를 가져옵니다. 사용 가능한 옵션을 모두 확인하려면 importPrivateKey -h 명령을 사용합니다.

```
Command: importPrivateKey -f rsa2048.key -l rsa2048-imported -w 524299
BER encoded key length is 1216

Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

Private Key Unwrapped.  Key Handle: 524301

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

퍼블릭 키 가져오기

importPubKey명령을 사용하여 공개 키를 가져옵니다. 사용 가능한 옵션을 모두 확인하려면 importPubKey -h 명령을 사용합니다.

다음 예제에서는 `rsa2048.pub`라는 파일에서 RSA 퍼블릭 키를 가져옵니다.

```
Command: importPubKey -f rsa2048.pub -l rsa2048-public-imported
Cfm3CreatePublicKey returned: 0x00 : HSM Return: SUCCESS

Public Key Handle: 524302

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

키 내보내기

보안 키, 즉 대칭 키와 비대칭 프라이빗 키를 HSM에서 내보내려면 먼저 래핑 키를 생성해야 합니다. 래핑 키 없이 퍼블릭 키를 바로 내보낼 수 있습니다.

키 소유자만 키를 내보낼 수 있습니다. 다른 사용자의 키를 공유하는 사용자는 해당 키를 암호화 작업에 사용할 수는 있지만 내보낼 수는 없습니다. 이 예제를 실행할 때 본인이 생성한 키를 내보내야 합니다.

> **Important**
>
> 이 exSymKey명령은 비밀 키의 일반 텍스트 (암호화되지 않은) 사본을 파일에 씁니다. 내보내기 절차에 래핑 키가 필요하지만, 파일의 키는 래핑된 키가 아닙니다. 키의 래핑된(암호화된) 사본을 내보내려면 wrapKey 명령을 사용합니다.

주제

- 보안 키 내보내기
- 퍼블릭 키 내보내기

보안 키 내보내기

다음 단계를 수행하여 보안 키를 내보냅니다.

보안 키를 내보내려면

1. genSymKey명령을 사용하여 래핑 키를 생성합니다. 다음 명령은 현재 세션에 대해서만 유효한 128비트 AES 래핑 키를 생성합니다.

```
Command: genSymKey -t 31 -s 16 -sess -l export-wrapping-key
Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created.  Key Handle: 524304

Cluster Error Status
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

2. 내보내는 보안 키의 유형에 따라 다음 명령 중 하나를 사용합니다.

   - 대칭 키를 내보내려면 exSymKey명령을 사용합니다. 다음 명령은 AES 키를 `aes256.key.exp`라는 파일로 내보냅니다. 사용 가능한 옵션을 모두 확인하려면 exSymKey -h 명령을 사용합니다.

```
Command: exSymKey -k 524295 -out aes256.key.exp -w 524304
Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS


Wrapped Symmetric Key written to file "aes256.key.exp"
```

   > **Note**
   >
   > 명령의 출력은 "Wrapped Symmetric Key"가 출력 파일에 기록되었음을 보여 줍니다. 하지만 출력 파일은 평문(래핑되지 않은) 키를 포함합니다. 래핑된(암호화된) 키를 파일로 내보내려면 wrapKey 명령을 사용합니다.

   - 프라이빗 키를 내보내려면 exportPrivateKey 명령을 사용합니다. 다음 명령은 프라이빗 키를 `rsa2048.key.exp`라는 파일로 내보냅니다. 사용 가능한 옵션을 모두 확인하려면 exportPrivateKey -h 명령을 사용합니다.

```
Command: exportPrivateKey -k 524296 -out rsa2048.key.exp -w 524304
Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS
```

```
Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS

PEM formatted private key is written to rsa2048.key.exp
```

퍼블릭 키 내보내기

exportPubKey 명령을 사용하여 퍼블릭 키를 내보냅니다. 사용 가능한 옵션을 모두 확인하려면 exportPubKey -h 명령을 사용합니다.

다음 예제에서는 `rsa2048.pub.exp`라는 파일로 RSA 퍼블릭 키를 내보냅니다.

```
Command: exportPubKey -k 524294 -out rsa2048.pub.exp
PEM formatted public key is written to rsa2048.pub.key

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

키 삭제

다음 예제에 나온 바와 같이 deleteKey 명령을 사용하여 키를 삭제합니다. 키 소유자만 키를 삭제할 수 있습니다.

```
Command: deleteKey -k 524300
Cfm3DeleteKey returned: 0x00 : HSM Return: SUCCESS

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

키 공유 및 공유 해제

AWS CloudHSM에서는 키를 생성한 CU가 키를 소유합니다. 소유자는 키를 관리하고, 키를 내보내거나 삭제할 수 있고, 암호화 작업에 키를 사용할 수 있습니다. 또한 사용자는 다른 CU 사용자와 키를 공유할 수 있습니다. 다른 사용자의 키를 공유하는 사용자는 해당 키를 암호화 작업에 사용할 수는 있지만 키를 내보내거나 삭제할 수 없고 다른 사용자와 공유할 수 없습니다.

키를 생성할 때 (예: genSymKey또는 GenRSA KeyPair 명령의 -u 파라미터 사용) 다른 CU 사용자와 키를 공유할 수 있습니다. 다른 HSM 사용자와 기존 키를 공유하려면 cloudhsm_mgmt_util 명령줄 도구를 사용합니다. key_mgmt_util 명령줄 도구를 사용하는 이 단원의 대부분의 작업과는 다른 작업입니다.

키를 공유하려면 먼저 cloudhsm_mgmt_util을 시작하고 암호화를 활성화한 다음 HSM에 로그인해야 합니다. end-to-end 키를 공유하려면 키를 소유하는 CU(Crypto User)로 HSM에 로그인합니다. 키 소유자만이 키를 공유할 수 있습니다.

shareKey 명령을 사용하여 키를 공유 또는 공유 해제하고, 키의 핸들과 사용자의 ID를 지정합니다. 한 명 이상의 사용자와 키를 공유 또는 공유 해제하려면 쉼표로 구분된 사용자 ID 목록을 지정합니다. 키를 공유하려면 다음 예제에 나온 바와 같이 1을 명령의 마지막 파라미터로 사용합니다. 공유를 해제하려면 0을 사용합니다.

```
aws-cloudhsm>shareKey 524295 4 1
************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)?y
shareKey success on server 0(10.0.2.9)
shareKey success on server 1(10.0.3.11)
shareKey success on server 2(10.0.1.12)
```

다음은 shareKey 명령의 구문을 보여 줍니다.

```
aws-cloudhsm>shareKey <key handle> <user ID> <Boolean: 1 for share, 0 for unshare>
```

CMU에서 키를 신뢰할 수 있는 것으로 표시하는 방법

이 섹션의 내용은 CMU를 사용하여 키를 신뢰할 수 있는 것으로 표시하는 방법에 대한 지침을 제공합니다.

1. loginHSM 명령을 사용하여 암호화폐 책임자 (CO) 로 로그인합니다.
2. OBJ_ATTR_TRUSTED(값134) 을 true () 로 설정한 상태에서 setAttribute 명령을 사용하십시오. 1

   ```
   setAttribute <Key Handle> 134 1
   ```

복제된 클러스터 관리

원격 리전의 클러스터가 원래 다른 리전의 클러스터 백업에서 생성된 경우, CloudHSM Management Utility(CMU) 를 사용하여 원격 리전의 클러스터를 동기화할 수 있습니다. 클러스터를 다른 리전(대상)에 복사한 다음 나중에 원래 클러스터(소스) 의 변경 내용을 동기화하려고 한다고 가정해 보겠습니다. 이와 같은 시나리오에서는 CMU를 사용하여 클러스터를 동기화합니다. 새 CMU 구성 파일을 만들고 새 파일에 두 클러스터의 HSM(하드웨어 보안 모듈)을 지정한 다음 CMU를 사용하여 해당 파일이 있는 클러스터에 연결하면 됩니다.

복제된 클러스터에서 CMU를 사용하려면

1. 현재 구성 파일의 복사본을 만들고 복사본 이름을 다른 이름으로 변경합니다.

   예를 들어, 다음 파일 위치를 사용하여 현재 구성 파일의 사본을 찾아 만든 다음 복사본의 이름을 `cloudhsm_mgmt_config.cfg`에서 `syncConfig.cfg`(으)로 변경합니다.

   - Linux: `/opt/cloudhsm/etc/cloudhsm_mgmt_config.cfg`
   - Windows: `C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_config.cfg`
2. 이름을 바꾼 사본에 대상 HSM(동기화가 필요한 외부 지역의 HSM) 의 ENI(엘라스틱 네트워크 인터페이스) IP를 추가합니다. 원본 HSM 아래에 대상 HSM을 추가하는 것이 좋습니다.

   ```
   {
       ...
       "servers": [
           {
               ...
               "hostname": "<ENI Source IP>",
               ...
           },
           {
               ...
               "hostname": "<ENI Destination IP>",
               ...
           }
       ]
   }
   ```

   IP 주소를 얻는 방법에 대한 자세한 내용은 the section called “HSM의 IP 주소 가져오기” 단원을 참조하세요.

3. 새 구성 파일을 사용하여 CMU를 초기화합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/userSync.cfg
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM>cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
   \CloudHSM\data\userSync.cfg
   ```

4. 반환된 상태 메시지를 보고 CMU이 필요한 모든 HSM에 연결되었는지 확인하고 반환된 ENI IP 중에서 각 클러스터에 해당하는 것이 어느 것인지 알아봅니다. SyncUser와 SyncKey를 사용하여 사용자와 키를 수동으로 동기화할 수 있습니다. 자세한 내용은 SyncUser 및 SyncKey를 참조하세요.

HSM의 IP 주소 가져오기

이 섹션을 통해 HSM의 IP 주소를 할당받습니다.

HSM의 IP 주소를 가져오려면 (콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. AWS 리전을 변경하려면 페이지의 오른쪽 상단에 있는 리전 선택기를 사용합니다.
3. 클러스터 세부 정보 페이지를 열려면 클러스터 테이블에서 클러스터 ID를 선택합니다.
4. IP 주소를 가져오려면 HSM 탭에서 ENI IP 주소 아래에 나열된 IP 주소 중 하나를 선택합니다.

HSM의 IP 주소를 가져오려면 ()AWS CLI

- AWS CLI의 describe-clusters 명령을 사용하여 HSM의 IP 주소를 가져옵니다. 명령의 출력에서 HSM의 IP 주소는 `EniIp`의 값입니다.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
```

```
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

관련 주제

- syncUser
- syncKey
- 리전 간 백업 복사

AWS CloudHSM 명령줄 도구

이 항목에서는 AWS CloudHSM을 관리하고 사용하는 데 활용할 수 있는 명령줄 도구에 대해 설명합니다.

주제

명령줄 도구 이해

AWS 리소스를 관리하는 데 사용하는 AWS Command Line Interface (AWS CLI) 외에도 HSM에서 HSM 사용자 및 키를 생성하고 관리하기 위한 명령줄 도구를 AWS CloudHSM 제공합니다. 에서는 AWS CloudHSM 익숙한 CLI를 사용하여 클러스터를 관리하고 CloudHSM 명령줄 도구를 사용하여 HSM을 관리합니다.

다음은 다양한 명령줄 도구입니다.

HSM 및 클러스터를 관리하려면

모듈의 CloudHSMv2 명령 및 HSM2 cmdlet AWS CLI PowerShell AWSPowerShell

- 이러한 도구는 클러스터와 HSM을 가져오고, 생성하고, 삭제하고, 태그를 지정합니다. AWS CloudHSM
- CLI에서 CloudHSMv2 명령에 있는 명령을 사용하려면 설치 및 구성해야 합니다. AWS CLI
- 모듈의 HSM2 PowerShell cmdlet은 Windows AWSPowerShell 모듈과 크로스 플랫폼 코어 모듈에서 사용할 수 있습니다. PowerShell PowerShell

HSM 사용자를 관리하려면

CloudHSM CLI

- CloudHSM CLI를 사용하여 사용자 생성, 사용자 삭제, 사용자 나열, 사용자 암호 변경, 사용자 멀티 팩터 인증(MFA)을 업데이트합니다. AWS CloudHSM 클라이언트 소프트웨어에는 포함되어 있지 않습니다. 이 도구 설치에 대한 지침은 CloudHSM CLI 설치 및 구성을 참조하십시오.

Helper 도구

도구 및 소프트웨어 라이브러리를 사용하는 AWS CloudHSM 데 도움이 되는 두 가지 도구는 다음과 같습니다.

- 구성 도구는 CloudHSM 클라이언트 구성 파일을 업데이트합니다. 이렇게 하면 클러스터의 AWS CloudHSM HSM을 동기화할 수 있습니다.

  AWS CloudHSM 두 가지 주요 버전을 제공하며 클라이언트 SDK 5는 최신 버전입니다. 클라이언트 SDK 3(이전 시리즈) 에 비해 다양한 이점을 제공합니다.
- pkpspeed는 소프트웨어 라이브러리와 독립적인 HSM 하드웨어의 성능을 측정합니다.

이전 SDK용 도구

키 관리 도구(KMU)를 사용하여 대칭 키와 비대칭 키 페어를 만들고, 삭제하고, 가져오고, 내보낼 수 있습니다.

- key_mgmt_util. 이 도구는 AWS CloudHSM 클라이언트 소프트웨어에 포함되어 있습니다.

CloudHSM 관리 도구(CMU)를 사용하여 사용자 관리 작업의 쿼럼 인증 구현을 포함하여 HSM 사용자를 생성하고 삭제합니다.

- cloudhsm_mgmt_util. 이 도구는 AWS CloudHSM 클라이언트 소프트웨어에 포함되어 있습니다.

구성 도구

AWS CloudHSM 클러스터의 모든 하드웨어 보안 모듈 (HSM) 간에 데이터를 자동으로 동기화합니다. configure 도구는 동기화 메커니즘이 사용하는 구성 파일의 HSM 데이터를 업데이트합니다. 특히 클러스터의 HSM이 변경된 경우, 명령줄 도구를 사용하기 전에 configure를 사용하여 HSM 데이터를 새로 고치십시오.

AWS CloudHSM 두 가지 주요 클라이언트 SDK 버전이 포함되어 있습니다.

- 클라이언트 SDK 5: 최신 기본 클라이언트 SDK입니다. 제공되는 이점 및 장점에 대한 자세한 내용은 Client SDK 5의 이점 섹션을 참조하세요.
- 클라이언트 SDK 3: 이전 클라이언트 SDK입니다. 여기에는 플랫폼 및 언어 기반 애플리케이션 호환성 및 관리 도구를 위한 전체 구성 요소 세트가 포함됩니다.

클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하는 방법에 대한 지침은 을 참조하십시오. 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션

주제

- 클라이언트 SDK 5 구성 도구
- 클라이언트 SDK 3 구성 도구

클라이언트 SDK 5 구성 도구

클라이언트 SDK 5 구성 도구를 사용하여 클라이언트측 구성 파일을 업데이트하세요.

클라이언트 SDK 5의 각 구성 요소에는 구성 도구의 파일 이름에 구성 요소 표기가 있는 구성 도구가 포함되어 있습니다. 예를 들어 클라이언트 SDK 5용 PKCS #11 라이브러리에는 Linux에는 `configure-pkcs11` 또는 Windows에는 `configure-pkcs11.exe`라는 구성 도구가 포함되어 있습니다.

구문

PKCS #11

```
configure-pkcs11[ .exe ]
              -a <ENI IP address>
              [--hsm-ca-cert <customerCA certificate file path>]
              [--cluster-id <cluster ID>]
              [--endpoint <endpoint>]
              [--region <region>]
              [--server-client-cert-file <client certificate file path>]
              [--server-client-key-file <client key file path>]
              [--log-level <error | warn | info | debug | trace>]
                   Default is <info>
              [--log-rotation <daily | weekly>]
                   Default is <daily>
```

```
             [--log-file <file name with path>]
                  Default is </opt/cloudhsm/run/cloudhsm-pkcs11.log>
                  Default for Windows is <C:\\Program Files\\Amazon\\CloudHSM\
\cloudhsm-pkcs11.log>
             [--log-type <file | term>]
                  Default is <file>
             [-h | --help]
             [-V | --version]
             [--disable-key-availability-check]
             [--enable-key-availability-check]
             [--disable-validate-key-at-init]
             [--enable-validate-key-at-init]
                  This is the default for PKCS #11
```

OpenSSL

```
configure-dyn[ .exe ]
             -a <ENI IP address>
             [--hsm-ca-cert <customerCA certificate file path>]
             [--cluster-id <cluster ID>]
             [--endpoint <endpoint>]
             [--region <region>]
             [--server-client-cert-file <client certificate file path>]
             [--server-client-key-file <client key file path>]
             [--log-level <error | warn | info | debug | trace>]
                  Default is <error>
             [--log-type <file | term>]
                  Default is <term>
             [-h | --help]
             [-V | --version]
             [--disable-key-availability-check]
             [--enable-key-availability-check]
             [--disable-validate-key-at-init]
                  This is the default for OpenSSL
             [--enable-validate-key-at-init]
```

JCE

```
configure-jce[ .exe ]
             -a <ENI IP address>
             [--hsm-ca-cert <customerCA certificate file path>]
             [--cluster-id <cluster ID>]
             [--endpoint <endpoint>]
```

```
            [--region <region>]
            [--server-client-cert-file <client certificate file path>]
            [--server-client-key-file <client key file path>]
            [--log-level <error | warn | info | debug | trace>]
                 Default is <info>
            [--log-rotation <daily | weekly>]
                 Default is <daily>
            [--log-file <file name with path>]
                 Default is </opt/cloudhsm/run/cloudhsm-jce.log>
                 Default for Windows is <C:\\Program Files\\Amazon\\CloudHSM\
\cloudhsm-jce.log>
            [--log-type <file | term>]
                 Default is <file>
            [-h | --help]
            [-V | --version]
            [--disable-key-availability-check]
            [--enable-key-availability-check]
            [--disable-validate-key-at-init]
                 This is the default for JCE
            [--enable-validate-key-at-init]
```

CloudHSM CLI

```
configure-cli[ .exe ]
            -a <ENI IP address>
            [--hsm-ca-cert <customerCA certificate file path>]
            [--cluster-id <cluster ID>]
            [--endpoint <endpoint>]
            [--region <region>]
            [--server-client-cert-file <client certificate file path>]
            [--server-client-key-file <client key file path>]
            [--log-level <error | warn | info | debug | trace>]
                 Default is <info>
            [--log-rotation <daily | weekly>]
                 Default is <daily>
            [--log-file <file name with path>]
                 Default for Linux is </opt/cloudhsm/run/cloudhsm-cli.log>
                 Default for Windows is <C:\\Program Files\\Amazon\\CloudHSM\
\cloudhsm-cli.log>
            [--log-type <file | term>]
                 Default setting is <file>
            [-h | --help]
            [-V | --version]
```

```
            [--disable-key-availability-check]
            [--enable-key-availability-check]
            [--disable-validate-key-at-init]
                 This is the default for CloudHSM CLI
            [--enable-validate-key-at-init]
```

고급 구성

클라이언트 SDK 5 구성 도구와 관련된 고급 구성 목록은 클라이언트 SDK 5 구성 도구의 고급 구성을 참조하세요.

> **Important**
>
> 구성을 변경한 후 애플리케이션을 다시 시작해야 변경 사항이 적용됩니다.

예

다음 예제에서는 클라이언트 SDK 5용 구성 도구를 사용하는 방법을 보여 줍니다.

부트스트래핑 클라이언트 SDK 5

Example

이 예제에서는 `-a` 파라미터를 사용하여 클라이언트 SDK 5 및 HSM 데이터를 업데이트합니다. `-a` 파라미터를 사용하려면 클러스터에 있는 HSM 중 하나의 IP 주소가 있어야 합니다.

PKCS #11 library

Client SDK 5용 Linux EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 -a <HSM IP addresses>
```

Client SDK 5용 Windows EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" -a <HSM IP
 addresses>
```

OpenSSL Dynamic Engine

Client SDK 5용 Linux EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-dyn -a <HSM IP addresses>
```

JCE provider

Client SDK 5용 Linux EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-jce -a <HSM IP addresses>
```

Client SDK 5용 Windows EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" -a <HSM IP addresses>
```

CloudHSM CLI

Client SDK 5용 Linux EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM(s)의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IP addresses of the HSMs>
```

Client SDK 5용 Windows EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM(s)의 IP 주소를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI IP
 addresses of the HSMs>
```

> **Note**
>
> `-a <HSM_IP_ADDRESSES>` 대신 `--cluster-id` 파라미터를 사용할 수 있습니다. `--cluster-id` 사용 요구 사항을 보려면 클라이언트 SDK 5 구성 도구 단원을 참조하십시오.

`-a` 파라미터에 대한 자세한 내용은 the section called “파라미터”을 참조하세요.

클라이언트 SDK 5의 클러스터, 리전, 엔드포인트를 지정하세요.

Example

이 예제에서는 `cluster-id` 파라미터를 사용하여 `DescribeClusters` 호출을 통해 클라이언트 SDK 5를 부트스트랩합니다.

PKCS #11 library

`cluster-id`를 사용하여 클라이언트 SDK 5용 Linux EC2 인스턴스를 부트스트랩하려면

- 클러스터 ID를 사용하여 `cluster-1234567` 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id cluster-1234567
```

cluster-id를 사용하여 클라이언트 SDK 5용 Windows EC2 인스턴스를 부트스트랩하려면

- 클러스터 ID를 사용하여 cluster-1234567 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\configure-pkcs11.exe" --cluster-
id cluster-1234567
```

OpenSSL Dynamic Engine

cluster-id를 사용하여 클라이언트 SDK 5용 Linux EC2 인스턴스를 부트스트랩하려면

- 클러스터 ID를 사용하여 cluster-1234567 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id cluster-1234567
```

JCE provider

cluster-id를 사용하여 클라이언트 SDK 5용 Linux EC2 인스턴스를 부트스트랩하려면

- 클러스터 ID를 사용하여 cluster-1234567 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id cluster-1234567
```

cluster-id를 사용하여 클라이언트 SDK 5용 Windows EC2 인스턴스를 부트스트랩하려면

- 클러스터 ID를 사용하여 cluster-1234567 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\configure-jce.exe" --cluster-
id cluster-1234567
```

CloudHSM CLI

cluster-id를 사용하여 클라이언트 SDK 5용 Linux EC2 인스턴스를 부트스트랩하려면

- 클러스터 ID를 사용하여 cluster-1234567 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id cluster-1234567
```

cluster-id를 사용하여 클라이언트 SDK 5용 Windows EC2 인스턴스를 부트스트랩하려면

- 클러스터 ID를 사용하여 cluster-1234567 클러스터에 있는 HSM의 IP 주소를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --cluster-
id cluster-1234567
```

--region 및 --endpoint 파라미터를 cluster-id 파라미터와 함께 사용하여 시스템에서 DescribeClusters를 호출하는 방식을 지정할 수 있습니다. 예를 들어 클러스터의 리전이 AWS CLI 기본값으로 구성된 리전과 다른 경우 --region 파라미터를 사용하여 해당 리전을 사용해야 합니다. 또한 호출에 사용할 AWS CloudHSM API 엔드포인트를 지정할 수 있습니다. 이는 기본 DNS 호스트 이름을 사용하지 않는 VPC 인터페이스 엔드포인트를 사용하는 등 다양한 네트워크 설정에 필요할 수 있습니다. AWS CloudHSM

PKCS #11 library

사용자 지정 엔드포인트 및 리전을 사용하여 Linux EC2 인스턴스를 부트스트래핑하려면

- 구성 도구를 사용하여 클러스터 내 HSM의 IP 주소를 사용자 지정 지역 및 엔드포인트로 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id cluster-1234567 --
region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

엔드포인트 및 리전을 사용하여 Windows EC2 인스턴스를 부트스트래핑하려면

- 구성 도구를 사용하여 클러스터 내 HSM의 IP 주소를 사용자 지정 지역 및 엔드포인트로 지정합니다.

```
C:\Program Files\Amazon\CloudHSM\configure-pkcs11.exe --cluster-
id cluster-1234567--region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

OpenSSL Dynamic Engine

사용자 지정 엔드포인트 및 리전을 사용하여 Linux EC2 인스턴스를 부트스트래핑하려면

- 구성 도구를 사용하여 클러스터 내 HSM의 IP 주소를 사용자 지정 지역 및 엔드포인트로 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id cluster-1234567 --region us-
east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

JCE provider

사용자 지정 엔드포인트 및 리전을 사용하여 Linux EC2 인스턴스를 부트스트래핑하려면

- 구성 도구를 사용하여 클러스터 내 HSM의 IP 주소를 사용자 지정 지역 및 엔드포인트로 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id cluster-1234567 --region us-
east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

엔드포인트 및 리전을 사용하여 Windows EC2 인스턴스를 부트스트래핑하려면

- 구성 도구를 사용하여 클러스터 내 HSM의 IP 주소를 사용자 지정 지역 및 엔드포인트로 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\configure-jce.exe" --cluster-
id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

CloudHSM CLI

사용자 지정 엔드포인트 및 리전을 사용하여 Linux EC2 인스턴스를 부트스트래핑하려면

- 구성 도구를 사용하여 클러스터 내 HSM의 IP 주소를 사용자 지정 지역 및 엔드포인트로 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id cluster-1234567 --region us-
east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

엔드포인트 및 리전을 사용하여 Windows EC2 인스턴스를 부트스트래핑하려면

- 구성 도구를 사용하여 클러스터 내 HSM의 IP 주소를 사용자 지정 지역 및 엔드포인트로 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\configure-cli.exe" --cluster-
id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

`--cluster-id`, `--region`, `--endpoint` 파라미터에 대한 자세한 내용은 the section called “파라미터” 섹션을 참조하십시오.

TLS 클라이언트-서버 상호 인증을 위한 클라이언트 인증서 및 키 업데이트

Example

이 예제에서는 사용자 지정 키와 SSL 인증서를 지정하여 `server-client-cert-file` 및 `--server-client-key-file` 매개변수를 사용하여 SSL을 재구성하는 방법을 보여줍니다. AWS CloudHSM

PKCS #11 library

Linux용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증을 위한 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. 구성 도구를 사용하여 `ssl-client.crt` 및 `ssl-client.key`을 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Windows용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증에 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. PowerShell 인터프리터의 경우 구성 도구를 사용하여 및 을 지정합니다`ssl-client.crt`. `ssl-client.key`

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
```

```
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

OpenSSL Dynamic Engine

Linux용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증을 위한 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. 구성 도구를 사용하여 `ssl-client.crt` 및 `ssl-client.key`을 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-dyn \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

JCE provider

Linux용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증을 위한 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. 구성 도구를 사용하여 `ssl-client.crt` 및 `ssl-client.key`을 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-jce \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Windows용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증에 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. PowerShell 인터프리터의 경우 구성 도구를 사용하여 및 을 지정합니다`ssl-client.crt`. `ssl-client.key`

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

CloudHSM CLI

Linux용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증을 위한 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
$ sudo cp ssl-client.crt /opt/cloudhsm/etc
sudo cp ssl-client.key /opt/cloudhsm/etc
```

2. 구성 도구를 사용하여 `ssl-client.crt` 및 `ssl-client.key`을 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli \
            --server-client-cert-file /opt/cloudhsm/etc/ssl-client.crt \
            --server-client-key-file /opt/cloudhsm/etc/ssl-client.key
```

Windows용 클라이언트 SDK 5를 사용하여 TLS 클라이언트-서버 상호 인증에 사용자 지정 인증서 및 키를 사용하려면

1. 키와 인증서를 적절한 디렉터리로 복사합니다.

```
cp ssl-client.crt C:\ProgramData\Amazon\CloudHSM\ssl-client.crt
```

```
cp ssl-client.key C:\ProgramData\Amazon\CloudHSM\ssl-client.key
```

2. PowerShell 인터프리터의 경우 구성 도구를 사용하여 및 을 지정합니다`ssl-client.crt`. `ssl-client.key`

```
& "C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" `
            --server-client-cert-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.crt `
            --server-client-key-file C:\ProgramData\Amazon\CloudHSM\ssl-
client.key
```

`server-client-cert-file` 및 `--server-client-key-file` 파라미터에 대한 자세한 내용은 the section called “파라미터” 단원을 참조하세요.

클라이언트 키 내구성 설정 비활성화

Example

이 예제에서는 `--disable-key-availability-check` 파라미터를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다. 단일 HSM으로 클러스터를 실행하려면 클라이언트 키 내구성 설정을 비활성화해야 합니다.

PKCS #11 library

Linux용 Client SDK 5의 클라이언트 키 내구성 비활성화

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Windows용 Client SDK 5의 클라이언트 키 내구성을 사용하지 않도록 설정

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-
availability-check
```

OpenSSL Dynamic Engine

Linux용 Client SDK 5의 클라이언트 키 내구성 비활성화

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

JCE provider

Linux용 Client SDK 5의 클라이언트 키 내구성 비활성화

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Windows용 Client SDK 5의 클라이언트 키 내구성을 사용하지 않도록 설정

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-
availability-check
```

CloudHSM CLI

Linux용 Client SDK 5의 클라이언트 키 내구성 비활성화

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Windows용 Client SDK 5의 클라이언트 키 내구성을 사용하지 않도록 설정

- 구성 도구를 사용하여 클라이언트 키 내구성 설정을 비활성화합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-
availability-check
```

`--disable-key-availability-check` 파라미터에 대한 자세한 내용은 the section called “파라미터”을 참조하세요.

로깅 옵션 관리

Example

클라이언트 SDK 5는`log-file`, `log-level`, `log-rotation`, `log-type` 파라미터를 사용하여 로깅을 관리합니다.

> **Note**
>
> AWS Fargate 또는 AWS Lambda와 같은 서버리스 환경에 맞게 SDK를 구성하려면 로그 유형을 로 구성하는 것이 좋습니다. AWS CloudHSM `term` 클라이언트 로그는 해당 환경에 대해 구성된 로그 CloudWatch 로그 그룹에 `stderr` 출력되고 캡처됩니다.

PKCS #11 library

기본 로깅 위치

- 파일 위치를 지정하지 않으면 시스템은 다음 기본 위치에 로그를 기록합니다.

  Linux

```
/opt/cloudhsm/run/cloudhsm-pkcs11.log
```

  Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
```

로깅 수준을 구성하고 다른 로깅 옵션은 기본값으로 설정하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-level info
  ```

파일 로깅 옵션을 구성하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type file --log-file <file name
   with path> --log-rotation daily --log-level info
  ```

터미널 로깅 옵션을 구성하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type term --log-level info
  ```

OpenSSL Dynamic Engine

기본 로깅 위치

- 파일 위치를 지정하지 않으면 시스템은 다음 기본 위치에 로그를 기록합니다.

  Linux

  ```
  stderr
  ```

로깅 수준을 구성하고 다른 로깅 옵션은 기본값으로 설정하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-level info
  ```

파일 로깅 옵션을 구성하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type <file name> --log-file file --
  log-rotation daily --log-level info
  ```
  ```

터미널 로깅 옵션을 구성하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type term --log-level info
  ```

JCE provider

기본 로깅 위치

- 파일 위치를 지정하지 않으면 시스템은 다음 기본 위치에 로그를 기록합니다.

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-jce.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
  ```

로깅 수준을 구성하고 다른 로깅 옵션은 기본값으로 설정하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-level info
  ```

파일 로깅 옵션을 구성하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type file --log-file <file name> --
  log-rotation daily --log-level info
  ```

터미널 로깅 옵션을 구성하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type term --log-level info
  ```

CloudHSM CLI

기본 로깅 위치

- 파일 위치를 지정하지 않으면 시스템은 다음 기본 위치에 로그를 기록합니다.

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-cli.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-cli.log
  ```

로깅 수준을 구성하고 다른 로깅 옵션은 기본값으로 설정하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-level info
  ```

파일 로깅 옵션을 구성하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type file --log-file <file name> --
  log-rotation daily --log-level info
  ```

터미널 로깅 옵션을 구성하려면

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type term --log-level info
  ```

`log-file`, `log-level`, `log-rotation`, `log-type` 파라미터에 대한 자세한 내용은 the section called “파라미터” 단원을 참조하세요.

클라이언트 SDK 5용 발급 인증서 배치

Example

이 예제에서는 `--hsm-ca-cert` 파라미터를 사용하여 클라이언트 SDK 5의 발급 인증서 위치를 업데이트합니다.
  ```

PKCS #11 library

Linux용 Client SDK 5에 발급 인증서를 배치하려면

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --hsm-ca-cert <customerCA certificate
 file>
```

Windows용 Client SDK 5에 발급 인증서 배치

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\configure-pkcs11.exe" --hsm-ca-
cert <customerCA certificate file>
```

OpenSSL Dynamic Engine

Linux용 Client SDK 5에 발급 인증서를 배치하려면

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --hsm-ca-cert <customerCA certificate
 file>
```

JCE provider

Linux용 Client SDK 5에 발급 인증서를 배치하려면

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-jce --hsm-ca-cert <customerCA certificate
 file>
```

Windows용 Client SDK 5에 발급 인증서 배치

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\configure-jce.exe" --hsm-ca-cert <customerCA
 certificate file>
```

CloudHSM CLI

Linux용 Client SDK 5에 발급 인증서를 배치하려면

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli --hsm-ca-cert <customerCA certificate
 file>
```

Windows용 Client SDK 5에 발급 인증서 배치

- 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\configure-cli.exe" --hsm-ca-cert <customerCA
 certificate file>
```

`--hsm-ca-cert` 파라미터에 대한 자세한 내용은 the section called “파라미터” 단원을 참조하세요.

파라미터

-a ***<ENI IP address>***

지정된 IP 주소를 클라이언트 SDK 5 구성 파일에 추가합니다. 클러스터에 있는 HSM의 모든 ENI IP 주소를 입력합니다. 이 옵션을 사용하는 방법에 관한 자세한 내용은 클라이언트 SDK 5 부트스트랩을 참조하세요.

필수 여부: 예

-- hsm-ca-cert <customerCA certificate file path>

EC2 클라이언트 인스턴스를 클러스터에 연결하는 데 사용되는 CA(인증 기관) 인증서를 저장하는 디렉터리 경로입니다. 클러스터를 초기화할 때 이 파일을 생성합니다. 기본적으로 시스템은 다음 위치에서 이 파일을 찾습니다.

Linux

```
/opt/cloudhsm/etc/customerCA.crt
```

Windows

```
C:\ProgramData\Amazon\CloudHSM\customerCA.crt
```

클러스터를 초기화하거나 인증서를 배치하는 방법에 대한 자세한 내용은 ??? 및 ??? 단원을 참조하세요.

필수 여부: 아니요

--cluster-id ***<cluster ID>***

클러스터 ID와 연결된 클러스터의 모든 HSM Elastic Network 인터페이스(ENI) IP 주소를 찾기 위해 `DescribeClusters`를 호출합니다. 시스템이 ENI IP 주소를 AWS CloudHSM 구성 파일에 추가합니다.

> **Note**
>
> 퍼블릭 인터넷에 액세스할 수 없는 VPC 내 EC2 인스턴스의 `--cluster-id` 파라미터를 사용하는 경우 연결할 인터페이스 VPC 엔드포인트를 만들어야 합니다. AWS CloudHSM VPC 엔드포인트에 대한 자세한 내용은 ??? 섹션을 참조하십시오.

필수 여부: 아니요

--endpoint ***<endpoint>***

AWS CloudHSM 호출에 사용되는 API 엔드포인트를 지정합니다. `DescribeClusters` 이 옵션은 `--cluster-id`와 조합하여 설정해야 합니다.

필수 여부: 아니요

--region ***<region>***

클러스터의 지역을 지정합니다. 이 옵션은 `--cluster-id`와 조합하여 설정해야 합니다.

`--region` 파라미터를 제공하지 않으면 시스템은 `AWS_DEFAULT_REGION` 또는 `AWS_REGION` 환경 변수를 읽으려고 시도하여 리전을 선택합니다. 이러한 변수가 설정되지 않은 경우, 시스템은 사용자가 `AWS_CONFIG_FILE` 환경 변수에 다른 파일을 지정하지 않는 한 AWS config 파일(일반적으로 `~/.aws/config`)의 프로필과 연결된 리전을 확인합니다. 위 항목 중 아무 것도 설정되지 않은 경우 시스템은 `us-east-1` 리전을 기본값으로 사용합니다.

필수 여부: 아니요

-- server-client-cert-file <client certificate file path>

TLS 클라이언트-서버 상호 인증에 사용되는 클라이언트 인증서의 경로입니다.

클라이언트 SDK 5에 포함된 기본 키 및 SSL/TLS 인증서를 사용하지 않으려는 경우에만 이 옵션을 사용하세요. 이 옵션은 `--server-client-key-file`와 조합하여 설정해야 합니다.

필수 여부: 아니요

-- server-client-key-file <client key file path>

TLS 클라이언트-서버 상호 인증에 사용되는 클라이언트 키의 경로.

클라이언트 SDK 5에 포함된 기본 키 및 SSL/TLS 인증서를 사용하지 않으려는 경우에만 이 옵션을 사용하세요. 이 옵션은 `--server-client-cert-file`와 조합하여 설정해야 합니다.

필수 여부: 아니요

--log-level ***<error | warn | info | debug | trace>***

시스템이 로그 파일에 기록해야 하는 최소 로깅 수준을 지정합니다. 각 수준에는 이전 수준이 포함되며, 오류는 최소 수준이고 최대 수준은 추적됩니다. 즉, 오류를 지정하는 경우 시스템은 오류만 로

그에 기록합니다. 추적을 지정하면 시스템에서 오류, 경고, 정보(정보) 및 디버그 메시지를 로그에 기록합니다. 자세한 내용은 클라이언트 SDK 5 로깅 단원을 참조하세요.

필수 여부: 아니요

--log-rotation ***<daily | weekly>***

시스템에서 로그를 순환하는 빈도를 지정합니다. 자세한 내용은 클라이언트 SDK 5 로깅 단원을 참조하세요.

필수 여부: 아니요

--log-file ***<file name with path>***

시스템에서 로그 파일을 기록할 위치를 지정합니다. 자세한 내용은 클라이언트 SDK 5 로깅 단원을 참조하세요.

필수 여부: 아니요

--log-type ***<term | file>***

시스템에서 로그를 파일 또는 터미널에 기록할지 여부를 지정합니다. 자세한 내용은 클라이언트 SDK 5 로깅 단원을 참조하세요.

필수 여부: 아니요

-h | --help

도움말을 표시합니다.

필수 여부: 아니요

-v | --version

버전을 표시합니다.

필수 여부: 아니요

--disable-key-availability-check

키 가용성 쿼럼을 비활성화하는 플래그입니다. 이 플래그를 사용하여 키 가용성 쿼럼을 AWS CloudHSM 비활성화해야 함을 나타내며 클러스터의 한 HSM에만 있는 키를 사용할 수 있습니다. 이 플래그를 사용하여 키 가용성 쿼럼을 설정하는 방법에 대한 자세한 내용은 ??? 단원을 참조하세요.

필수 여부: 아니요

--enable-key-availability-check

키 가용성 쿼럼을 활성화하는 플래그 이 플래그를 사용하여 키 가용성 쿼럼을 AWS CloudHSM 사용해야 하고 해당 키가 클러스터의 두 HSM에 존재할 때까지 키 사용을 허용하지 않도록 지정하십시오. 이 플래그를 사용하여 키 가용성 쿼럼을 설정하는 방법에 대한 자세한 내용은 ??? 단원을 참조하세요.

기본적으로 활성화됩니다.

필수 여부: 아니요

-- -init disable-validate-key-at

초기화 호출을 건너뛰고 후속 호출 시 키에 대한 권한을 확인할 수 있도록 지정하여 성능을 개선합니다. 주의해서 사용하세요.

배경: PKCS #11 라이브러리의 일부 메커니즘은 초기화 호출에서 후속 호출에 키를 사용할 수 있는지 확인하는 멀티파트 작업을 지원합니다. 이를 위해서는 HSM에 대한 확인 호출이 필요하며, 이로 인해 전체 작업에 지연 시간이 늘어납니다. 이 옵션을 사용하면 후속 호출을 비활성화하고 잠재적으로 성능을 개선할 수 있습니다.

필수 여부: 아니요

-- -초기화 enable-validate-key-at

초기화 호출을 사용하여 후속 호출 시 키에 대한 권한을 확인해야 한다고 지정합니다. 이는 기본 옵션입니다. `enable-validate-key-at-init`를 사용하여 초기화 호출을 일시 중단한 후 다시 시작하는 데 `disable-validate-key-at-init`를 사용합니다.

필수 여부: 아니요

관련 주제

- DescribeClusters API 연산
- describe-clusters AWS CLI
- Get-HSM2Cluster PowerShell cmdlet
- 클라이언트 SDK 5 부트스트랩
- AWS CloudHSM VPC 엔드포인트
- 클라이언트 SDK 5 키 내구성 설정 관리

- 클라이언트 SDK 5 로깅

클라이언트 SDK 5 구성 도구의 고급 구성

클라이언트 SDK 5 구성 도구에는 대부분의 고객이 활용하는 일반 기능에 속하지 않는 고급 구성이 포함되어 있습니다. 고급 구성은 추가 기능을 제공합니다.

- PKCS #11의 고급 구성
  - PKCS #11를 사용하여 여러 슬롯에 연결
  - PKCS #11 재시도 명령
- JCE용 고급 구성
  - JCE 공급자를 통해 여러 클러스터에 연결
  - JCE 재시도 명령
  - JCE를 사용한 키 추출
- OpenSSL용 고급 구성
  - OpenSSL용 재시도 명령
- AWS CloudHSM 명령줄 인터페이스 (CLI) 의 고급 구성
  - CloudHSM CLI를 사용하여 여러 클러스터에 연결

클라이언트 SDK 3 구성 도구

클라이언트 SDK 3 구성 도구를 사용하여 클라이언트 데몬을 부트스트래핑하고 CloudHSM 관리 유틸리티를 구성합니다.

구문

```
configure -h | --help
          -a <ENI IP address>
          -m [-i <daemon_id>]
          --ssl --pkey <private key file> --cert <certificate file>
          --cmu <ENI IP address>
```

예

이러한 예제는 configure 도구를 사용하는 방법을 보여 줍니다.

Example : 클라이언트 및 key_mgmt_util의 HSM 데이터를 업데이트합니다 AWS CloudHSM .

이 예제에서는 의 -a 파라미터를 사용하여 클라이언트와 key_mgmt_util의 configure HSM 데이터를 업데이트합니다. AWS CloudHSM -a 파라미터를 사용하려면 클러스터에 있는 HSM 중 하나의 IP 주소가 있어야 합니다. 콘솔 또는 AWS CLI를 사용하여 IP 주소를 가져오세요.

HSM의 IP 주소를 가져오려면 (콘솔)

1. https://console.aws.amazon.com/cloudhsm/home 에서 AWS CloudHSM 콘솔을 엽니다.
2. AWS 리전을 변경하려면 페이지의 오른쪽 상단에 있는 리전 선택기를 사용합니다.
3. 클러스터 세부 정보 페이지를 열려면 클러스터 테이블에서 클러스터 ID를 선택합니다.
4. IP 주소를 가져오려면 HSM 탭에서 ENI IP 주소 아래에 나열된 IP 주소 중 하나를 선택합니다.

HSM의 IP 주소를 가져오려면 ()AWS CLI

- AWS CLI의 describe-clusters 명령을 사용하여 HSM의 IP 주소를 가져옵니다. 명령의 출력에서 HSM의 IP 주소는 `EniIp`의 값입니다.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
            "Hsms": [
                {
...
                    "EniIp": "10.0.0.9",
...
                },
                {
...
                    "EniIp": "10.0.1.6",
...
```

HSM 데이터를 업데이트하려면

1. -a파라미터를 업데이트하기 전에 AWS CloudHSM 클라이언트를 중지하십시오. 이렇게 하면 configure가 클라이언트의 구성 파일을 편집하는 동안 발생할 수 있는 충돌이 방지됩니다. 클라이언트가 이미 중지된 경우, 이 명령은 아무런 악영향이 없으므로 스크립트에서 사용할 수 있습니다.

   Amazon Linux

   ```
   $ sudo stop cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client stop
   ```

   CentOS 7

   ```
   $ sudo service cloudhsm-client stop
   ```

   CentOS 8

   ```
   $ sudo service cloudhsm-client stop
   ```

   RHEL 7

   ```
   $ sudo service cloudhsm-client stop
   ```

   RHEL 8

   ```
   $ sudo service cloudhsm-client stop
   ```

   Ubuntu 16.04 LTS

   ```
   $ sudo service cloudhsm-client stop
   ```

   Ubuntu 18.04 LTS

   ```
   $ sudo service cloudhsm-client stop
   ```

Windows

- Windows 클라이언트 1.1.2+의 경우:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Windows 클라이언트 1.1.1 이상의 경우:

  클라이언트를 시작한 명령 창에서 Ctrl + C를 사용합니다. AWS CloudHSM

2. 이 단계에서는 configure의 `-a` 파라미터를 사용하여 `10.0.0.9` ENI IP 주소를 구성 파일에 추가합니다.

Amazon Linux

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Amazon Linux 2

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

CentOS 7

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

CentOS 8

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

RHEL 7

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

RHEL 8

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Ubuntu 16.04 LTS

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Ubuntu 18.04 LTS

```
$ sudo /opt/cloudhsm/bin/configure -a 10.0.0.9
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\ configure.exe -a 10.0.0.9
```

3. 그런 다음 클라이언트를 다시 시작합니다. AWS CloudHSM 클라이언트를 시작할 때 구성 파일의 ENI IP 주소를 사용하여 클러스터를 쿼리합니다. 그런 다음 클러스터에 있는 모든 HSM의 ENI IP 주소를 `cluster.info` 파일에 씁니다.

   Amazon Linux

   ```
   $ sudo start cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client start
   ```

   CentOS 7

   ```
   $ sudo service cloudhsm-client start
   ```

   CentOS 8

   ```
   $ sudo service cloudhsm-client start
   ```

   RHEL 7

   ```
   $ sudo service cloudhsm-client start
   ```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Windows 클라이언트 1.1.2+의 경우:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows 클라이언트 1.1.1 이상의 경우:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

명령이 완료되면 AWS CloudHSM 클라이언트와 key_mgmt_util이 사용하는 HSM 데이터가 완전하고 정확합니다.

Example : 클라이언트 SDK 3.2.1 및 이전 버전에서 CMU용 HSM 데이터 업데이트

이 예제에서는 `-m` configure 명령을 사용하여 업데이트된 HSM 데이터의 사본을 `cluster.info` 파일에서 cloudhsm_mgmt_util이 사용하는 `cloudhsm_mgmt_util.cfg` 파일로 복사합니다. 클라이언트 SDK 3.2.1 및 이전 버전과 함께 제공되는 CMU와 함께 사용할 수 있습니다.

- 를 `-m` 실행하기 전에 이전 예와 같이 AWS CloudHSM 클라이언트를 중지하고 `-a` 명령을 실행한 다음 클라이언트를 다시 시작합니다. AWS CloudHSM 이렇게 하면 `cluster.info` 파일에서 `cloudhsm_mgmt_util.cfg` 파일로 복사되는 데이터가 완전하고 정확해집니다.

Linux

```
$ sudo /opt/cloudhsm/bin/configure -m
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\ configure.exe -m
```

Example : 클라이언트 SDK 3.3.0 이상에서 CMU용 HSM 데이터 업데이트

이 예제에서는 configure 명령의 `--cmu` 파라미터를 사용하여 CMU용 HSM 데이터를 업데이트합니다. 클라이언트 SDK 3.3.0 이상과 함께 제공되는 CMU와 함께 이 기능을 사용하세요. CMU 사용에 대한 자세한 내용은 CloudHSM 관리 유틸리티(CMU) 를 사용하여 사용자 관리 및 클라이언트 SDK 3.2.1 및 이전 버전과 함께 CMU 사용을 참조하세요.

- `--cmu` 파라미터를 사용하여 클러스터에 있는 HSM의 IP 주소를 전달할 수 있습니다.

  Linux

  ```
  $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
  ```

파라미터

-h | --help

명령 구문을 표시합니다.

필수 여부: 예

-a *<ENI IP address>*

지정된 HSM ENI(탄력적 네트워크 인터페이스) IP 주소를 AWS CloudHSM 구성 파일에 추가합니다. 클러스터에 있는 HSM 중 하나의 ENI IP 주소를 입력합니다. 어떤 HSM을 선택해도 상관이 없습니다.

클러스터에 있는 HSM의 ENI IP 주소를 가져오려면 DescribeClusters작업, describe-cluster AWS CLI 명령 또는 cmdlet을 사용하십시오. Get-HSM2Cluster PowerShell

> **Note**
>
> 명령을 실행하기 전에 클라이언트를 중지하십시오. -a configure AWS CloudHSM 그런 다음 -a 명령이 완료되면 AWS CloudHSM 클라이언트를 다시 시작합니다. 자세한 내용은 예제를 참조하세요.

이 파라미터는 다음 구성 파일을 편집합니다.

- `/opt/cloudhsm/etc/cloudhsm_client.cfg`: AWS CloudHSM 클라이언트와 key_mgmt_util에서 사용합니다.
- `/opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg`: cloudhsm_mgmt_util에서 사용합니다.

AWS CloudHSM 클라이언트가 시작되면 구성 파일의 ENI IP 주소를 사용하여 클러스터를 쿼리하고 클러스터의 모든 HSM에 대한 올바른 ENI IP 주소로 `cluster.info` 파일 (`/opt/cloudhsm/daemon/1/cluster.info`) 을 업데이트합니다.

필수 여부: 예

-m

CMU가 사용하는 구성 파일의 HSM ENI IP 주소를 업데이트합니다.

> **Note**
>
> -m 파라미터는 클라이언트 SDK 3.2.1 및 이전 버전의 CMU와 함께 사용하기 위한 것입니다. 클라이언트 SDK 3.3.0 이상의 CMU에 대해서는 CMU의 HSM 데이터 업데이트 프로세스를 간소화하는 --cmu 파라미터를 참조하세요.

의 -a configure 매개변수를 업데이트한 다음 클라이언트를 시작하면 AWS CloudHSM 클라이언트 데몬이 클러스터를 쿼리하고 클러스터의 모든 HSM에 대해 올바른 HSM IP 주소로 `cluster.info` 파일을 업데이트합니다. -m configure 명령을 실행하면 `cluster.info`에서 cloudhsm_mgmt_util이 사용하는 `cloudhsm_mgmt_util.cfg` 구성 파일로 HSM IP 주소를 복사하여 업데이트가 완료됩니다.

-aconfigure명령을 실행하기 전에 명령을 실행하고 AWS CloudHSM 클라이언트를 다시 시작해야 합니다. -m 이렇게 하면 cluster.info에서 cloudhsm_mgmt_util.cfg로 복사되는 데이터가 완전하고 정확해집니다.

필수 여부: 예

-i

대체 클라이언트 데몬을 지정합니다. 기본값은 AWS CloudHSM 클라이언트를 나타냅니다.

기본값: 1

필수 여부: 아니요

--ssl

클러스터의 SSL 키와 인증서를 지정된 프라이빗 키와 인증서로 대체합니다. 이 파라미터를 사용할 경우 --pkey 및 --cert 파라미터가 필요합니다.

필수 여부: 아니요

--pkey

새 프라이빗 키를 지정합니다. 프라이빗 키를 포함하는 파일의 경로 및 파일 이름을 입력합니다.

필수: if --ssl을 지정한 경우 필수입니다. 그렇지 않은 경우에는 사용해서는 안 됩니다.

--cert

새 인증서를 지정합니다. 인증서를 포함하는 파일의 경로 및 파일 이름을 입력합니다. 클러스터를 초기화하는 데 사용된 자체 서명 인증서인 customerCA.crt에 새 인증서를 연결해야 합니다. 자세한 내용은 클러스터 초기화를 참조하세요.

필수: if --ssl을 지정한 경우 필수입니다. 그렇지 않은 경우에는 사용해서는 안 됩니다.

--cmu ***<ENI IP address>***

및 파라미터를 하나의 파라미터로 결합합니다. -a -m 지정된 HSM ENI (엘라스틱 네트워크 인터페이스) IP 주소를 AWS CloudHSM 구성 파일에 추가한 다음 CMU 구성 파일을 업데이트합니다. 클러스터에 있는 모든 HSM의 IP 주소를 입력합니다. 클라이언트 SDK 3.2.1 및 이전 버전의 경우 클라이언트 SDK 3.2.1 및 이전 버전에서 CMU 사용을 참조하세요.

필수: 예

관련 주제

- key_mgmt_util 설정

CloudHSM 명령줄 인터페이스(CLI)

CloudHSM CLI는 관리자가 사용자를 관리하고 암호화 사용자는 클러스터의 키를 관리하는 데 도움이 됩니다. 여기에는 사용자를 생성, 삭제 및 나열하고, 사용자 암호를 변경하고, 사용자 다단계 인증(MFA) 을 업데이트하는 데 사용할 수 있는 도구가 포함되어 있습니다. 또한 키 생성, 삭제, 가져오기 및 내보내기, 속성 가져오기 및 설정, 키 찾기, 암호화 작업 수행 등의 명령도 포함됩니다.

CloudHSM CLI 사용자의 정의된 목록은 CloudHSM CLI를 사용한 HSM 사용자 관리 을 참조하십시오. CloudHSM CLI의 정의된 주요 속성 목록은 을 참조하십시오. CloudHSM CLI의 키 속성 CloudHSM CLI를 사용하여 키를 관리하는 방법에 대한 자세한 내용은 을 참조하십시오. CloudHSM CLI를 사용한 키 관리

빠른 시작은 CloudHSM 명령줄 인터페이스(CLI) 시작하기을 참조하십시오. CloudHSM CLI 명령 및 명령 사용 예시에 대한 자세한 내용은 CloudHSM CLI 명령에 대한 참조을 참조하십시오.

주제

- CloudHSM 명령줄 인터페이스(CLI)를 지원하는 플랫폼
- CloudHSM 명령줄 인터페이스(CLI) 시작하기
- 대화형 및 단일 명령 모드
- CloudHSM CLI의 키 속성
- 클라이언트 SDK 3 CMU 및 KMU에서 클라이언트 SDK 5 CloudHSM CLI로 마이그레이션
- CLI를 위한 고급 구성
- CloudHSM CLI 명령에 대한 참조

CloudHSM 명령줄 인터페이스(CLI)를 지원하는 플랫폼

Linux 지원

지원하는 플랫폼	X86_64 아키텍처	ARM 아키텍처
Amazon Linux 2	예	예

지원하는 플랫폼	X86_64 아키텍처	ARM 아키텍처
Amazon Linux 2023	예	예
센토스 7 (7.8+)	예	아니요
레드햇 엔터프라이즈 리눅스 7 (7.8+)	예	아니요
레드햇 엔터프라이즈 리눅스 8 (8.3+)	예	아니요
레드햇 엔터프라이즈 리눅스 9 (9.2+)	예	예
Ubuntu 20.04 LTS	예	아니요
Ubuntu 22.04 LTS	예	예

참고: SDK 5.4.2는 CentOS 8 플랫폼 지원을 제공하는 마지막 릴리스입니다. 자세한 내용은 CentOS 웹 사이트를 참조하세요.

윈도우 지원

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

CloudHSM 명령줄 인터페이스(CLI) 시작하기

CloudHSM CLI (명령줄 인터페이스) 를 사용하면 클러스터의 사용자를 관리할 수 있습니다. AWS CloudHSM 이 주제를 통해 사용자 생성, 사용자 목록 작성, CloudHSM CLI를 클러스터에 연결 등 기본적인 HSM 사용자 관리 작업을 시작할 수 있습니다.

클라우드HSM CLI 설치

다음 명령을 사용하여 CloudHSM CLI를 다운로드하고 설치합니다.

Amazon Linux 2

x86_64 아키텍처의 Amazon Linux 2:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

ARM64 아키텍처의 Amazon Linux 2:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.aarch64.rpm
```

Amazon Linux 2023

x86_64 아키텍처의 아마존 리눅스 2023:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.x86_64.rpm
```

ARM64 아키텍처의 아마존 리눅스 2023:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.amzn2023.aarch64.rpm
```

CentOS 7 (7.8+)

x86_64 아키텍처 기반 CentOS 7:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

RHEL 7 (7.8+)

x86_64 아키텍처 기반 RHEL 7:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-cli-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

x86_64 아키텍처 기반 RHEL 8:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-cli-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el8.x86_64.rpm
```

RHEL 9 (9.2+)

x86_64 아키텍처의 RHEL 9:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.x86_64.rpm
```

ARM64 아키텍처의 RHEL 9:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-cli-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-cli-latest.el9.aarch64.rpm
```

Ubuntu 20.04 LTS

x86_64 아키텍처 기반 Ubuntu 20.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-cli_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u20.04_amd64.deb
```

Ubuntu 22.04 LTS

x86_64 아키텍처 기반 Ubuntu 22.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-cli_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_amd64.deb
```

ARM64 아키텍처의 우분투 22.04 LTS:

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/cloudhsm-cli_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-cli_latest_u22.04_arm64.deb
```

Windows Server 2016

x86_64 아키텍처의 Windows Server 2016의 경우 관리자 PowerShell 권한으로 열고 다음 명령을 실행합니다.

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2019

x86_64 아키텍처의 Windows Server 2019의 경우 관리자 PowerShell 권한으로 열고 다음 명령을 실행합니다.

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Windows/AWSCloudHSMCLI-latest.msi -Outfile C:\AWSCloudHSMCLI-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMCLI-latest.msi /
quiet /norestart /log C:\client-install.txt' -Wait
```

다음 명령을 사용하여 CloudHSM CLI를 구성합니다.

Client SDK 5용 Linux EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM(s)의 IP 주소를 지정합니다.

```
$ sudo /opt/cloudhsm/bin/configure-cli -a <The ENI IP addresses of the HSMs>
```

Client SDK 5용 Windows EC2 인스턴스 부트스트랩

- 구성 도구를 사용하여 클러스터에 있는 HSM(s)의 IP 주소를 지정합니다.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" -a <The ENI IP addresses
 of the HSMs>
```

CloudHSM CLI 사용

1. 다음 명령을 사용하여 CloudHSM CLI를 시작합니다.

   Linux

   ```
   $ /opt/cloudhsm/bin/cloudhsm-cli interactive
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
   ```

2. login 명령을 사용하여 클러스터에 로그인합니다. 모든 사용자가 이 명령을 사용할 수 있습니다.

   다음 예의 명령은 기본 관리자 계정인 admin에 로그인합니다. 클러스터를 활성화한 경우에만 이 사용자의 암호를 설정할 수 있습니다.

```
aws-cloudhsm > login --username admin --role admin
```

시스템이 암호를 묻는 메시지를 표시합니다. 암호를 입력하면 명령이 성공했다는 결과가 출력됩니다.

```
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin",
    "role": "admin"
  }
}
```

3. user list 명령을 실행하여 클러스터의 모든 사용자를 나열합니다.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

4. user create을 사용하여 **example_user**라는 CU 사용자를 생성합니다.

이전 단계에서 관리자로 로그인했으므로 CU를 생성할 수 있습니다. 관리자만 사용자 생성 및 삭제, 다른 사용자의 암호 변경과 같은 사용자 관리 작업을 수행할 수 있습니다.

```
aws-cloudhsm > user create --username example_user --role crypto-user
Enter password:
Confirm password:
{
 "error_code": 0,
 "data": {
   "username": "example_user",
   "role": "crypto-user"
 }
}
```

5. user list을 사용하여 클러스터의 모든 사용자를 나열합니다.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "example_user",
        "role": "crypto_user",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
```

```
    ]
  }
}
```

6. logout명령을 사용하여 클러스터에서 로그아웃합니다. AWS CloudHSM

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

7. quit 명령을 사용하여 CLI를 중지합니다.

```
aws-cloudhsm > quit
```

대화형 및 단일 명령 모드

CloudHSM CLI에서는 단일 명령 모드와 대화형 모드의 두 가지 방법으로 명령을 실행할 수 있습니다. 대화형 모드는 사용자를 위해 설계되었으며 단일 명령 모드는 스크립트용으로 설계되었습니다.

> **Note**
>
> 모든 명령은 대화형 모드 및 단일 명령 모드에서 작동합니다.

대화형 모드

다음 명령을 사용하여 CloudHSM CLI 대화형 모드를 시작합니다.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm-cli interactive
```

Windows

```
C:\Program Files\Amazon\CloudHSM\bin\> .\cloudhsm-cli.exe interactive
```

대화형 모드에서 CLI를 사용하는 경우 login 명령을 사용하여 사용자 계정에 로그인할 수 있습니다.

모든 CloudHSM CLI 명령을 나열하려면 다음 명령을 실행합니다.

```
aws-cloudhsm > help
```

CloudHSM CLI 명령의 구문을 가져오려면 다음 명령을 실행합니다.

```
aws-cloudhsm >  help <command-name>
```

HSM의 사용자 목록을 가져오려면 user list을 입력합니다.

```
aws-cloudhsm > user list
```

CloudHSM CLI 세션을 종료하려면 다음 명령을 실행합니다.

```
aws-cloudhsm > quit
```

단일 명령 모드

단일 명령 모드를 사용하여 CloudHSM CLI를 실행하는 경우 자격 증명을 제공할 두 가지 환경 변수, 즉 CLOUDHSM_PIN과 CLOUDHSM_ROLE을 설정해야 합니다.

```
$ export CLOUDHSM_ROLE=admin
```

```
$ export CLOUDHSM_PIN=admin_username:admin_password
```

이렇게 한 후에는 환경에 저장된 자격 증명을 사용하여 명령을 실행할 수 있습니다.

```
$ cloudhsm-cli user change-password --username alice --role crypto-user
Enter password:
Confirm password:
{
    "error_code": 0,
    "data": {
      "username": "alice",
      "role": "crypto-user"
    }
}
```

CloudHSM CLI의 키 속성

이 주제에서는 CloudHSM CLI를 사용하여 키 속성을 설정하는 방법을 설명합니다. CloudHSM CLI의 키 속성은 키 유형, 키 작동 방식 또는 키 레이블 지정 방식을 정의할 수 있습니다. 일부 속성은 고유한 특성(예: 키 유형)을 정의합니다. 다른 속성은 true 또는 false로 설정할 수 있습니다. 속성을 변경하면 키 기능의 일부가 활성화되거나 비활성화됩니다.

키 속성 사용 방법을 보여주는 예제는 상위 명령 키 아래에 나열된 명령을 참조하십시오.

지원되는 속성

제한적으로 만들려는 속성의 값만 설정하는 것이 좋습니다. 값을 지정하지 않으면 CloudHSM CLI는 아래 표에 지정된 기본값을 사용합니다.

다음 표에는 키 속성, 가능한 값, 기본값 및 관련 참고 사항이 나열되어 있습니다. 값 열의 빈 셀은 속성에 할당된 특정 기본값이 없음을 나타냅니다.

CloudHSM CLI 속성	값	키 세트 속성으로 수정 가능	키 생성 시 설정 가능
`always-sensitive`	`sensitive` 가 항상 `True`로 설정되어 있고 변경된 적이 없는 경우 값은 `True`입니다.	아니요	아니요
`check-value`	키의 확인 값입니다. 자세한 내용은 추가 세부 정보를 참조하십시오.	아니요	아니요
`class`	가능한 값: `secret-key` , `public-key` 및 `private-key`	아니요	예
`curve`	EC 키 페어를 생성하는 데 사용되는 타원 곡선입니다. 유효한 값: `secp224r1` ,	아니요	RSA로 설정 가능, EC로 설정 불가

CloudHSM CLI 속성	값	키 세트 속성으로 수정 가능	키 생성 시 설정 가능
	secp256r1 , prime256v1 , secp384r1 , secp256k1 및 secp521r1		
decrypt	기본값: False	예	예
derive	기본값: False	예	예
destroyable	기본값: True	예	예
ec-point	EC 키의 경우 ANSI X9.62 ECPoint 값 "Q"를 16진수 형식으로 DER 인코딩합니다. 다른 키 유형의 경우, 이 속성이 존재하지 않습니다.	아니요	아니요
encrypt	기본값: False	예	예
extractable	기본값: True	아니요	예
id	기본값: 비어 있음	아니요	예
key-length-bytes	AES 키를 생성하는 데 필요합니다. 유효한 값: 16, 24 및 32 바이트	아니요	아니요
key-type	가능한 값: aes, rsa 및 ec	아니요	예
label	기본값: 비어 있음	예	예

CloudHSM CLI 속성	값	키 세트 속성으로 수정 가능	키 생성 시 설정 가능
local	기본값: HSM에서 생성된 키의 경우 True, HSM으로 가져온 키의 경우 False입니다.	아니요	아니요
modifiable	기본값: True	아니요	아니요
modulus	RSA 키 페어를 생성하는 데 사용된 모듈러스입니다. 다른 키 유형의 경우, 이 속성이 존재하지 않습니다.	아니요	아니요
modulus-size-bits	RSA 키 페어를 생성하는 데 필요합니다. 최소값은 2048입니다.	아니요	RSA로 설정 가능, EC로 설정 불가
never-extractable	추출 가능 항목이 False로 설정된 적이 없는 경우 값은 True입니다. 추출 가능 항목이 True로 설정된 적이 있는 경우 값은 False입니다.	아니요	아니요
private	기본값: True	아니요	예
public-exponent	RSA 키 페어를 생성하는 데 필요합니다. 유효한 값: 값은 65537 이상의 홀수여야 합니다.	아니요	RSA로 설정 가능, EC로 설정 불가

CloudHSM CLI 속성	값	키 세트 속성으로 수정 가능	키 생성 시 설정 가능
sensitive	기본값: • AES 키와 EC 및 RSA 프라이빗 키의 값은 True입니다. • EC 및 RSA 퍼블릭 키의 값은 False입니다.	아니요	프라이빗 키로 설정 가능하고 퍼블릭 키로는 설정할 수 없습니다.
sign	기본값: • AES 키의 값은 True입니다. • RSA 및 EC 키의 값은 False입니다.	예	예
token	기본값: False	아니요	예
trusted	기본값: False	예	아니요
unwrap	기본값: False	예	예
unwrap-template	값은 이 래핑 키를 사용하여 언래핑된 모든 키에 적용되는 속성 템플릿을 사용해야 합니다.	예	아니요
verify	기본값: • AES 키의 값은 True입니다. • RSA 및 EC 키의 값은 False입니다.	예	예
wrap	기본값: False	예	예

CloudHSM CLI 속성	값	키 세트 속성으로 수정 가능	키 생성 시 설정 가능
`wrap-template`	값은 이 래핑 키를 사용하여 래핑된 키와 일치하도록 속성 템플릿을 사용해야 합니다.	예	아니요
`wrap-with-trusted`	기본값: `False`	예	예

추가 세부 정보

값 확인

확인 값은 HSM이 키를 가져오거나 생성할 때 생성된 키의 3바이트 해시 또는 체크섬입니다. 또한 키를 내보낸 후와 같이 HSM 밖에서 확인 값을 계산할 수 있습니다. 그 다음에는 확인 값을 비교하여 키의 자격 증명 및 무결성을 확인할 수 있습니다. 키의 확인 값을 가져오려면 verbose 플래그와 함께 키 목록을 사용하십시오.

AWS CloudHSM 는 다음과 같은 표준 방법을 사용하여 체크 값을 생성합니다.

- 대칭 키: 키를 사용하여 0 블록을 암호화한 결과의 처음 3바이트.
- 비대칭 키 페어: 퍼블릭 키 SHA-1 해시의 처음 3바이트.
- HMAC 키: HMAC 키의 KVC는 현재 지원되지 않습니다.

관련 주제

- 키
- CloudHSM CLI 명령에 대한 참조

클라이언트 SDK 3 CMU 및 KMU에서 클라이언트 SDK 5 CloudHSM CLI로 마이그레이션

이 주제를 사용하여 클라이언트 SDK 3 명령줄 도구인 CloudHSM 관리 유틸리티 (CMU) 및 키 관리 유틸리티 (KMU) 를 사용하는 워크플로를 대신 클라이언트 SDK 5 명령줄 도구인 CloudHSM CLI를 사용하도록 마이그레이션할 수 있습니다.

에서 고객 애플리케이션은 AWS CloudHSM클라이언트 소프트웨어 개발 키트 (SDK) 를 사용하여 암호화 작업을 수행합니다. AWS CloudHSM 클라이언트 SDK 5는 계속해서 새로운 기능과 플랫폼 지원이 추가되는 기본 SDK입니다. 이 항목에서는 명령줄 도구를 위해 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하는 방법에 대한 세부 정보를 제공합니다.

클라이언트 SDK 3에는 사용자 관리를 위한 CMU와 키를 관리하고 키로 작업을 수행하는 KMU라는 두 개의 별도 명령줄 도구가 포함되어 있습니다. 클라이언트 SDK 5는 CMU 및 KMU (클라이언트 SDK 3와 함께 제공되었던 도구) 의 기능을 단일 도구인 로 통합합니다. CloudHSM 명령줄 인터페이스(CLI) 사용자 관리 작업은 하위 명령 및 에서 찾을 수 있습니다. 사용자 쿼럼 키 관리 작업은 key 하위 명령에서, 암호화 작업은 crypto 하위 명령에서 찾을 수 있습니다. 전체 명령 목록은 을 CloudHSM CLI 명령에 대한 참조 참조하십시오.

> **Note**
>
> 클러스터 간 동기화를 위해 클라이언트 SDK syncKey 3와 syncUser 기능을 사용했다면 CMU를 계속 사용하십시오. 클라이언트 SDK 5의 CloudHSM CLI는 현재 이 기능을 지원하지 않습니다.

클라이언트 SDK 5로 마이그레이션하는 방법에 대한 지침은 을 참조하십시오. 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션 마이그레이션의 이점은 을 참조하십시오. Client SDK 5의 이점

CLI를 위한 고급 구성

AWS CloudHSM 명령줄 인터페이스 (CLI) 에는 대부분의 고객이 사용하는 일반 구성에는 포함되지 않는 다음과 같은 고급 구성이 포함됩니다. 이러한 구성은 추가 기능을 제공합니다.

- 여러 클러스터에 연결

CloudHSM CLI를 사용하여 여러 클러스터에 연결

클라이언트 SDK 5를 사용하면 단일 CLI 인스턴스에서 여러 CloudHSM 클러스터에 연결할 수 있도록 CloudHSM CLI를 구성할 수 있습니다.

CloudHSM CLI를 사용하여 다중 클러스터 기능을 사용하여 여러 클러스터에 연결하려면 이 항목의 지침을 따르십시오.

주제

- 다중 클러스터 사전 요구 사항

- 다중 클러스터 기능을 위한 CloudHSM CLI를 구성합니다.
- 구성-cli 추가 클러스터
- 구성-cli 제거-클러스터
- 여러 클러스터 사용

다중 클러스터 사전 요구 사항

- 연결하려는 두 개 이상의 AWS CloudHSM 클러스터와 해당 클러스터 인증서
- 위의 모든 클러스터에 연결하도록 보안 그룹이 올바르게 구성된 EC2 인스턴스. 클러스터 및 클라이언트 인스턴스를 설정하는 방법에 대한 자세한 내용은 시작하기를 참조하십시오 AWS CloudHSM.
- 다중 클러스터 기능을 설정하려면 CloudHSM CLI를 이미 다운로드하여 설치해야 합니다. 아직 수행하지 않은 경우 ???의 지침을 참조하십시오.
- 로 구성된 클러스터는 a와 연결되지 `./configure-cli[.exe] -a` 않으므로 액세스할 수 없습니다. `cluster-id` 이 가이드에 설명된 `config-cli add-cluster` 대로 재구성할 수 있습니다.

다중 클러스터 기능을 위한 CloudHSM CLI를 구성합니다.

다중 클러스터 기능을 위해 CloudHSM CLI를 구성하려면 다음 단계를 따르십시오.

1. 연결하려는 클러스터를 식별하십시오.
2. 아래 설명과 같이 configure-cli 하위 명령을 사용하여 CloudHSM CLI 구성에 이러한 클러스터를 추가합니다. `add-cluster`
3. 새 구성을 적용하려면 모든 CloudHSM CLI 프로세스를 다시 시작하십시오.

구성-cli 추가 클러스터

여러 클러스터에 연결하는 경우 `configure-cli add-cluster` 명령을 사용하여 구성에 클러스터를 추가합니다.

구문

```
configure-cli add-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [--region <REGION>]
        [--endpoint <ENDPOINT>]
        [--hsm-ca-cert <HSM CA CERTIFICATE FILE>]
```

```
        [--server-client-cert-file <CLIENT CERTIFICATE FILE>]
        [--server-client-key-file <CLIENT KEY FILE>]
        [-h, --help]
```

예

cluster-id 파라미터를 사용하여 클러스터를 추가합니다.

Example

cluster-id 파라미터와 함께 configure-cli add-cluster을 사용하여 구성에 클러스터(ID cluster-1234567)를 추가합니다.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli add-cluster --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-cli.exe add-cluster --cluster-
id cluster-1234567
```

> **Tip**
>
> cluster-id 파라미터와 함께 configure-cli add-cluster을 사용해도 클러스터가 추가되지 않는 경우, 추가되는 클러스터를 식별하기 위해 --region 및 --endpoint 파라미터도 필요한 이 명령의 더 긴 버전에 대한 다음 예를 참조하십시오. 예를 들어 클러스터의 리전이 AWS CLI 기본값으로 구성된 리전과 다른 경우 --region 파라미터를 사용하여 올바른 리전을 사용해야 합니다. 또한 호출에 사용할 AWS CloudHSM API 엔드포인트를 지정할 수 있습니다. 이는 기본 DNS 호스트 이름을 사용하지 않는 VPC 인터페이스 엔드포인트를 사용하는 등 다양한 네트워크 설정에 필요할 수 있습니다. AWS CloudHSM

cluster-id, **endpoint**, **region** 파라미터를 사용하여 클러스터를 추가합니다.

Example

cluster-id, endpoint, region 파라미터와 함께 configure-cli add-cluster을 사용하여 구성에 클러스터(ID cluster-1234567)를 추가합니다.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli add-cluster --cluster-id cluster-1234567 --
region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-cli.exe add-cluster --cluster-
id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

`--cluster-id`, `--region`, `--endpoint` 파라미터에 대한 자세한 내용은 the section called “파라미터” 섹션을 참조하십시오.

파라미터

--cluster-id ***<Cluster ID>***

클러스터 ID와 연결된 클러스터의 모든 HSM Elastic Network 인터페이스(ENI) IP 주소를 찾기 위해 `DescribeClusters`를 호출합니다. 시스템은 ENI IP 주소를 구성 파일에 추가합니다. AWS CloudHSM

> **Note**
>
> 퍼블릭 인터넷에 액세스할 수 없는 VPC 내 EC2 인스턴스의 `--cluster-id` 파라미터를 사용하는 경우 연결할 인터페이스 VPC 엔드포인트를 만들어야 합니다. AWS CloudHSM VPC 엔드포인트에 대한 자세한 내용은 ??? 섹션을 참조하십시오.

필수 여부: 예

--endpoint ***<Endpoint>***

AWS CloudHSM 호출에 사용되는 API 엔드포인트를 지정합니다. `DescribeClusters` 이 옵션은 `--cluster-id`와 조합하여 설정해야 합니다.

필수 여부: 아니요

-- hsm-ca-cert <HsmCA Certificate Filepath>

HSM CA 인증서의 파일 경로를 지정합니다.

필수 여부: 아니요

--region *<Region>*

클러스터의 지역을 지정합니다. 이 옵션은 `--cluster-id`와 조합하여 설정해야 합니다.

`--region` 파라미터를 제공하지 않으면 시스템은 `AWS_DEFAULT_REGION` 또는 `AWS_REGION` 환경 변수를 읽으려고 시도하여 리전을 선택합니다. 이러한 변수가 설정되지 않은 경우, 시스템은 사용자가 `AWS_CONFIG_FILE` 환경 변수에 다른 파일을 지정하지 않는 한 AWS config 파일(일반적으로 `~/.aws/config`)의 프로필과 연결된 리전을 확인합니다. 위 항목 중 아무 것도 설정되지 않은 경우 시스템은 `us-east-1` 리전을 기본값으로 사용합니다.

필수 여부: 아니요

-- server-client-cert-file <Client Certificate Filepath>

TLS 클라이언트-서버 상호 인증에 사용되는 클라이언트 인증서의 경로입니다.

클라이언트 SDK 5에 포함된 기본 키 및 SSL/TLS 인증서를 사용하지 않으려는 경우에만 이 옵션을 사용하세요. 이 옵션은 `--server-client-key-file`와 조합하여 설정해야 합니다.

필수 여부: 아니요

-- server-client-key-file <Client Key Filepath>

TLS 클라이언트-서버 상호 인증에 사용되는 클라이언트 키의 경로

클라이언트 SDK 5에 포함된 기본 키 및 SSL/TLS 인증서를 사용하지 않으려는 경우에만 이 옵션을 사용하세요. 이 옵션은 `--server-client-cert-file`와 조합하여 설정해야 합니다.

필수 여부: 아니요

구성-cli 제거-클러스터

CloudHSM CLI로 여러 클러스터에 연결하는 경우 명령을 `configure-cli remove-cluster` 사용하여 구성에서 클러스터를 제거합니다.

구문

```
configure-cli remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

예

cluster-id 파라미터를 사용하여 클러스터를 삭제합니다.

Example

cluster-id 파라미터와 함께 configure-cli remove-cluster을 사용하여 구성에서 클러스터 (ID cluster-1234567)를 삭제합니다.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-cli remove-cluster --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-cli.exe remove-cluster --cluster-
id cluster-1234567
```

--cluster-id 파라미터에 대한 자세한 내용은 the section called “파라미터”을 참조하세요.

파라미터

--cluster-id *<Cluster ID>*

구성에서 제거할 클러스터의 ID.

필수 여부: 예

여러 클러스터 사용

CloudHSM CLI로 여러 클러스터를 구성한 후 명령을 cloudhsm-cli 사용하여 해당 클러스터와 상호 작용합니다.

예

대화형 모드 사용 **cluster-id** 시 기본값 설정

Example

cluster-id매개 변수와 ??? 함께 를 사용하여 구성에서 기본 클러스터 (ID 포 함cluster-1234567) 를 설정합니다.

Linux

```
$ cloudhsm-cli interactive --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\cloudhsm-cli.exe interactive --cluster-
id cluster-1234567
```

단일 명령 실행 **cluster-id** 시 설정

Example

`cluster-id`파라미터를 사용하여 ??? 가져올 클러스터 (ID 포함`cluster-1234567`) 를 설정합니다.

Linux

```
$ cloudhsm-cli cluster hsm-info --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\cloudhsm-cli.exe cluster hsm-info --cluster-
id cluster-1234567
```

CloudHSM CLI 명령에 대한 참조

CloudHSM CLI는 관리자가 클러스터의 사용자를 관리하는 데 도움이 됩니다. AWS CloudHSM CloudHSM CLI는 대화형 모드와 단일 명령 모드의 두 가지 모드로 실행할 수 있습니다. 빠른 시작은 CloudHSM 명령줄 인터페이스(CLI) 시작하기 섹션을 참조하십시오.

대부분의 CloudHSM CLI 명령을 실행하려면 CloudHSM CLI를 시작하고 HSM에 로그인해야 합니다. HSM을 추가하거나 삭제하는 경우 CloudHSM CLI의 구성 파일을 업데이트합니다. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

다음 주제에서는 CloudHSM CLI의 명령에 대해 설명합니다.

Command	설명	사용자 유형
클러스터 활성화	CloudHSM 클러스터를 활성화하고 클러스터가 새 클러스터인지 확인합니다. 다른 작업을 수행하기 전에 이 작업을 먼저 수행해야 합니다.	활성화되지 않은 관리자
클러스터 hsm-정보	클러스터의 HSM을 나열하십시오.	모든 [1](인증되지 않은 사용자 포함) 로그인은 필요하지 않습니다.
크립토 사인 ecdsa	EC 개인 키와 ECDSA 서명 메커니즘을 사용하여 서명을 생성합니다.	암호화 사용자(Crypto User)
크립토 사인 rsa-pkcs	RSA 개인 키와 RSA-PKCS 서명 메커니즘을 사용하여 서명을 생성합니다.	CU
암호 기호 rsa-pkcs-pss	RSA 개인 키와 RSA-PKCS-PSS 서명 메커니즘을 사용하여 서명을 생성합니다.	CU
크립토 검증 ecdsa	지정된 공개 키로 HSM에서 파일이 서명되었는지 확인합니다. ECDSA 서명 메커니즘을 사용하여 서명이 생성되었는지 확인합니다. 서명된 파일을 소스 파일과 비교하고 지정된 ecdsa 공개 키 및 서명 메커니즘을 기반으로 두 파일이 암호학적으로 관련되어 있는지 확인합니다.	CU
크립토 검증 rsa-pkcs	지정된 공개 키로 HSM에서 파일이 서명되었는지 확인합니다. RSA-PKCS 서명 메커니즘	CU

Command	설명	사용자 유형
	을 사용하여 서명이 생성되었는지 확인합니다. 서명된 파일을 소스 파일과 비교하고 지정된 rsa 공개 키 및 서명 메커니즘을 기반으로 두 파일이 암호학적으로 관련되어 있는지 확인합니다.	
크립토 검증 rsa-pkcs-pss	지정된 공개 키로 HSM에서 파일이 서명되었는지 확인합니다. RSA-PKCS-PSS 서명 메커니즘을 사용하여 서명이 생성되었는지 확인합니다. 서명된 파일을 소스 파일과 비교하고 지정된 rsa 공개 키 및 서명 메커니즘을 기반으로 두 파일이 암호학적으로 관련되어 있는지 확인합니다.	CU
키 삭제	클러스터에서 키를 삭제합니다. AWS CloudHSM	CU
키 생성-파일	AWS CloudHSM 클러스터에서 키 파일을 생성합니다.	CU
키 generate-asymmetric-pair rsa	클러스터에서 비대칭 RSA 키 쌍을 생성합니다. AWS CloudHSM	CU
키 ec generate-asymmetric-pair	클러스터에 비대칭 EC (타원곡선) 키 쌍을 생성합니다. AWS CloudHSM	CU
키 생성-대칭 aes	클러스터에서 대칭 AES 키를 생성합니다. AWS CloudHSM	CU

Command	설명	사용자 유형
키 생성-대칭 일반 보안	클러스터에서 대칭형 일반 비밀 키를 생성합니다. AWS CloudHSM	CU
키 임포트 pem	PEM 형식 키를 HSM으로 가져옵니다. 이 명령을 사용하여 HSM 외부에서 생성된 퍼블릭 키를 가져올 수 있습니다.	CU
키 목록	클러스터에 있는 현재 사용자의 모든 키를 찾습니다. AWS CloudHSM	CU
키 복제	소스 클러스터에서 복제된 대상 클러스터로 키를 복제합니다.	CU
키 세트-속성	클러스터에 있는 키의 속성을 설정합니다. AWS CloudHSM	CU는 이 명령을 실행할 수 있고, 관리자는 신뢰할 수 있는 속성을 설정할 수 있습니다.
키 공유	AWS CloudHSM 클러스터의 다른 CU와 키를 공유합니다.	CU
키 공유 취소	AWS CloudHSM 클러스터의 다른 CU와 키를 공유 해제합니다.	CU
키 언랩 aes-gcm	AES 래핑 키와 AES-GCM 언래핑 메커니즘을 사용하여 페이로드 키를 클러스터로 언래핑합니다.	CU
키 언랩 aes-no-pad	AES 래핑 키와 AES-NO-PAD 언래핑 메커니즘을 사용하여 페이로드 키를 클러스터로 언래핑합니다.	CU

Command	설명	사용자 유형
키 언랩 aes-pkcs5-패드	AES 래핑 키와 AES-PKCS5-PAD 언래핑 메커니즘을 사용하여 페이로드 키를 언래핑합니다.	CU
키 언랩 aes-zero-pad	AES 래핑 키와 AES-ZERO-PAD 언래핑 메커니즘을 사용하여 페이로드 키를 클러스터로 언래핑합니다.	CU
키 언랩 cloudhsm-aes-gcm	AES 래핑 키와 CLOUDHSM-AES-GCM 언래핑 메커니즘을 사용하여 페이로드 키를 클러스터로 언래핑합니다.	CU
키 언랩 rsa-aes	RSA 개인 키와 RSA-AES 언래핑 메커니즘을 사용하여 페이로드 키를 언래핑합니다.	CU
키 언랩 rsa-oaep	RSA 개인 키와 RSA-OAEP 언래핑 메커니즘을 사용하여 페이로드 키를 언래핑합니다.	CU
키 언랩 rsa-pkcs	RSA 개인 키와 RSA-PKCS 언래핑 메커니즘을 사용하여 페이로드 키를 언래핑합니다.	CU
키 랩 aes-gcm	HSM의 AES 키와 AES-GCM 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다.	CU
키 랩 aes-no-pad	HSM의 AES 키와 AES-NO-PAD 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다.	CU

Command	설명	사용자 유형
키 랩 AES-pkcs5-패드	HSM의 AES 키와 AES-PKCS5-PAD 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다.	CU
키 랩 aes-zero-pad	HSM의 AES 키와 AES-ZERO-PAD 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다.	CU
키 랩 cloudhsm-aes-gcm	HSM의 AES 키와 CLOUDHSM-AES-GCM 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다.	CU
키 랩 라사-에스	HSM의 RSA 공개 키와 RSA-AES 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다.	CU
키 랩 랩-비누	HSM의 RSA 공개 키와 RSA-OAEP 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다.	CU

Command	설명	사용자 유형
이 key wrap rsa-pkcs 명령은 HSM의 RSA 공개 키와 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다. RSA-PKCS 페이로드 키의 `extractable` 속성을 로 설정해야 합니다. `true` 키 소유자, 즉 키를 생성한 암호화 사용자 (CU) 만 키를 래핑할 수 있습니다. 키를 공유하는 사용자는 암호화 작업에 키를 사용할 수 있습니다. key wrap rsa-pkcs명령을 사용하려면 먼저 클러스터에 RSA 키가 있어야 합니다. AWS CloudHSM 키 generate-asymmetric-pair 명령과 `wrap` 속성 설정을 로 사용하여 RSA 키페어를 생성할 수 있습니다. `true` 사용자 유형 다음 사용자 유형이 이 명령을 실행할 수 있습니다. • CU(Crypto User) 요구 사항 • 이 명령을 실행하려면 CU로 로그인해야 합니다. 구문 `aws-cloudhsm > help key wrap rsa-pkcs`	HSM의 RSA 공개 키와 RSA-PKCS 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다.	CU

Command	설명	사용자 유형
로그인	클러스터에 AWS CloudHSM 로그인합니다.	관리자, CU(Crypto User), AU(Appliance User)
로그아웃	AWS CloudHSM 클러스터에서 로그아웃하세요.	관리자, CU, AU(Appliance User)
쿼럼 토큰-서명 삭제	쿼럼 인증 서비스의 토큰을 하나 이상 삭제합니다.	관리자
쿼럼 토큰-서명 생성	쿼럼 인증 서비스를 위한 토큰을 생성합니다.	관리자
쿼럼 토큰-서명 목록	CloudHSM 클러스터에 있는 모든 토큰-서명 쿼럼 토큰을 나열합니다.	모든 [1](인증되지 않은 사용자 포함) 로그인은 필요하지 않습니다.
쿼럼 토큰 사인 list-quorum-values	CloudHSM 클러스터에 설정된 쿼럼 값을 나열합니다.	모든 [1](인증되지 않은 사용자 포함) 로그인은 필요하지 않습니다.
쿼럼 토큰-서명 목록-제한 시간	모든 토큰 유형의 토큰 제한 시간(초)을 가져옵니다.	관리자 및 CU(Crypto User)
쿼럼 토큰 사인 set-quorum-value	쿼럼 인증 서비스의 새 쿼럼 값을 설정합니다.	관리자
쿼럼 토큰-서명 설정-제한 시간	각 토큰 유형의 토큰 제한 시간을 초 단위로 설정합니다.	관리자
사용자 변경-mfa	사용자의 다중 인증(MFA) 전략을 변경합니다.	관리자, CU
사용자 암호-변경	HSM 사용자의 암호를 변경합니다. 모든 사용자는 본인의 암호를 변경할 수 있습니다. 관리자는 모든 사용자의 암호를 변경할 수 있습니다.	관리자, CU

Command	설명	사용자 유형
사용자 생성	클러스터에 사용자를 생성합니다. AWS CloudHSM	관리자
사용자 삭제	AWS CloudHSM 클러스터에서 사용자를 삭제합니다.	관리자
사용자 목록	AWS CloudHSM 클러스터의 사용자를 나열합니다.	모든 [1](인증되지 않은 사용자 포함) 로그인은 필요하지 않습니다.
사용자 변경-쿼럼 토큰-서명 등록	사용자의 쿼럼 토큰-서명 쿼럼 전략을 등록합니다.	관리자

주석

- [1] 모든 사용자에는 나열된 모든 역할과 로그인하지 않은 사용자가 포함됩니다.

cluster

cluster은 상위 카테고리와 결합될 때 사용자에게 특정한 명령을 생성하는 명령 그룹의 상위 카테고리입니다. 현재 사용자 범주는 다음과 같은 명령으로 구성되어 있습니다.

- 클러스터 활성화
- 클러스터 hsm-정보

클러스터 활성화

CloudHSM CLI에서 cluster activate 명령을 사용하여 새 클러스터를 활성화합니다. 클러스터를 사용하여 암호화 작업을 수행하려면 먼저 해당 명령을 실행해야 합니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- 활성화되지 않은 관리자

구문

이 명령에는 파라미터가 없습니다.

```
aws-cloudhsm > help cluster activate
Activate a cluster

This command will set the initial Admin password. This process will cause your CloudHSM
 cluster to
move into the ACTIVE state.

USAGE:
    cloudhsm-cli cluster activate [OPTIONS] [--password <PASSWORD>]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --password <PASSWORD>
          Optional: Plaintext activation password If you do not include this argument
 you will be prompted for it

  -h, --help
          Print help (see a summary with '-h')
```

예

이 명령은 관리자 사용자의 초기 암호를 설정하여 클러스터를 활성화합니다.

```
aws-cloudhsm > cluster activate
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": "Cluster activation successful"
}
```

관련 주제

- 사용자 생성
- 사용자 삭제

- 사용자 변경-암호

클러스터 hsm-정보

CloudHSM CLI에서 cluster hsm-info 명령을 사용하여 클러스터의 HSM을 나열합니다 이 명령을 실행하기 위해 CloudHSM CLI에 로그인할 필요는 없습니다.

> **Note**
>
> HSM을 추가하거나 삭제하는 경우 AWS CloudHSM 클라이언트와 명령줄 도구가 사용하는 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- 모든 사용자. 이 명령을 실행하기 위해 로그인할 필요는 없습니다.

구문

```
aws-cloudhsm > help cluster hsm-info
List info about each HSM in the cluster

Usage: cloudhsm-cli cluster hsm-info [OPTIONS]

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

예

이 명령은 클러스터에 있는 HSM을 나열합니다. AWS CloudHSM

```
aws-cloudhsm > cluster hsm-info
{
```

```
  "error_code": 0,
  "data": {
    "hsms": [
      {
        "vendor": "Marvell Semiconductors, Inc.",
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000590",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      },
      {
        "vendor": "Marvell Semiconductors, Inc.",
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000625",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      },
      {
        "vendor": "Marvell Semiconductors, Inc.",
        "model": "NITROX-III CNN35XX-NFBE",
        "serial-number": "5.3G1941-ICM000663",
        "hardware-version-major": "5",
        "hardware-version-minor": "3",
        "firmware-version-major": "2",
        "firmware-version-minor": "6",
        "firmware-build-number": "16",
        "firmware-id": "CNN35XX-NFBE-FW-2.06-16"
        "fips-state": "2 [FIPS mode with single factor authentication]"
      }
    ]
  }
}
```

출력에는 다음 속성이 있습니다.

- 공급업체: HSM의 공급업체 이름입니다.
- 모델: HSM의 모델 번호입니다.
- 일련 번호: HSM의 일련 번호입니다. 교체로 인해 변경될 수 있습니다.
- H ardware-version-major: 메이저 하드웨어 버전.
- H ardware-version-minor: 마이너 하드웨어 버전.
- F irmware-version-major: 메이저 펌웨어 버전.
- F irmware-version-minor: 마이너 펌웨어 버전.
- F irmware-build-number: 펌웨어 빌드 번호.
- Firmware-id: 빌드와 함께 메이저 버전과 마이너 버전이 포함된 펌웨어 ID입니다.
- FIPS 상태: FIPS 모드, 클러스터 및 클러스터 내 HSM. FIPS 모드인 경우 출력은 “2 [단일 요소 인증을 사용하는 FIPS 모드]”입니다. 비 FIPS 모드인 경우 출력은 “0 [단일 요소 인증을 사용하는 비 FIPS 모드]”입니다.

관련 주제

- 클러스터 활성화

crypto

crypto명령 그룹의 상위 범주로, 상위 범주와 결합하면 암호화 작업과 관련된 명령을 만들 수 있습니다. 현재 이 범주는 다음과 같은 명령으로 구성되어 있습니다.

- 크립토 사인
  - 크립토 사인 ecdsa
  - 크립토 사인 rsa-pkcs
  - 암호 기호 rsa-pkcs-pss
- 크립토 검증
  - 크립토 검증 ecdsa
  - 크립토 검증 rsa-pkcs
  - 크립토 검증 rsa-pkcs-pss

크립토 사인

crypto sign명령 그룹의 상위 범주로, 상위 범주와 결합할 경우 AWS CloudHSM 클러스터에서 선택한 개인 키를 사용하여 서명을 생성합니다. crypto sign에는 다음과 같은 하위 명령이 있습니다.

- 크립토 사인 ecdsa
- 크립토 사인 rsa-pkcs
- 암호 기호 rsa-pkcs-pss

사용하려면 crypto sign HSM에 개인 키가 있어야 합니다. 다음 명령으로 개인 키를 생성할 수 있습니다.

- 키 generate-asymmetric-pair ec
- 키 generate-asymmetric-pair RSA

크립토 사인 ecdsa

이 crypto sign ecdsa 명령은 EC 개인 키와 ECDSA 서명 메커니즘을 사용하여 서명을 생성합니다.

crypto sign ecdsa명령을 사용하려면 먼저 클러스터에 EC 개인 키가 있어야 합니다. AWS CloudHSM `sign`속성이 로 설정된 키 generate-asymmetric-pair 등 명령을 사용하여 EC 개인 키를 생성할 수 `true` 있습니다.

> **Note**
>
> 서명은 크립토 검증 하위 AWS CloudHSM 명령으로 확인할 수 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help crypto sign ecdsa
Sign with the ECDSA mechanism

Usage: crypto sign ecdsa --key-filter [<KEY_FILTER>>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be signed
      --data <DATA>
          Base64 Encoded data to be signed
  -h, --help
          Print help
```

예

이 예제는 ECDSA 서명 메커니즘과 해시 함수를 사용하여 서명을 생성하는 crypto sign ecdsa 데 사용하는 방법을 보여줍니다. SHA256 이 명령은 HSM의 개인 키를 사용합니다.

Example 예: Base 64로 인코딩된 데이터에 대한 서명 생성

```
aws-cloudhsm > crypto sign ecdsa --key-filter attr.label=ec-private --hash-function
 sha256 --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw=="
  }
}
```

Example 예: 데이터 파일의 서명 생성

```
aws-cloudhsm > crypto sign ecdsa --key-filter attr.label=ec-private --hash-function
 sha256 --data-path data.txt
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000007808dd",
    "signature": "4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw=="
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<DATA>

서명할 Base64로 인코딩된 데이터.

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<DATA_PATH>

서명할 데이터의 위치를 지정합니다.

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<HASH_FUNCTION>

해시 함수를 지정합니다.

유효한 값:

- sha1
- sha224
- sha256
- sha384
- sha512

필수 여부: 예

<KEY_FILTER>

키 참조 (예: 키-참조=0xabc) 또는 Attr.key_attribute_name=key_attribute_value 형식의 공백으로 구분된 키 속성 목록을 사용하여 일치하는 키를 선택합니다.

지원되는 CloudHSM CLI 키 속성 목록은 CloudHSM CLI의 주요 속성을 참조하십시오.

필수: 예

관련 주제

- 크립토 사인
- 크립토 검증

크립토 사인 rsa-pkcs

이 crypto sign rsa-pkcs 명령은 RSA 개인 키와 RSA-PKCS 서명 메커니즘을 사용하여 서명을 생성합니다.

crypto sign rsa-pkcs명령을 사용하려면 먼저 클러스터에 RSA 개인 키가 있어야 합니다. AWS CloudHSM `sign`속성이 로 설정된 키 generate-asymmetric-pair RSA 명령을 사용하여 RSA 개인 키를 생성할 수 있습니다. `true`

> **Note**
>
> 서명은 하위 AWS CloudHSM 명령으로 크립토 검증 확인할 수 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help crypto sign rsa-pkcs
Sign with the RSA-PKCS mechanism

Usage: crypto sign rsa-pkcs --key-filter [<KEY_FILTER>>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be signed
      --data <DATA>
          Base64 Encoded data to be signed
  -h, --help
          Print help
```

예

이 예제는 RSA-PKCS 서명 메커니즘과 해시 함수를 사용하여 서명을 생성하는 crypto sign rsa-pkcs 데 사용하는 방법을 보여줍니다. SHA256 이 명령은 HSM의 개인 키를 사용합니다.

Example 예: Base 64로 인코딩된 데이터에 대한 서명 생성

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private --hash-function
 sha256 --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
```

```
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

Example 예: 데이터 파일의 서명 생성

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private --hash-function
 sha256 --data-path data.txt
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<DATA>

서명할 Base64로 인코딩된 데이터

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<DATA_PATH>

서명할 데이터의 위치를 지정합니다.

필수: 예 (데이터를 통해 제공한 경우 제외)

<HASH_FUNCTION>

해시 함수를 지정합니다.

유효한 값:

- sha1
- sha224
- sha256
- sha384
- sha512

필수 여부: 예

<KEY_FILTER>

일치하는 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록

지원되는 CloudHSM CLI 키 속성 목록은 CloudHSM CLI의 주요 속성을 참조하십시오.

필수: 예

관련 주제

- 크립토 사인
- 크립토 검증

암호 기호 rsa-pkcs-pss

이 crypto sign rsa-pkcs-pss 명령은 RSA 개인 키와 서명 메커니즘을 사용하여 서명을 생성합니다. `RSA-PKCS-PSS`

crypto sign rsa-pkcs-pss명령을 사용하려면 먼저 클러스터에 RSA 개인 키가 있어야 합니다. AWS CloudHSM `sign`속성이 로 설정된 키 generate-asymmetric-pair RSA 명령을 사용하여 RSA 개인 키를 생성할 수 있습니다. `true`

> **Note**
>
> 서명은 하위 AWS CloudHSM 명령으로 크립토 검증 확인할 수 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help crypto sign rsa-pkcs-pss
Sign with the RSA-PKCS-PSS mechanism

Usage: crypto sign rsa-pkcs-pss [OPTIONS] --key-filter [<KEY_FILTER>...] --
hash-function <HASH_FUNCTION> --mgf <MGF> --salt-length <SALT_LENGTH> <--data-
path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>        Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]   Key reference (e.g. key-
reference=0xabc) or space separated list of key attributes in the form of
 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key
      --hash-function <HASH_FUNCTION>  [possible values: sha1, sha224, sha256, sha384,
 sha512]
      --data-path <DATA_PATH>          The path to the file containing the data to be
 signed
      --data <DATA>                    Base64 Encoded data to be signed
      --mgf <MGF>                      The mask generation function [possible values:
 mgf1-sha1, mgf1-sha224, mgf1-sha256, mgf1-sha384, mgf1-sha512]
      --salt-length <SALT_LENGTH>      The salt length
  -h, --help                           Print help
```

예

이 예제는 RSA-PKCS-PSS 서명 메커니즘과 SHA256 해시 함수를 사용하여 서명을 생성하는 crypto sign rsa-pkcs-pss 데 사용하는 방법을 보여줍니다. 이 명령은 HSM의 개인 키를 사용합니다.

Example 예: Base 64로 인코딩된 데이터에 대한 서명 생성

```
aws-cloudhsm > crypto sign rsa-pkcs-pss --key-filter attr.label=rsa-private --hash-
function sha256 --data YWJjMTIz --salt-length 10 --mgf mgf1-sha256
```

```
{
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg=="
  }
}
```

Example 예: 데이터 파일의 서명 생성

```
aws-cloudhsm > crypto sign rsa-pkcs-pss --key-filter attr.label=rsa-private --hash-
function sha256 --data-path data.txt --salt-length 10 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg=="
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<DATA>

서명할 Base64로 인코딩된 데이터

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<DATA_PATH>

서명할 데이터의 위치를 지정합니다.

필수: 예 (데이터를 통해 제공한 경우 제외)

<HASH_FUNCTION>

해시 함수를 지정합니다.

유효한 값:

- sha1
- sha224
- sha256
- sha384
- sha512

필수 여부: 예

<KEY_FILTER>

일치하는 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록.

지원되는 CloudHSM CLI 키 속성 목록은 CloudHSM CLI의 주요 속성을 참조하십시오.

필수 여부: 예

<MGF>

마스크 생성 함수를 지정합니다.

> **Note**
>
> 마스크 생성 함수 해시 함수는 서명 메커니즘 해시 함수와 일치해야 합니다.

유효한 값:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

필수 여부: 예

<SALT_LENGTH>

소금 길이를 지정합니다.

필수: 예

관련 주제

- 크립토 사인
- 크립토 검증

관련 주제

- 크립토 검증

크립토 검증

crypto verify명령 그룹의 상위 범주로, 상위 범주와 결합하면 파일이 지정된 키로 서명되었는지 여부를 확인할 수 있습니다. crypto verify에는 다음과 같은 하위 명령이 있습니다.

- 크립토 검증 ecdsa
- 크립토 검증 rsa-pkcs
- 크립토 검증 rsa-pkcs-pss

이 crypto verify 명령은 서명된 파일을 소스 파일과 비교하고 지정된 공개 키 및 서명 메커니즘을 기반으로 이들 파일이 암호학적으로 관련되어 있는지 분석합니다.

> **Note**
>
> 작업을 통해 파일에 AWS CloudHSM 로그인할 수 있습니다. 크립토 사인

크립토 검증 ecdsa

이 crypto verify ecdsa 명령은 다음 작업을 완료하는 데 사용됩니다.

- 지정된 공개 키로 HSM에서 파일이 서명되었는지 확인합니다.
- ECDSA 서명 메커니즘을 사용하여 서명이 생성되었는지 확인하십시오.

- 서명된 파일을 소스 파일과 비교하고 지정된 ecdsa 공개 키 및 서명 메커니즘을 기반으로 두 파일이 암호학적으로 관련되어 있는지 확인합니다.

crypto verify ecdsa명령을 사용하려면 먼저 클러스터에 EC 공개 키가 있어야 합니다. AWS CloudHSM `verify`속성이 로 설정된 키 임포트 펜 명령을 사용하여 EC 공개 키를 가져올 수 `true` 있습니다.

> **Note**
>
> CloudHSM 크립토 사인 CLI에서 하위 명령을 사용하여 서명을 생성할 수 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help crypto verify ecdsa
Verify with the ECDSA mechanism

Usage: crypto verify ecdsa --key-filter [<KEY_FILTER>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>> <--signature-
path <SIGNATURE_PATH>|--signature <SIGNATURE>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
```

```
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be verified
      --data <DATA>
          Base64 encoded data to be verified
      --signature-path <SIGNATURE_PATH>
          The path to where the signature is located
      --signature <SIGNATURE>
          Base64 encoded signature to be verified
  -h, --help
          Print help
```

예

이 예제는 ECDSA 서명 메커니즘과 해시 함수를 사용하여 생성된 서명을 확인하는 crypto verify ecdsa 데 사용하는 방법을 보여줍니다. SHA256 이 명령은 HSM의 공개 키를 사용합니다.

Example 예: Base64로 인코딩된 데이터를 사용하여 Base64로 인코딩된 서명 확인

```
aws-cloudhsm > crypto verify ecdsa --hash-function sha256 --key-filter attr.label=ec-
public --data YWJjMTIz --signature 4zki+FzjhP7Z/KqoQvh4ueMAxQQVp7FQguZ2wOS3Q5bzk
+Hc5irV5iTkuxQbropPttVFZ8V6FgR2fz+sPegwCw==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example 예: 서명 파일을 데이터 파일로 확인

```
aws-cloudhsm > crypto verify ecdsa --hash-function sha256 --key-filter attr.label=ec-
public --data-path data.txt --signature-path signature-file
{
   "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example 예: 허위 서명 관계 증명

이 명령은 ECDSA 서명 메커니즘을 ecdsa-public 사용하여 에 있는 데이터가 레이블이 있는 공개 키로 서명되었는지 여부를 확인하여 서명을 생성합니다. /home/data /home/signature 지정된 인수가 실제 서명 관계를 구성하지 않기 때문에 이 명령은 오류 메시지를 반환합니다.

```
aws-cloudhsm > crypto verify ecdsa --hash-function sha256 --
key-filter attr.label=ec-public --data aW52YWxpZA== --signature
 +ogk7M7S3iTqFg3SndJfd91dZFr5Qo6YixJl8JwcvqqVgsVuO6o+VKvTRjz0/V05kf3JJbBLr87Q
+wLWcMAJfA==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<DATA>

서명할 Base64로 인코딩된 데이터입니다.

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<DATA_PATH>

서명할 데이터의 위치를 지정합니다.

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<HASH_FUNCTION>

해시 함수를 지정합니다.

유효한 값:

- sha1
- sha224
- sha256
- sha384

- sha512

필수 여부: 예

<KEY_FILTER>

일치하는 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록.

지원되는 CloudHSM CLI 키 속성 목록은 CloudHSM CLI의 주요 속성을 참조하십시오.

필수 여부: 예

<SIGNATURE>

Base64로 인코딩된 서명.

필수: 예 (서명 경로를 통해 제공한 경우 제외)

<SIGNATURE_PATH>

서명 위치를 지정합니다.

필수: 예 (서명 경로를 통해 제공한 경우 제외)

관련 주제

- 크립토 사인
- 크립토 검증

크립토 검증 rsa-pkcs

이 crypto verify rsa-pkcs 명령은 다음 작업을 완료하는 데 사용됩니다.

- 지정된 공개 키로 HSM에서 파일이 서명되었는지 확인합니다.
- `RSA-PKCS`서명 메커니즘을 사용하여 서명이 생성되었는지 확인하십시오.
- 서명된 파일을 소스 파일과 비교하고 지정된 rsa 공개 키 및 서명 메커니즘을 기반으로 두 파일이 암호학적으로 관련되어 있는지 확인합니다.

crypto verify rsa-pkcs명령을 사용하려면 먼저 클러스터에 RSA 공개 키가 있어야 합니다. AWS CloudHSM

> **Note**
>
> 하위 명령과 함께 CloudHSM CLI를 사용하여 서명을 생성할 수 있습니다. 크립토 사인

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help crypto verify rsa-pkcs
Verify with the RSA-PKCS mechanism

Usage: crypto verify rsa-pkcs --key-filter [<KEY_FILTER>...] --hash-
function <HASH_FUNCTION> <--data-path <DATA_PATH>|--data <DATA>> <--signature-
path <SIGNATURE_PATH>|--signature <SIGNATURE>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be verified
      --data <DATA>
          Base64 encoded data to be verified
      --signature-path <SIGNATURE_PATH>
          The path to where the signature is located
```

```
      --signature <SIGNATURE>
          Base64 encoded signature to be verified
  -h, --help
          Print help
```

예

이 예제는 RSA-PKCS 서명 메커니즘 및 해시 함수를 사용하여 생성된 서명을 확인하는 crypto verify rsa-pkcs 데 사용하는 방법을 보여줍니다. SHA256 이 명령은 HSM의 공개 키를 사용합니다.

Example 예: Base64로 인코딩된 데이터를 사용하여 Base64로 인코딩된 서명 확인

```
aws-cloudhsm > crypto verify rsa-pkcs --hash-function sha256 --key-filter
 attr.label=rsa-public --data YWJjMTIz --signature XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example 예: 서명 파일을 데이터 파일로 확인

```
aws-cloudhsm > crypto verify rsa-pkcs --hash-function sha256 --key-filter
 attr.label=rsa-public --data-path data.txt --signature-path signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example 예: 허위 서명 관계 증명

이 명령은 RSAPKCS 서명 메커니즘을 rsa-public 사용하여 레이블이 있는 공개 키로 잘못된 데이터에 서명되었는지 확인하여 서명을 생성합니다. /home/signature 지정된 인수가 진정한 서명 관계를 구성하지 않기 때문에 이 명령은 오류 메시지를 반환합니다.

```
aws-cloudhsm > crypto verify rsa-pkcs --hash-function sha256 --key-filter
 attr.label=rsa-public --data aW52YWxpZA== --signature XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<DATA>

서명할 Base64로 인코딩된 데이터.

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<DATA_PATH>

서명할 데이터의 위치를 지정합니다.

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<HASH_FUNCTION>

해시 함수를 지정합니다.

유효한 값:

- sha1
- sha224
- sha256
- sha384
- sha512

필수 여부: 예

<KEY_FILTER>

일치하는 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록.

지원되는 CloudHSM CLI 키 속성 목록은 CloudHSM CLI의 주요 속성을 참조하십시오.

필수 여부: 예

<SIGNATURE>

Base64로 인코딩된 서명.

필수: 예 (서명 경로를 통해 제공한 경우 제외)

<SIGNATURE_PATH>

서명 위치를 지정합니다.

필수: 예 (서명 경로를 통해 제공한 경우 제외)

관련 주제

- 크립토 사인
- 크립토 검증

크립토 검증 rsa-pkcs-pss

이 crypto sign rsa-pkcs-pss 명령은 다음 작업을 완료하는 데 사용됩니다.

- 지정된 공개 키로 HSM에서 파일이 서명되었는지 확인합니다.
- RSA-PKCS-PSS 서명 메커니즘을 사용하여 서명이 생성되었는지 확인하십시오.
- 서명된 파일을 소스 파일과 비교하고 지정된 rsa 공개 키 및 서명 메커니즘을 기반으로 두 파일이 암호학적으로 관련되어 있는지 확인합니다.

crypto verify rsa-pkcs-pss명령을 사용하려면 먼저 클러스터에 RSA 공개 키가 있어야 합니다. AWS CloudHSM 속성이 로 설정된 키 가져오기 pem 명령 (ADD UNWRAP LINK HERE) 을 사용하여 RSA 공개 키를 가져올 수 있습니다. `verify true`

> **Note**
>
> 하위 명령과 함께 CloudHSM CLI를 사용하여 서명을 생성할 수 있습니다. 크립토 사인

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help crypto verify rsa-pkcs-pss
Verify with the RSA-PKCS-PSS mechanism

Usage: crypto verify rsa-pkcs-pss --key-filter [<KEY_FILTER>...] --hash-
function <HASH_FUNCTION> --mgf <MGF> --salt-length >SALT_LENGTH< <--data-
path <DATA_PATH>|--data <DATA> <--signature-path <SIGNATURE_PATH>|--
signature <SIGNATURE>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --key-filter [<KEY_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key
      --hash-function <HASH_FUNCTION>
          [possible values: sha1, sha224, sha256, sha384, sha512]
      --data-path <DATA_PATH>
          The path to the file containing the data to be verified
      --data <DATA>
          Base64 encoded data to be verified
      --signature-path <SIGNATURE_PATH>
```

```
          The path to where the signature is located
      --signature <SIGNATURE>
          Base64 encoded signature to be verified
      --mgf <MGF>
          The mask generation function [possible values: mgf1-sha1, mgf1-sha224, mgf1-
sha256, mgf1-sha384, mgf1-sha512]
      --salt-length <SALT_LENGTH>
          The salt length
  -h, --help
          Print help
```

예

이 예제는 RSA-PKCS-PSS 서명 메커니즘 및 해시 함수를 사용하여 생성된 서명을 확인하는 crypto verify rsa-pkcs-pss 데 사용하는 방법을 보여줍니다. SHA256 이 명령은 HSM의 공개 키를 사용합니다.

Example 예: Base64로 인코딩된 데이터를 사용하여 Base64로 인코딩된 서명 확인

```
aws-cloudhsm > crypto verify rsa-pkcs-pss --key-filter attr.label=rsa-public
 --hash-function sha256 --data YWJjMTIz --salt-length 10 --mgf mgf1-sha256
 --signature H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg==
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
}
```

Example 예: 서명 파일을 데이터 파일로 확인

```
aws-cloudhsm > crypto verify rsa-pkcs-pss --key-filter attr.label=rsa-public --hash-
function sha256 --data-path data.txt --salt-length 10 --mgf mgf1-sha256 --signature
 signature-file
{
  "error_code": 0,
  "data": {
    "message": "Signature verified successfully"
  }
```

```
}
```

Example 예: 허위 서명 관계 증명

이 명령은 RSAPKCSPSS 서명 메커니즘을 `rsa-public` 사용하여 레이블이 있는 공개 키로 잘못된 데이터에 서명되었는지 확인하여 서명을 생성합니다. `/home/signature` 지정된 인수가 진정한 서명 관계를 구성하지 않기 때문에 이 명령은 오류 메시지를 반환합니다.

```
aws-cloudhsm > crypto verify rsa-pkcs-pss --key-filter attr.label=rsa-public
 --hash-function sha256 --data aW52YWxpZA== --salt-length 10 --mgf mgf1-sha256
 --signature H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg==
{
  "error_code": 1,
  "data": "Signature verification failed"
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<DATA>

서명할 Base64로 인코딩된 데이터

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<DATA_PATH>

서명할 데이터의 위치를 지정합니다.

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<HASH_FUNCTION>

해시 함수를 지정합니다.

유효한 값:

- sha1
- sha224
- sha256
- sha384
- sha512

필수 여부: 예

<KEY_FILTER>

일치하는 키를 선택하기 위한 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록

지원되는 CloudHSM CLI 키 속성 목록은 CloudHSM CLI의 주요 속성을 참조하십시오.

필수 여부: 예

<MFG>

마스크 생성 함수를 지정합니다.

> **Note**
>
> 마스크 생성 함수 해시 함수는 서명 메커니즘 해시 함수와 일치해야 합니다.

유효한 값:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

필수 여부: 예

<SIGNATURE>

Base64로 인코딩된 서명

필수: 예 (서명 경로를 통해 제공한 경우 제외)

<SIGNATURE_PATH>

서명의 위치를 지정합니다.

필수: 예 (서명 경로를 통해 제공한 경우 제외)

관련 주제

- 크립토 사인
- 크립토 검증

키

key은 상위 카테고리와 결합될 때 키에 특정한 명령을 생성하는 명령 그룹의 상위 카테고리입니다. 현재 이 범주는 다음과 같은 명령으로 구성되어 있습니다.

- 키 삭제
- 키 생성-파일
- 키 generate-asymmetric-pair
  - 키 generate-asymmetric-pair RSA
  - 키 generate-asymmetric-pair 등
- 키 생성-대칭
  - 키 생성-대칭 aes
  - 키 생성-대칭 일반 보안
- 키 임포트 펜
- 키 목록
- 키 리플리케이트
- 키 세트-속성
- 키 공유
- 키 공유 취소
- 키 언랩
- 키 랩

키 삭제

CloudHSM CLI의 key delete 명령을 사용하여 클러스터에서 키를 삭제합니다. AWS CloudHSM 키는 한 번에 하나만 삭제할 수 있습니다. 키 페어에서 키 하나를 삭제해도 해당 페어의 다른 키에는 아무 영향을 미치지 않습니다.

키를 생성하고 결과적으로 소유한 CU만 키를 삭제할 수 있습니다. 키를 공유하지만 소유하지 않은 사용자는 암호화 작업에 키를 사용할 수 있지만 삭제할 수는 없습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key delete
Delete a key in the HSM cluster

Usage: key delete [OPTIONS] --filter [<FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --filter [<FILTER>...]     Key reference (e.g. key-reference=0xabc)
 or space separated list of key attributes in the form of
 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key for deletion
  -h, --help                     Print help
```

예

```
aws-cloudhsm > key delete --filter attr.label="ec-test-public-key"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
```

```
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<FILTER>

삭제할 일치하는 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록입니다.

지원되는 CloudHSM CLI 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수: 예

관련 주제

- 키 목록
- 키 생성-파일
- 키 공유 취소
- CloudHSM CLI의 키 속성
- CloudHSM CLI를 사용하여 키를 필터링

키 생성-파일

이 key generate-file 명령은 HSM에서 비대칭 키를 내보냅니다. 대상이 개인 키인 경우 개인 키에 대한 참조는 가짜 PEM 형식으로 내보내집니다. 대상이 공개 키인 경우 공개 키 바이트는 PEM 형식으로 내보내집니다.

실제 개인 키 자료는 포함하지 않고 대신 HSM의 개인 키를 참조하는 가짜 PEM 파일은 웹 서버에서 SSL/TLS 오프로드를 설정하는 데 사용될 수 있습니다. AWS CloudHSM자세한 내용은 SSL/TLS 오프로드를 참조하십시오.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key generate-file
Generate a key file from a key in the HSM cluster. This command does not export any
 private key data from the HSM

Usage: key generate-file --encoding <ENCODING> --path <PATH> --filter [<FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --encoding <ENCODING>
          Encoding format for the key file

          Possible values:
          - reference-pem: PEM formatted key reference (supports private keys)
          - pem:           PEM format (supports public keys)

      --path <PATH>
          Filepath where the key file will be written

      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key for file generation

  -h, --help
          Print help (see a summary with '-h')
```

예

이 예제는 클러스터에서 키 파일을 생성하는 key generate-file 데 사용하는 방법을 보여줍니다. AWS CloudHSM

Example

```
aws-cloudhsm > key generate-file --encoding reference-pem --path /tmp/ec-private-
key.pem --filter attr.label="ec-test-private-key"
{
  "error_code": 0,
  "data": {
    "message": "Successfully generated key file"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<FILTER>

삭제할 일치하는 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록입니다.

지원되는 CloudHSM CLI 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 아니요

<ENCODING>

키 파일의 인코딩 형식을 지정합니다.

필수 여부: 예

<PATH>

키 파일이 기록될 파일 경로를 지정합니다.

필수: 예

관련 주제

- CloudHSM CLI의 키 속성
- CloudHSM CLI를 사용하여 키를 필터링

- 키 generate-asymmetric-pair
- 키 생성-대칭

키 generate-asymmetric-pair

key generate-asymmetric-pair은 명령 그룹의 상위 범주로, 상위 범주와 결합하면 비대칭 키 쌍을 생성하는 명령이 생성됩니다. 현재 이 범주는 다음과 같은 명령으로 구성되어 있습니다.

- 키 generate-asymmetric-pair 등
- 키 generate-asymmetric-pair RSA

키 generate-asymmetric-pair 등

CloudHSM CLI의 key asymmetric-pair ec 명령을 사용하여 클러스터에서 비대칭 EC (타원곡선) 키 쌍을 생성합니다. AWS CloudHSM

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key generate-asymmetric-pair ec
Generate an Elliptic-Curve Cryptography (ECC) key pair

Usage: key generate-asymmetric-pair ec [OPTIONS] --public-label <PUBLIC_LABEL> --
private-label <PRIVATE_LABEL> --curve <CURVE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --public-label <PUBLIC_LABEL>
          Label for the public key
```

```
      --private-label <PRIVATE_LABEL>
          Label for the private key
      --session
          Creates a session key pair that exists only in the current session. The key
 cannot be recovered after the session ends
      --curve <CURVE>
          Elliptic curve used to generate the key pair [possible values: prime256v1,
 secp256r1, secp224r1, secp384r1, secp256k1, secp521r1]
      --public-attributes [<PUBLIC_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated EC public key
 in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
      --private-attributes [<PRIVATE_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated EC private
 key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
  -h, --help
          Print help
```

예제

이 예에서는 key generate-asymmetric-pair ec 명령을 사용하여 EC 키 페어를 생성하는 방법을 보여줍니다.

Example 예: EC 키 페어 생성

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp224r1 \
    --public-label ec-public-key-example \
    --private-label ec-private-key-example
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x000000000012000b",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
```

```
        "key-type": "ec",
        "label": "ec-public-key-example",
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 57,
        "ec-point":
 "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f38a
        "curve": "secp224r1"
      }
    },
"private_key": {
      "key-reference": "0x000000000012000c",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-private-key-example",
```

```
        "id": "",
        "check-value": "0xd7c1a7",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 122,
        "ec-point":
 "0x047096513df542250a6b228fd9cb67fd0c903abc93488467681974d6f371083fce1d79da8ad1e9ede745fb9f38a
        "curve": "secp224r1"
      }
    }
  }
}
```

Example 예: 선택적 속성을 사용하여 EC 키 페어 생성

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp224r1 \
    --public-label ec-public-key-example \
    --private-label ec-private-key-example \
    --public-attributes token=true encrypt=true \
    --private-attributes token=true decrypt=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x00000000002806eb",
```

```
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-public-key-example",
        "id": "",
        "check-value": "0xedef86",
        "class": "public-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 57,
        "ec-point":
"0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514
        "curve": "secp224r1"
      }
    },
    "private_key": {
      "key-reference": "0x0000000000280c82",
      "key-info": {
        "key-owners": [
```

```
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "ec",
        "label": "ec-private-key-example",
        "id": "",
        "check-value": "0xedef86",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 122,
        "ec-point":
 "0x0487af31882189ec29eddf17a48e8b9cebb075b7b5afc5522fe9c83a029a450cc68592889a1ebf45f32240da514
        "curve": "secp224r1"
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<CURVE>

타원 곡선의 식별자를 지정합니다.

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1

필수 여부: 예

<PUBLIC_KEY_ATTRIBUTES>

생성된 EC 퍼블릭 키에 대해 설정할 키 속성의 공백으로 구분된 목록을 `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`(예: `token=true`) 형식으로 지정합니다.

지원되는 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 아니요

<PUBLIC_LABEL>

퍼블릭 키에 대해 사용자 정의 레이블을 지정합니다. 클라이언트 SDK 5.11 이상의 경우 허용되는 최대 `label` 크기는 127자입니다. 클라이언트 SDK 5.10 이전 버전은 126자로 제한됩니다.

필수 여부: 예

<PRIVATE_KEY_ATTRIBUTES>

생성된 EC 프라이빗 키에 대해 설정할 키 속성의 공백으로 구분된 목록을 `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`(예: `token=true`) 형식으로 지정합니다.

지원되는 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 아니요

<PRIVATE_LABEL>

프라이빗 키에 대해 사용자 정의 레이블을 지정합니다. 클라이언트 SDK 5.11 이상의 경우 허용되는 최대 `label` 크기는 127자입니다. 클라이언트 SDK 5.10 이전 버전은 126자로 제한됩니다.

필수 여부: 예

<SESSION>

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

기본적으로 생성되는 키는 영구(토큰) 키입니다. <SESSION>을 전달하면 이를 변경하여 이 인수로 생성된 키가 세션(임시) 키인지 확인합니다.

필수 여부: 아니요

관련 주제

- CloudHSM CLI의 키 속성
- CloudHSM CLI를 사용하여 키를 필터링

키 generate-asymmetric-pair RSA

key generate-asymmetric-pair rsa명령을 사용하면 클러스터에 비대칭 RSA 키 쌍이 생성됩니다. AWS CloudHSM

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key generate-asymmetric-pair rsa
Generate an RSA key pair

Usage: key generate-asymmetric-pair rsa [OPTIONS] --public-label <PUBLIC_LABEL>
 --private-label <PRIVATE_LABEL> --modulus-size-bits <MODULUS_SIZE_BITS> --public-
exponent <PUBLIC_EXPONENT>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --public-label <PUBLIC_LABEL>
          Label for the public key
      --private-label <PRIVATE_LABEL>
          Label for the private key
      --session
          Creates a session key pair that exists only in the current session. The key
 cannot be recovered after the session ends
      --modulus-size-bits <MODULUS_SIZE_BITS>
          Modulus size in bits used to generate the RSA key pair
      --public-exponent <PUBLIC_EXPONENT>
          Public exponent used to generate the RSA key pair
      --public-attributes [<PUBLIC_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated RSA public
 key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
      --private-attributes [<PRIVATE_KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated RSA private
 key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
  -h, --help
          Print help
```

예제

이 예시에서는 `key generate-asymmetric-pair rsa`를 사용하여 RSA 키 쌍을 생성하는 방법을 보여줍니다.

Example 예시: RSA 키 쌍 생성하기

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
```

```
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000160010",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-public-key-example",
        "id": "",
        "check-value": "0x498e1f",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
```

```
        "modulus":
 "0xdfca0669dc8288ed3bad99509bd21c7e6192661407021b3f4cdf4a593d939dd24f4d641af8e4e73b04c847731c6
e89a065e7d1a46ced96b46b909db2ab6be871ee700fd0a448b6e975bb64cae77c49008749212463e37a577baa57ce3e
bcebb7d20bd6df1948ae336ae23b52d73b7f3b6acc2543edb6358e08d326d280ce489571f4d34e316a2ea1904d513ca
        "modulus-size-bits": 2048
      }
    },
"private_key": {
      "key-reference": "0x0000000000160011",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-private-key-example",
        "id": "",
        "check-value": "0x498e1f",
        "class": "private-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1217,
```

```
        "public-exponent": "0x010001",
        "modulus":
 "0xdfca0669dc8288ed3bad99509bd21c7e6192661407021b3f4cdf4a593d939dd24f4d641af8e4e73b04c847731c6
        "modulus-size-bits": 2048
      }
    }
  }
}
```

Example 예시: 선택적 속성을 사용하여 RSA 키 쌍 생성하기

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
--public-exponent 65537 \
--modulus-size-bits 2048 \
--public-label rsa-public-key-example \
--private-label rsa-private-key-example \
--public-attributes token=true encrypt=true \
--private-attributes token=true decrypt=true
{
  "error_code": 0,
  "data": {
    "public_key": {
      "key-reference": "0x0000000000280cc8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-public-key-example",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "public-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
```

```
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4
73a80fdb457aa7b20cd61e486c326e2cfd5e124a7f6a996437437812b542e3caf85928aa866f0298580f7967ee6aa01
f6e6296d6c116d5744c6d60d14d3bf3cb978fe6b75ac67b7089bafd50d8687213b31abc7dc1bad422780d29c851d510
133022653225bd129f8491101725e9ea33e1ded83fb57af35f847e532eb30cd7e726f23910d2671c6364092e834697e
ac3160f0ca9725d38318b7",
        "modulus-size-bits": 2048
      }
    },
    "private_key": {
      "key-reference": "0x0000000000280cc7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa-private-key-example",
        "id": "",
        "check-value": "0x01fe6e",
        "class": "private-key",
        "encrypt": false,
```

```
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 1217,
        "public-exponent": "0x010001",
        "modulus":
 "0xb1d27e857a876f4e9fd5de748a763c539b359f937eb4b4260e30d1435485a732c878cdad9c72538e2215351b1d4
        "modulus-size-bits": 2048
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<MODULUS_SIZE_BITS>

모듈러스 길이를 비트 단위로 지정합니다. 최소값은 2048입니다.

필수 여부: 예

<PRIVATE_KEY_ATTRIBUTES>

생성된 RSA 프라이빗 키에 대해 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE의 형식(예: token=true)으로 설정할 키 속성의 공백으로 구분된 목록을 지정합니다.

지원되는 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 아니요

<PRIVATE_LABEL>

프라이빗 키에 대해 사용자 정의 레이블을 지정합니다. 클라이언트 SDK 5.11 이상의 경우 허용되는 최대 `label` 크기는 127자입니다. 클라이언트 SDK 5.10 이전 버전은 126자로 제한됩니다.

필수 여부: 예

<PUBLIC_EXPONENT>

퍼블릭 지수를 지정합니다. 값은 65537 이상의 홀수여야 합니다.

필수 여부: 예

<PUBLIC_KEY_ATTRIBUTES>

생성된 RSA 퍼블릭 키에 대해 `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`의 형식(예: `token=true`)으로 설정할 키 속성의 공백으로 구분된 목록을 지정합니다.

지원되는 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 아니요

<PUBLIC_LABEL>

퍼블릭 키에 대해 사용자 정의 레이블을 지정합니다. 클라이언트 SDK 5.11 이상의 경우 허용되는 최대 `label` 크기는 127자입니다. 클라이언트 SDK 5.10 이전 버전은 126자로 제한됩니다.

필수 여부: 예

<SESSION>

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

기본적으로 생성되는 키는 영구(토큰) 키입니다. <SESSION>을 전달하면 이를 변경하여 이 인수로 생성된 키가 세션(임시) 키인지 확인합니다.

필수 여부: 아니요

관련 주제

- CloudHSM CLI의 키 속성
- CloudHSM CLI를 사용하여 키를 필터링

키 생성-대칭

key generate-symmetric은 상위 범주와 결합할 경우 대칭 키를 생성하는 명령을 생성하는 명령 그룹의 상위 범주입니다. 현재 이 범주는 다음과 같은 명령으로 구성되어 있습니다.

- 키 생성-대칭 aes
- 키 생성-대칭 일반 보안

키 생성-대칭 aes

이 key generate-symmetric aes 명령은 클러스터에 대칭 AES 키를 생성합니다. AWS CloudHSM

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key generate-symmetric aes
Generate an AES key

Usage: key generate-symmetric aes [OPTIONS] --label <LABEL> --key-length-
bytes <KEY_LENGTH_BYTES>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
```

```
      --label <LABEL>
          Label for the key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
      --key-length-bytes <KEY_LENGTH_BYTES>
          Key length in bytes
      --attributes [<KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated AES key in
 the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
  -h, --help
          Print help
```

예제

다음 예시에서는 key generate-symmetric aes 명령을 사용하여 AES 키를 생성하는 방법을 보여 줍니다.

Example 예시: AES 키 생성하기

```
aws-cloudhsm > key generate-symmetric aes \
--label example-aes \
--key-length-bytes 24
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e06bf",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "aes",
        "label": "example-aes",
        "id": "",
        "check-value": "0x9b94bd",
```

```
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 24
      }
    }
  }
}
```

Example 예시: 선택적 속성을 사용하여 AES 키 생성하기

```
aws-cloudhsm > key generate-symmetric aes \
--label example-aes \
--key-length-bytes 24 \
--attributes decrypt=true encrypt=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e06bf",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
```

```
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "aes",
        "label": "example-aes",
        "id": "",
        "check-value": "0x9b94bd",
        "class": "secret-key",
        "encrypt": true,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 24
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<KEY_ATTRIBUTES>

생성된 AES 키에 대해 설정할 키 속성의 공백으로 구분된 목록을 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (예: `token=true`)의 형식으로 지정합니다.

지원되는 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 아니요

<KEY-LENGTH-BYTES>

키 크기를 바이트 단위로 지정합니다.

유효한 값:

- 16, 24 및 32

필수 여부: 예

<LABEL>

AES 키에 대해 사용자 정의 레이블을 지정합니다. 클라이언트 SDK 5.11 이상의 경우 허용되는 최대 `label` 크기는 127자입니다. 클라이언트 SDK 5.10 이전 버전은 126자로 제한됩니다.

필수 여부: 예

<SESSION>

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

기본적으로 생성되는 키는 영구(토큰) 키입니다. <SESSION>을 전달하면 이를 변경하여 이 인수로 생성된 키가 세션(임시) 키인지 확인합니다.

필수 여부: 아니요

관련 주제

- CloudHSM CLI의 키 속성

- CloudHSM CLI를 사용하여 키를 필터링

키 생성-대칭 일반 보안 암호

key generate-asymmetric-pair 명령은 AWS CloudHSM 클러스터에 대칭형 일반 비밀 키를 생성합니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > key help generate-symmetric generic-secret
Generate a generic secret key

Usage: key generate-symmetric generic-secret [OPTIONS] --label <LABEL> --key-length-
bytes <KEY_LENGTH_BYTES>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --label <LABEL>
          Label for the key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
      --key-length-bytes <KEY_LENGTH_BYTES>
          Key length in bytes
      --attributes [<KEY_ATTRIBUTES>...]
          Space separated list of key attributes to set for the generated generic
 secret key in the form of KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE
  -h, --help
          Print help
```

예제

다음 예제에서는 key generate-symmetric generic-secret 명령을 사용하여 일반 비밀 키를 생성하는 방법을 보여 줍니다.

Example 예: 일반 비밀 키 생성

```
aws-cloudhsm > key generate-symmetric generic-secret \
--label example-generic-secret \
--key-length-bytes 256
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e08fd",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "generic-secret",
        "label": "example-generic-secret",
        "id": "",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
```

```
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 256
      }
    }
  }
}
```

Example 예: 선택적 속성을 포함하는 일반 비밀 키 생성

```
aws-cloudhsm > key generate-symmetric generic-secret \
--label example-generic-secret \
--key-length-bytes 256 \
--attributes token=true encrypt=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000002e08fd",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "generic-secret",
        "label": "example-generic-secret",
        "id": "",
        "class": "secret-key",
        "encrypt": true,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
```

```
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 256
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<KEY_ATTRIBUTES>

생성된 AES 키에 대해 설정할 키 속성의 공백으로 구분된 목록을 `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (예: `token=true`)의 형식으로 지정합니다.

지원되는 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 아니요

<KEY-LENGTH-BYTES>

키 크기를 바이트 단위로 지정합니다.

유효한 값:

- 1~800

필수 여부: 예

<LABEL>

일반 비밀 키에 대한 사용자 정의 레이블을 지정합니다. 클라이언트 SDK 5.11 이상의 경우 허용되는 최대 `label` 크기는 127자입니다. 클라이언트 SDK 5.10 이전 버전은 126자로 제한됩니다.

필수 여부: 예

<SESSION>

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

기본적으로 생성되는 키는 영구(토큰) 키입니다. <SESSION>을 전달하면 이를 변경하여 이 인수로 생성된 키가 세션(임시) 키인지 확인합니다.

필수 여부: 아니요

관련 주제

- CloudHSM CLI의 키 속성
- CloudHSM CLI를 사용하여 키를 필터링

키 임포트 펜

의 key import pem 명령은 PEM 형식 키를 HSM으로 AWS CloudHSM 가져옵니다. 이 명령을 사용하여 HSM 외부에서 생성된 퍼블릭 키를 가져올 수 있습니다.

> **Note**
>
> 키 생성-파일명령을 사용하여 공개 키에서 표준 PEM 파일을 만들거나 개인 키에서 참조 PEM 파일을 만들 수 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key import pem
Import key from a PEM file

Usage: key import pem [OPTIONS] --path <PATH> --label <LABEL> --key-type-
class <KEY_TYPE_CLASS>
Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --path PATH>
          Path where the key is located in PEM format
      --label LABEL>
          Label for the imported key
      --key-type-class KEY_TYPE_CLASS>
          Key type and class of the imported key [possible values: ec-public, rsa-
public]
      --attributes [IMPORT_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the imported key
  -h, --help
          Print help
```

예제

이 예제에서는 key import pem 명령을 사용하여 파일에서 PEM 형식의 RSA 공개 키를 가져오는 방법을 보여줍니다.

Example 예: RSA 퍼블릭 키 가져오기

```
aws-cloudhsm > key import pem --path /home/example --label example-imported-key --key-
type-class rsa-public
{
  "error_code": 0,
  "data": {
    "key": {
```

```
      "key-reference": "0x00000000001e08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "example-imported-key",
        "id": "0x",
        "check-value": "0x99fe93",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": false,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
"0x8e9c172c37aa22ed1ce25f7c3a7c936dadc532201400128b044ebb4b96#··3e4930ab910df5a2896eaeb8853cfe
        "modulus-size-bits": 2048
      }
    },
    "message": "Successfully imported key"
  }
```

```
}
```

Example 예: 선택적 속성이 있는 RSA 공개 키 가져오기

```
aws-cloudhsm > key import pem --path /home/example --label example-imported-key-with-
attributes --key-type-class rsa-public --attributes verify=true
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001e08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "session"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "example-imported-key-with-attributes",
        "id": "0x",
        "check-value": "0x99fe93",
        "class": "public-key",
        "encrypt": false,
        "decrypt": false,
        "token": false,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": false,
        "sign": false,
        "trusted": false,
        "unwrap": false,
        "verify": true,
```

```
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 512,
        "public-exponent": "0x010001",
        "modulus":
 "0x8e9c172c37aa22ed1ce25f7c3a7c936dadc532201400128b044ebb4b96#··3e4930ab910df5a2896eaeb8853cfe
        "modulus-size-bits": 2048
      }
    },
    "message": "Successfully imported key"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<PATH>

키 파일이 있는 파일 경로를 지정합니다.

필수 여부: 예

<LABEL>

가져온 키의 사용자 정의 레이블을 지정합니다. `label`에 허용되는 최대 크기는 126자입니다.

필수 여부: 예

<KEY_TYPE_CLASS>

래핑된 키의 키 유형 및 클래스

가능한 값은 다음과 같습니다.

- ec-퍼블릭
- RS-퍼블릭

필수 여부: 예

<IMPORT_KEY_ATTRIBUTES>

가져온 키에 대해 설정할 키 속성의 공백으로 구분된 목록 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (예:token=true) 을 지정합니다. 지원되는 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 아니요

관련 주제

- 크립토 사인
- 크립토 검증

키 목록

이 key list 명령은 AWS CloudHSM 클러스터에 있는 현재 사용자의 모든 키를 찾습니다. 출력에는 사용자가 소유 및 공유하는 키는 물론 CloudHSM 클러스터의 모든 퍼블릭 키가 포함됩니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

구문

```
aws-cloudhsm > help key list
List the keys the current user owns, shares, and all public keys in the HSM cluster

Usage: key list [OPTIONS]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select
 matching key(s) to list
      --max-items <MAX_ITEMS>
```

```
          The total number of items to return in the command's output. If the total
number of items available is more than the value specified, a next-token is provided
in the command's output. To resume pagination, provide the next-token value in the
starting-token argument of a subsequent command [default: 10]
     --starting-token <STARTING_TOKEN>
          A token to specify where to start paginating. This is the next-token from a
previously truncated response
 -v, --verbose
          If included, prints all attributes and key information for each matched key.
By default each matched key only displays its key-reference and label attribute
 -h, --help
          Print help
```

예제

다음 예에서는 key list 명령을 실행하는 다양한 방법을 보여줍니다.

Example 예: 모든 키 찾기 - 기본값

이 명령은 AWS CloudHSM 클러스터에 있는 로그인한 사용자의 키를 나열합니다.

> **Note**
>
> 기본적으로 현재 로그인한 사용자의 키 10개만 표시되고 `key-reference` 및 `label`만 출력으로 표시됩니다. 적절한 페이지 매김 옵션을 사용하여 출력으로 더 많거나 적은 키를 표시할 수 있습니다.

```
aws-cloudhsm > key list
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000000003d5",
        "attributes": {
          "label": "test_label_1"
        }
      },
      {
        "key-reference": "0x0000000000000626",
        "attributes": {
          "label": "test_label_2"
```

```
        }
      },.
      ...8 keys later...
    ],
    "total_key_count": 56,
    "returned_key_count": 10,
    "next_token": "10"
  }
}
```

Example 예: 모든 키 찾기 - verbose

출력에는 사용자가 소유 및 공유하는 키는 물론 HSM의 모든 퍼블릭 키가 포함됩니다.

> **Note**
>
> 참고: 기본적으로 현재 로그인한 사용자의 키 10개만 표시됩니다. 적절한 페이지 매김 옵션을 사용하여 출력으로 더 많거나 적은 키를 표시할 수 있습니다.

```
aws-cloudhsm > key list --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x000000000012000c",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "session"
        },
        "attributes": {
          "key-type": "ec",
          "label": "ec-test-private-key",
          "id": "",
          "check-value": "0x2a737d",
```

```
          "class": "private-key",
          "encrypt": false,
          "decrypt": false,
          "token": false,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": false,
          "trusted": false,
          "unwrap": false,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 122,
          "ec-point":
"0x0442d53274a6c0ec1a23c165dcb9ccdd72c64e98ae1a9594bb5284e752c746280667e11f1e983493c1c605e0a80
          "curve": "secp224r1"
        }
      },
      {
        "key-reference": "0x000000000012000d",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "session"
        },
        "attributes": {
          "key-type": "ec",
          "label": "ec-test-public-key",
          "id": "",
          "check-value": "0x2a737d",
          "class": "public-key",
          "encrypt": false,
```

```
          "decrypt": false,
          "token": false,
          "always-sensitive": false,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": false,
          "sign": false,
          "trusted": false,
          "unwrap": false,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 57,
          "ec-point":
 "0x0442d53274a6c0ec1a23c165dcb9ccdd72c64e98ae1a9594bb5284e752c746280667e11f1e983493c1c605e0a80
          "curve": "secp224r1"
        }
      }
    ],
      ...8 keys later...
    "total_key_count": 1580,
    "returned_key_count": 10
  }
}
```

Example 예: 페이지를 매긴 반환

다음 예에서는 두 개의 키만 표시하는 페이지가 매겨진 키 하위 집합을 표시합니다. 그런 다음 예제에서는 다음 두 키를 표시하기 위한 후속 호출을 제공합니다.

```
aws-cloudhsm > key list --verbose --max-items 2
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000000030",
        "key-info": {
```

```
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "98a6688d1d964ed7b45b9cec5c4b1909",
          "id": "",
          "check-value": "0xb28a46",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": false,
          "verify": true,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      },
      {
        "key-reference": "0x0000000000000042",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
```

```
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "4ad6cdcbc02044e09fa954143efde233",
          "id": "",
          "check-value": "0xc98104",
          "class": "secret-key",
          "encrypt": true,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": true,
          "wrap": true,
          "wrap-with-trusted": false,
          "key-length-bytes": 16
        }
      }
    ],
    "total_key_count": 1580,
    "returned_key_count": 2,
    "next_token": "2"
  }
}
```

다음 두 키를 표시하려면 후속 호출을 수행하면 됩니다.

```
aws-cloudhsm > key list --verbose --max-items 2 --starting-token 2
{
  "error_code": 0,
```

```
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x0000000000000081",
        "key-info": {
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "6793b8439d044046982e5b895791e47f",
          "id": "",
          "check-value": "0x3f986f",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": false,
          "verify": true,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      },
      {
        "key-reference": "0x0000000000000089",
        "key-info": {
```

```
          "key-owners": [
            {
              "username": "cu1",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "56b30fa05c6741faab8f606d3b7fe105",
          "id": "",
          "check-value": "0xe9201a",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": false,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": false,
          "verify": true,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      }
    ],
    "total_key_count": 1580,
    "returned_key_count": 2,
    "next_token": "4"
  }
}
```

CloudHSM CLI에서 키 필터링 메커니즘이 작동하는 방식을 보여주는 추가 예제는 CloudHSM CLI를 사용하여 키를 필터링을 참조하십시오.

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<FILTER>

일치하는 키를 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 나열하기 위한 형식의 키 참조 (예:`key-reference=0xabc`) 또는 공백으로 구분된 키 속성 목록

지원되는 CloudHSM CLI 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 아니요

<MAX_ITEMS>

명령의 출력에서 반환되는 항목의 총 수입니다. 사용 가능한 총 항목 수가 지정된 값을 초과하는 경우 명령의 출력에 다음 토큰이 제공됩니다. 페이지 매김을 재개하려면 후속 명령의 시작 토큰에 다음 토큰 값을 제공합니다.

필수 여부: 아니요

<STARTING_TOKEN>

페이지 매김을 시작할 위치를 지정하기 위한 토큰입니다. 이는 이전에 잘린 응답의 다음 토큰입니다.

필수 여부: 아니요

<VERBOSE>

포함된 경우 일치하는 각 키의 모든 속성 및 키 정보를 인쇄합니다. 기본적으로 일치하는 각 키는 해당 키 참조 및 레이블 속성만 표시합니다.

필수 여부: 아니요

관련 주제

- 키 삭제

- 키 생성-파일
- 키 공유 취소
- CloudHSM CLI의 키 속성
- CloudHSM CLI를 사용하여 키를 필터링

키 리플리케이트

이 key replicate 명령은 원본 AWS CloudHSM 클러스터에서 대상 클러스터로 키를 복제합니다. AWS CloudHSM

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

> **Note**
>
> Crypto 사용자는 키를 소유해야 이 명령을 사용할 수 있습니다.

요구 사항

- 소스 및 대상 클러스터는 클론이어야 합니다. 즉, 하나가 다른 백업에서 생성되었거나 둘 다 공통 백업에서 생성되었음을 의미합니다. 자세한 정보는 백업에서 클러스터 생성을 참조하세요.
- 키 소유자가 대상 클러스터에 있어야 합니다. 또한 키를 다른 사용자와 공유하는 경우 해당 사용자가 대상 클러스터에도 있어야 합니다.
- 이 명령을 실행하려면 소스 클러스터와 대상 클러스터 모두에 CU로 로그인해야 합니다.
  - 단일 명령 모드에서 명령은 CLOUDHSM_PIN 및 CLOUDHSM_ROLE 환경 변수를 사용하여 소스 클러스터에서 인증합니다. 자세한 정보는 단일 명령 모드을 참조하세요. 대상 클러스터에 자격 증명을 제공하려면 DESTINATION_CLOUDHSM_PIN과 DESTINATION_CLOUDHSM_ROLE이라는 두 가지 추가 환경 변수를 설정해야 합니다.

```
$ export DESTINATION_CLOUDHSM_ROLE=crypto-user
```

```
$ export DESTINATION_CLOUDHSM_PIN=username:password
```

- 대화형 모드에서는 사용자가 소스 클러스터와 대상 클러스터 모두에 명시적으로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key replicate
Replicate a key from a source to a destination cluster

Usage: key replicate --filter [<FILTER>...] --source-cluster-id <SOURCE_CLUSTER_ID> --
destination-cluster-id <DESTINATION_CLUSTER_ID>

Options:
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select
 matching key on the source cluster
      --source-cluster-id <SOURCE_CLUSTER_ID>
          Source cluster ID
      --destination-cluster-id <DESTINATION_CLUSTER_ID>
          Destination cluster ID
  -h, --help
          Print help
```

예제

Example 예: 복제 키

이 명령은 원본 클러스터의 키를 복제된 대상 클러스터에 복제합니다.

```
crypto-user-1@cluster-1234abcdefg > key replicate \
      --filter attr.label=example-key \
      --source-cluster-id cluster-1234abcdefg \
      --destination-cluster-id cluster-2345bcdefgh
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x0000000000300006",
      "key-info": {
        "key-owners": [
          {
            "username": "crypto-user-1",
```

```
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "example-key",
        "id": "0x",
        "check-value": "0x5e118e",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": true,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    },
    "message": "Successfully replicated key"
  }
}
```

인수

<FILTER>

소스 클러스터에서 일치하는 키를 선택하기 위한 형식의 키 참조 (예:key-reference=0xabc) 또는 공백으로 구분된 키 속성 목록 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE

지원되는 CloudHSM CLI 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 예

<SOURCE_CLUSTER_ID>

소스 클러스터 ID

필수 여부: 예

<DESTINATION_CLUSTER_ID>

대상 클러스터 ID

필수: 예

관련 주제

- CloudHSM CLI를 사용하여 여러 클러스터에 연결

키 세트-속성

key set-attribute명령을 사용하여 AWS CloudHSM 클러스터의 키 속성을 설정합니다. 키를 생성하고 소유한 CU만 키 속성을 변경할 수 있습니다.

CloudHSM CLI에서 사용할 수 있는 키 속성의 목록은 CloudHSM CLI의 키 속성을 참조하십시오.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- 암호화 사용자(Crypto User)가 이 명령을 실행할 수 있습니다.
- 관리자는 신뢰할 수 있는 속성을 설정할 수 있습니다.

요구 사항

이 명령을 실행하려면 CU로 로그인해야 합니다. 신뢰할 수 있는 속성을 설정하려면 관리자 사용자로 로그인한 상태이어야 합니다.

구문

```
aws-cloudhsm > help key set-attribute
Set an attribute for a key in the HSM cluster
```

```
Usage: cloudhsm-cli key set-attribute [OPTIONS] --filter [<FILTER>...] --
name <KEY_ATTRIBUTE> --value <KEY_ATTRIBUTE_VALUE>

Options:
      --cluster-id <CLUSTER_ID>         Unique Id to choose which of the clusters in
 the config file to run the operation against. If not provided, will fall back to the
 value provided when interactive mode was started, or error
      --filter [<FILTER>...]            Key reference (e.g. key-
reference=0xabc) or space separated list of key attributes in the form of
 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a matching key to modify
      --name <KEY_ATTRIBUTE>            Name of attribute to be set
      --value <KEY_ATTRIBUTE_VALUE>...  Attribute value to be set
  -h, --help                            Print help
```

예시: 키 속성 설정

다음 예시는 key set-attribute 명령을 사용하여 레이블을 설정하는 방법을 보여줍니다.

Example

1. 다음과 같이 my_key 레이블과 함께 키를 사용합니다.

```
aws-cloudhsm > key set-attribute --filter attr.label=my_key --name encrypt --value
 false
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

2. key list 명령을 사용하여 encrypt 속성이 변경되었는지 확인합니다.

```
aws-cloudhsm > key list --filter attr.label=my_key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000006400ec",
        "key-info": {
          "key-owners": [
```

```
            {
              "username": "bob",
              "key-coverage": "full"
            }
          ],
          "shared-users": [],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "aes",
          "label": "my_key",
          "id": "",
          "check-value": "0x6bd9f7",
          "class": "secret-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": true,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": true,
          "unwrap": true,
          "verify": true,
          "wrap": true,
          "wrap-with-trusted": false,
          "key-length-bytes": 32
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<KEY_ATTRIBUTE>

키의 속성 이름을 지정합니다.

필수 여부: 예

<FILTER>

삭제할 일치하는 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록.

지원되는 CloudHSM CLI 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 아니요

<KEY_ATTRIBUTE_VALUE>

키 속성 값을 지정합니다.

필수 여부: 예

<KEY_REFERENCE>

키의 16진수 또는 10진수 표현입니다. (예: 키 핸들).

필수 여부: 아니요

관련 주제

- CloudHSM CLI를 사용하여 키를 필터링
- CloudHSM CLI의 키 속성

키 공유

이 key share 명령은 AWS CloudHSM 클러스터의 다른 CU와 키를 공유합니다.

키를 생성하고 결과적으로 소유한 CU만 키를 공유할 수 있습니다. 키를 공유하는 사용자는 해당 키를 암호화 작업에 사용할 수는 있지만 키를 삭제, 내보내기, 공유 또는 공유 해제할 수 없습니다. 또한 이러한 사용자는 키 속성을 변경할 수 없습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key share
Share a key in the HSM cluster with another user

Usage: key share --filter [<FILTER>...] --username <USERNAME> --role <ROLE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 matching key for sharing

      --username <USERNAME>
          A username with which the key will be shared

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

  -h, --help
```

```
          Print help (see a summary with '-h')
```

예: 다른 CU와 키 공유

다음 예에서는 key share 명령을 사용하여 CU `alice`와 키를 공유하는 방법을 보여줍니다.

Example

1. key share 명령을 실행하여 `alice`와 키를 공유합니다.

```
aws-cloudhsm > key share --filter attr.label="rsa_key_to_share" attr.class=private-
key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key shared successfully"
  }
}
```

2. key list 명령을 실행합니다.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
```

```
            "key-coverage": "full"
          },
          {
            "username": "cu4",
            "key-coverage": "full"
          },
          {
            "username": "cu5",
            "key-coverage": "full"
          },
          {
            "username": "cu6",
            "key-coverage": "full"
          },
          {
            "username": "cu7",
            "key-coverage": "full"
          },
          {
            "username": "alice",
            "key-coverage": "full"
          }
        ],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "rsa",
        "label": "rsa_key_to_share",
        "id": "",
        "check-value": "0xae8ff0",
        "class": "private-key",
        "encrypt": false,
        "decrypt": true,
        "token": true,
        "always-sensitive": true,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": true,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
```

```
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

3. 위 목록에서 `alice`이 `shared-users` 목록에 있는지 확인하십시오.

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<FILTER>

삭제할 일치하는 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록입니다.

지원되는 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 예

<USERNAME>

사용자의 친숙한 이름을 지정합니다. 최대 길이는 31자입니다. 허용되는 유일한 특수 문자는 밑줄 (_)입니다. 이 명령에서 사용자 이름은 대소문자를 구분하지 않으며, 사용자 이름은 항상 소문자로 표시됩니다.

필수: 예

<ROLE>

이 사용자에게 할당된 역할을 지정합니다. 이 파라미터는 필수 사항입니다. 사용자 역할을 가져오려면 user list 명령을 사용합니다. HSM에서의 사용자 유형에 대한 자세한 내용은 HSM 사용자 이해 섹션을 참조하십시오.

필수: 예

관련 주제

- CloudHSM CLI를 사용하여 키를 필터링
- CloudHSM CLI의 키 속성

키 공유 취소

이 key unshare 명령은 클러스터의 다른 CU와 키 공유를 AWS CloudHSM 해제합니다.

키를 생성하고 결과적으로 소유한 CU만 키를 공유 해제할 수 있습니다. 키를 공유하는 사용자는 해당 키를 암호화 작업에 사용할 수는 있지만 키를 삭제, 내보내기, 공유 또는 공유 해제할 수 없습니다. 또한 이러한 사용자는 키 속성을 변경할 수 없습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key unshare
Unshare a key in the HSM cluster with another user

Usage: key unshare --filter [<FILTER>...] --username <USERNAME> --role <ROLE>

Options:
      --cluster-id <CLUSTER_ID>
```

```
        Unique Id to choose which of the clusters in the config file to run the
operation against. If not provided, will fall back to the value provided when
interactive mode was started, or error

     --filter [<FILTER>...]
        Key reference (e.g. key-reference=0xabc) or space separated list of key
attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
matching key for unsharing

     --username <USERNAME>
        A username with which the key will be unshared

     --role <ROLE>
        Role the user has in the cluster

        Possible values:
        - crypto-user: A CryptoUser has the ability to manage and use keys
        - admin:       An Admin has the ability to manage user accounts

 -h, --help
        Print help (see a summary with '-h')
```

예: 다른 CU와 키 공유 해제

다음 예에서는 key unshare 명령을 사용하여 CU `alice`와 키 공유를 해제하는 방법을 보여줍니다.

Example

1. key list 명령을 실행하고 `alice`와의 공유를 해제하려는 특정 키를 기준으로 필터링합니다.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
```

```
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
            },
            {
              "username": "alice",
              "key-coverage": "full"
            }
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
```

```
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```

2. alice이 shared-users 출력에 있는지 확인하고 다음 key unshare 명령을 실행하여 키를 alice와 공유 해제합니다.

```
aws-cloudhsm > key unshare --filter attr.label="rsa_key_to_share"
 attr.class=private-key --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "message": "Key unshared successfully"
  }
}
```

3. key list 명령을 다시 실행하여 alice와 키가 공유되지 않았는지 확인합니다.

```
aws-cloudhsm > key list --filter attr.label="rsa_key_to_share" attr.class=private-
key --verbose
{
```

```
  "error_code": 0,
  "data": {
    "matched_keys": [
      {
        "key-reference": "0x00000000001c0686",
        "key-info": {
          "key-owners": [
            {
              "username": "cu3",
              "key-coverage": "full"
            }
          ],
          "shared-users": [
            {
              "username": "cu2",
              "key-coverage": "full"
            },
            {
              "username": "cu1",
              "key-coverage": "full"
            },
            {
              "username": "cu4",
              "key-coverage": "full"
            },
            {
              "username": "cu5",
              "key-coverage": "full"
            },
            {
              "username": "cu6",
              "key-coverage": "full"
            },
            {
              "username": "cu7",
              "key-coverage": "full"
            },
          ],
          "cluster-coverage": "full"
        },
        "attributes": {
          "key-type": "rsa",
          "label": "rsa_key_to_share",
          "id": "",
```

```
          "check-value": "0xae8ff0",
          "class": "private-key",
          "encrypt": false,
          "decrypt": true,
          "token": true,
          "always-sensitive": true,
          "derive": false,
          "destroyable": true,
          "extractable": true,
          "local": true,
          "modifiable": true,
          "never-extractable": false,
          "private": true,
          "sensitive": true,
          "sign": true,
          "trusted": false,
          "unwrap": true,
          "verify": false,
          "wrap": false,
          "wrap-with-trusted": false,
          "key-length-bytes": 1219,
          "public-exponent": "0x010001",
          "modulus":
 "0xa8855cba933cec0c21a4df0450ec31675c024f3e65b2b215a53d2bda6dcd191f75729150b59b4d86df58254
          "modulus-size-bits": 2048
        }
      }
    ],
    "total_key_count": 1,
    "returned_key_count": 1
  }
}
```


인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<FILTER>

삭제할 일치하는 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록입니다.

지원되는 키 속성 목록은 CloudHSM CLI의 키 속성 섹션을 참조하십시오.

필수 여부: 예

<USERNAME>

사용자의 친숙한 이름을 지정합니다. 최대 길이는 31자입니다. 허용되는 유일한 특수 문자는 밑줄 (_)입니다. 이 명령에서 사용자 이름은 대소문자를 구분하지 않으며, 사용자 이름은 항상 소문자로 표시됩니다.

필수: 예

<ROLE>

이 사용자에게 할당된 역할을 지정합니다. 이 파라미터는 필수 사항입니다. 사용자 역할을 가져오려면 user list 명령을 사용합니다. HSM에서의 사용자 유형에 대한 자세한 내용은 HSM 사용자 이해 섹션을 참조하십시오.

필수: 예

관련 주제

- CloudHSM CLI를 사용하여 키를 필터링
- CloudHSM CLI의 키 속성

키 언랩

CloudHSM CLI의 key unwrap parent 명령은 암호화된 (래핑된) 대칭 또는 비대칭 개인 키를 파일에서 HSM으로 가져옵니다. 이 명령은 명령으로 래핑된 암호화된 키를 가져오도록 설계되었지만 다른 도구로 래핑된 키의 래핑을 해제하는 데에도 사용할 수 있습니다. 키 랩 하지만 이런 상황에서는 PKCS#11 또는 JCE 소프트웨어 라이브러리를 사용하여 키를 언래핑하는 것이 좋습니다.

- 키 언랩 aes-gcm
- 키 언랩 aes-no-pad
- 키 언랩 aes-pkcs5-패드
- 키 언랩 aes-zero-pad

- 키 언랩 cloudhsm-aes-gcm
- 키 언랩 rsa-aes
- 키 언랩 rsa-oaep
- 키 언랩 rsa-pkcs

키 언랩 aes-gcm

이 key unwrap aes-gcm 명령은 AES 래핑 키와 언래핑 메커니즘을 사용하여 페이로드 키를 클러스터로 언래핑합니다. `AES-GCM`

래핑되지 않은 키는 에서 생성한 키와 동일한 방식으로 사용할 수 있습니다. AWS CloudHSM로컬에서 생성되지 않았음을 나타내기 위해 `false` 해당 `local` 속성은 로 설정됩니다.

key unwrap aes-gcm명령을 사용하려면 AWS CloudHSM 클러스터에 AES 래핑 키가 있어야 하고 해당 `unwrap` 속성을 로 설정해야 합니다`true`.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key unwrap aes-gcm
Usage: key unwrap aes-gcm [OPTIONS] --filter [<FILTER>...] --tag-length-
bits <TAG_LENGTH_BITS> --key-type-class <KEY_TYPE_CLASS> --label <LABEL> --iv <IV> <--
data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
```

```
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
      --iv <IV>
          Initial value used to wrap the key, in hex
  -h, --help
          Print help
```

예제

이 예제는 unwrap 속성 값이 로 설정된 AES 키를 사용하여 key unwrap aes-gcm 명령을 사용하는 방법을 보여줍니다`true`.

Example 예: Base64로 인코딩된 래핑된 키 데이터에서 페이로드 키 래핑 해제

```
aws-cloudhsm > key unwrap aes-gcm --key-type-class aes --label aes-unwrapped
 --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --iv
 0xf90613bb8e337ec0339aad21 --data xvslgrtg8kHzrvekny97tLSIeokpPwV8
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e4",
      "key-info": {
```

```
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example 예: 데이터 경로를 통해 제공된 페이로드 키의 래핑 해제

```
aws-cloudhsm > key unwrap aes-gcm --key-type-class aes --label aes-unwrapped
 --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --iv
 0xf90613bb8e337ec0339aad21 --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e4",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
```

```
        "key-length-bytes": 16
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<FILTER>

래핑을 해제할 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 공백으로 구분된 키 속성 목록. `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`

필수 여부: 예

<DATA_PATH>

래핑된 키 데이터를 포함하는 바이너리 파일의 경로입니다.

필수: 예 (Base64로 인코딩된 데이터를 통해 제공되지 않는 경우)

<DATA>

Base64로 인코딩된 래핑된 키 데이터.

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<ATTRIBUTES>

래핑된 키 형식의 `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록입니다.

필수 여부: 아니요

<AAD>

Aes GCM 추가 인증 데이터 (AAD) 값 (16진수)

필수 여부: 아니요

<TAG_LENGTH_BITS>

Aes GCM 태그 길이 (비트)

필수 여부: 예

<KEY_TYPE_CLASS>

래핑된 키의 키 유형 및 클래스 [가능한 값: `aes des3ec-private,generic-secret,,rsa-private`]

필수 여부: 예

<LABEL>

래핑되지 않은 키의 레이블.

필수 여부: 예

<SESSION>

현재 세션에만 있는 세션 키를 만듭니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

필수 여부: 아니요

<IV>

키를 래핑하는 데 사용되는 초기 값 (16진수).

필수 여부: 아니요

관련 주제

- 키 랩
- 키 언랩

키 언랩 aes-no-pad

이 key unwrap aes-no-pad 명령은 AES 래핑 키와 언래핑 메커니즘을 사용하여 페이로드 키를 클러스터로 언래핑합니다. `AES-NO-PAD`

래핑되지 않은 키는 에서 생성한 키와 동일한 방식으로 사용할 수 있습니다. AWS CloudHSM로컬에서 생성되지 않았음을 나타내기 위해 `false` 해당 `local` 속성은 로 설정됩니다.

key unwrap aes-no-pad명령을 사용하려면 AWS CloudHSM 클러스터에 AES 래핑 키가 있어야 하고 해당 unwrap 속성을 로 설정해야 합니다true.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key unwrap aes-no-pad
Usage: key unwrap aes-no-pad [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
```

```
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
  -h, --help
          Print help
```

예제

이 예제는 unwrap 속성 값이 로 설정된 AES 키를 사용하여 key unwrap aes-no-pad 명령을 사용하는 방법을 보여줍니다true.

Example 예: Base64로 인코딩된 래핑된 키 데이터에서 페이로드 키 래핑 해제

```
aws-cloudhsm > key unwrap aes-no-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data eXK3PMAOnKM9y3YX6brbhtMoC060EOH9
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ec",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
```

```
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example 예: 데이터 경로를 통해 제공된 페이로드 키의 래핑 해제

```
aws-cloudhsm > key unwrap aes-no-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ec",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
```

```
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<FILTER>

래핑을 해제할 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 공백으로 구분된 키 속성 목록입니다. `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`

필수 여부: 예

<DATA_PATH>

래핑된 키 데이터를 포함하는 바이너리 파일의 경로입니다.

필수: 예 (Base64로 인코딩된 데이터를 통해 제공되지 않는 경우)

<DATA>

Base64로 인코딩된 래핑된 키 데이터

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<ATTRIBUTES>

래핑된 키 형식의 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 공백으로 구분된 키 속성 목록입니다.

필수 여부: 아니요

<KEY_TYPE_CLASS>

래핑된 키의 키 유형 및 클래스 [가능한 값: aes des3ec-private,generic-secret,,rsa-private]

필수 여부: 예

<LABEL>

래핑되지 않은 키의 레이블.

필수 여부: 예

<SESSION>

현재 세션에만 있는 세션 키를 만듭니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

필수 여부: 아니요

관련 주제

- 키 랩
- 키 언랩

키 언랩 aes-pkcs5-패드

이 key unwrap aes-pkcs5-pad 명령은 AES 래핑 키와 언래핑 메커니즘을 사용하여 페이로드 키를 언래핑합니다. AES-PKCS5-PAD

래핑되지 않은 키는 에서 생성한 키와 동일한 방식으로 사용할 수 있습니다. AWS CloudHSM로컬에서 생성되지 않았음을 나타내기 위해 false 해당 local 속성은 로 설정됩니다.

key unwrap aes-pkcs5-pad명령을 사용하려면 AWS CloudHSM 클러스터에 AES 래핑 키가 있어야 하고 해당 unwrap 속성을 로 설정해야 합니다true.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key unwrap aes-pkcs5-pad
Usage: key unwrap aes-pkcs5-pad [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
  -h, --help
          Print help
```

예제

이 예제는 unwrap 속성 값이 로 설정된 AES 키를 사용하여 key unwrap aes-pkcs5-pad 명령을 사용하는 방법을 보여줍니다true.

Example 예: Base64로 인코딩된 래핑된 키 데이터에서 페이로드 키 래핑 해제

```
aws-cloudhsm > key unwrap aes-pkcs5-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data MbuYNresfOKyGNnxKWen88nSfX+uUE/0qmGofSisicY=
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
```

```
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example 예: 데이터 경로를 통해 제공된 페이로드 키의 래핑 해제

```
aws-cloudhsm > key unwrap aes-pkcs5-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e3",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
```

```
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<FILTER>

래핑을 해제할 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 공백으로 구분된 키 속성 목록. `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`

필수 여부: 예

<DATA_PATH>

래핑된 키 데이터를 포함하는 바이너리 파일의 경로입니다.

필수: 예 (Base64로 인코딩된 데이터를 통해 제공되지 않는 경우)

<DATA>

Base64로 인코딩된 래핑된 키 데이터.

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<ATTRIBUTES>

래핑된 키 형식의 `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록입니다.

필수 여부: 아니요

<KEY_TYPE_CLASS>

래핑된 키의 키 유형 및 클래스 [가능한 값: aesdes3,ec-private,generic-secret,rsa-private].

필수 여부: 예

<LABEL>

래핑되지 않은 키의 레이블.

필수 여부: 예

<SESSION>

현재 세션에만 있는 세션 키를 만듭니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

필수 여부: 아니요

관련 주제

- 키 랩
- 키 언랩

키 언랩 aes-zero-pad

이 key unwrap aes-zero-pad 명령은 AES 래핑 키와 언래핑 메커니즘을 사용하여 페이로드 키를 클러스터로 언래핑합니다. AES-ZERO-PAD

래핑되지 않은 키는 에서 생성한 키와 동일한 방식으로 사용할 수 있습니다. AWS CloudHSM로컬에서 생성되지 않았음을 나타내기 위해 false 해당 local 속성은 로 설정됩니다.

key unwrap aes-no-pad명령을 사용하려면 AWS CloudHSM 클러스터에 AES 래핑 키가 있어야 하고 해당 unwrap 속성을 로 설정해야 합니다true.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key unwrap aes-zero-pad
Usage: key unwrap aes-zero-pad [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
  -h, --help
          Print help
```

예제

이 예제는 unwrap 속성 값이 로 설정된 AES 키를 사용하여 key unwrap aes-zero-pad 명령을 사용하는 방법을 보여줍니다`true`.

Example 예: Base64로 인코딩된 래핑된 키 데이터에서 페이로드 키 래핑 해제

```
aws-cloudhsm > key unwrap aes-zero-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data L1wVlL/YeBNVAw6Mpk3owFJZXBzDLONt
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
```

```
      }
    }
  }
}
```

Example 예: 데이터 경로를 통해 제공된 페이로드 키의 래핑 해제

```
aws-cloudhsm > key unwrap aes-zero-pad --key-type-class aes --label aes-unwrapped --
filter attr.label=aes-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08e7",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
```

```
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<FILTER>

래핑을 해제할 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 공백으로 구분된 키 속성 목록입니다. `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`

필수 여부: 예

<DATA_PATH>

래핑된 키 데이터를 포함하는 바이너리 파일의 경로입니다.

필수: 예 (Base64로 인코딩된 데이터를 통해 제공되지 않는 경우)

<DATA>

Base64로 인코딩된 래핑된 키 데이터

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<ATTRIBUTES>

래핑된 키 형식의 `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록입니다.

필수 여부: 아니요

<KEY_TYPE_CLASS>

래핑된 키의 키 유형 및 클래스 [가능한 값: `aes` `des3ec-private`,`generic-secret`,,`rsa-private`]

필수 여부: 예

<LABEL>

래핑되지 않은 키의 레이블.

필수 여부: 예

<SESSION>

현재 세션에만 있는 세션 키를 만듭니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

필수 여부: 아니요

관련 주제

- 키 랩
- 키 언랩

키 언랩 cloudhsm-aes-gcm

이 key unwrap cloudhsm-aes-gcm 명령은 AES 래핑 키와 언래핑 메커니즘을 사용하여 페이로드 키를 클러스터로 언래핑합니다. `CLOUDHSM-AES-GCM`

래핑되지 않은 키는 에서 생성한 키와 동일한 방식으로 사용할 수 있습니다. AWS CloudHSM로컬에서 생성되지 않았음을 나타내기 위해 `false` 해당 `local` 속성은 로 설정됩니다.

key unwrap cloudhsm-aes-gcm명령을 사용하려면 AWS CloudHSM 클러스터에 AES 래핑 키가 있어야 하고 해당 `unwrap` 속성이 로 설정되어야 합니다`true`.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key unwrap cloudhsm-aes-gcm
Usage: key unwrap cloudhsm-aes-gcm [OPTIONS] --filter [<FILTER>...] --tag-length-
bits <TAG_LENGTH_BITS> --key-type-class <KEY_TYPE_CLASS> --label <LABEL> <--data-
path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
  -h, --help
          Print help
```

예제

이 예제는 unwrap 속성 값이 로 설정된 AES 키를 사용하여 key unwrap cloudhsm-aes-gcm 명령을 사용하는 방법을 보여줍니다true.

Example 예: Base64로 인코딩된 래핑된 키 데이터에서 페이로드 키 래핑 해제

```
aws-cloudhsm > key unwrap cloudhsm-aes-gcm --key-type-class aes --label aes-
unwrapped --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --data
 6Rn8nkjEriDYlnP3P8nPkYQ8hplOEJ899zsrF+aTB0i/fIlZ
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001408e8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
```

```
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example 예: 데이터 경로를 통해 제공된 페이로드 키의 래핑 해제

```
aws-cloudhsm > key unwrap cloudhsm-aes-gcm --key-type-class aes --label aes-unwrapped
 --filter attr.label=aes-example --tag-length-bits 64  --aad 0x10 --data-path payload-
key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001408e8",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
```

```
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<FILTER>

래핑을 해제할 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 공백으로 구분된 키 속성 목록. `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`

필수 여부: 예

<DATA_PATH>

래핑된 키 데이터를 포함하는 바이너리 파일의 경로입니다.

필수: 예 (Base64로 인코딩된 데이터를 통해 제공되지 않는 경우)

<DATA>

Base64로 인코딩된 래핑된 키 데이터.

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<ATTRIBUTES>

래핑된 키 형식의 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 공백으로 구분된 키 속성 목록입니다.

필수 여부: 아니요

<AAD>

Aes GCM 추가 인증 데이터 (AAD) 값 (16진수).

필수 여부: 아니요

<TAG_LENGTH_BITS>

Aes GCM 태그 길이 (비트).

필수 여부: 예

<KEY_TYPE_CLASS>

래핑된 키의 키 유형 및 클래스 [가능한 값: aesdes3,ec-private,generic-secret,rsa-private].

필수 여부: 예

<LABEL>

래핑되지 않은 키의 레이블.

필수 여부: 예

<SESSION>

현재 세션에만 있는 세션 키를 만듭니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

필수 여부: 아니요

관련 주제

- 키 랩
- 키 언랩

키 언랩 rsa-aes

이 key unwrap rsa-aes 명령은 RSA 개인 키와 언래핑 메커니즘을 사용하여 페이로드 키를 언래핑합니다. `RSA-AES`

래핑되지 않은 키는 에서 생성한 키와 동일한 방식으로 사용할 수 있습니다. AWS CloudHSM로컬에서 생성되지 않았음을 나타내기 위해 `false` 해당 `local` 속성은 로 설정됩니다.

를 사용하려면 AWS CloudHSM 클러스터에 key unwrap rsa-aes RSA 퍼블릭 래핑 키의 RSA 프라이빗 키가 있어야 하고 해당 `unwrap` 속성을 로 설정해야 합니다. `true`

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key unwrap rsa-aes
Usage: key unwrap rsa-aes [OPTIONS] --filter [<FILTER>...] --hash-
function <HASH_FUNCTION> --mgf <MGF> --key-type-class <KEY_TYPE_CLASS> --label <LABEL>
 <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
```

```
          Space separated list of key attributes in the form of
KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --hash-function <HASH_FUNCTION>
          Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
      --mgf <MGF>
          Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
mgf1-sha256, mgf1-sha384, mgf1-sha512]
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
be recovered after the session ends
  -h, --help
          Print help
```

예

이 예제는 unwrap 속성 값이 로 설정된 RSA 개인 키를 사용하여 key unwrap rsa-aes 명령을 사용하는 방법을 보여줍니다. `true`

Example 예: Base64로 인코딩된 래핑된 키 데이터에서 페이로드 키 래핑 해제

```
aws-cloudhsm > key unwrap rsa-aes --key-type-class aes --label aes-unwrapped
 --filter attr.label=rsa-private-key-example --hash-function sha256 --
mgf mgf1-sha256 --data HrSE1DEyLjIeyGdPa9R+ebiqB5TIJGyamPker31ZebPwRA
+NcerbAJO8DJ1lXPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4DOmU4meSofqREYvtb3EoIKwjy
+RL5WGXKe4nAboAkC5GO7veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCSlw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/OWUd6l4d97BdgtaEz6ueg==
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e2",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
```

```
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example 예: 데이터 경로를 통해 제공된 페이로드 키의 래핑 해제

```
aws-cloudhsm > key unwrap rsa-aes --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-key-example --hash-function sha256 --mgf mgf1-sha256 --data-
path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e2",
```

```
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<FILTER>

래핑을 해제할 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 공백으로 구분된 키 속성 목록입니다. `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`

필수 여부: 예

<DATA_PATH>

래핑된 키 데이터를 포함하는 바이너리 파일의 경로입니다.

필수: 예 (Base64로 인코딩된 데이터를 통해 제공되지 않는 경우)

<DATA>

Base64로 인코딩된 래핑된 키 데이터

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<ATTRIBUTES>

래핑된 키 형식의 `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록입니다.

필수 여부: 아니요

<KEY_TYPE_CLASS>

래핑된 키의 키 유형 및 클래스 [가능한 값: `aes des3ec-private,generic-secret,,rsa-private`]

필수 여부: 예

<HASH_FUNCTION>

해시 함수를 지정합니다.

유효한 값:

- sha1

- sha224
- sha256
- sha384
- sha512

필수 여부: 예

<MGF>

마스크 생성 함수를 지정합니다.

> **Note**
>
> 마스크 생성 함수 해시 함수는 서명 메커니즘 해시 함수와 일치해야 합니다.

유효한 값:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

필수 여부: 예

<LABEL>

래핑되지 않은 키의 라벨.

필수 여부: 예

<SESSION>

현재 세션에만 있는 세션 키를 만듭니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

필수 여부: 아니요

관련 주제

- 키 랩
- 키 언랩

키 언랩 rsa-oaep

이 key unwrap rsa-oaep 명령은 RSA 개인 키와 언래핑 메커니즘을 사용하여 페이로드 키를 언래핑합니다. `RSA-OAEP`

래핑되지 않은 키는 에서 생성한 키와 동일한 방식으로 사용할 수 있습니다. AWS CloudHSM로컬에서 생성되지 않았음을 나타내기 위해 `false` 해당 `local` 속성은 로 설정됩니다.

key unwrap rsa-oaep명령을 사용하려면 AWS CloudHSM 클러스터에 RSA 퍼블릭 래핑 키의 RSA 프라이빗 키가 있어야 하고 해당 `unwrap` 속성을 로 설정해야 합니다. `true`

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key unwrap rsa-oaep
Usage: key unwrap rsa-oaep [OPTIONS] --filter [<FILTER>...] --hash-
function <HASH_FUNCTION> --mgf <MGF> --key-type-class <KEY_TYPE_CLASS> --label <LABEL>
 <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <<DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
```

```
          Space separated list of key attributes in the form of
KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --hash-function <HASH_FUNCTION>
          Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
      --mgf <MGF>
          Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
mgf1-sha256, mgf1-sha384, mgf1-sha512]
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
be recovered after the session ends
  -h, --help
          Print help
```

예제

이 예제는 unwrap 속성 값이 로 설정된 RSA 개인 키를 사용하여 key unwrap rsa-oaep 명령을 사용하는 방법을 보여줍니다. `true`

Example 예: Base64로 인코딩된 래핑된 키 데이터에서 페이로드 키 래핑 해제

```
aws-cloudhsm > key unwrap rsa-oaep --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-example-key --hash-function sha256 --mgf mgf1-sha256 --data
 OjJe4msobPLz9TuSAdULEu17T5rMDWtSlLyBSkLbaZnYzzpdrhsbGLbwZJCtB/jGkDNdB4qyTAOQwEpggGf6v
+Yx6JcesNeKKNU8XZal/YBoHC8noTGUSDI2qr+u2tDc84NPv6d+F2KOONXsSxMhmxzzNG/
gzTVIJhOuy/B1yHjGP4mOXoDZf5+7f5M1CjxBmz4Vva/wrWHGCSG0yOaWblEvOiHAIt3UBdyKmU+/
My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw==
{
  "data": {
    "key": {
      "key-reference": "0x00000000001808e9",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
```

```
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example 예: 데이터 경로를 통해 제공된 페이로드 키의 래핑 해제

```
aws-cloudhsm > key unwrap rsa-oaep --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-example-key --hash-function sha256 --mgf mgf1-sha256 --data-
path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001808e9",
```

```
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<FILTER>

래핑을 해제할 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 공백으로 구분된 키 속성 목록입니다. `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`

필수 여부: 예

<DATA_PATH>

래핑된 키 데이터를 포함하는 바이너리 파일의 경로입니다.

필수: 예 (Base64로 인코딩된 데이터를 통해 제공되지 않는 경우)

<DATA>

Base64로 인코딩된 래핑된 키 데이터

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<ATTRIBUTES>

래핑된 키 형식의 `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록입니다.

필수 여부: 아니요

<KEY_TYPE_CLASS>

래핑된 키의 키 유형 및 클래스 [가능한 값: `aes des3ec-private,generic-secret,,rsa-private`]

필수 여부: 예

<HASH_FUNCTION>

해시 함수를 지정합니다.

유효한 값:

- sha1

- sha224
- sha256
- sha384
- sha512

필수 여부: 예

<MGF>

마스크 생성 함수를 지정합니다.

> **Note**
>
> 마스크 생성 함수 해시 함수는 서명 메커니즘 해시 함수와 일치해야 합니다.

유효한 값:

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

필수 여부: 예

<LABEL>

래핑되지 않은 키의 라벨.

필수 여부: 예

<SESSION>

현재 세션에만 있는 세션 키를 만듭니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

필수 여부: 아니요

관련 주제

- 키 랩
- 키 언랩

키 언랩 rsa-pkcs

이 key unwrap rsa-pkcs 명령은 RSA 개인 키와 언래핑 메커니즘을 사용하여 페이로드 키를 언래핑합니다. RSA-PKCS

래핑되지 않은 키는 에서 생성한 키와 동일한 방식으로 사용할 수 있습니다. AWS CloudHSM로컬에서 생성되지 않았음을 나타내기 위해 `false` 해당 `local` 속성은 로 설정됩니다.

키 unwrap rsa-pkcs 명령을 사용하려면 AWS CloudHSM 클러스터에 RSA 퍼블릭 래핑 키의 RSA 개인 키가 있어야 하고 해당 `unwrap` 속성을 로 설정해야 합니다. `true`

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key unwrap rsa-pkcs
Usage: key unwrap rsa-pkcs [OPTIONS] --filter [<FILTER>...] --key-type-
class <KEY_TYPE_CLASS> --label <LABEL> <--data-path <DATA_PATH>|--data <DATA>>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --filter [<FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a key
 to unwrap with
      --data-path <DATA_PATH>
          Path to the binary file containing the wrapped key data
      --data <DATA>
          Base64 encoded wrapped key data
      --attributes [<UNWRAPPED_KEY_ATTRIBUTES>...]
```

```
          Space separated list of key attributes in the form of
 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE for the unwrapped key
      --key-type-class <KEY_TYPE_CLASS>
          Key type and class of wrapped key [possible values: aes, des3, ec-private,
 generic-secret, rsa-private]
      --label <LABEL>
          Label for the unwrapped key
      --session
          Creates a session key that exists only in the current session. The key cannot
 be recovered after the session ends
  -h, --help
          Print help
```

예제

이 예제는 unwrap 속성 값이 로 설정된 AES 키를 사용하여 key unwrap rsa-oaep 명령을 사용하는 방법을 보여줍니다. `true`

Example 예: Base64로 인코딩된 래핑된 키 데이터에서 페이로드 키 래핑 해제

```
aws-cloudhsm > key unwrap rsa-pkcs --key-type-class aes --label
 aes-unwrapped --filter attr.label=rsa-private-key-example --data
 am0Nc7+YE8FWs+5HvU7sIBcXVb24QA0l65nbNAD+1bK+e18BpSfnaI3P+r8Dp+pLu1ofoUy/
vtzRjZoCiDofcz4EqCFnGl4GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYRO1lGlfq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcpOOobJlITCoXhtMRepSdO4OggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg==
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ef",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
```

```
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

Example 예: 데이터 경로를 통해 제공된 페이로드 키의 래핑 해제

```
aws-cloudhsm > key unwrap rsa-pkcs --key-type-class aes --label aes-unwrapped --filter
 attr.label=rsa-private-key-example --data-path payload-key.pem
{
  "error_code": 0,
  "data": {
    "key": {
      "key-reference": "0x00000000001c08ef",
      "key-info": {
        "key-owners": [
          {
            "username": "cu1",
            "key-coverage": "full"
          }
```

```
        ],
        "shared-users": [],
        "cluster-coverage": "full"
      },
      "attributes": {
        "key-type": "aes",
        "label": "aes-unwrapped",
        "id": "0x",
        "check-value": "0x8d9099",
        "class": "secret-key",
        "encrypt": false,
        "decrypt": false,
        "token": true,
        "always-sensitive": false,
        "derive": false,
        "destroyable": true,
        "extractable": true,
        "local": false,
        "modifiable": true,
        "never-extractable": false,
        "private": true,
        "sensitive": true,
        "sign": true,
        "trusted": false,
        "unwrap": false,
        "verify": true,
        "wrap": false,
        "wrap-with-trusted": false,
        "key-length-bytes": 16
      }
    }
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<FILTER>

래핑을 해제할 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 공백으로 구분된 키 속성 목록입니다. `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`

필수 여부: 예

<DATA_PATH>

래핑된 키 데이터를 포함하는 바이너리 파일의 경로입니다.

필수: 예 (Base64로 인코딩된 데이터를 통해 제공되지 않는 경우)

<DATA>

Base64로 인코딩된 래핑된 키 데이터

필수: 예 (데이터 경로를 통해 제공하지 않는 한)

<ATTRIBUTES>

래핑된 키 형식의 `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록입니다.

필수 여부: 아니요

<KEY_TYPE_CLASS>

래핑된 키의 키 유형 및 클래스 [가능한 값: `aes des3ec-private,generic-secret,,rsa-private`]

필수 여부: 예

<LABEL>

래핑되지 않은 키의 레이블.

필수 여부: 예

<SESSION>

현재 세션에만 있는 세션 키를 만듭니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

필수 여부: 아니요

관련 주제

- 키 랩

- 키 언랩

키 랩

CloudHSM CLI의 key wrap 명령은 HSM에서 대칭 또는 비대칭 개인 키의 암호화된 사본을 파일로 내보냅니다. 를 실행할 때 key wrap 내보낼 키와 출력 파일이라는 두 가지를 지정합니다. 내보낼 키는 내보내려는 키를 암호화 (래핑) 하는 HSM의 키입니다.

이 key wrap 명령은 HSM에서 키를 제거하거나 암호화 작업에 키를 사용하는 것을 막지 않습니다. 동일한 키를 여러 번 내보낼 수 있습니다. 암호화된 키를 HSM으로 다시 가져오려면 를 사용하십시오. 키 언랩 키 소유자, 즉 키를 생성한 암호화 사용자 (CU) 만 키를 래핑할 수 있습니다. 키를 공유하는 사용자는 암호화 작업에서만 키를 사용할 수 있습니다.

key wrap명령은 다음과 같은 하위 명령으로 구성됩니다.

- 키 랩 aes-gcm
- 키 랩 aes-no-pad
- 키 랩 AES-pkcs5-패드
- 키 랩 aes-zero-pad
- 키 랩 cloudhsm-aes-gcm
- 키 랩 라사-에스
- 키 랩 랩-비누
- 키 랩 rsa-pcs

키 랩 aes-gcm

이 key wrap aes-gcm 명령은 HSM의 AES 키와 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다. `AES-GCM` 페이로드 키의 `extractable` 속성을 로 설정해야 합니다. `true`

키 소유자, 즉 키를 생성한 암호화 사용자 (CU) 만 키를 래핑할 수 있습니다. 키를 공유하는 사용자는 암호화 작업에 키를 사용할 수 있습니다.

key wrap aes-gcm명령을 사용하려면 먼저 AWS CloudHSM 클러스터에 AES 키가 있어야 합니다. 키 생성-대칭 aes명령과 `wrap` 속성을 로 설정하여 래핑용 AES 키를 생성할 수 `true` 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key wrap aes-gcm
Usage: key wrap aes-gcm [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...] --tag-length-bits <TAG_LENGTH_BITS>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
  -h, --help
          Print help
```

예

이 예제에서는 AES 키를 사용하여 key wrap aes-gcm 명령을 사용하는 방법을 보여줍니다.

Example

```
aws-cloudhsm > key wrap aes-gcm --payload-filter attr.label=payload-key --wrapping-
filter attr.label=aes-example --tag-length-bits 64  --aad 0x10
```

```
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "iv": "0xf90613bb8e337ec0339aad21",
    "wrapped_key_data": "xvslgrtg8kHzrvekny97tLSIeokpPwV8"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<PAYLOAD_FILTER>

페이로드 키를 선택하기 위한 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록

필수 여부: 예

<PATH>

래핑된 키 데이터가 저장될 바이너리 파일의 경로입니다.

필수 여부: 아니요

<WRAPPING_FILTER>

래핑 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록

필수 여부: 예

<AAD>

AES GCM 추가 인증 데이터 (AAD) 값 (16진수)

필수 여부: 아니요

<TAG_LENGTH_BITS>

AES GCM 태그의 길이 (비트)

필수: 예

관련 주제

- 키 랩
- 키 언랩

키 랩 aes-no-pad

이 key wrap aes-no-pad 명령은 HSM의 AES 키와 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다. AES-NO-PAD 페이로드 키의 extractable 속성을 로 설정해야 합니다. true

키 소유자, 즉 키를 생성한 암호화 사용자 (CU) 만 키를 래핑할 수 있습니다. 키를 공유하는 사용자는 암호화 작업에 키를 사용할 수 있습니다.

key wrap aes-no-pad명령을 사용하려면 먼저 AWS CloudHSM 클러스터에 AES 키가 있어야 합니다. 키 생성-대칭 aes명령을 사용하고 wrap 속성을 로 설정하여 래핑용 AES 키를 생성할 수 true 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key wrap aes-no-pad
Usage: key wrap aes-no-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
```

```
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
  -h, --help
          Print help
```

예

이 예제에서는 `wrap` 속성 값이 로 설정된 AES 키를 사용하여 key wrap aes-no-pad 명령을 사용하는 방법을 보여줍니다`true`.

Example

```
aws-cloudhsm > key wrap aes-no-pad --payload-filter attr.label=payload-key --wrapping-
filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "eXK3PMAOnKM9y3YX6brbhtMoC060EOH9"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<PAYLOAD_FILTER>

페이로드 키를 선택하기 위한 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록

필수 여부: 예

<PATH>

래핑된 키 데이터가 저장될 바이너리 파일의 경로입니다.

필수 여부: 아니요

<WRAPPING_FILTER>

래핑 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록

필수: 예

관련 주제

- 키 랩
- 키 언랩

키 랩 AES-pkcs5-패드

이 key wrap aes-pkcs5-pad 명령은 HSM의 AES 키와 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다. `AES-PKCS5-PAD` 페이로드 키의 `extractable` 속성을 로 설정해야 합니다. `true`

키 소유자, 즉 키를 생성한 암호화 사용자 (CU) 만 키를 래핑할 수 있습니다. 키를 공유하는 사용자는 암호화 작업에 키를 사용할 수 있습니다.

key wrap aes-pkcs5-pad명령을 사용하려면 먼저 AWS CloudHSM 클러스터에 AES 키가 있어야 합니다. 키 생성-대칭 aes명령을 사용하고 `wrap` 속성을 로 설정하여 래핑용 AES 키를 생성할 수 `true` 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key wrap aes-pkcs5-pad
Usage: key wrap aes-pkcs5-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --
wrapping-filter [<WRAPPING_FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
  -h, --help
          Print help
```

예

이 예제에서는 `wrap` 속성 값이 로 설정된 AES 키를 사용하여 key wrap aes-pkcs5-pad 명령을 사용하는 방법을 보여줍니다`true`.

Example

```
aws-cloudhsm > key wrap aes-pkcs5-pad --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "MbuYNresfOKyGNnxKWen88nSfX+uUE/0qmGofSisicY="
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<PAYLOAD_FILTER>

페이로드 키를 선택하기 위한 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록

필수 여부: 예

<PATH>

래핑된 키 데이터가 저장될 바이너리 파일의 경로입니다.

필수 여부: 아니요

<WRAPPING_FILTER>

래핑 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록

필수: 예

관련 주제

- 키 랩
- 키 언랩

키 랩 aes-zero-pad

이 key wrap aes-zero-pad 명령은 HSM의 AES 키와 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다. `AES-ZERO-PAD` 페이로드 키의 `extractable` 속성을 로 설정해야 합니다. `true`

키 소유자, 즉 키를 생성한 암호화 사용자 (CU) 만 키를 래핑할 수 있습니다. 키를 공유하는 사용자는 암호화 작업에 키를 사용할 수 있습니다.

key wrap aes-zero-pad명령을 사용하려면 먼저 AWS CloudHSM 클러스터에 AES 키가 있어야 합니다. `wrap`속성이 로 설정된 키 생성-대칭 aes 명령을 사용하여 래핑용 AES 키를 생성할 수 `true` 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key wrap aes-zero-pad
Usage: key wrap aes-zero-pad [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --
wrapping-filter [<WRAPPING_FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
  -h, --help
          Print help
```

예

이 예제에서는 `wrap` 속성 값이 로 설정된 AES 키를 사용하여 key wrap aes-zero-pad 명령을 사용하는 방법을 보여줍니다`true`.

Example

```
aws-cloudhsm > key wrap aes-zero-pad --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "L1wVlL/YeBNVAw6Mpk3owFJZXBzDLONt"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<PAYLOAD_FILTER>

페이로드 키를 선택하기 위한 형식의 키 참조 (예:key-reference=0xabc) 또는 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 공백으로 구분된 키 속성 목록

필수 여부: 예

<PATH>

래핑된 키 데이터가 저장될 바이너리 파일의 경로입니다.

필수 여부: 아니요

<WRAPPING_FILTER>

래핑 키를 선택하는 형식의 키 참조 (예:key-reference=0xabc) 또는 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 공백으로 구분된 키 속성 목록

필수: 예

관련 주제

- 키 랩

- 키 언랩

키 랩 cloudhsm-aes-gcm

이 key wrap cloudhsm-aes-gcm 명령은 HSM의 AES 키와 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다. `CLOUDHSM-AES-GCM` 페이로드 키의 `extractable` 속성을 로 설정해야 합니다. `true`

키 소유자, 즉 키를 생성한 암호화 사용자 (CU) 만 키를 래핑할 수 있습니다. 키를 공유하는 사용자는 암호화 작업에 키를 사용할 수 있습니다.

key wrap cloudhsm-aes-gcm명령을 사용하려면 먼저 AWS CloudHSM 클러스터에 AES 키가 있어야 합니다. 키 생성-대칭 aes명령과 `wrap` 속성을 로 설정하여 래핑용 AES 키를 생성할 수 `true` 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key wrap cloudhsm-aes-gcm
Usage: key wrap cloudhsm-aes-gcm [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --
wrapping-filter [<WRAPPING_FILTER>...] --tag-length-bits <TAG_LENGTH_BITS>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
```

```
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --aad <AAD>
          Aes GCM Additional Authenticated Data (AAD) value, in hex
      --tag-length-bits <TAG_LENGTH_BITS>
          Aes GCM tag length in bits
  -h, --help
          Print help
```

예

이 예제에서는 AES 키를 사용하여 key wrap cloudhsm-aes-gcm 명령을 사용하는 방법을 보여줍니다.

Example

```
aws-cloudhsm > key wrap cloudhsm-aes-gcm --payload-filter attr.label=payload-key --
wrapping-filter attr.label=aes-example --tag-length-bits 64 --aad 0x10
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
    "wrapping_key_reference": "0x00000000001c08ea",
    "wrapped_key_data": "6Rn8nkjEriDYlnP3P8nPkYQ8hplOEJ899zsrF+aTB0i/fIlZ"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<PAYLOAD_FILTER>

페이로드 키를 선택하기 위한 형식의 키 참조 (예:key-reference=0xabc) 또는 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 공백으로 구분된 키 속성 목록

필수 여부: 예

<PATH>

래핑된 키 데이터가 저장될 바이너리 파일의 경로입니다.

필수 여부: 아니요

<WRAPPING_FILTER>

래핑 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록

필수 여부: 예

<AAD>

AES GCM 추가 인증 데이터 (AAD) 값 (16진수)

필수 여부: 아니요

<TAG_LENGTH_BITS>

AES GCM 태그의 길이 (비트)

필수: 예

관련 주제

- 키 랩
- 키 언랩

키 랩 라사-에스

이 key wrap rsa-aes 명령은 HSM의 RSA 공개 키와 RSA-AES 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다. 페이로드 키의 속성을 로 설정해야 합니다. `extractable true`

키 소유자, 즉 키를 생성한 암호화 사용자 (CU) 만 키를 래핑할 수 있습니다. 키를 공유하는 사용자는 암호화 작업에 키를 사용할 수 있습니다.

key wrap rsa-aes명령을 사용하려면 먼저 클러스터에 RSA 키가 있어야 합니다. AWS CloudHSM 키 generate-asymmetric-pair 명령과 `wrap` 속성 설정을 로 사용하여 RSA 키페어를 생성할 수 있습니다. `true`

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key wrap rsa-aes
Usage: key wrap rsa-aes [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...] --hash-function <HASH_FUNCTION> --mgf <MGF>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --hash-function <HASH_FUNCTION>
          Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
      --mgf <MGF>
          Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
 mgf1-sha256, mgf1-sha384, mgf1-sha512]
  -h, --help
          Print help
```

예

이 예제에서는 wrap 속성 값이 로 설정된 RSA 공개 키를 사용하여 key wrap rsa-ae 명령을 사용하는 방법을 보여줍니다. true

Example

```
aws-cloudhsm > key wrap rsa-aes --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example --hash-function sha256 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "payload-key-reference": "0x00000000001c08f1",
    "wrapping-key-reference": "0x00000000007008da",
    "wrapped-key-data": "HrSE1DEyLjIeyGdPa9R+ebiqB5TIJGyamPker31ZebPwRA
+NcerbAJO8DJ1lXPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4DOmU4meSofqREYvtb3EoIKwjy
+RL5WGXKe4nAboAkC5GO7veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCSlw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/OWUd6l4d97BdgtaEz6ueg=="
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<PAYLOAD_FILTER>

페이로드 키를 선택하기 위한 형식의 키 참조 (예:key-reference=0xabc) 또는 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 공백으로 구분된 키 속성 목록

필수 여부: 예

<PATH>

래핑된 키 데이터가 저장될 바이너리 파일의 경로입니다.

필수 여부: 아니요

<WRAPPING_FILTER>

래핑 키를 선택하는 형식의 키 참조 (예:key-reference=0xabc) 또는 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 공백으로 구분된 키 속성 목록

필수 여부: 예

<MGF>

마스크 생성 함수를 지정합니다.

> **Note**
>
> 마스크 생성 함수 해시 함수는 서명 메커니즘 해시 함수와 일치해야 합니다.

유효값

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

필수: 예

관련 주제

- 키 랩
- 키 언랩

키 랩 랩-비누

이 key wrap rsa-oaep 명령은 HSM의 RSA 공개 키와 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다. RSA-OAEP 페이로드 키의 `extractable` 속성을 로 설정해야 합니다. `true`

키 소유자, 즉 키를 생성한 암호화 사용자 (CU) 만 키를 래핑할 수 있습니다. 키를 공유하는 사용자는 암호화 작업에 키를 사용할 수 있습니다.

key wrap rsa-oaep명령을 사용하려면 먼저 클러스터에 RSA 키가 있어야 합니다. AWS CloudHSM 키 generate-asymmetric-pair 명령과 `wrap` 속성 설정을 로 사용하여 RSA 키페어를 생성할 수 있습니다. `true`

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key wrap rsa-oaep
Usage: key wrap rsa-oaep [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...] --hash-function <HASH_FUNCTION> --mgf <MGF>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
      --hash-function <HASH_FUNCTION>
          Hash algorithm [possible values: sha1, sha224, sha256, sha384, sha512]
      --mgf <MGF>
          Mask Generation Function algorithm [possible values: mgf1-sha1, mgf1-sha224,
 mgf1-sha256, mgf1-sha384, mgf1-sha512]
  -h, --help
          Print help
```

예

이 예제에서는 `wrap` 속성 값이 로 설정된 RSA 공개 키를 사용하여 key wrap rsa-oaep 명령을 사용하는 방법을 보여줍니다. `true`

Example

```
aws-cloudhsm > key wrap rsa-oaep --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example --hash-function sha256 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "payload-key-reference": "0x00000000001c08f1",
    "wrapping-key-reference": "0x00000000007008da",
    "wrapped-key-data": "OjJe4msobPLz9TuSAdULEu17T5rMDWtSlLyBSkLbaZnYzzpdrhsbGLbwZJCtB/
jGkDNdB4qyTAOQwEpggGf6v+Yx6JcesNeKKNU8XZal/YBoHC8noTGUSDI2qr+u2tDc84NPv6d
+F2KOONXsSxMhmxzzNG/gzTVIJhOuy/B1yHjGP4mOXoDZf5+7f5M1CjxBmz4Vva/
wrWHGCSG0yOaWblEvOiHAIt3UBdyKmU+/
My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw=="
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<PAYLOAD_FILTER>

페이로드 키를 선택하기 위한 형식의 키 참조 (예:key-reference=0xabc) 또는 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 공백으로 구분된 키 속성 목록

필수 여부: 예

<PATH>

래핑된 키 데이터가 저장될 바이너리 파일의 경로입니다.

필수 여부: 아니요

<WRAPPING_FILTER>

래핑 키를 선택하는 형식의 키 참조 (예:key-reference=0xabc) 또는 attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE 공백으로 구분된 키 속성 목록

필수 여부: 예

<MGF>

마스크 생성 함수를 지정합니다.

> **Note**
>
> 마스크 생성 함수 해시 함수는 서명 메커니즘 해시 함수와 일치해야 합니다.

유효값

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

필수: 예

관련 주제

- 키 랩
- 키 언랩

키 랩 rsa-pcs

이 key wrap rsa-pkcs 명령은 HSM의 RSA 공개 키와 래핑 메커니즘을 사용하여 페이로드 키를 래핑합니다. RSA-PKCS 페이로드 키의 `extractable` 속성을 로 설정해야 합니다. `true`

키 소유자, 즉 키를 생성한 암호화 사용자 (CU) 만 키를 래핑할 수 있습니다. 키를 공유하는 사용자는 암호화 작업에 키를 사용할 수 있습니다.

key wrap rsa-pkcs명령을 사용하려면 먼저 클러스터에 RSA 키가 있어야 합니다. AWS CloudHSM 키 generate-asymmetric-pair 명령과 `wrap` 속성 설정을 로 사용하여 RSA 키페어를 생성할 수 있습니다. `true`

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CU(Crypto User)

요구 사항

- 이 명령을 실행하려면 CU로 로그인해야 합니다.

구문

```
aws-cloudhsm > help key wrap rsa-pkcs
Usage: key wrap rsa-pkcs [OPTIONS] --payload-filter [<PAYLOAD_FILTER>...] --wrapping-
filter [<WRAPPING_FILTER>...]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
      --payload-filter [<PAYLOAD_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 payload key
      --wrapping-filter [<WRAPPING_FILTER>...]
          Key reference (e.g. key-reference=0xabc) or space separated list of key
 attributes in the form of attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE to select a
 wrapping key
      --path <PATH>
          Path to the binary file where the wrapped key data will be saved
  -h, --help
          Print help
```

예

이 예제에서는 RSA 공개 키를 사용하여 key wrap rsa-pkcs 명령을 사용하는 방법을 보여줍니다.

Example

```
aws-cloudhsm > key wrap rsa-pkcs --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example
{
  "error_code": 0,
  "data": {
    "payload_key_reference": "0x00000000001c08f1",
```

```
    "wrapping_key_reference": "0x00000000007008da",
    "wrapped_key_data": "am0Nc7+YE8FWs+5HvU7sIBcXVb24QA0l65nbNAD+1bK+e18BpSfnaI3P+r8Dp
+pLu1ofoUy/
vtzRjZoCiDofcz4EqCFnGl4GdcJ1/3W/5WRvMatCa2d7cx02swaeZcjKsermPXYRO1lGlfq6NskwMeeTkV8R7Rx9artFrs1
c3XdFJ2+0Bo94c6og/
yfPcpOOobJlITCoXhtMRepSdO4OggYq/6nUDuHCtJ86pPGnNahyr7+sAaSI3a5ECQLUjwaIARUCyoRh7EFK3qPXcg=="
  }
```


인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<PAYLOAD_FILTER>

페이로드 키를 선택하기 위한 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록.

필수 여부: 예

<PATH>

래핑된 키 데이터가 저장될 바이너리 파일의 경로입니다.

필수 여부: 아니요

<WRAPPING_FILTER>

래핑 키를 선택하는 형식의 키 참조 (예:`key-reference=0xabc`) 또는 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 공백으로 구분된 키 속성 목록.

필수: 예

관련 주제

- 키 랩
- 키 언랩

login

CloudHSM CLI의 login명령을 사용하여 클러스터의 각 HSM에 로그인/로그아웃할 수 있습니다.

> **Note**
>
> 잘못된 로그인 시도 횟수가 5회를 초과하면 계정이 잠깁니다. 계정 잠금을 해제하려면 관리자는 cloudhsm_cli의 사용자 변경-비밀번호 명령을 사용하여 비밀번호를 재설정해야 합니다.

로그인 및 로그아웃 문제를 해결하려면

클러스터에 HSM이 둘 이상인 경우 계정이 잠기기 전에 잘못된 로그인 시도가 추가로 허용될 수 있습니다. 이는 CloudHSM 클라이언트가 다양한 HSM 간에 로드 균형을 조정하기 때문입니다. 따라서 매번 동일한 HSM에서 로그인 시도가 시작되지 않을 수 있습니다. 이 기능을 테스트하는 경우 활성화된 HSM이 하나 뿐인 클러스터에서 수행하는 것이 좋습니다.

2018년 2월 이전에 클러스터를 만든 경우 로그인 시도가 20회 실패한 후에 계정이 잠깁니다.

사용자 유형

다음 사용자가 이러한 명령을 실행할 수 있습니다.

- 활성화되지 않은 관리자
- 관리자
- CU(Crypto User)

구문

```
aws-cloudhsm > help login
Login to your cluster

USAGE:
    cloudhsm-cli login [OPTIONS] --username <USERNAME> --role <ROLE> [COMMAND]

Commands:
  mfa-token-sign  Login with token-sign mfa
  help            Print this message or the help of the given subcommand(s)

OPTIONS:
        --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
```

```
        --username <USERNAME>
            Username to access the Cluster

        --role <ROLE>
            Role the user has in the Cluster

           Possible values:
           - crypto-user: A CryptoUser has the ability to manage and use keys
           - admin:       An Admin has the ability to manage user accounts

        --password <PASSWORD>
           Optional: Plaintext user's password. If you do not include this argument you
 will be prompted for it

  -h, --help
          Print help (see a summary with '-h')
```

예

Example

이 명령은 admin1이라는 이름인 관리자 사용자의 자격 증명을 사용하여 클러스터의 모든 HSM에 로그인합니다.

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<USERNAME>

사용자의 친숙한 이름을 지정합니다. 최대 길이는 31자입니다. 허용되는 유일한 특수 문자는 밑줄(_)입니다. 이 명령에서 사용자 이름은 대소문자를 구분하지 않으며, 사용자 이름은 항상 소문자로 표시됩니다.

필수: 예

<ROLE>

이 사용자에게 할당된 역할을 지정합니다. 이 파라미터는 필수 사항입니다. 유효값은 admin, crypto-user입니다.

사용자 역할을 가져오려면 user list 명령을 사용합니다. HSM에서의 사용자 유형에 대한 자세한 내용은 HSM 사용자 이해를 참조하십시오.

<PASSWORD>

HSM에 로그인 중인 사용자의 암호를 지정합니다.

관련 주제

- CloudHSM CLI로 시작하기
- 클러스터 활성화

로그인 mfa-token-sign

멀티 팩터 인증을 사용하여 AWS CloudHSM CloudHSM CLI 로그인에서 login mfa-token-sign 명령을 사용합니다. 이 명령을 사용하려면 먼저 CloudHSM CLI용 MFA를 설정해야 합니다.

사용자 유형

다음 사용자가 이러한 명령을 실행할 수 있습니다.

- 관리자
- CU(Crypto User)

구문

```
aws-cloudhsm > help login mfa-token-sign
Login with token-sign mfa
```

```
USAGE:
    login --username <USERNAME> --role <ROLE> mfa-token-sign --token <TOKEN>

OPTIONS:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --token <TOKEN>            Filepath where the unsigned token file will be written
  -h, --help                     Print help
```

예

Example

```
aws-cloudhsm > login --username test_user --role admin mfa-token-sign --token /home/
valid.token
Enter password:
Enter signed token file path (press enter if same as the unsigned token file):
{
  "error_code": 0,
  "data": {
    "username": "test_user",
    "role": "admin"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<TOKEN>

서명되지 않은 토큰 파일이 작성되는 파일 경로입니다.

필수: 예

관련 주제

- CloudHSM CLI로 시작하기

- 클러스터 활성화
- CloudHSM CLI를 사용하여 MFA 관리

로그아웃

CloudHSM CLI의 logout 명령을 사용하여 클러스터의 각 HSM에서 로그아웃할 수 있습니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 관리자
- CU(Crypto User)

구문

```
aws-cloudhsm > help logout
Logout of your cluster

USAGE:
    logout

OPTIONS:
        --cluster-id <CLUSTER_ID> Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
    -h, --help                    Print help information
    -V, --version                 Print version information
```

예

Example

이 명령은 클러스터의 모든 HSM에서 로그아웃하게 합니다.

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

관련 주제

- CloudHSM CLI로 시작하기
- 클러스터 활성화

사용자

user은 상위 카테고리와 결합될 때 사용자에게 특정한 명령을 생성하는 명령 그룹의 상위 카테고리입니다. 현재 사용자 범주는 다음과 같은 명령으로 구성되어 있습니다.

- 사용자 변경-mfa
- 사용자의 암호 변경
- 사용자 생성
- 사용자 삭제
- 사용자 목록

사용자 변경-mfa

현재 이 범주는 다음과 같은 하위 명령으로 구성되어 있습니다.

- 사용자 변경-mfa 토큰-사인

사용자 변경-mfa 토큰-사인

CloudHSM CLI에 있는 user change-mfa 명령을 사용하여 사용자 계정의 Multi-Factor Authentication(MFA) 설정을 업데이트합니다. 모든 사용자 계정에서 이 명령을 실행할 수 있습니다. 관리자 역할이 있는 계정은 다른 사용자를 위해 이 명령을 실행할 수 있습니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 관리자
- Crypto 사용자

구문

현재 사용자가 사용할 수 있는 멀티팩터 전략은 토큰 사인이라는 단 하나뿐입니다.

```
aws-cloudhsm > help user change-mfa
Change a user's Mfa Strategy

Usage:
    user change-mfa <COMMAND>

Commands:
  token-sign  Register or Deregister a public key using token-sign mfa strategy
  help        Print this message or the help of the given subcommand(s)
```

토큰 사인 전략에서는 사인되지 않은 토큰을 기록할 토큰 파일을 요청합니다.

```
aws-cloudhsm > help user change-mfa token-sign
Register or Deregister a public key using token-sign mfa strategy

Usage: user change-mfa token-sign [OPTIONS] --username <USERNAME> --role <ROLE> <--
token <TOKEN>|--deregister>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --username <USERNAME>
          Username of the user that will be modified

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --change-password <CHANGE_PASSWORD>
          Optional: Plaintext user's password. If you do not include this argument you
 will be prompted for it

      --token <TOKEN>
          Filepath where the unsigned token file will be written. Required for enabling
 MFA for a user
```

```
      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

      --deregister
          Deregister the MFA public key, if present

      --change-quorum
          Change the Quorum public key along with the MFA key

  -h, --help
          Print help (see a summary with '-h')
```

예시

이 명령은 클러스터의 HSM당 서명되지 않은 토큰 하나를 token으로 지정된 파일에 기록합니다. 메시지가 나타나면 파일에 있는 토큰에 사인됩니다.

Example : 클러스터의 HSM당 사인서명되지 않은 토큰 하나를 작성합니다.

```
aws-cloudhsm > user change-mfa token-sign --username cu1 --change-password password --
role crypto-user --token /path/myfile
Enter signed token file path (press enter if same as the unsigned token file):
Enter public key PEM file path:/path/mypemfile
{
  "error_code": 0,
  "data": {
    "username": "test_user",
    "role": "admin"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<ROLE>

사용자 계정에 부여된 역할을 지정합니다. 이 파라미터는 필수 사항입니다. HSM에서의 사용자 유형에 대한 자세한 내용은 HSM 사용자 이해를 참조하십시오.

유효값

- 관리자: 관리자는 사용자를 관리할 수 있지만 키는 관리할 수 없습니다.
- Crypto user: crypto user는 키를 관리하고 암호화 작업에서 키를 사용할 수 있습니다.

<USERNAME>

사용자의 친숙한 이름을 지정합니다. 최대 길이는 31자입니다. 허용되는 유일한 특수 문자는 밑줄(_)입니다.

사용자를 생성한 후에는 사용자 이름을 변경할 수 없습니다. CloudHSM CLI 명령에서 역할과 암호는 대소문자를 구분하지만 사용자 이름은 그렇지 않습니다.

필수: 예

<CHANGE_PASSWORD>

MFA가 등록 취소하려는 사용자의 일반 텍스트 새 암호를 지정합니다.

필수: 예

<TOKEN>

사인되지 않은 토큰 파일이 기록될 파일 경로입니다.

필수 항목 여부: 예

<APPROVAL>

작업을 승인할 서명된 쿼럼 토큰 파일의 파일 경로를 지정합니다. 쿼럼 사용자 서비스 쿼럼 값이 1보다 큰 경우에만 필요합니다.

<DEREGISTER>

MFA 퍼블릭 키(있는 경우)의 등록을 취소합니다.

<CHANGE-QUORUM>

쿼럼 퍼블릭 키를 MFA 키와 함께 변경합니다.

관련 주제

- HSM 사용자를 위한 2FA 이해

사용자-암호 변경

CloudHSM CLI의 user change-password 명령을 사용하여 클러스터에 있는 기존 사용자의 비밀번호를 변경합니다. AWS CloudHSM 사용자에 대해 MFA를 활성화하려면 `user change-mfa` 명령을 사용합니다.

모든 사용자는 본인의 암호를 변경할 수 있습니다. 또한 관리자 역할을 가진 사용자는 클러스터에 있는 다른 사용자의 암호를 변경할 수 있습니다. 변경하기 위해 현재 암호를 입력할 필요는 없습니다.

> **Note**
>
> 현재 클러스터에 로그인한 사용자의 암호는 변경할 수 없습니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 관리자
- CU(Crypto User)

구문

> **Note**
>
> 사용자에 대해 멀티팩터 인증(MFA)을 활성화하려면 user change-mfa 명령을 사용합니다.

```
aws-cloudhsm > help user change-password
Change a user's password

    Usage:
        cloudhsm-cli user change-password [OPTIONS] --username <USERNAME> --role <ROLE>
 [--password <PASSWORD>]

    Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error
```

```
    --username <USERNAME>
        Username of the user that will be modified

    --role <ROLE>
        Role the user has in the cluster

        Possible values:
        - crypto-user: A CryptoUser has the ability to manage and use keys
        - admin:       An Admin has the ability to manage user accounts

    --password <PASSWORD>
        Optional: Plaintext user's password. If you do not include this argument you
will be prompted for it

    --approval <APPROVAL>
        Filepath of signed quorum token file to approve operation

    --deregister-mfa <DEREGISTER-MFA>
        Deregister the user's mfa public key, if present

    --deregister-quorum <DEREGISTER-QUORUM>
        Deregister the user's quorum public key, if present
-h, --help
        Print help (see a summary with '-h')
```

예

다음 예제에서는 user change-password를 사용하여 클러스터의 현재 사용자나 다른 사용자의 암호를 재설정하는 방법을 보여줍니다.

Example : 사용자의 암호 변경

클러스터의 모든 사용자는 user change-password를 사용하여 암호를 변경할 수 있습니다.

다음 출력은 Bob이 현재 CU(Crypto User)로 로그인되어 있음을 보여줍니다.

```
aws-cloudhsm > user change-password --username bob --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "bob",
```

```
    "role": "crypto-user"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<APPROVAL>

작업을 승인할 서명된 쿼럼 토큰 파일의 파일 경로를 지정합니다. 쿼럼 사용자 서비스 쿼럼 값이 1보다 큰 경우에만 필요합니다.

<DEREGISTER-MFA>

MFA 퍼블릭 키(있는 경우)의 등록을 취소합니다.

<DEREGISTER-QUORUM>

쿼럼 퍼블릭 키가 있는 경우 등록을 취소합니다.

<PASSWORD>

사용자의 일반 텍스트 새 암호를 지정합니다.

필수: 예

<ROLE>

사용자 계정에 부여된 역할을 지정합니다. 이 파라미터는 필수 사항입니다. HSM에서의 사용자 유형에 대한 자세한 내용은 HSM 사용자 이해를 참조하십시오.

유효값

- 관리자: 관리자는 사용자를 관리할 수 있지만 키는 관리할 수 없습니다.
- Crypto user: crypto user는 키를 관리하고 암호화 작업에서 키를 사용할 수 있습니다.

<USERNAME>

사용자의 친숙한 이름을 지정합니다. 최대 길이는 31자입니다. 허용되는 유일한 특수 문자는 밑줄(_)입니다.

사용자를 생성한 후에는 사용자 이름을 변경할 수 없습니다. CloudHSM CLI 명령에서 역할과 암호는 대소문자를 구분하지만 사용자 이름은 그렇지 않습니다.

필수: 예

관련 주제

- 사용자 목록
- 사용자 생성
- 사용자 삭제

사용자 변경-쿼럼

user change-quorum은 명령 그룹의 상위 범주로, 상위 범주와 결합할 경우 사용자의 쿼럼 변경과 관련된 명령을 생성할 수 있습니다.

user change-quorum은 지정된 쿼럼 전략을 사용하여 사용자 쿼럼 인증을 등록하는 데 사용됩니다. SDK 5.8.0부터는 아래와 같이 사용자가 사용할 수 있는 쿼럼 전략이 하나뿐입니다.

현재 이 범주는 다음과 같은 범주와 하위 명령으로 구성되어 있습니다.

- 토큰 사인
  - 등록

사용자 변경-쿼럼 토큰-사인

user change-quorum token-sign은 이 상위 범주와 결합할 경우 토큰 사인 쿼럼 작업과 관련된 명령을 생성하는 명령의 상위 범주입니다.

현재 이 항목은 다음과 같은 명령으로 구성되어 있습니다.

- 등록

사용자 변경-쿼럼 토큰-사인 등록

CloudHSM CLI의 user change-quorum token-sign register 명령을 사용하여 관리자 사용자를 위한 토큰 사인 쿼럼 전략을 등록할 수 있습니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 관리자

구문

```
aws-cloudhsm > help user change-quorum token-sign register
Register a user for quorum authentication with a public key

Usage: user change-quorum token-sign register --public-key <PUBLIC_KEY> --signed-
token <SIGNED_TOKEN>

Options:
      --cluster-id <CLUSTER_ID>      Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --public-key <PUBLIC_KEY>      Filepath to public key PEM file
      --signed-token <SIGNED_TOKEN>  Filepath with token signed by user private key
  -h, --help Print help (see a summary with '-h')
```

예

Example

이 명령을 실행하려면 register quorum token-sign을 원하는 사용자로 로그인해야 합니다.

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

이 user change-quorum token-sign register 명령은 퍼블릭 키를 HSM에 등록합니다. 따라서 필요한 쿼럼 값 임계값을 충족하기 위해 사용자가 쿼럼 사인을 받아야 하는 쿼럼 필수 작업에 대한 쿼럼 승인자 자격이 부여됩니다.

```
aws-cloudhsm > user change-quorum token-sign register \
    --public-key /home/mypemfile \
    --signed-token /home/mysignedtoken
{
  "error_code": 0,
```

```
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

이제 user list 명령을 실행하여 쿼럼 토큰 사인이 이 사용자에 대해 등록되었는지 확인할 수 있습니다.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [],
        "cluster-coverage": "full"
      },
      {
        "username": "admin1",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "quorum": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      }
    ]
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<PUBLIC-KEY>

퍼블릭 키 PEM 파일의 파일 경로

필수 항목 여부: 예

<SIGNED-TOKEN>

사용자 개인 키로 사인된 토큰이 있는 파일 경로

필수: 예

관련 주제

- CloudHSM CLI를 사용하여 쿼럼 인증을 관리합니다.
- 관리자에 대한 쿼럼 인증 사용: 처음 설정
- 관리자의 쿼럼 최소값 변경
- 쿼럼 인증을 지원하는 서비스 이름 및 유형

사용자 생성

CloudHSM CLI의 user create 명령은 클러스터에 사용자를 생성합니다. AWS CloudHSM 관리자 역할을 가진 사용자 계정만 이 명령을 실행할 수 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- 관리자

요구 사항

이 명령을 실행하려면 관리자로 로그인해야 합니다.

구문

```
aws-cloudhsm > help user create
Create a new user
```

```
Usage: cloudhsm-cli user create [OPTIONS] --username <USERNAME> --role <ROLE> [--
password <PASSWORD>]

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --username <USERNAME>
          Username to access the HSM cluster

      --role <ROLE>
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --password <PASSWORD>
          Optional: Plaintext user's password. If you do not include this argument you
 will be prompted for it

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

  -h, --help
          Print help (see a summary with '-h')
```

예

다음 예제에서는 user create를 사용하여 HSM에서 새 사용자를 생성하는 방법을 보여 줍니다.

Example : crypto user 생성

이 예시에서는 암호화 사용자 역할을 가진 계정을 AWS CloudHSM 클러스터에 생성합니다.

```
aws-cloudhsm > user create --username alice --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
```

```
    "username": "alice",
    "role": "crypto-user"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<USERNAME>

사용자의 친숙한 이름을 지정합니다. 최대 길이는 31자입니다. 허용되는 유일한 특수 문자는 밑줄(_)입니다. 이 명령에서 사용자 이름은 대소문자를 구분하지 않으며, 사용자 이름은 항상 소문자로 표시됩니다.

필수: 예

<ROLE>

이 사용자에게 할당된 역할을 지정합니다. 이 파라미터는 필수 사항입니다. 유효값은 admin, crypto-user입니다.

사용자 역할을 가져오려면 user list 명령을 사용합니다. HSM에서의 사용자 유형에 대한 자세한 내용은 HSM 사용자 이해를 참조하십시오.

<PASSWORD>

HSM에 로그인 중인 사용자의 암호를 지정합니다.

필수: 예

<APPROVAL>

작업을 승인할 서명된 쿼럼 토큰 파일의 파일 경로를 지정합니다. 쿼럼 사용자 서비스 쿼럼 값이 1보다 큰 경우에만 필요합니다.

관련 주제

- 사용자 목록

- 사용자 삭제
- 사용자 변경-암호

사용자 삭제

CloudHSM CLI의 user delete 명령은 클러스터에서 사용자를 삭제합니다. AWS CloudHSM 관리자 역할을 가진 사용자 계정만 이 명령을 실행할 수 있습니다. 현재 HSM에 로그인한 사용자는 삭제할 수 없습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- 관리자

요구 사항

- 키를 소유한 사용자 계정은 삭제할 수 없습니다.
- 이 명령을 실행하려면 사용자 계정에 관리자 역할이 있어야 합니다.

구문

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
aws-cloudhsm > help user delete
Delete a user

Usage: user delete [OPTIONS] --username <USERNAME> --role <ROLE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --username <USERNAME>
          Username to access the HSM cluster

      --role <ROLE>
```

```
          Role the user has in the cluster

          Possible values:
          - crypto-user: A CryptoUser has the ability to manage and use keys
          - admin:       An Admin has the ability to manage user accounts

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation
```

예

```
aws-cloudhsm > user delete --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<USERNAME>

사용자의 친숙한 이름을 지정합니다. 최대 길이는 31자입니다. 허용되는 유일한 특수 문자는 밑줄(_)입니다. 이 명령에서 사용자 이름은 대소문자를 구분하지 않으며, 사용자 이름은 항상 소문자로 표시됩니다.

필수: 예

<ROLE>

이 사용자에게 할당된 역할을 지정합니다. 이 파라미터는 필수 사항입니다. 유효값은 admin, crypto-user입니다.

사용자 역할을 가져오려면 user list 명령을 사용합니다. HSM에서의 사용자 유형에 대한 자세한 내용은 HSM 사용자 이해를 참조하십시오.

필수 여부: 예

<APPROVAL>

작업을 승인할 서명된 쿼럼 토큰 파일의 파일 경로를 지정합니다. 쿼럼 사용자 서비스 쿼럼 값이 1보다 큰 경우에만 필요합니다.

필수: 예

관련 주제

- 사용자 목록
- 사용자 생성
- 사용자의 변경-암호

사용자 목록

CloudHSM CLI의 user list 명령은 CloudHSM 클러스터에 있는 사용자 계정을 나열합니다. 이 명령을 실행하기 위해 CloudHSM CLI에 로그인할 필요는 없습니다.

> **Note**
>
> HSM을 추가하거나 삭제하는 경우 AWS CloudHSM 클라이언트와 명령줄 도구가 사용하는 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- 모든 사용자. 이 명령을 실행하기 위해 로그인할 필요는 없습니다.

구문

```
aws-cloudhsm > help user list
List the users in your cluster

USAGE:
    user list
```

```
Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

예

이 명령은 CloudHSM 클러스터에 있는 사용자를 나열합니다.

```
aws-cloudhsm > user list
{
  "error_code": 0,
  "data": {
    "users": [
      {
        "username": "admin",
        "role": "admin",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      },
      {
        "username": "test_user",
        "role": "admin",
        "locked": "false",
        "mfa": [
          {
            "strategy": "token-sign",
            "status": "enabled"
          }
        ],
        "cluster-coverage": "full"
      },
      {
        "username": "app_user",
        "role": "internal(APPLIANCE_USER)",
        "locked": "false",
        "mfa": [],
        "cluster-coverage": "full"
      }
    ]
  }
```

```
}
```

출력에는 다음의 사용자 속성이 포함됩니다.

- 사용자 이름: 사용자의 사용자 정의 이름을 표시합니다. 사용자 이름은 항상 소문자로 표시됩니다.
- 역할: 사용자가 HSM에서 수행할 수 있는 작업을 결정합니다.
- 잠금: 이 사용자 계정이 잠겼는지 여부를 나타냅니다.
- MFA: 이 사용자 계정에 지원되는 다단계 인증 메커니즘을 나타냅니다.
- 클러스터 적용 범위: 이 사용자 계정의 클러스터 전체 가용성을 나타냅니다.

관련 주제

- key_mgmt_util의 listUsers
- 사용자 생성
- 사용자 삭제
- 사용자 변경-암호

쿼럼

quorum은 quorum와 결합하면 쿼럼 인증 또는 M/N 작업과 관련된 명령을 생성하는 명령 그룹의 상위 카테고리입니다. 현재 이 카테고리는 자체 명령으로 구성된 token-sign 하위 카테고리로 구성되어 있습니다. 자세한 내용을 보려면 아래 링크를 클릭하십시오.

- 토큰 사인

관리 서비스: 쿼럼 인증은 사용자 생성, 사용자 삭제, 사용자 비밀번호 변경, 쿼럼 값 설정, 쿼럼 및 MFA 기능 비활성화와 같은 관리자 권한 서비스에 사용됩니다.

각 서비스 유형은 수행할 수 있는 특정 쿼럼 지원 서비스 작업 세트를 포함하는 적격 서비스 이름으로 더 세분화됩니다.

서비스 이름	서비스 유형	서비스 작업
사용자	관리자	• 사용자 생성 • 사용자 삭제

서비스 이름	서비스 유형	서비스 작업
		• 사용자의 암호-변경 • 사용자 변경-mfa
쿼럼	관리자	• 쿼럼 토큰 사인 set-quorum-value

관련 주제

- 관리자를 위한 쿼럼 인증 사용: 최초 설정
- CloudHSM CLI를 사용하여 쿼럼 인증 관리(M/N 액세스 제어)

쿼럼 토큰-서명

quorum token-sign은 quorum token-sign와 결합될 때 쿼럼 인증 또는 M/N 작업과 관련된 명령을 생성하는 명령 그룹의 카테고리입니다.

현재 이 범주는 다음과 같은 명령으로 구성되어 있습니다.

- delete
- 생성
- list
- list-quorum-values
- 목록-제한 시간
- set-quorum-value
- 시간 초과 설정

쿼럼 토큰-서명 삭제

CloudHSM CLI의 quorum token-sign delete 명령을 사용하여 쿼럼 인증 서비스에 대한 하나 이상의 토큰을 삭제합니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 관리자

구문

```
aws-cloudhsm > help quorum token-sign delete
Delete one or more Quorum Tokens

Usage: quorum token-sign delete --scope <SCOPE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --scope <SCOPE>
          Scope of which token(s) will be deleted

          Possible values:
          - user: Deletes all token(s) of currently logged in user
          - all:  Deletes all token(s) on the HSM
  -h, --help
          Print help (see a summary with '-h')
```

예

다음 예시는 CloudHSM CLI의 quorum token-sign delete 명령을 사용하여 쿼럼 인증 서비스의 토큰을 하나 이상 삭제하는 방법을 보여줍니다.

Example : 쿼럼 인증 서비스의 토큰을 하나 이상 삭제합니다.

```
aws-cloudhsm > quorum token-sign delete --scope all
{
  "error_code": 0,
  "data": "Deletion of quorum token(s) successful"
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우

<SCOPE>

AWS CloudHSM 클러스터에서 토큰이 삭제되는 범위입니다.

유효값

- User: 로그인한 사용자가 소유한 토큰만 삭제하는 데 사용됨.
- 모두: AWS CloudHSM 클러스터의 모든 토큰을 삭제하는 데 사용됩니다.

관련 주제

- 사용자 목록
- 사용자 생성
- 사용자 삭제

쿼럼 토큰-사인 생성

CloudHSM CLI의 quorum token-sign generate 명령을 사용하여 쿼럼 인증 서비스를 위한 토큰을 생성합니다.

서비스 사용자 및 쿼럼을 위한 HSM 클러스터의 서비스당 사용자당 하나의 활성 토큰을 획득하는 데에는 제한이 있습니다.

> **Note**
>
> 관리자만 서비스 토큰을 생성할 수 있습니다.

관리 서비스: 쿼럼 인증은 사용자 생성, 사용자 삭제, 사용자 비밀번호 변경, 쿼럼 값 설정, 쿼럼 및 MFA 기능 비활성화와 같은 관리자 권한 서비스에 사용됩니다.

각 서비스 유형은 수행할 수 있는 특정 쿼럼 지원 서비스 작업 세트를 포함하는 적격 서비스 이름으로 더 세분화됩니다.

서비스 이름	서비스 유형	서비스 작업
사용자	관리자	• 사용자 생성

서비스 이름	서비스 유형	서비스 작업
		• 사용자 삭제 • 사용자의 암호-변경 • 사용자 변경-mfa
쿼럼	관리자	• 쿼럼 토큰 사인 set-quorum-value

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 관리자
- CU(Crypto User)

구문

```
aws-cloudhsm > help quorum token-sign generate
Generate a token

Usage: quorum token-sign generate --service <SERVICE> --token <TOKEN>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --service <SERVICE>
          Service the token will be used for

          Possible values:
          - user:
            User management service is used for executing quorum authenticated user
 management operations
          - quorum:
            Quorum management service is used for setting quorum values for any quorum
 service
          - registration:
```

```
            Registration service is used for registering a public key for quorum
 authentication

      --token <TOKEN>
          Filepath where the unsigned token file will be written
  -h, --help                      Print help
```

예

이 명령은 클러스터의 HSM당 서명되지 않은 토큰 하나를 token으로 지정된 파일에 기록합니다.

Example : 클러스터의 HSM당 사인서명되지 않은 토큰 하나를 작성합니다.

```
aws-cloudhsm > quorum token-sign generate --service user --token /home/tfile
{
  "error_code": 0,
  "data": {
    "filepath": "/home/tfile"
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<SERVICE>

토큰을 생성할 쿼럼 인증 서비스를 지정합니다. 이 파라미터는 필수 사항입니다.

유효값

- 사용자: 쿼럼의 승인된 사용자 관리 작업을 실행하는 데 사용되는 사용자 관리 서비스입니다.
- 쿼럼: 모든 쿼럼 인증 서비스에 대해 쿼럼 인증 쿼럼 값을 설정하는 데 사용되는 쿼럼 관리 서비스입니다.
- 등록: 쿼럼 인증을 위한 공개 키를 등록하는 데 사용할 사인되지 않은 토큰을 생성합니다.

필수: 예

<TOKEN>

서명되지 않은 토큰 파일이 작성되는 파일 경로입니다.

필수: 예

관련 주제

- 쿼럼 인증을 지원하는 서비스 이름 및 유형

쿼럼 토큰-서명 목록

CloudHSM CLI의 quorum token-sign list 명령을 사용하여 클러스터에 있는 모든 토큰 서명 쿼럼 토큰을 나열합니다. AWS CloudHSM

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 관리자
- CU(Crypto User)

구문

```
aws-cloudhsm > help quorum token-sign list
List the token-sign tokens in your cluster

Usage: quorum token-sign list

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

예

이 명령은 클러스터에 있는 모든 토큰 서명 토큰을 나열합니다. AWS CloudHSM

Example

```
aws-cloudhsm > quorum token-sign list
{
  "error_code": 0,
```

```
  "data": {
    "tokens": [
      {
        "username": "admin",
        "service": "quorum",
        "approvals-required": 2
        "number-of-approvals": 0
        "token-timeout-seconds": 397
        "cluster-coverage": "full"
      },
      {
        "username": "admin",
        "service": "user",
        "approvals-required": 2
        "number-of-approvals": 2
        "token-timeout-seconds": 588
        "cluster-coverage": "full"
      }
    ]
  }
}
```

관련 주제

- 쿼럼 토큰-서명 생성

쿼럼 토큰 사인 list-quorum-values

CloudHSM CLI의 quorum token-sign list-quorum-values 명령을 사용하여 클러스터에 설정된 쿼럼 값을 나열합니다. AWS CloudHSM

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 모든 사용자. 이 명령을 실행하기 위해 로그인할 필요는 없습니다.

구문

```
aws-cloudhsm > help quorum token-sign list-quorum-values
List current quorum values
```

```
Usage: quorum token-sign list-quorum-values

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

예

이 명령은 각 서비스에 대해 클러스터에 설정된 쿼럼 값을 나열합니다. AWS CloudHSM

Example

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 1
  }
}
```

관련 주제

- 쿼럼 인증을 지원하는 서비스 이름 및 유형

쿼럼 토큰-서명 목록-제한 시간

모든 토큰 유형에 대한 토큰 제한 시간(초)을 얻으려면 CloudHSM CLI에서 quorum token-sign list-timeouts 명령을 사용합니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 모든 사용자. 이 명령을 실행하기 위해 로그인할 필요는 없습니다.

구문

```
aws-cloudhsm > help quorum token-sign list-timeouts
List timeout durations in seconds for token validity
```

```
Usage: quorum token-sign list-timeouts

Options:
      --cluster-id <CLUSTER_ID>  Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
  -h, --help                     Print help
```

예

Example

```
aws-cloudhsm > quorum token-sign list-timeouts
{
  "error_code": 0,
  "data": {
    "generated": 600,
    "approved": 600
  }
}
```

출력에는 다음이 포함됩니다.

- 생성됨: 생성된 토큰이 승인되는 데 걸리는 제한 시간(초).
- 승인됨: 승인된 토큰을 사용하여 쿼럼 승인 작업을 실행하는 데 걸리는 제한 시간(초).

관련 주제

- 쿼럼 토큰-서명 설정-제한 시간

쿼럼 토큰 사인 set-quorum-value

CloudHSM CLI의 quorum token-sign set-quorum-value 명령을 사용하여 쿼럼 인증 서비스의 새 쿼럼 값을 설정합니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 관리자

구문

```
aws-cloudhsm > help quorum token-sign set-quorum-value
Set a quorum value

Usage: quorum token-sign set-quorum-value [OPTIONS] --service <SERVICE> --value <VALUE>

Options:
      --cluster-id <CLUSTER_ID>
          Unique Id to choose which of the clusters in the config file to run the
 operation against. If not provided, will fall back to the value provided when
 interactive mode was started, or error

      --service <SERVICE>
          Service the token will be used for

          Possible values:
          - user:
            User management service is used for executing quorum authenticated user
 management operations
          - quorum:
            Quorum management service is used for setting quorum values for any quorum
 service

      --value <VALUE>
          Value to set for service

      --approval <APPROVAL>
          Filepath of signed quorum token file to approve operation

  -h, --help
          Print help (see a summary with '-h')
```

예

Example

다음 예제에서 이 명령은 클러스터의 HSM당 하나의 서명되지 않은 토큰을 토큰으로 지정된 파일에 씁니다. 메시지가 나타나면 파일에 있는 토큰에 사인됩니다.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2
{
  "error_code": 0,
```

```
  "data": "Set Quorum Value successful"
}
```

그런 다음 list-quorum-values 명령을 실행하여 쿼럼 관리 서비스의 쿼럼 값이 설정되었는지 확인할 수 있습니다.

```
aws-cloudhsm > quorum token-sign list-quorum-values
{
  "error_code": 0,
  "data": {
    "user": 1,
    "quorum": 2
  }
}
```

인수

<CLUSTER_ID>

이 작업을 실행할 클러스터의 ID입니다.

필수: 여러 클러스터가 구성된 경우.

<APPROVAL>

HSM에서 승인할 서명된 토큰 파일의 파일 경로입니다.

<SERVICE>

토큰을 생성할 쿼럼 인증 서비스를 지정합니다. 이 파라미터는 필수 사항입니다. 서비스 유형 및 이름에 대한 자세한 내용은 쿼럼 인증을 지원하는 서비스 이름 및 유형을 참조하십시오.

유효값

- 사용자: 사용자 관리 서비스입니다. 쿼럼 인증된 사용자 관리 작업을 실행하는 데 사용되는 서비스입니다.
- 쿼럼: 쿼럼 관리 서비스입니다. 모든 쿼럼 인증 서비스에 대해 쿼럼 인증된 쿼럼 값을 설정하는 데 사용되는 서비스입니다.
- 등록: 쿼럼 인증을 위한 퍼블릭 키를 등록하는 데 사용할 서명되지 않은 토큰을 생성합니다.

필수 항목 여부: 예

<VALUE>

설정할 쿼럼 값을 지정합니다. 최대 쿼럼 값은 8입니다.

필수 항목 여부: 예

관련 주제

- 쿼럼 토큰 서명 list-quorum-values
- 쿼럼 인증을 지원하는 서비스 이름 및 유형

쿼럼 토큰-서명 설정-제한 시간

CloudHSM CLI의 quorum token-sign set-timeout 명령을 사용하여 각 토큰 유형에 대한 토큰 타임아웃 기간을 초 단위로 설정합니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 관리자

구문

```
aws-cloudhsm > help quorum token-sign set-timeout
Set timeout duration in seconds for token validity

Usage: quorum token-sign set-timeout <--generated <GENERATED> |--approved <APPROVED>>

Options:
      --cluster-id <CLUSTER_ID>   Unique Id to choose which of the clusters in the
 config file to run the operation against. If not provided, will fall back to the value
 provided when interactive mode was started, or error
      --generated <GENERATED>     Timeout period in seconds for a generated (non-
approved) token to be approved
      --approved <APPROVED>      Timeout period in seconds for an approved token to be
 used to execute a quorum operation
  -h, --help    Print help (see a summary with '-h')
```

예

다음 예제에서는 quorum token-sign set-timeout 명령을 통해 토큰 제한 시간을 설정하는 방법을 보여 줍니다.

```
aws-cloudhsm > quorum token-sign set-timeout --generated 900
```

```
{
  "error_code": 0,
  "data": "Set token timeout successful"
}
```

관련 주제

- 쿼럼 토큰-서명 목록-타임아웃

CloudHSM 관리 유틸리티(CMU)

cloudhsm_mgmt_util 명령줄 도구는 CO(Crypto Officer)가 HSM에서 사용자를 관리하는 데 사용할 수 있습니다. 사용자를 생성, 삭제 및 나열하고 사용자 암호를 변경하는 도구가 여기에 포함됩니다.

KMU와 CMU는 클라이언트 SDK 3 제품군의 일부입니다. 클라이언트 SDK 3 및 관련 명령줄 도구 (키 관리 유틸리티 및 CloudHSM 관리 유틸리티) 는 HSM 유형 hsm1.medium에서만 사용할 수 있습니다.

cloudhsm_mgmt_util에는 CU(Crypto User)가 키를 공유하거나 키 속성을 가져오고 설정할 수 있는 명령도 포함됩니다. 이러한 명령은 프라이머리 키 관리 도구 key_mgmt_util의 키 관리 명령을 보완합니다.

빠른 시작은 복제된 클러스터 관리 섹션을 참조하십시오. cloudhsm_mgmt_util 명령 및 명령 사용 예제에 대한 자세한 내용은 cloudhsm_mgmt_util 명령 참조 섹션을 참조하십시오.

주제

- AWS CloudHSM 관리 유틸리티 지원 플랫폼
- CloudHSM 관리 유틸리티(CMU) 시작하기
- AWS CloudHSM 클라이언트 설치 및 구성 (Linux)
- AWS CloudHSM 클라이언트 설치 및 구성 (Windows)
- cloudhsm_mgmt_util 명령 참조

AWS CloudHSM 관리 유틸리티 지원 플랫폼

Linux 지원

- Amazon Linux
- Amazon Linux 2

- CentOS 6.10+
- CentOS 7.3+
- CentOS 8
- Red Hat Enterprise Linux(RHEL) 6.10+
- Red Hat Enterprise Linux(RHEL) 7.9+
- Red Hat Enterprise Linux(RHEL) 8
- Ubuntu 16.04 LTS
- Ubuntu 18.04 LTS

윈도우 지원

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

CloudHSM 관리 유틸리티(CMU) 시작하기

CloudHSM 관리 유틸리티(CMU)를 사용하면 하드웨어 보안 모듈(HSM) 사용자를 관리할 수 있습니다. 이 주제를 사용하여 사용자 생성, 사용자 등록 및 CMU를 클러스터에 연결하는 등 기본적인 HSM 사용자 관리 태스크를 시작할 수 있습니다.

1. CMU를 사용하려면 먼저 구성 도구를 사용하여 클러스터에 있는 HSM 중 하나의 `--cmu` 파라미터와 IP 주소로 로컬 CMU 구성을 업데이트해야 합니다. CMU를 사용할 때마다 이 작업을 수행하여 클러스터의 모든 HSM에서 HSM 사용자를 관리하고 있는지 확인하십시오.

   Linux

   ```
   $ sudo /opt/cloudhsm/bin/configure --cmu <IP address>
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe --cmu <IP address>
   ```

2. 다음 명령을 사용하여 대화형 모드에서 CLI를 시작합니다.

Linux

```
$ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

Windows

```
C:\Program Files\Amazon\CloudHSM> .\cloudhsm_mgmt_util.exe C:\ProgramData\Amazon
\CloudHSM\data\cloudhsm_mgmt_util.cfg
```

보유한 HSM의 수에 따라 다음과 같이 출력됩니다.

```
Connecting to the server(s), it may take time
depending on the server(s) load, please wait...

Connecting to server '10.0.2.9': hostname '10.0.2.9', port 2225...
Connected to server '10.0.2.9': hostname '10.0.2.9', port 2225.

Connecting to server '10.0.3.11': hostname '10.0.3.11', port 2225...
Connected to server '10.0.3.11': hostname '10.0.3.11', port 2225.

Connecting to server '10.0.1.12': hostname '10.0.1.12', port 2225...
Connected to server '10.0.1.12': hostname '10.0.1.12', port 2225.
```

프롬프트는 cloudhsm_mgmt_util이 실행 중일 때 `aws-cloudhsm>`로 변경됩니다.

3. loginHSM 명령을 사용하여 클러스터에 로그인합니다. 모든 유형의 사용자가 이 명령을 사용하여 클러스터에 로그인할 수 있습니다.

   다음 예시의 명령은 기본 crypto officer (CO)인 관리자(admin)에 로그인합니다. 클러스터를 활성화한 경우에만 이 사용자의 암호를 설정할 수 있습니다. `-hpswd` 파라미터를 사용하여 비밀번호를 숨길 수 있습니다.

   ```
   aws-cloudhsm>loginHSM CO admin -hpswd
   ```

   시스템이 암호를 묻는 메시지를 표시합니다. 암호를 입력하면 시스템이 암호를 숨기고 출력에는 명령이 성공적으로 수행되었으며 클러스터의 모든 HSM에 연결되었음이 표시됩니다.

   ```
   Enter password:
   ```

```
loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)
loginHSM success on server 2(10.0.1.12)
```

4. 클러스터의 모든 사용자를 나열하는 데 listUsers을 사용합니다.

```
aws-cloudhsm>listUsers
```

CMU는 클러스터의 모든 사용자를 나열합니다.

```
Users on server 0(10.0.2.9):
Number of users found:2

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                   NO
               0               NO
         2              AU              app_user                                NO
               0               NO
Users on server 1(10.0.3.11):
Number of users found:2

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                   NO
               0               NO
         2              AU              app_user                                NO
               0               NO
Users on server 2(10.0.1.12):
Number of users found:2

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                   NO
               0               NO
         2              AU              app_user                                NO
               0               NO
```

5. **password1**의 암호를 사용하여 **example_user**라는 이름의 CU 사용자를 생성하기 위해 createUser을 사용합니다.

애플리케이션에서 CU 사용자를 사용하여 암호화 및 키 관리 작업을 수행합니다. 3단계에서 CO 사용자로 로그인했기 때문에 CU 사용자를 생성할 수 있습니다. CO 사용자만 CMU를 사용하여 사용자 생성 및 삭제, 다른 사용자의 암호 변경과 같은 사용자 관리 태스크를 수행할 수 있습니다.

```
aws-cloudhsm>createUser CU example_user password1
```

CMU는 사용자 생성 작업에 대한 메시지를 표시합니다.

```
**********************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
**************************************************************

Do you want to continue(y/n)?
```

6. CU 사용자 **example_user**을 생성하려면 **y**를 입력합니다.
7. 클러스터의 모든 사용자를 나열하는 데 listUsers을 사용합니다.

```
aws-cloudhsm>listUsers
```

CMU는 방금 생성한 새 CU 사용자를 포함하여 클러스터의 모든 사용자를 나열합니다.

```
Users on server 0(10.0.2.9):
Number of users found:3

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                NO
               0               NO
         2              AU              app_user                             NO
               0               NO
         3              CU              example_user                         NO
               0               NO
Users on server 1(10.0.3.11):
Number of users found:3
```

```
    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                   NO
               0              NO
         2              AU              app_user                                NO
               0              NO
         3              CU              example_user                            NO
               0              NO
Users on server 2(10.0.1.12):
Number of users found:3

    User Id             User Type       User Name
 MofnPubKey    LoginFailureCnt         2FA
         1              CO              admin                                   NO
               0              NO
         2              AU              app_user                                NO
               0              NO
         3              CU              example_user                            NO
               0              NO
```

8. logoutHSM 명령을 사용하여 HSM에서 로그아웃합니다.

```
aws-cloudhsm>logoutHSM


logoutHSM success on server 0(10.0.2.9)
logoutHSM success on server 1(10.0.3.11)
logoutHSM success on server 2(10.0.1.12)
```

9. quit 명령을 사용하여 cloudhsm_mgmt_util를 중지합니다.

```
aws-cloudhsm>quit


disconnecting from servers, please wait...
```

AWS CloudHSM 클라이언트 설치 및 구성 (Linux)

AWS CloudHSM 클러스터의 HSM과 상호 작용하려면 Linux용 AWS CloudHSM 클라이언트 소프트웨어가 필요합니다. 이전에 생성한 Linux EC2 클라이언트 인스턴스에 설치해야 합니다. Windows를 사용

하는 경우에도 클라이언트를 설치할 수 있습니다. 자세한 내용은 AWS CloudHSM 클라이언트 설치 및 구성 (Windows) 섹션을 참조하십시오.

Tasks

- AWS CloudHSM 클라이언트 및 명령줄 도구를 설치합니다.
- 클라이언트 구성 편집

AWS CloudHSM 클라이언트 및 명령줄 도구를 설치합니다.

클라이언트 인스턴스에 연결하고 다음 명령을 실행하여 AWS CloudHSM 클라이언트 및 명령줄 도구를 다운로드하고 설치합니다.

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-latest.el6.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb
```

클라이언트 구성 편집

AWS CloudHSM 클라이언트를 사용하여 클러스터에 연결하려면 먼저 클라이언트 구성을 편집해야 합니다.

클라이언트 구성을 편집하려면

1. cloudhsm_mgmt_util에 클라이언트 SDK 3을 설치하는 경우 다음 단계를 완료하여 클러스터의 모든 노드가 동기화되는지 확인합니다.

   a. configure -a ***<IP of one of the HSMs>***를 실행합니다.

   b. 클라이언트 서비스를 다시 시작합니다.

   c. config -m를 실행합니다.

2. 발급 인증서 복사 - 클러스터 인증서에 서명하는 데 사용한 인증서를 클라이언트 인스턴스의 `/opt/cloudhsm/etc/customerCA.crt` 위치에 복사합니다. 인증서를 이 위치로 복사하려면 클라이언트 인스턴스에 대해 인스턴스 루트 사용자 권한이 필요합니다.

3. 다음 configure 명령을 사용하여 클러스터에 있는 HSM의 IP 주소를 지정하여 AWS CloudHSM 클라이언트 및 명령줄 도구의 구성 파일을 업데이트하십시오. HSM의 IP 주소를 가져오려면 AWS CloudHSM 콘솔에서 클러스터를 보거나 명령을 실행하십시오. describe-clusters AWS CLI 명령의 출력에서 HSM의 IP 주소는 `EniIp` 필드의 값입니다. HSM이 두 개 이상인 경우 임의 HSM의 IP 주소를 선택합니다.

   ```
   sudo /opt/cloudhsm/bin/configure -a <IP address>

   Updating server config in /opt/cloudhsm/etc/cloudhsm_client.cfg
   Updating server config in /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

4. 클러스터 활성화로 이동합니다.

AWS CloudHSM 클라이언트 설치 및 구성 (Windows)

Windows의 AWS CloudHSM 클러스터에서 HSM을 사용하려면 Windows용 AWS CloudHSM 클라이언트 소프트웨어가 필요합니다. 이전에 생성한 Windows 서버 인스턴스에 설치해야 합니다.

최신 Windows 클라이언트 및 명령줄 도구를 설치(또는 업데이트)하려면

1. Windows 서버 인스턴스에 연결합니다.
2. AWSCloudHSMClient-latest.msi 설치 프로그램을 다운로드하세요.
3. cloudhsm_mgmt_util에 클라이언트 SDK 3을 설치하는 경우 다음 단계를 완료하여 클러스터의 모든 노드가 동기화되는지 확인합니다.
   a. configure -a ***<IP of one of the HSMs>***를 실행합니다.
   b. 클라이언트 서비스를 다시 시작합니다.
   c. config -m를 실행합니다.
4. 다운로드 위치로 이동하여 관리자 권한으로 설치 프로그램 (AWSCloudHSMClient-latest.msi) 을 실행합니다.
5. 설치 프로그램 지침을 따르고 설치 프로그램이 완료되면 닫기를 선택합니다.
6. 자체 서명된 발급 인증서 복사 - 클러스터 인증서에 서명하는 데 사용된 인증서를 `C:\ProgramData\Amazon\CloudHSM` 폴더로 복사합니다.
7. 다음 명령을 실행하여 구성 파일을 업데이트합니다. 업데이트하려면 재구성 중에 클라이언트를 중지했다 시작해야 합니다.

```
C:\Program Files\Amazon\CloudHSM\bin\ configure.exe -a <HSM IP address>
```

8. 클러스터 활성화로 이동합니다.

참고:

- 클라이언트를 업데이트하려는 경우, 이전 설치의 기존 구성 파일을 덮어쓰지 않습니다.
- Windows용 AWS CloudHSM 클라이언트 설치 프로그램은 암호화 API: 차세대 (CNG) 및 KSP (키 저장소 공급자) 를 자동으로 등록합니다. 클라이언트를 제거하려면 설치 프로그램을 다시 실행하고 제거 지침을 따르십시오.
- Linux를 사용하는 경우, Linux 클라이언트를 설치할 수 있습니다. 자세한 내용은 AWS CloudHSM 클라이언트 설치 및 구성 (Linux) 섹션을 참조하십시오.

cloudhsm_mgmt_util 명령 참조

cloudhsm_mgmt_util 명령줄 도구는 CO(Crypto Officer)가 HSM에서 사용자를 관리하는 데 사용할 수 있습니다. 이 도구에는 CU(Crypto User)가 키를 공유하거나 키 속성을 가져오고 설정할 수 있는 명령도 포함됩니다. 이러한 명령은 key_mgmt_util 명령줄 도구의 기본 키 관리 명령을 보완합니다.

빠른 시작은 복제된 클러스터 관리 섹션을 참조하십시오.

cloudhsm_mgmt_util 명령을 실행하기 전에 cloudhsm_mgmt_util을 시작하고 HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 계정 유형으로 로그인해야 합니다.

모든 cloudhsm_mgmt_util 명령을 나열하려면 다음 명령을 실행합니다.

```
aws-cloudhsm> help
```

cloudhsm_mgmt_util 명령의 구문을 가져오려면 다음 명령을 실행합니다.

```
aws-cloudhsm> help <command-name>
```

> **Note**
>
> 설명서에 따라 구문을 사용합니다. 기본 제공 소프트웨어 도움말이 추가 옵션을 제공하기도 하지만 이 옵션이 꼭 지원되는 것은 아니므로 프로덕션 코드에 활용해서는 안 됩니다.

명령을 실행하려면 명령 이름을 입력하거나 다른 cloudhsm_mgmt_util 명령의 이름과 구별하는 데 충분한 정도의 이름을 입력합니다.

예를 들어, HSM의 사용자 목록을 가져오려면 listUsers 또는 listU를 입력합니다.

```
aws-cloudhsm> listUsers
```

cloudhsm_mgmt_util 세션을 종료하려면 다음 명령을 실행합니다.

```
aws-cloudhsm> quit
```

키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

다음 주제에서는 cloudhsm_mgmt_util의 명령에 대해 설명합니다.

> **Note**
>
> key_mgmt_util과 cloudhsm_mgmt_util의 일부 명령은 이름이 같습니다. 하지만 일반적으로 명령은 구문, 출력 및 기능이 조금씩 다릅니다.

Command	설명	사용자 유형
changePswd	HSM 사용자의 암호를 변경합니다. 모든 사용자는 본인의 암호를 변경할 수 있습니다. CO는 모든 사용자의 암호를 변경할 수 있습니다.	CO
createUser	HSM에서 모든 유형의 사용자를 생성합니다.	CO
deleteUser	HSM에서 모든 유형의 사용자를 삭제합니다.	CO
findAllKeys	사용자가 소유하거나 공유하는 키를 가져옵니다. 또한 각 HSM의 모든 키에 대한 키 소유권 및 공유 데이터의 해시를 가져옵니다.	CO, AU
getAttribute	AWS CloudHSM 키의 속성 값을 가져와서 파일이나 stdout에 씁니다 (표준 출력).	CU
getCert	특정 HSM의 인증서를 가져와서 원하는 인증서 형식으로 저장합니다.	모두.
getHSMInfo	특정 HSM이 실행되는 하드웨어에 대한 정보를 가져옵니다.	모두. 로그인은 필요하지 않습니다.

Command	설명	사용자 유형
getKeyInfo	키의 소유자, 공유 사용자 및 쿼럼 인증 상태를 가져옵니다.	모두. 로그인은 필요하지 않습니다.
info	HSM에 대한 정보를 가져옵니다(IP 주소, 호스트 이름, 포트, 현재 사용자 등).	모두. 로그인은 필요하지 않습니다.
listUsers	각 HSM의 사용자, 해당 사용자 유형/ID 및 기타 속성을 가져옵니다.	모두. 로그인은 필요하지 않습니다.
loginHSM 및 logoutHSM	HSM에 로그인하고 로그아웃합니다.	모두.
quit	cloudhsm_mgmt_util을 종료합니다.	모두. 로그인은 필요하지 않습니다.
서버	HSM에서 서버 모드에 들어가고 나옵니다.	모두.
registerQuorumPub키	HSM 사용자를 비대칭 RSA-2048 키 페어와 연결합니다.	CO
setAttribute	기존 키의 레이블, 암호화, 암호 해독, 래핑 및 언래핑 속성의 값을 변경합니다.	CU
shareKey	기존 키를 다른 사용자와 공유합니다.	CU
syncKey	복제된 클러스터 AWS CloudHSM 간에 키를 동기화합니다.	CU, CO

Command	설명	사용자 유형
syncUser	복제된 클러스터 간에 사용자를 동기화합니다. AWS CloudHSM	CO

changePswd

cloudhsm_mgmt_util에서 changePswd 명령은 클러스터의 HSM에 있는 기존 사용자의 암호를 변경합니다.

모든 사용자는 본인의 암호를 변경할 수 있습니다. 또한 Crypto Officer(CO 및 PCO)는 다른 CO 또는 Crypto User(CU)의 암호를 변경할 수 있습니다. 변경하기 위해 현재 암호를 입력할 필요는 없습니다.

> **Note**
>
> 현재 AWS CloudHSM 클라이언트에 로그인한 사용자 또는 key_mgmt_util의 비밀번호는 변경할 수 없습니다.

changePswd 문제를 해결하려면

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- Crypto officers(CO)
- 암호화 사용자(Crypto User)

구문

구문 다이어그램에 지정된 순서대로 인수를 입력합니다. -hpswd 파라미터를 사용하여 암호를 숨길 수 있습니다. CO 사용자에 대해 2단계 인증(2FA)를 활성화하려면 -2fa 파라미커를 사용하고 파일 경로를 포함합니다. 자세한 정보는 the section called “인수”을 참조하세요.

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

예제

다음 예제에서는 changePassword를 사용하여 HSM의 현재 사용자나 다른 사용자의 암호를 재설정하는 방법을 보여줍니다.

Example : 사용자의 암호 변경

HSM의 모든 사용자는 changePswd를 사용하여 자신의 암호를 변경할 수 있습니다. 암호를 변경하기 전에 info를 사용하여 로그인한 사용자의 사용자 이름 및 사용자 유형을 비롯해 클러스터의 각 HSM에 대한 정보를 가져오십시오.

다음 출력은 Bob이 현재 CU(Crypto User)로 로그인되어 있음을 보여줍니다.

```
        aws-cloudhsm> info server 0

Id      Name                    Hostname         Port   State           Partition
  LoginState
0       10.1.9.193              10.1.9.193      2225    Connected       hsm-jqici4covtv
  Logged in as 'bob(CU)'

aws-cloudhsm> info server 1

Id      Name                    Hostname         Port   State           Partition
  LoginState
1       10.1.10.7               10.1.10.7       2225    Connected       hsm-ogi3sywxbqx
  Logged in as 'bob(CU)'
```

암호를 변경하기 위해 Bob은 changePswd, 사용자 유형, 사용자 이름 및 새 암호 순서로 실행합니다.

```
aws-cloudhsm> changePswd CU bob newPassword
```

```
************************CAUTION**********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?y
Changing password for bob(CU) on 2 nodes
```

Example : 다른 사용자의 암호 변경하기

HSM에서 다른 CO 또는 CU의 암호를 변경하려면 CO 또는 PCO여야 합니다. 다른 사용자의 암호를 변경하기 전에 info 명령을 사용하여 사용자 유형이 CO 또는 PCO임을 확인하십시오.

다음 출력에서는 CO인 Alice가 현재 로그인되어 있음을 확인합니다.

```
aws-cloudhsm>info server 0

Id      Name             Hostname         Port   State           Partition
 LoginState
0       10.1.9.193       10.1.9.193        2225   Connected      hsm-jqici4covtv
 Logged in as 'alice(CO)'


aws-cloudhsm>info server 1

Id      Name             Hostname         Port   State           Partition
 LoginState
0       10.1.10.7        10.1.10.7        2225   Connected       hsm-ogi3sywxbqx
 Logged in as 'alice(CO)'
```

Alice는 다른 사용자 John의 암호를 재설정하려고 합니다. 암호를 변경하기 전 listUsers 명령을 사용하여 John의 사용자 유형을 확인합니다.

다음 출력에 John이 CO 사용자로 나열됩니다.

```
aws-cloudhsm> listUsers
Users on server 0(10.1.9.193):
Number of users found:5
```

```
   User Id             User Type       User Name            MofnPubKey
 LoginFailureCnt         2FA
        1              PCO             admin                   YES              0
            NO
        2              AU              jane                     NO              0
            NO
        3              CU              bob                      NO              0
            NO
        4              CU              alice                    NO              0
            NO
        5              CO              john                     NO              0
            NO
Users on server 1(10.1.10.7):
Number of users found:5

   User Id             User Type       User Name            MofnPubKey
 LoginFailureCnt         2FA
        1              PCO             admin                   YES              0
            NO
        2              AU              jane                     NO              0
            NO
        3              CU              bob                      NO              0
            NO
        4              CO              alice                    NO              0
            NO
        5              CO              john                     NO              0
            NO
```

암호를 변경하기 위해 Alice는 changePswd, John의 사용자 유형, 사용자 이름 및 새 암호 순서로 실행합니다.

```
aws-cloudhsm>changePswd CO john newPassword

*************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?y
Changing password for john(CO) on 2 nodes
```

인수

구문 다이어그램에 지정된 순서대로 인수를 입력합니다. -hpswd 파라미터를 사용하여 암호를 숨길 수 있습니다. CO 사용자에 대해 2FA를 활성화하려면 -2fa 파라미터를 사용하고 파일 경로를 포함하십시오. 2FA 작업에 대한 자세한 내용은 CMU를 사용하여 2FA 관리 섹션을 참조하세요.

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user-type>

암호를 변경할 사용자의 현재 유형을 지정합니다. changePswd를 사용하여 사용자 유형을 변경할 수는 없습니다.

유효한 값은 CO, CU, PCO 및 PRECO입니다.

사용자 유형을 가져오려면 listUsers를 사용합니다. HSM에서의 사용자 유형에 대한 자세한 내용은 HSM 사용자 이해 섹션을 참조하십시오.

필수 여부: 예

<user-name>

사용자의 표시 이름을 지정합니다. 이 파라미터는 대소문자를 구분하지 않습니다. changePswd를 사용하여 사용자 이름을 변경할 수는 없습니다.

필수 여부: 예

<password | -hpswd >

사용자의 새 암호를 지정합니다. 7~32자의 문자열을 입력합니다. 이 값은 대소문자를 구분합니다. 암호를 입력하면 일반 텍스트로 암호가 나타납니다. 암호를 숨기려면 암호 대신 -hpswd 파라미터를 사용하고 표시되는 프롬프트를 따르십시오.

필수 여부: 예

[-2fa </path/to/authdata>]

이 CO 사용자에 대해 2FA를 활성화하도록 지정합니다. 2FA 설정에 필요한 데이터를 가져오려면 파일 시스템의 위치에 대한 경로를 -2fa 파라미터 뒤에 파일 이름과 함께 포함시키십시오. 2FA 작업에 대한 자세한 내용은 CMU를 사용하여 2FA 관리 섹션을 참조하세요.

필수 여부: 아니요

관련 주제

- info
- listUsers
- createUser
- deleteUser

createUser

cloudhsm_mgmt_util 에서 createUser 명령은 HSM에 사용자를 생성합니다. crypto officer (CO 및 PRECO)만이 이 명령을 실행할 수 있습니다. 명령이 성공하면 클러스터의 모든 HSM에서 사용자가 생성됩니다.

CreateUser 문제를 해결하려면

하지만 HSM 구성이 정확하지 않으면 일부 HSM에서 사용자가 생성되지 않을 수 있습니다. 사용자가 누락된 HSM에 사용자를 추가하려면 이 사용자가 누락된 HSM에서만 syncUser 또는 createUser 명령을 사용하십시오. 구성 오류를 방지하려면 옵션과 함께 구성 도구를 실행하십시오. `-m`

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- crypto officers (CO, PRECO)

구문

구문 다이어그램에 지정된 순서대로 인수를 입력합니다. `-hpswd` 파라미터를 사용하여 암호를 숨길 수 있습니다. 2단계 인증 (2FA) 을 사용하는 CO 사용자를 만들려면 `-2fa` 매개변수를 사용하고 파일 경로를 포함하세요. 자세한 정보는 the section called “인수”을 참조하세요.

```
createUser <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

예제

다음 예제에서는 createUser를 사용하여 HSM에서 새 사용자를 생성하는 방법을 보여 줍니다.

Example : crypto officer 생성

이 예제에서는 클러스터의 HSM에서 CO(Crypto Officer)를 생성합니다. 첫 번째 명령은 loginHSM을 사용하여 HSM에 CO(Crypto Officer)로 로그인합니다.

```
aws-cloudhsm> loginHSM CO admin 735782961

loginHSM success on server 0(10.0.0.1)
loginHSM success on server 1(10.0.0.2)
loginHSM success on server 1(10.0.0.3)
```

두 번째 명령은 createUser 명령을 사용하여 HSM에서 새 CO(Crypto Officer)인 `alice`를 생성합니다.

주의 메시지에서는 명령이 클러스터의 모든 HSM에서 사용자를 생성한다고 설명합니다. 하지만 명령이 실패하는 HSM이 있는 경우, 해당 HSM에는 사용자가 존재하지 않습니다. 계속하려면 y를 입력합니다.

출력은 클러스터의 3개 HSM 모두에서 새 사용자가 생성되었음을 보여 줍니다.

```
aws-cloudhsm> createUser CO alice 391019314

*************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?Invalid option, please type 'y' or 'n'

Do you want to continue(y/n)?y
Creating User alice(CO) on 3 nodes
```

명령이 완료되면 `alice`는 HSM의 사용자 암호 변경 등 `admin` CO 사용자와 동일한 권한을 HSM에서 갖습니다.

마지막 명령은 listUsers 명령을 사용하여 alice가 클러스터의 3개 HSM 모두에 존재함을 확인합니다. 출력은 alice에게 사용자 ID 3이 할당되었다는 것도 보여 줍니다.. 사용자 ID를 사용하여 다른 명령 (예:) alice 에서 식별할 수 findAllKeys있습니다.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
         3              CO              alice                                 NO
           0               NO
Users on server 1(10.0.0.2):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
         3              CO              alice                                 NO
           0               NO

Users on server 1(10.0.0.3):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PRECO           admin                                YES
           0               NO
         2              AU              app_user                              NO
           0               NO
         3              CO              alice                                 NO
           0               NO
```

Example : crypto user 생성

이 예제는 HSM에서 CU(crypto user)인 bob을 생성합니다. crypto user는 키를 생성하고 관리할 수 있지만 사용자를 관리할 수는 없습니다.

주의 메시지에 응답하여 y를 입력한 후 출력은 클러스터의 3개 HSM 모두에서 bob이 생성되었음을 보여 줍니다. 새 CU는 HSM에 로그인하여 키를 생성하고 관리할 수 있습니다.

이 명령에서 사용한 암호 값은 `defaultPassword`입니다. 나중에 bob 또는 아무 CO나 changePswd 명령을 사용하여 bob의 암호를 변경할 수 있습니다.

```
aws-cloudhsm> createUser CU bob defaultPassword

************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)?Invalid option, please type 'y' or 'n'

Do you want to continue(y/n)?y
Creating User bob(CU) on 3 nodes
```

인수

구문 다이어그램에 지정된 순서대로 인수를 입력합니다. `-hpswd` 파라미터를 사용하여 암호를 숨길 수 있습니다. 2FA가 활성화된 CO 사용자를 생성하려면 `-2fa` 매개변수를 사용하고 파일 경로를 포함하십시오. 2FA에 대한 자세한 내용은 CMU를 사용하여 2FA 관리 단원을 참조하세요.

```
createUser <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user-type>

사용자 유형을 지정합니다. 이 파라미터는 필수 사항입니다.

HSM에서의 사용자 유형에 대한 자세한 내용은 HSM 사용자 이해 섹션을 참조하십시오.

유효값:

- CO: Crypto officer는 사용자를 관리할 수 있지만 키를 관리할 수는 없습니다.

- CU: crypto user는 키를 관리하고 암호화 작업에서 키를 사용할 수 있습니다.

PRECO는 HSM 활성화 도중 암호를 할당할 때 CO로 변환됩니다.

필수 여부: 예

<user-name>

사용자의 친숙한 이름을 지정합니다. 최대 길이는 31자입니다. 허용되는 유일한 특수 문자는 밑줄(_)입니다.

사용자를 생성한 후에는 사용자 이름을 변경할 수 없습니다. cloudhsm_mgmt_util 명령에서 사용자 유형과 암호는 대소문자를 구분하지만 사용자 이름은 대소문자를 구분하지 않습니다.

필수 여부: 예

<password | -hpswd >

사용자의 암호를 지정합니다. 7~32자의 문자열을 입력합니다. 이 값은 대소문자를 구분합니다. 암호를 입력하면 일반 텍스트로 암호가 나타납니다. 암호를 숨기려면 암호 대신 -hpswd 파라미터를 사용하고 표시되는 프롬프트를 따르십시오.

사용자 암호를 변경하려면 changePswd를 사용합니다. HSM 사용자는 자신의 암호를 변경할 수 있지만, CO 사용자는 HSM에 있는(유형 불문하고) 모든 사용자의 암호를 변경할 수 있습니다.

필수 여부: 예

[-2fa </path/to/authdata>]

2FA가 활성화된 CO 사용자 생성을 지정합니다. 2FA 인증을 설정하는 데 필요한 데이터를 가져오려면 파일 시스템의 특정 위치에 대한 경로를 매개변수 뒤에 파일 이름과 함께 포함시키십시오. -2fa 2FA로 작업 및 설정하는 법에 대한 자세한 내용은 CMU를 사용하여 2FA 관리 단원을 참조하십시오.

필수 여부: 아니요

관련 주제

- listUsers
- deleteUser
- syncUser
- changePswd

deleteUser

cloudhsm_mgmt_util의 deleteUser 명령은 하드웨어 보안 모듈(HSM)에서 사용자를 삭제합니다. crypto officers(CO)만이 이 명령을 실행할 수 있습니다. 현재 HSM에 로그인한 사용자는 삭제할 수 없습니다. 사용자 삭제에 대한 자세한 내용은 HSM 사용자 삭제 방법을 참조하십시오.

> **Tip**
>
> 키를 소유한 CU(Crypto User)는 삭제할 수 없습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- CO

구문

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
deleteUser <user-type> <user-name>
```

예

이 예제는 클러스터의 HSM에서 CO(Crypto Officer)를 삭제합니다. 첫 번째 명령은 listUsers를 사용하여 HSM의 모든 사용자를 나열합니다.

출력은 사용자 3인 alice가 HSM에서 CO임을 보여줍니다.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:3

    User Id             User Type       User Name                      MofnPubKey
  LoginFailureCnt         2FA
        1               PCO             admin                              YES
          0               NO
        2               AU              app_user                            NO
          0               NO
```

```
         3              CO             alice                                  NO
           0               NO
Users on server 1(10.0.0.2):
Number of users found:3

    User Id             User Type       User Name                         MofnPubKey
  LoginFailureCnt         2FA
         1              PCO            admin                                  YES
           0               NO
         2              AU             app_user                               NO
           0               NO
         3              CO             alice                                  NO
           0               NO

Users on server 1(10.0.0.3):
Number of users found:3

    User Id             User Type       User Name                         MofnPubKey
  LoginFailureCnt         2FA
         1              PCO            admin                                  YES
           0               NO
         2              AU             app_user                               NO
           0               NO
         3              CO             alice                                  NO
           0               NO
```

두 번째 명령은 deleteUser 명령을 사용하여 HSM에서 `alice`를 삭제합니다.

출력은 클러스터의 3개 HSM 모두에서 명령이 성공했음을 보여줍니다.

```
aws-cloudhsm> deleteUser CO alice
Deleting user alice(CO) on 3 nodes
deleteUser success on server 0(10.0.0.1)
deleteUser success on server 0(10.0.0.2)
deleteUser success on server 0(10.0.0.3)
```

마지막 명령은 listUsers 명령을 사용하여 `alice`가 클러스터의 3개 HSM 모두에서 삭제되었음을 확인합니다.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:2
```

```
    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                               YES
           0               NO
         2              AU              app_user                             NO
           0               NO
Users on server 1(10.0.0.2):
Number of users found:2

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                               YES
           0               NO
         2              AU              app_user                             NO
           0               NO
Users on server 1(10.0.0.3):
Number of users found:2

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
         1              PCO             admin                               YES
           0               NO
         2              AU              app_user                             NO
           0               NO
```

인수

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
deleteUser <user-type> <user-name>
```

<user-type>

사용자 유형을 지정합니다. 이 파라미터는 필수 사항입니다.

> **Tip**
>
> 키를 소유한 CU(Crypto User)는 삭제할 수 없습니다.

유효한 값은 CO, CU입니다.

사용자 유형을 가져오려면 listUsers를 사용합니다. HSM에서의 사용자 유형에 대한 자세한 내용은 HSM 사용자 이해 섹션을 참조하십시오.

필수 여부: 예

<user-name>

사용자의 친숙한 이름을 지정합니다. 최대 길이는 31자입니다. 허용되는 유일한 특수 문자는 밑줄(_)입니다.

사용자를 생성한 후에는 사용자 이름을 변경할 수 없습니다. cloudhsm_mgmt_util 명령에서 사용자 유형과 암호는 대소문자를 구분하지만 사용자 이름은 대소문자를 구분하지 않습니다.

필수: 예

관련 주제

- listUsers
- createUser
- syncUser
- changePswd

findAllKeys

cloudhsm_mgmt_util의 findAllKeys 명령은 지정된 암호화 사용자(crypto user)가 소유하거나 공유하는 키를 가져옵니다. 또한 각각의 HSM에 있는 사용자 데이터의 해시를 반환합니다. 해시를 사용하면 사용자, 키 소유권, 키 공유 데이터가 클러스터의 모든 HSM에서 동일한지 한눈에 확인할 수 있습니다. 출력에서, 사용자가 소유한 키는 (o)에 의해 주석이 첨부되고 공유 키는 (s)에 의해 주석이 첨부됩니다.

HSM의 모든 CU는 어떤 퍼블릭 키도 사용할 수 있지만, findAllKeys는 지정된 CU가 키를 소유할 때만 퍼블릭 키를 반환합니다. 이 동작은 모든 CU 사용자에 대해 퍼블릭 키를 반환하는 key_mgmt_util의 FindKey와 다릅니다.

Crypto Officer(CO 및 PCO)와 AU(어플라이언스 사용자)만 이 명령을 실행할 수 있습니다. CU(Crypto User)는 다음 명령을 실행할 수 있습니다.

- listUsers - 모든 사용자 찾기
- 사용할 수 있는 키를 찾을 수 있는 key_mgmt_util의 findKey

- getKeyInfokey_mgmt_util에서 소유하거나 공유하는 특정 키의 소유자 및 공유 사용자를 찾을 수 있습니다.

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- crypto officer(CO, PCO)
- 어플라이언스 사용자(AU)

구문

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

예제

이러한 예제는 `findAllKeys`를 사용하여 사용자의 모든 키를 찾는 방법과 각 HSM에서 키 사용자 정보 해시를 가져오는 방법을 보여 줍니다.

Example : CU의 키 찾기

이 예제는 findAllKeys를 사용하여 사용자 4가 소유하고 공유하는 HSM의 키를 찾습니다. 이 명령은 두 번째 인수에 `0` 값을 사용하여 해시 값을 억제합니다. 이 명령은 선택적 파일 이름을 생략하기 때문에 stdout에 씁니다(표준 출력).

출력은 사용자 4가 8, 9, 17, 262162, 19, 31 등 여섯 개의 키를 사용할 수 있음을 보여 줍니다. 출력은 `(s)`를 사용하여 사용자가 명시적으로 공유하는 키를 나타냅니다. 사용자가 소유한 키는 `(o)`으로 표시되며 사용자가 공유하지 않는 대칭 및 프라이빗 키와 모든 crypto user가 사용할 수 있는 퍼블릭 키를 포함합니다.

```
aws-cloudhsm> findAllKeys 4 0
```

```
Keys on server 0(10.0.0.1):
Number of keys found 6
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 0(10.0.0.1)

Keys on server 1(10.0.0.2):
Number of keys found 6
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 1(10.0.0.2)

Keys on server 1(10.0.0.3):
Number of keys found 6
number of keys matched from start index 0::6
8(s),9(s),17,262162(s),19(o),31(o)
findAllKeys success on server 1(10.0.0.3)
```

Example : 사용자 데이터가 동기화되어 있는지 확인하기

이 예제는 findAllKeys를 사용하여 클러스터의 모든 HSM에 동일한 사용자, 키 소유권, 키 공유 값이 포함되어 있다는 것을 확인합니다. 이를 위해 각 HSM에서 키 사용자 데이터 해시를 가져와 해시 값을 비교합니다.

이 명령은 키 해시를 가져오기 위해 두 번째 인수에서 1 값을 사용합니다. 선택적 파일 이름이 생략되기 때문에 이 명령은 키 해시를 stdout에 씁니다.

예제는 사용자 6을 지정하지만 HSM에 있는 키를 소유하거나 공유하는 모든 사용자에게 해시 값은 동일합니다. 지정된 사용자가 CO와 같이 키를 소유하지 않거나 공유하지 않는 경우, 명령은 해시 값을 반환하지 않습니다.

출력은 클러스터의 두 HSM 모두에서 키 해시가 동일함을 보여 줍니다. HSM 중 하나에 다른 사용자, 다른 키 소유자 또는 다른 공유 사용자가 있는 경우, 키 값이 동일하지 않습니다.

```
aws-cloudhsm> findAllKeys 6 1
Keys on server 0(10.0.0.1):
Number of keys found 3
number of keys matched from start index 0::3
8(s),9(s),11,17(s)
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 0(10.0.0.1)
```

```
Keys on server 1(10.0.0.2):
Number of keys found 3
number of keys matched from start index 0::3
8(s),9(s),11(o),17(s)
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 1(10.0.0.2)
```

이 명령은 해시 값이 HSM의 모든 키에 대한 사용자 데이터를 나타냄을 보여 줍니다. 이 명령은 사용자 3에 대해 findAllKeys를 사용합니다. 3개의 키만 소유하거나 공유하는 사용자 6과는 달리 사용자 3은 17개의 키를 소유 또는 공유하지만 키 해시 값은 동일합니다.

```
aws-cloudhsm> findAllKeys 3 1
Keys on server 0(10.0.0.1):
Number of keys found 17
number of keys matched from start index 0::17
6(o),7(o),8(s),11(o),12(o),14(o),262159(o),262160(o),17(s),262162(s),19(s),20(o),21(o),262177(o
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 0(10.0.0.1)
Keys on server 1(10.0.0.2):
Number of keys found 17
number of keys matched from start index 0::17
6(o),7(o),8(s),11(o),12(o),14(o),262159(o),262160(o),17(s),262162(s),19(s),20(o),21(o),262177(o
Key Hash:
55655676c95547fd4e82189a072ee1100eccfca6f10509077a0d6936a976bd49

findAllKeys success on server 1(10.0.0.2)
```

인수

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

<user id>

지정된 사용자가 소유하거나 공유하는 모든 키를 가져옵니다. HSM에 있는 사용자의 사용자 ID를 입력합니다. 모든 사용자의 사용자 ID를 찾으려면 listUsers를 사용하십시오.

모든 사용자 ID가 유효하지만 `findAllKeys`는 CU(Crypto User)에 대해서만 키를 반환합니다.

필수 여부: 예

<key hash>

각 HSM의 모든 키에 대한 사용자 소유권 및 공유 데이터의 해시를 포함(1)하거나 제외(0)합니다.

`user id` 인수가 키를 소유 또는 공유하는 사용자를 나타내는 경우, 키 해시가 채워집니다. HSM에서 키를 소유하거나 공유하는 모든 사용자의 경우, 서로 다른 키를 소유하고 공유하더라도 키 해시 값은 동일합니다. 하지만 `user id`가 CO와 같이 아무 키도 소유 또는 공유하지 않는 사용자를 나타내는 경우, 해시 값이 채워지지 않습니다.

필수 여부: 예

<output file>

지정된 파일에 출력을 기록합니다.

필수 여부: 아니요

기본값: Stdout

관련 주제

- changePswd
- deleteUser
- listUsers
- syncUser
- key_mgmt_util의 findKey
- getKeyInfokey_mgmt_util에서

getAttribute

cloudhsm_mgmt_util의 getAttribute 명령은 클러스터의 모든 HSM에서 키에 대한 하나의 속성 값을 가져와서 stdout(표준 출력) 또는 파일에 씁니다. CU(Crypto User)만 이 명령을 실행할 수 있습니다.

키 속성은 키의 특성입니다. 여기에는 키 유형, 클래스, 레이블, ID 같은 특성과 암호화, 복호화, 래핑, 서명, 확인 등 키에서 수행할 수 있는 작업을 나타내는 값이 포함됩니다.

사용자가 소유 및 공유하는 키에 대해서만 getAttribute를 사용할 수 있습니다. 이 명령이나 키의 속성 값 중 하나 또는 전부를 파일에 기록하는 key_mgmt_util에서 getAttribute 명령을 실행할 수 있습니다.

속성과 각 속성을 나타내는 상수의 목록을 가져오려면 listAttributes 명령을 사용합니다. 기존 키의 속성 값을 변경하려면 key_mgmt_util에서 setAttribute를 사용하고 cloudhsm_mgmt_util에서 setAttribute를 사용합니다. 키 속성 해석에 대한 도움말은 키 속성 참조을 참조하십시오.

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 암호화 사용자(Crypto User)

구문

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
getAttribute <key handle> <attribute id> [<filename>]
```

예

이 예제는 HSM의 키에서 추출 가능 속성의 값을 가져옵니다. 이러한 명령을 사용하여 HSM에서 키를 내보낼 수 있는지 여부를 확인할 수 있습니다.

첫 번째 명령은 listAttributes를 사용하여 추출 가능 속성을 나타내는 상수를 찾습니다. 출력은 OBJ_ATTR_EXTRACTABLE 상수가 354임을 보여줍니다. 또한 키 속성 참조의 속성 및 해당 값에 대한 설명을 사용하여 이 정보를 찾을 수 있습니다.

```
aws-cloudhsm> listAttributes

Following are the possible attribute values for getAttribute:

      OBJ_ATTR_CLASS                  = 0
```

```
    OBJ_ATTR_TOKEN                  = 1
    OBJ_ATTR_PRIVATE                = 2
    OBJ_ATTR_LABEL                  = 3
    OBJ_ATTR_TRUSTED                = 134
    OBJ_ATTR_KEY_TYPE               = 256
    OBJ_ATTR_ID                     = 258
    OBJ_ATTR_SENSITIVE              = 259
    OBJ_ATTR_ENCRYPT                = 260
    OBJ_ATTR_DECRYPT                = 261
    OBJ_ATTR_WRAP                   = 262
    OBJ_ATTR_UNWRAP                 = 263
    OBJ_ATTR_SIGN                   = 264
    OBJ_ATTR_VERIFY                 = 266
    OBJ_ATTR_DERIVE                 = 268
    OBJ_ATTR_LOCAL                  = 355
    OBJ_ATTR_MODULUS                = 288
    OBJ_ATTR_MODULUS_BITS           = 289
    OBJ_ATTR_PUBLIC_EXPONENT        = 290
    OBJ_ATTR_VALUE_LEN              = 353
    OBJ_ATTR_EXTRACTABLE            = 354
    OBJ_ATTR_NEVER_EXTRACTABLE      = 356
    OBJ_ATTR_ALWAYS_SENSITIVE       = 357
    OBJ_ATTR_DESTROYABLE            = 370
    OBJ_ATTR_KCV                    = 371
    OBJ_ATTR_WRAP_WITH_TRUSTED      = 528
    OBJ_ATTR_WRAP_TEMPLATE          = 1073742353
    OBJ_ATTR_UNWRAP_TEMPLATE        = 1073742354
    OBJ_ATTR_ALL                    = 512
```

두 번째 명령은 getAttribute를 사용하여 HSM에서 키 핸들이 262170인 키의 추출 가능 속성 값을 가져옵니다. 추출 가능 속성을 지정하기 위해 이 명령은 해당 속성을 나타내는 상수인 354를 사용합니다. 이 명령은 파일 이름을 지정하지 않기 때문에 getAttribute는 출력을 stdout에 기록합니다.

출력은 모든 HSM에서 추출 가능 속성의 값이 1임을 보여 줍니다. 이 값은 키의 소유자가 키를 내보낼 수 있음을 나타냅니다. 값이 0(0x0)이라면 키를 HSM에서 내보낼 수 없습니다. 키를 생성할 때 추출 가능 속성의 값을 설정하지만 값을 변경할 수는 없습니다.

```
aws-cloudhsm> getAttribute 262170 354

Attribute Value on server 0(10.0.1.10):
OBJ_ATTR_EXTRACTABLE
0x00000001
```

```
Attribute Value on server 1(10.0.1.12):
OBJ_ATTR_EXTRACTABLE
0x00000001

Attribute Value on server 2(10.0.1.7):
OBJ_ATTR_EXTRACTABLE
0x00000001
```

인수

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
getAttribute <key handle> <attribute id> [<filename>]
```

<key-handle>

대상 키의 키 핸들을 지정합니다. 각 명령에서 키를 하나만 지정할 수 있습니다. 키의 키 핸들을 가져오려면 key_mgmt_util에서 findKey를 사용합니다.

사용자가 지정된 키를 소유 또는 공유해야 합니다. 키의 사용자를 찾으려면 getKeyInfokey_mgmt_util에서 사용하십시오.

필수 여부: 예

<attribute id>

속성을 식별합니다. 모든 속성을 나타내는 상수, 즉 모든 속성을 나타내는 512를 입력합니다. 예를 들어 키 유형을 가져오려면 OBJ_ATTR_KEY_TYPE 속성의 상수인 256을 입력합니다.

속성과 해당 상수를 나열하려면 listAttributes를 사용합니다. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

필수 여부: 예

<filename>

지정된 파일에 출력을 기록합니다. 파일 경로를 입력합니다.

지정된 파일이 존재하면 getAttribute가 경고 없이 파일을 덮어씁니다.

필수 여부: 아니요

기본값: Stdout

관련 주제

- key_mgmt_util의 getAttribute
- listAttributes
- cloudhsm_mgmt_util의 setAttribute
- key_mgmt_util의 setAttribute
- 키 속성 참조

getCert

cloudhsm_mgmt_util의 getCert 명령을 사용하여 클러스터에서 특정 HSM의 인증서를 검색할 수 있습니다. 명령을 실행할 때 검색할 인증서의 유형을 지정합니다. 이렇게 하려면 아래 인수 단원의 설명과 같이 해당 정수 중 하나를 사용합니다. 이 인증서의 각 역할을 알아보려면 HSM 자격 증명 확인 섹션을 참조하십시오.

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 모든 사용자.

필수 조건

시작하기 전에 대상 HSM에서 서버 모드에 들어가야 합니다. 자세한 내용은 서버를 참조하십시오.

구문

서버 모드에 들어간 후 getCert 명령을 사용하려면

```
server> getCert <file-name> <certificate-type>
```

예

먼저 서버 모드에 들어갑니다. 이 명령은 서버 번호가 0인 HSM에서 서버 모드에 들어갑니다.

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

그런 다음 getCert 명령을 사용합니다. 이 예제에서는 /tmp/P0.crt를 인증서가 저장된 파일의 이름으로 사용하고 4(고객 루트 인증서)를 원하는 인증서 유형으로 사용합니다.

```
server0> getCert /tmp/P0.crt 4
getCert Success
```

인수

```
getCert <file-name> <certificate-type>
```

<file-name>

인증서가 저장되는 파일의 이름을 지정합니다.

필수 여부: 예

<certificate-type>

검색할 인증서의 유형을 지정하는 정수입니다. 정수 및 해당 인증서 유형은 다음과 같습니다.

- 1 – 제조업체 루트 인증서
- 2 – 제조업체 하드웨어 인증서
- 4 – 고객 루트 인증서
- 8 – 클러스터 인증서(고객 루트 인증서로 서명됨)
- 16 – 클러스터 인증서(제조업체 루트 인증서에 묶임)

필수: 예

관련 주제

- 서버

getHSMInfo

cloudhsm_mgmt_util의 getHSMInfo 명령은 모델, 일련 번호, FIPS 상태, 메모리, 온도, 하드웨어 및 펌웨어 버전 번호 등 각 HSM이 실행되는 하드웨어에 대한 정보를 가져옵니다. 정보에는 cloudhsm_mgmt_util이 HSM을 참조하기 위해 사용하는 서버 ID도 포함한다.

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- 모든 사용자. 이 명령을 실행하기 위해 로그인할 필요는 없습니다.

구문

이 명령에는 파라미터가 없습니다.

```
getHSMInfo
```

예

이 예제는 getHSMInfo를 사용하여 클러스터의 HSM에 대한 정보를 가져옵니다.

```
aws-cloudhsm> getHSMInfo
Getting HSM Info on 3 nodes
                *** Server 0 HSM Info ***

        Label                :cavium
        Model                :NITROX-III CNN35XX-NFBE

        Serial Number        :3.0A0101-ICM000001
        HSM Flags            :0
        FIPS state           :2 [FIPS mode with single factor authentication]

        Manufacturer ID      :
        Device ID            :10
        Class Code           :100000
```

```
        System vendor ID     :177D
        SubSystem ID         :10


        TotalPublicMemory    :560596
        FreePublicMemory     :294568
        TotalPrivateMemory   :0
        FreePrivateMemory    :0

        Hardware Major       :3
        Hardware Minor       :0

        Firmware Major       :2
        Firmware Minor       :03

        Temperature          :56 C

        Build Number         :13

        Firmware ID          :xxxxxxxxxxxxxx

...
```

관련 주제

- info

getKeyInfo

key_mgmt_util의 getKeyInfo 명령은 키를 공유하는 소유자 및 암호화 사용자(Crypto User)를 포함하여 키를 사용할 수 있는 사용자의 HSM 사용자 ID를 반환합니다. 키에서 쿼럼 인증이 활성화되면 getKeyInfo는 키를 사용하는 암호화 작업을 승인해야 하는 사용자의 수도 반환합니다. getKeyInfo는 소유한 키 및 공유된 키에서만 실행할 수 있습니다.

퍼블릭 키에서 getKeyInfo를 실행하면 HSM의 모든 사용자가 퍼블릭 키를 사용할 수 있더라도 getKeyInfo는 키 소유자만 반환합니다. HSM에 있는 사용자의 HSM 사용자 ID를 찾으려면 listUsers를 사용하십시오. 특정 사용자의 키를 찾으려면 key_mgmt_util의 findKey -u를 사용합니다. 암호화폐 담당자는 findAllKeyscloudhsm_mgmt_util에서 사용할 수 있습니다.

생성하는 키는 본인의 소유입니다. 키를 생성하면 다른 사용자와 공유할 수 있습니다. 그런 다음 기존 키를 공유하거나 공유 해제하려면 cloudhsm_mgmt_util에서 shareKey를 사용합니다.

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- 암호화 사용자(Crypto User)

구문

```
getKeyInfo -k <key-handle> [<output file>]
```

예제

이러한 예제에서는 getKeyInfo를 사용하여 키의 사용자에 대한 정보를 가져오는 방법을 보여 줍니다.

Example : 비대칭 키의 사용자 가져오기

이 명령은 키 핸들이 262162인 AES(비대칭) 키를 사용할 수 있는 사용자를 가져옵니다. 출력은 사용자 3이 키를 소유하고 사용자 4 및 6과 공유함을 보여 줍니다.

사용자 3, 4, 6만이 키 262162에서 getKeyInfo를 실행할 수 있습니다.

```
aws-cloudhsm>getKeyInfo 262162
Key Info on server 0(10.0.0.1):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 2 user(s):

                4
                6
Key Info on server 1(10.0.0.2):

        Token/Flash Key,
```

```
        Owned by user 3

        also, shared to following 2 user(s):

                4
                6
```

Example : 대칭 키 쌍의 사용자 가져오기

이러한 명령은 getKeyInfo를 사용하여 ECC(대칭) 키 페어에서 키를 사용할 수 있는 사용자를 가져옵니다. 퍼블릭 키에는 키 핸들 262179이 있습니다. 프라이빗 키에는 키 핸들 262177이 있습니다.

프라이빗 키(262177)에서 getKeyInfo를 실행하면 키 소유자(3)와 키가 공유되는 CU(Crypto User) 4가 반환됩니다.

```
aws-cloudhsm>getKeyInfo -k 262177
Key Info on server 0(10.0.0.1):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
Key Info on server 1(10.0.0.2):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
```

퍼블릭 키(262179)에서 getKeyInfo를 실행하면 키 소유자인 사용자 3만 반환됩니다.

```
aws-cloudhsm>getKeyInfo -k 262179
Key Info on server 0(10.0.3.10):

        Token/Flash Key,

        Owned by user 3
```

```
Key Info on server 1(10.0.3.6):

        Token/Flash Key,

        Owned by user 3
```

사용자 4가 퍼블릭 키(및 HSM의 모든 퍼블릭 키)를 사용할 수 있는지 확인하려면 -u에서 findKey의 파라미터를 사용하십시오.

출력은 사용자 4가 키 페어의 퍼블릭 키(262179)와 프라이빗 키(262177)를 모두 사용할 수 있음을 보여 줍니다. 사용자 4는 다른 모든 퍼블릭 키 및 생성되거나 공유된 어떤 프라이빗 키도 사용할 수 있습니다.

```
Command:  findKey -u 4

Total number of keys present 8

 number of keys matched from start index 0::7
11, 12, 262159, 262161, 262162, 19, 20, 21, 262177, 262179

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example : 키의 쿼럼 인증 값(m_value) 가져오기

이 예제는 키의 m_value를 가져오는 방법을 보여 줍니다. m_value는 키를 사용하는 암호화 작업과 키를 공유 또는 공유 해제하는 작업을 승인해야 하는 쿼럼 내 사용자의 수입니다.

키에서 쿼럼 인증이 활성화되면 사용자들의 쿼럼은 해당 키를 사용하는 모든 암호화 작업을 승인해야 합니다. 쿼럼 인증을 활성화하고 쿼럼 크기를 설정하려면 키를 생성할 때 -m_value 파라미터를 사용합니다.

이 명령은 사용자 4와 공유하는 256비트 genSymKeyAES 키를 생성하는 데 사용됩니다. 이 명령은 m_value 파라미터를 사용하여 쿼럼 인증을 활성화하고 쿼럼 크기를 2명의 사용자로 설정합니다. 사용자의 수는 필요한 승인을 제공할 수 있을 만큼 커야 합니다.

출력은 명령이 키 10을 생성했음을 보여 줍니다.

```
Command:  genSymKey -t 31 -s 32 -l aes256m2 -u 4 -m_value 2
```

```
        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 10

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

이 명령은 cloudhsm_mgmt_util의 getKeyInfo를 사용하여 키 10의 사용자에 대한 정보를 가져옵니다. 출력은 이 키를 사용자 3이 소유하고 사용자 4와 공유하고 있음을 보여 줍니다. 또한 2명의 사용자로 이루어진 쿼럼이 해당 키를 사용하는 모든 암호화 작업을 승인해야 함을 보여 줍니다.

```
aws-cloudhsm>getKeyInfo 10

Key Info on server 0(10.0.0.1):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
         2 Users need to approve to use/manage this key
Key Info on server 1(10.0.0.2):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
         2 Users need to approve to use/manage this key
```

인수

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
getKeyInfo -k <key-handle> <output file>
```

<key-handle>

HSM에서 한 키의 키 핸들을 지정합니다. 소유하거나 공유하는 키의 키 핸들을 입력합니다. 이 파라미터는 필수 사항입니다.

필수 여부: 예

<output file>

stdout 대신 지정된 파일에 출력을 기록합니다. 파일이 존재하는 경우, 이 명령은 경고 없이 파일에 덮어씁니다.

필수 여부: 아니요

기본값: stdout

관련 주제

- getKeyInfokey_mgmt_util에서
- key_mgmt_util의 findKey
- findAllKayscloudhsm_mgmt_util에서
- listUsers
- shareKey

info

cloudhsm_mgmt_util의 info 명령은 호스트 이름, 포트, IP 주소, HSM에서 cloudhsm_mgmt_util로 로그인되어 있는 사용자의 이름과 유형을 포함하여 클러스터의 각 HSM에 대한 정보를 가져옵니다.

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- 모든 사용자. 이 명령을 실행하기 위해 로그인할 필요는 없습니다.

구문

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
info server <server ID>
```

예

이 예제는 info를 사용하여 클러스터의 HSM에 대한 정보를 가져옵니다. 이 명령은 클러스터의 첫 번째 HSM을 가리키는 데 0을 사용합니다. 출력은 IP 주소, 포트 및 현재 사용자의 유형과 이름을 보여 줍니다.

```
aws-cloudhsm> info server 0
Id      Name                    Hostname          Port    State           Partition
         LoginState
0       10.0.0.1                10.0.0.1         2225    Connected       hsm-udw0tkfg1ab
         Logged in as 'testuser(CU)'
```

인수

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
info server <server ID>
```

<server id>

HSM의 서버 ID를 지정합니다. HSM에는 0부터 시작해 클러스터에 추가되는 순서를 나타내는 서수가 할당됩니다. HSM의 서버 ID를 찾으려면 getHSMInfo를 사용하십시오.

필수: 예

관련 주제

- getHSMInfo
- loginHSM 및 logoutHSM

listAttributes

listAttributescloudhsm_mgmt_util의 명령은 키의 속성과 이를 나타내는 상수를 나열합니다. AWS CloudHSM 이러한 상수를 사용하여 getAttribute 및 setAttribute 명령에서 속성을 식별합니다.

키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 암호화 사용자(Crypto User)로 HSM에 로그인해야 합니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 모든 사용자. 이 명령을 실행하기 위해 로그인할 필요는 없습니다.

구문

```
listAttributes [-h]
```

예

이 명령은 key_mgmt_util에서 가져오고 변경할 수 있는 키 속성과 속성을 나타내는 상수를 나열합니다. 키 속성 해석에 대한 도움말은 키 속성 참조을 참조하십시오. 모든 속성을 나타내려면 512을 사용합니다.

```
Command: listAttributes

    Description
===========
The following are all of the possible attribute values for getAttribute.

      OBJ_ATTR_CLASS                = 0
      OBJ_ATTR_TOKEN                = 1
      OBJ_ATTR_PRIVATE              = 2
      OBJ_ATTR_LABEL                = 3
      OBJ_ATTR_TRUSTED              = 134
      OBJ_ATTR_KEY_TYPE             = 256
      OBJ_ATTR_ID                   = 258
      OBJ_ATTR_SENSITIVE            = 259
      OBJ_ATTR_ENCRYPT              = 260
```

```
    OBJ_ATTR_DECRYPT                = 261
    OBJ_ATTR_WRAP                   = 262
    OBJ_ATTR_UNWRAP                 = 263
    OBJ_ATTR_SIGN                   = 264
    OBJ_ATTR_VERIFY                 = 266
    OBJ_ATTR_DERIVE                 = 268
    OBJ_ATTR_LOCAL                  = 355
    OBJ_ATTR_MODULUS                = 288
    OBJ_ATTR_MODULUS_BITS           = 289
    OBJ_ATTR_PUBLIC_EXPONENT        = 290
    OBJ_ATTR_VALUE_LEN              = 353
    OBJ_ATTR_EXTRACTABLE            = 354
    OBJ_ATTR_NEVER_EXTRACTABLE      = 356
    OBJ_ATTR_ALWAYS_SENSITIVE       = 357
    OBJ_ATTR_DESTROYABLE            = 370
    OBJ_ATTR_KCV                     = 371
    OBJ_ATTR_WRAP_WITH_TRUSTED       = 528
    OBJ_ATTR_WRAP_TEMPLATE           = 1073742353
    OBJ_ATTR_UNWRAP_TEMPLATE        = 1073742354
    OBJ_ATTR_ALL                    = 512
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수: 예

관련 주제

- getAttribute
- setAttribute
- 키 속성 참조

listUsers

cloudhsm_mgmt_util에서 listUsers 명령은 각 HSM의 사용자와 함께 사용자 유형 및 기타 속성을 가져옵니다. 모든 사용자 유형이 이 명령을 실행할 수 있습니다. 이 명령을 실행하기 위해 cloudhsm_mgmt_util에 로그인할 필요도 없습니다.

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- 모든 사용자. 이 명령을 실행하기 위해 로그인할 필요는 없습니다.

구문

이 명령에는 파라미터가 없습니다.

```
listUsers
```

예

이 명령은 클러스터에 있는 각 HSM에서 사용자를 나열하고 그 속성을 표시합니다. `User ID` 속성을 사용하여 deleteUser, changePswd, findAllKeys 같은 다른 명령에서 사용자를 식별할 수 있습니다.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:6

    User Id             User Type       User Name            MofnPubKey
 LoginFailureCnt         2FA
         1              PCO             admin                   YES              0
             NO
         2              AU              app_user                 NO              0
             NO
         3              CU              crypto_user1             NO              0
             NO
         4              CU              crypto_user2             NO              0
             NO
         5              CO              officer1                YES              0
             NO
         6              CO              officer2                 NO              0
             NO
```

```
Users on server 1(10.0.0.2):
Number of users found:5

   User Id             User Type       User Name            MofnPubKey
 LoginFailureCnt         2FA
        1              PCO             admin                   YES              0
            NO
        2              AU              app_user                 NO              0
            NO
        3              CU              crypto_user1             NO              0
            NO
        4              CU              crypto_user2             NO              0
            NO
        5              CO              officer1                YES              0
            NO
```

출력에는 다음의 사용자 속성이 포함됩니다.

- User ID: key_mgmt_util 및 cloudhsm_mgmt_util 명령에서 사용자를 식별합니다.
- User type: 사용자가 HSM에서 수행할 수 있는 작업을 결정합니다.
- User Name: 해당 사용자의 사용자 정의 표시 이름을 표시합니다.
- MofnPubKey: 사용자가 쿼럼 인증 토큰에 서명하기 위해 키 쌍을 등록했는지 여부를 나타냅니다.
- LoginFailureCnt: 사용자가 로그인에 실패한 횟수를 나타냅니다.
- 2FA: 사용자가 멀티 팩터 인증을 활성화했는지 여부를 나타냅니다.

관련 주제

- key_mgmt_util의 listUsers
- createUser
- deleteUser
- changePswd

loginHSM 및 logoutHSM

cloudhsm_mgmt_util의 loginHSM 및 logoutHSM 명령을 사용하여 클러스터의 각 HSM에 로그인/로그아웃합니다. 모든 유형의 사용자는 이러한 명령을 사용할 수 있습니다.

> **Note**
>
> 잘못된 로그인 시도 횟수가 5회를 초과하면 계정이 잠깁니다. 계정 잠금을 해제하려면 CO(암호화 관리자)가 cloudhsm_mgmt_util의 changePswd 명령을 사용하여 암호를 재설정해야 합니다.

loginHSM 및 logoutHSM 문제를 해결하려면

이러한 cloudhsm_mgmt_util 명령을 실행하기 전에 cloudhsm_mgmt_util을 시작해야 합니다.

HSM을 추가하거나 삭제하는 경우 AWS CloudHSM 클라이언트와 명령줄 도구가 사용하는 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

클러스터에 HSM이 둘 이상인 경우 계정이 잠기기 전에 잘못된 로그인 시도가 추가로 허용될 수 있습니다. 이는 CloudHSM 클라이언트가 다양한 HSM 간에 로드 균형을 조정하기 때문입니다. 따라서 매번 동일한 HSM에서 로그인 시도가 시작되지 않을 수 있습니다. 이 기능을 테스트하는 경우 활성화된 HSM이 하나 뿐인 클러스터에서 수행하는 것이 좋습니다.

2018년 2월 이전에 클러스터를 만든 경우 로그인 시도가 20회 실패한 후에 계정이 잠깁니다.

사용자 유형

다음 사용자가 이러한 명령을 실행할 수 있습니다.

- PRESCO(Precrypto Officer)
- CO(Crypto Officer)
- CU(Crypto User)

구문

구문 다이어그램에 지정된 순서대로 인수를 입력합니다. -hpswd 파라미터를 사용하여 암호를 숨길 수 있습니다. 2팩터 인증(2FA)으로 로그인하려면 -2fa 파라미터를 사용하고 파일 경로를 포함하십시오. 자세한 내용은 the section called “인수” 섹션을 참조하십시오.

```
loginHSM <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

```
logoutHSM
```

예제

이러한 예제에서는 loginHSM 및 logoutHSM을 사용하여 클러스터의 모든 HSM에 로그인/로그아웃하는 방법을 보여줍니다.

Example : 클러스터의 HSM에 로그인

이 명령은 자격 증명으로 CO 사용자 admin 및 암호 co12345를 사용하여 클러스터의 모든 HSM에 로그인합니다. 이 출력은 명령이 성공적으로 수행되었으며 사용자가 HSM(이 경우, server 0 및 server 1)에 연결되었음을 보여줍니다.

```
aws-cloudhsm>loginHSM CO admin co12345

loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)
```

Example : 숨겨진 암호로 로그인

이 명령은 시스템이 암호를 숨기도록 지정한다는 점을 제외하면 위의 예와 동일합니다.

```
aws-cloudhsm>loginHSM CO admin -hpswd
```

시스템이 암호를 묻는 메시지를 표시합니다. 암호를 입력하면 시스템은 암호를 숨기며 출력에는 명령이 성공했고 HSM에 연결되었다는 메시지가 표시됩니다.

```
Enter password:

loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)

aws-cloudhsm>
```

Example : HSM에서 로그아웃

이 명령은 현재 로그인되어 있는 HSM(이 경우, server 0 및 server 1)에서 로그아웃합니다. 이 출력은 명령이 성공적으로 수행되었고 사용자가 HSM에서 연결 해제되었음을 보여줍니다.

```
aws-cloudhsm>logoutHSM

logoutHSM success on server 0(10.0.2.9)
```

```
logoutHSM success on server 1(10.0.3.11)
```

인수

구문 다이어그램에 지정된 순서대로 인수를 입력합니다. -hpswd 파라미터를 사용하여 암호를 숨길 수 있습니다. 2팩터 인증(2FA)으로 로그인하려면 -2fa 파라미터를 사용하고 파일 경로를 포함하십시오. 2FA 작업에 대한 자세한 내용은 CMU를 사용하여 2FA 관리 섹션을 참조하세요.

```
loginHSM <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user type>

HSM에 로그인 중인 사용자의 유형을 지정합니다. 자세한 내용은 위의 사용자 유형을 참조하십시오.

필수 여부: 예

<user name>

HSM에 로그인 중인 사용자의 사용자 이름을 지정합니다.

필수 여부: 예

<password | -hpswd >

HSM에 로그인 중인 사용자의 암호를 지정합니다. 암호를 숨기려면 암호 대신 -hpswd 파라미터를 사용하고 프롬프트를 따르십시오.

필수 여부: 예

[-2fa </path/to/authdata>]

시스템이 두 번째 요소를 사용하여 이 2FA 지원 CO 사용자를 인증하도록 지정합니다. 2FA로 로그인하는 데 필요한 데이터를 가져오려면 -2fa 파라미터 뒤에 파일 이름이 있는 파일 시스템의 위치에 대한 경로를 포함시키십시오. 2FA 작업에 대한 자세한 내용은 CMU를 사용하여 2FA 관리 섹션을 참조하세요.

필수 여부: 아니요

관련 주제

- cloudhsm_mgmt_util 시작하기
- 클러스터 활성화

registerQuorumPub키

cloudhsm_mgmt_util의 registerQuorumPubKey 명령은 하드웨어 보안 모듈 (HSM) 사용자를 비대칭 RSA-2048 키 페어와 연결합니다. HSM 사용자를 키에 연결하면 해당 사용자가 프라이빗 키를 사용하여 쿼럼 요청을 승인하고 클러스터는 등록된 퍼블릭 키를 사용하여 사용자의 서명이 맞는지 확인할 수 있습니다. 쿼럼 인증에 대한 자세한 내용은 쿼럼 인증 관리(N의 M 액세스 제어)를 참조하십시오.

> **Tip**
>
> AWS CloudHSM 설명서에서는 쿼럼 인증을 M of N (MoFn) 이라고도 하는데, 이는 총 승인자 수 N명 중 최소 M명의 승인자를 의미합니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- Crypto officers(CO)

구문

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

예제

이 예제에서는 쿼럼 인증 요청에 대해 CO(Crypto Officer)를 승인자로 등록하는 데 registerQuorumPubKey를 사용하는 방법을 보여줍니다. 이 명령을 실행하려면 비대칭 RSA-2048 키 페어, 서명된 토큰 및 서명되지 않은 토큰이 있어야 합니다. 이러한 요구 사항에 대한 자세한 내용은 the section called “인수” 단원을 참조하십시오.

Example : 쿼럼 인증을 위한 HSM 사용자 등록

이 예에서는 `quorum_officer`이라는 CO를 쿼럼 인증 승인자로 등록합니다.

```
aws-cloudhsm> registerQuorumPubKey CO <quorum_officer> </path/to/quorum_officer.token>
 </path/to/quorum_officer.token.sig> </path/to/quorum_officer.pub>
```

```
************************CAUTION**********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
*****************************************************************

Do you want to continue(y/n)?y
registerQuorumPubKey success on server 0(10.0.0.1)
```

마지막 명령은 ListUsers 명령을 사용하여 quorum_officer가 MofN 사용자로 등록되었는지 확인합니다.

```
aws-cloudhsm> listUsers
Users on server 0(10.0.0.1):
Number of users found:3

    User Id             User Type       User Name                       MofnPubKey
  LoginFailureCnt         2FA
        1               PCO             admin                             NO
          0               NO
        2               AU              app_user                          NO
          0               NO
        3               CO              quorum_officer                    YES
          0               NO
```

인수

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

<user-type>

사용자 유형을 지정합니다. 이 파라미터는 필수 사항입니다.

HSM에서의 사용자 유형에 대한 자세한 내용은 HSM 사용자 이해 섹션을 참조하십시오.

유효값:

- CO: Crypto officer는 사용자를 관리할 수 있지만 키를 관리할 수는 없습니다.

필수: 예

<user-name>

사용자의 친숙한 이름을 지정합니다. 최대 길이는 31자입니다. 허용되는 유일한 특수 문자는 밑줄(_)입니다.

사용자를 생성한 후에는 사용자 이름을 변경할 수 없습니다. cloudhsm_mgmt_util 명령에서 사용자 유형과 암호는 대소문자를 구분하지만 사용자 이름은 대소문자를 구분하지 않습니다.

필수 여부: 예

<registration-token>

서명되지 않은 등록 토큰이 포함된 파일의 경로를 지정합니다. 최대 파일 크기가 245바이트인 임의의 데이터를 포함할 수 있습니다. 서명되지 않은 등록 토큰을 만드는 방법에 대한 자세한 내용은 등록 토큰 만들기 및 서명을 참조하십시오.

필수 여부: 예

<signed-registration-token>

등록 토큰의 SHA256_PKCS 메커니즘 서명 해시가 포함된 파일의 경로를 지정합니다. 자세한 내용은 등록 토큰 생성 및 서명을 참조하십시오.

필수 여부: 예

<public-key>

비대칭 RSA-2048 키 페어의 퍼블릭 키가 포함된 파일의 경로를 지정합니다. 프라이빗 키를 사용하여 등록 토큰에 서명합니다. 자세한 내용은 RSA 키 페어 생성을 참조하십시오.

필수 여부: 예

> **Note**
>
> 클러스터는 쿼럼 인증과 2단계 인증(2FA)에 동일한 키를 사용합니다. 즉, registerQuorumPubKey를 사용하여 2FA를 활성화한 사용자에 대해서는 쿼럼 키를 교체할 수 없습니다. 키를 교체하려면 changePswd를 사용해야 합니다. 쿼럼 인증 및 2FA 사용에 대한 자세한 내용은 쿼럼 인증 및 2FA를 참조하십시오.

관련 주제

- RSA 키 페어 생성

- 등록 토큰 생성 및 서명
- HSM에 퍼블릭 키 등록
- 쿼럼 인증 관리(N의 M 액세스 제어)
- 쿼럼 인증 및 2FA
- listUsers

서버

일반적으로 cloudhsm_mgmt_util에서 명령을 실행하면 해당 명령은 지정된 클러스터에 있는 모든 HSM에 영향을 미칩니다(글로벌 모드) 하지만 단일 HSM에 대해 명령을 실행해야 하는 상황이 있을 수 있습니다. 예를 들어, 자동 동기화에 실패하는 경우 클러스터 전체에서 일관성을 유지하기 위해 HSM에서 키와 사용자를 동기화해야 할 수 있습니다. cloudhsm_mgmt_util에서 server 명령을 사용하여 서버 모드에 들어간 다음 특정 HSM 인스턴스와 직접 상호 작용할 수 있습니다.

성공적으로 시작하면 `aws-cloudhsm>` 명령 프롬프트가 `server>` 명령 프롬프트로 바뀝니다.

서버 모드에서 나가려면 `exit` 명령을 사용합니다. 성공적으로 종료하면 cloudhsm_mgmt_util 명령 프롬프트로 돌아갑니다.

cloudhsm_mgmt_util 명령을 실행하기 전에 cloudhsm_mgmt_util을 시작해야 합니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 모든 사용자.

필수 조건

서버 모드에 들어가려면 먼저 대상 HSM의 서버 번호를 알아야 합니다. 서버 번호는 시작할 때 cloudhsm_mgmt_util에서 생성된 추적 출력에 나열됩니다. 서버 번호는 구성 파일에 HSM이 나타나는 것과 동일한 순서대로 할당됩니다. 이 예제의 경우 `server 0`은 원하는 HSM에 해당하는 서버라고 가정합니다.

구문

서버 모드를 시작하려면:

```
server <server-number>
```

서버 모드를 종료하려면:

```
server> exit
```

예

이 명령은 서버 번호가 0인 HSM에서 서버 모드에 들어갑니다.

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

서버 모드에서 나가려면 exit 명령을 사용합니다.

```
server0> exit
```

인수

```
server <server-number>
```

<server-number>

대상 HSM의 서버 번호를 지정합니다.

필수 여부: 예

exit 명령에 대한 인수는 없습니다.

관련 주제

- syncKey
- createUser
- deleteUser

setAttribute

cloudhsm_mgmt_util에서 setAttribute 명령은 HSM에 있는 키의 label, encrypt, decrypt, wrap 및 unwrap 속성 값을 변경합니다. key_mgmt_util에서 setAttribute 명령을 사용하여 세션 키를 영구 키로 변환할 수도 있습니다. 본인이 소유한 키의 속성만 변경할 수 있습니다.

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 암호화 사용자(Crypto User)

구문

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
setAttribute <key handle> <attribute id>
```

예

이 예제는 대칭 키의 암호 해독 기능을 비활성화하는 방법을 보여줍니다. 이와 같은 명령을 사용하여 래핑 키를 구성할 수 있습니다. 래핑 키는 다른 키를 래핑 및 언래핑할 수 있지만 데이터를 암호화하거나 암호화 해제할 수 없습니다.

첫 번째 단계는 래핑 키를 생성하는 것입니다. 이 명령은 inkey_mgmt_util을 사용하여 genSymKey256 비트 AES 대칭 키를 생성합니다. 출력은 새 키에 키 핸들 14가 있음을 보여 줍니다.

```
$  genSymKey -t 31 -s 32 -l aes256

Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 14

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

다음으로 decrypt 속성의 현재 값을 확인합니다. decrypt 속성의 속성 ID를 가져오려면 listAttributes를 사용합니다. 출력은 OBJ_ATTR_DECRYPT 속성을 나타내는 상수가 261임을 보여줍니다. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

```
aws-cloudhsm> listAttributes

Following are the possible attribute values for getAttribute:

      OBJ_ATTR_CLASS                  = 0
      OBJ_ATTR_TOKEN                  = 1
      OBJ_ATTR_PRIVATE                = 2
      OBJ_ATTR_LABEL                  = 3
      OBJ_ATTR_TRUSTED                = 134
      OBJ_ATTR_KEY_TYPE               = 256
      OBJ_ATTR_ID                     = 258
      OBJ_ATTR_SENSITIVE              = 259
      OBJ_ATTR_ENCRYPT                = 260
      OBJ_ATTR_DECRYPT                = 261
      OBJ_ATTR_WRAP                   = 262
      OBJ_ATTR_UNWRAP                 = 263
      OBJ_ATTR_SIGN                   = 264
      OBJ_ATTR_VERIFY                 = 266
      OBJ_ATTR_DERIVE                 = 268
      OBJ_ATTR_LOCAL                  = 355
      OBJ_ATTR_MODULUS                = 288
      OBJ_ATTR_MODULUS_BITS           = 289
      OBJ_ATTR_PUBLIC_EXPONENT        = 290
      OBJ_ATTR_VALUE_LEN              = 353
      OBJ_ATTR_EXTRACTABLE            = 354
      OBJ_ATTR_NEVER_EXTRACTABLE      = 356
      OBJ_ATTR_ALWAYS_SENSITIVE       = 357
      OBJ_ATTR_DESTROYABLE            = 370
      OBJ_ATTR_KCV                    = 371
      OBJ_ATTR_WRAP_WITH_TRUSTED      = 528
      OBJ_ATTR_WRAP_TEMPLATE          = 1073742353
      OBJ_ATTR_UNWRAP_TEMPLATE        = 1073742354
      OBJ_ATTR_ALL                    = 512
```

키 14에 대한 암호 해독 속성의 현재 값을 가져오기 위해 다음 명령은 cloudhsm_mgmt_util의 getAttribute를 사용합니다.

출력은 클러스터의 두 HSM 모두에서 decrypt 속성의 값이 true(1)임을 보여 줍니다.

```
aws-cloudhsm> getAttribute 14 261

Attribute Value on server 0(10.0.0.1):
```

```
OBJ_ATTR_DECRYPT
0x00000001

Attribute Value on server 1(10.0.0.2):
OBJ_ATTR_DECRYPT
0x00000001
```

이 명령은 setAttribute를 사용하여 키 14의 decrypt 속성(속성 261) 값을 0으로 변경합니다. 이렇게 하면 키에서 암호화 해제 기능이 비활성화됩니다.

출력은 클러스터의 두 HSM 모두에서 명령이 성공했음을 보여 줍니다.

```
aws-cloudhsm> setAttribute 14 261 0
************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)? y
setAttribute success on server 0(10.0.0.1)
setAttribute success on server 1(10.0.0.2)
```

마지막 명령은 getAttribute 명령을 반복합니다. 이 경우에도 명령은 키 14의 decrypt 속성(속성 261)을 가져옵니다.

이번 출력은 클러스터에 있는 두 HSM 모두에서 decrypt 속성의 값이 false(0)임을 보여 줍니다.

```
aws-cloudhsm>getAttribute 14 261
Attribute Value on server 0(10.0.3.6):
OBJ_ATTR_DECRYPT
0x00000000

Attribute Value on server 1(10.0.1.7):
OBJ_ATTR_DECRYPT
0x00000000
```

인수

```
setAttribute <key handle> <attribute id>
```

<key-handle>

소유하는 키의 키 핸들을 지정합니다. 각 명령에서 키를 하나만 지정할 수 있습니다. 키의 키 핸들을 가져오려면 key_mgmt_util에서 findKey를 사용합니다. 키 사용자를 찾으려면 를 getKeyInfo사용하십시오.

필수 여부: 예

<attribute id>

변경하려는 속성을 나타내는 상수를 지정합니다. 각 명령에서 하나의 속성만 지정할 수 있습니다. 속성과 해당 정수 값을 가져오려면 listAttributes를 사용합니다. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

유효한 값:

- 3 – OBJ_ATTR_LABEL.
- 134 – OBJ_ATTR_TRUSTED.
- 260 – OBJ_ATTR_ENCRYPT.
- 261 – OBJ_ATTR_DECRYPT.
- 262 – OBJ_ATTR_WRAP.
- 263 – OBJ_ATTR_UNWRAP.
- 264 – OBJ_ATTR_SIGN.
- 266 – OBJ_ATTR_VERIFY.
- 268 – OBJ_ATTR_DERIVE.
- 370 – OBJ_ATTR_DESTROYABLE.
- 528 – OBJ_ATTR_WRAP_WITH_TRUSTED.
- 1073742353 – OBJ_ATTR_WRAP_TEMPLATE.
- 1073742354 – OBJ_ATTR_UNWRAP_TEMPLATE.

필수: 예

관련 주제

- key_mgmt_util의 setAttribute
- getAttribute
- listAttributes

- 키 속성 참조

종료

cloudhsm_mgmt_util의 quit 명령은 cloudhsm_mgmt_util을 종료합니다. 모든 유형의 사용자가 이 명령을 사용할 수 있습니다.

cloudhsm_mgmt_util 명령을 실행하기 전에 cloudhsm_mgmt_util을 시작해야 합니다.

사용자 유형

다음 사용자가 이 명령을 실행할 수 있습니다.

- 모든 사용자. 이 명령을 실행하기 위해 로그인할 필요는 없습니다.

구문

```
quit
```

예

이 명령은 cloudhsm_mgmt_util을 종료합니다. 성공적으로 완료되면 일반 명령줄로 돌아갑니다. 이 명령에는 출력 파라미터가 없습니다.

```
aws-cloudhsm> quit

disconnecting from servers, please wait...
```

관련 주제

- cloudhsm_mgmt_util 시작하기

shareKey

cloudhsm_mgmt_util의 shareKey 명령은 소유하는 키를 다른 암호화 사용자와 공유 및 공유 해제합니다. 키 소유자만 키를 공유 및 공유 해제할 수 있습니다. 키를 생성할 때 키를 공유할 수도 있습니다.

다른 사용자의 키를 공유하는 사용자는 해당 키를 암호화 작업에 사용할 수는 있지만 키에 대해 삭제, 내보내기, 공유, 공유 해제 또는 속성 변경을 할 수는 없습니다. 키에서 쿼럼 인증이 활성화되면 쿼럼이 해당 키를 공유 또는 공유 해제하는 모든 작업을 승인해야 합니다.

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- 암호화 사용자(Crypto User)

구문

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

사용자 유형: CU(Crypto User)

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

예

다음 예제는 shareKey를 사용하여 소유하는 키를 다른 Crypto User와 공유 및 공유 해제하는 방법을 보여줍니다.

Example : 키 공유하기

이 예제에서는 shareKey를 사용하여 현재 사용자가 소유하는 ECC 프라이빗 키를 HSM의 다른 Crypto User와 공유합니다. 퍼블릭 키는 HSM의 모든 사용자가 사용할 수 있으므로, 공유 또는 공유 해제할 필요가 없습니다.

첫 번째 명령은 HSM의 ECC 개인 262177 키인 키에 대한 사용자 정보를 가져오는 getKeyInfo데 사용합니다.

출력은 키 262177을 사용자 3이 소유하지만 공유되지 않고 있음을 보여줍니다.

```
aws-cloudhsm>getKeyInfo 262177

Key Info on server 0(10.0.3.10):

        Token/Flash Key,
```

```
        Owned by user 3

Key Info on server 1(10.0.3.6):

        Token/Flash Key,

        Owned by user 3
```

이 명령은 shareKey를 사용하여 키 262177을 HSM의 다른 Crypto User인 사용자 4와 공유합니다. 마지막 인수는 값 1을 사용하여 공유 작업을 표시합니다.

출력은 클러스터의 두 HSM 모두에서 작업이 성공했음을 보여 줍니다.

```
aws-cloudhsm>shareKey 262177 4 1
**************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
******************************************************************

Do you want to continue(y/n)?y
shareKey success on server 0(10.0.3.10)
shareKey success on server 1(10.0.3.6)
```

작업이 성공했는지 확인하기 위해 예제에서는 첫 번째 getKeyInfo 명령을 반복합니다.

출력은 이제 키 262177가 사용자 4와 공유되고 있음을 보여 줍니다.

```
aws-cloudhsm>getKeyInfo 262177

Key Info on server 0(10.0.3.10):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
Key Info on server 1(10.0.3.6):
```

```
        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
```

Example : 키 공유 해제하기

이 예제는 대칭 키를 공유 해제, 즉 키의 공유 사용자 목록에서 Crypto User를 제거합니다.

이 명령은 shareKey를 사용하여 키 6의 공유 사용자 목록에서 사용자 4를 제거합니다. 마지막 인수는 값 0을 사용하여 공유 해제 작업을 표시합니다.

출력은 두 HSM 모두에서 명령이 성공했음을 보여줍니다. 따라서 사용자 4는 더 이상 암호화 작업에 키 6을 사용할 수 없습니다.

```
aws-cloudhsm>shareKey 6 4 0
************************CAUTION*********************************
This is a CRITICAL operation, should be done on all nodes in the
cluster. AWS does NOT synchronize these changes automatically with the
nodes on which this operation is not executed or failed, please
ensure this operation is executed on all nodes in the cluster.
****************************************************************

Do you want to continue(y/n)?y
shareKey success on server 0(10.0.3.10)
shareKey success on server 1(10.0.3.6)
```

인수

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

<key-handle>

소유하는 키의 키 핸들을 지정합니다. 각 명령에서 키를 하나만 지정할 수 있습니다. 키의 키 핸들을 가져오려면 key_mgmt_util에서 findKey를 사용합니다. 키를 소유하고 있는지 확인하려면 를 사용하십시오. getKeyInfo

필수 여부: 예

<user id>

키를 공유 또는 공유 해제하는 CU(Crypto User)의 사용자 ID를 지정합니다. 사용자의 사용자 ID를 찾으려면 listUsers를 사용합니다.

필수 여부: 예

<share 1 or unshare 0>

지정된 사용자와 키를 공유하려면 1을 입력합니다. 키를 공유 해제하려면, 즉 키의 공유 사용자 목록에서 지정된 사용자를 제거하려면 0을 입력합니다.

필수: 예

관련 주제

- getKeyInfo

syncKey

cloudhsm_mgmt_util의 syncKey 명령을 사용하여 클러스터 내 HSM 인스턴스 간에 또는 복제된 클러스터 간에 키를 수동으로 동기화할 수 있습니다. 일반적으로는 클러스터에 있는 HSM 인스턴스가 자동으로 키를 동기화하므로 이 명령을 사용할 필요가 없습니다. 하지만 복제된 클러스터 간의 키 동기화는 수동으로 수행해야 합니다. 클론 클러스터는 일반적으로 글로벌 규모 조정 및 재해 복구 프로세스를 단순화하기 위해 여러 지역에 생성됩니다. AWS

syncKey를 사용하여 임의 클러스터 간에 키를 동기화할 수 없습니다. 클러스터 중 하나가 다른 클러스터의 백업에서 생성되어야 하기 때문입니다. 또한, 이 작업이 성공적으로 수행되려면 두 클러스터의 CO 및 CU 자격 증명이 일치해야 합니다. 자세한 내용은 HSM 사용자 단원을 참조하십시오.

사용하려면 syncKey 먼저 원본 클러스터의 HSM과 대상 클러스터의 HSM을 각각 지정하는 AWS CloudHSM 구성 파일을 만들어야 합니다. 이렇게 하면 cloudhsm_mgmt_util은 두 HSM 인스턴스에 모두 연결할 수 있습니다. 이 구성 파일을 사용하여 cloudhsm_mgmt_util을 시작합니다. 그런 다음 동기화할 키를 소유하는 CO 또는 CU의 자격 증명을 사용하여 로그인합니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- Crypto officers(CO)

- 암호화 사용자(Crypto User)

> **Note**
>
> CO는 모든 키에서 syncKey를 사용할 수 있는 반면, CU는 자신이 소유하는 키에 대해서만 이 명령을 사용할 수 있습니다. 자세한 정보는 the section called “HSM 사용자 이해”을 참조하세요.

사전 조건

시작하기 전에 대상 HSM과 동기화할 소스 HSM에서 키의 `key handle`을 알아야 합니다. `key handle`을 찾으려면 listUsers 명령을 사용하여 이름이 지정된 사용자에 대한 모든 식별자를 나열합니다. 그런 다음 findAllKeys명령을 사용하여 특정 사용자에 속하는 모든 키를 찾을 수 있습니다.

소스 및 대상 HSM에 할당된 `server IDs`도 알아야 합니다. 이러한 ID는 시작 시 cloudhsm_mgmt_util에서 반환하는 추적 출력에 표시됩니다. 이러한 ID는 HSM이 구성 파일에 나타나는 것과 동일한 순서대로 할당됩니다.

복제된 클러스터 간 CMU 사용 지침에 따라 cloudhsm_mgmt_util을 새 구성 파일로 초기화합니다. 그런 다음 server 명령을 실행하여 소스 HSM에서 서버 모드로 들어갑니다.

구문

> **Note**
>
> syncKey를 실행하려면 먼저 동기화할 키를 포함하는 HSM에서 서버 모드로 들어갑니다.

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

사용자 유형: CU(Crypto User)

```
syncKey <key handle> <destination hsm>
```

예

server 명령을 실행하여 소스 HSM에 로그인하고 서버 모드로 들어갑니다. 이 예제에서는 `server 0`을 소스 HSM으로 가정합니다.

```
aws-cloudhsm> server 0
```

이제 syncKey 명령을 실행합니다. 이 예제에서는 키 261251이 server 1에 동기화된다고 가정합니다.

```
aws-cloudhsm> syncKey 261251 1
syncKey success
```

인수

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
syncKey <key handle> <destination hsm>
```

<key handle>

동기화할 키의 key handle을 지정합니다. 각 명령에서 키를 하나만 지정할 수 있습니다. 키의 키 핸들을 가져오려면 HSM 서버에 로그인한 findAllKeys상태에서 사용하십시오.

필수 여부: 예

<destination hsm>

키를 동기화하려고 하는 서버의 번호를 지정합니다.

필수: 예

관련 주제

- listUsers
- findAllKeys
- 클러스터에 대한 설명 AWS CLI
- 서버

syncUser

cloudhsm_mgmt_util의 syncUser 명령을 사용하여 클러스터 내 HSM 인스턴스 또는 복제된 클러스터 전반에서 암호 사용자 (CU) 또는 암호화 책임자 (CO) 를 수동으로 동기화할 수 있습니다. AWS

CloudHSM 사용자를 자동으로 동기화하지 않습니다. 일반적으로 클러스터의 모든 HSM이 함께 업데이트되도록 글로벌 모드에서 사용자를 관리합니다. 암호 변경 등의 이유로 잘못해서 HSM이 동기화 해제된 경우나 복제된 클러스터 간에 사용자 자격 증명을 교체하려는 경우 syncUser를 사용해야 할 수도 있습니다. 클론 클러스터는 일반적으로 글로벌 규모 조정 및 재해 복구 프로세스를 단순화하기 위해 여러 AWS 지역에 생성됩니다.

CMU 명령을 실행하려면 먼저 CMU를 시작하고, HSM에 로그인해야 합니다. 사용하려는 명령을 실행할 수 있는 사용자 유형으로 로그인해야 합니다.

HSM을 추가하거나 삭제하는 경우 CMU의 구성 파일을 업데이트하십시오. 그렇지 않으면 변경 내용이 클러스터의 모든 HSM에서 유효하지 않을 수도 있습니다.

사용자 유형

다음 사용자 유형이 이 명령을 실행할 수 있습니다.

- Crypto officers(CO)

사전 조건

시작하기 전에 대상 HSM과 동기화할 소스 HSM에서 사용자의 `user ID`를 알아야 합니다. `user ID`를 찾으려면 listUsers 명령을 사용하여 클러스터에 있는 HSM의 모든 사용자를 나열합니다.

소스 및 대상 HSM에 할당된 `server ID`도 알아야 합니다. 이러한 ID는 시작 시 cloudhsm_mgmt_util에서 반환하는 추적 출력에 표시됩니다. 이러한 ID는 HSM이 구성 파일에 나타나는 것과 동일한 순서대로 할당됩니다.

복제된 클러스터 간에 HSM을 동기화하는 경우 복제된 클러스터 간에 CMU 사용 지침에 따라 새 구성 파일로 cloudhsm_mgmt_util을 초기화합니다.

syncUser를 실행할 준비가 되면 server 명령을 실행하여 소스 HSM에서 서버 모드로 들어갑니다.

구문

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
syncUser <user ID> <server ID>
```

예

server 명령을 실행하여 소스 HSM에 로그인하고 서버 모드로 들어갑니다. 이 예제에서는 `server 0`을 소스 HSM으로 가정합니다.

```
aws-cloudhsm> server 0
```

이제 syncUser 명령을 실행합니다. 이 예제에서는 사용자 6이 동기화될 사용자이고 `server 1`이 대상 HSM이라고 가정합니다.

```
server 0> syncUser 6 1
ExtractMaskedObject: 0x0 !
InsertMaskedObject: 0x0 !
syncUser success
```

인수

이 명령에는 이름이 지정된 파라미터가 없으므로 구문 다이어그램에 지정된 순서대로 인수를 입력해야 합니다.

```
syncUser <user ID> <server ID>
```

<user ID>

동기화할 사용자의 ID를 지정합니다. 각 명령에서 사용자를 하나만 지정할 수 있습니다. 사용자의 ID를 가져오려면 listUsers를 사용합니다.

필수 여부: 예

<server ID>

사용자를 동기화하려고 하는 HSM의 서버 번호를 지정합니다.

필수: 예

관련 주제

- listUsers
- 클러스터에 대해 설명하십시오. AWS CLI
- 서버

키 관리 유틸리티(KMU)

키 관리 유틸리티(KMU)는 암호화 사용자 (CU)가 하드웨어 보안 모듈(HSM)의 키를 관리하는 데 도움이 되는 명령줄 도구입니다. KMU에는 키 생성, 삭제, 가져오기 및 내보내기, 속성 가져오기 및 설정, 키 찾기와 암호화 작업을 수행하는 다수의 명령이 포함됩니다.

KMU와 CMU는 클라이언트 SDK 3 제품군의 일부입니다.

빠른 시작은 key_mgmt_util 시작하기 섹션을 참조하십시오. 명령에 대한 자세한 내용은 key_mgmt_util 명령 참조을 참조하십시오. 키 속성 해석에 대한 도움말은 키 속성 참조을 참조하십시오.

Linux를 사용하는 경우 key_mgmt_util을 사용하려면 클라이언트 인스턴스에 연결한 후 AWS CloudHSM 클라이언트 설치 및 구성 (Linux) 단원을 참조하십시오. Windows를 사용하는 경우 AWS CloudHSM 클라이언트 설치 및 구성 (Windows) 단원을 참조하십시오.

주제

- key_mgmt_util 시작하기
- AWS CloudHSM 클라이언트 설치 및 구성 (Linux)
- AWS CloudHSM 클라이언트 설치 및 구성 (Windows)
- key_mgmt_util 명령 참조

key_mgmt_util 시작하기

AWS CloudHSM AWS CloudHSM 클라이언트 소프트웨어와 함께 두 개의 명령줄 도구가 포함되어 있습니다. cloudhsm_mgmt_util 도구에는 HSM 사용자를 관리하기 위한 명령이 포함됩니다. key_mgmt_util 도구에는 키를 관리하기 위한 명령이 포함됩니다. key_mgmt_util 명령줄 도구를 시작하려면 다음 주제를 참조하십시오.

주제

- key_mgmt_util 설정
- key_mgmt_util의 기본적인 사용법

명령에서 오류 메시지 또는 예상치 못한 결과가 발생하는 경우 문제 해결 AWS CloudHSM 항목을 참조하십시오. key_mgmt_util 명령에 대한 자세한 내용은 key_mgmt_util 명령 참조 섹션을 참조하십시오.

key_mgmt_util 설정

key_mgmt_util을 사용하기 전에 다음 설정 작업을 완료합니다.

AWS CloudHSM 클라이언트를 시작합니다.

key_mgmt_util을 사용하기 전에 먼저 클라이언트를 시작해야 합니다. AWS CloudHSM 클라이언트는 클러스터의 HSM과 암호화된 통신을 설정하는 데몬입니다. end-to-end key_mgmt_util 도구는 클라이언트 연결을 사용하여 클러스터의 HSM과 통신을 합니다. 이 도구가 없으면 key_mgmt_util이 작동하지 않습니다.

클라이언트를 시작하려면 AWS CloudHSM

다음 명령을 사용하여 AWS CloudHSM 클라이언트를 시작합니다.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Windows 클라이언트 1.1.2+의 경우:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows 클라이언트 1.1.1 이상의 경우:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

key_mgmt_util 시작

AWS CloudHSM 클라이언트를 시작한 후 다음 명령을 사용하여 key_mgmt_util을 시작합니다.

Amazon Linux

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Amazon Linux 2

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

CentOS 7

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

CentOS 8

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

RHEL 7

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

RHEL 8

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Ubuntu 16.04 LTS

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Ubuntu 18.04 LTS

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

Windows

```
c:\Program Files\Amazon\CloudHSM> .\key_mgmt_util.exe
```

프롬프트는 key_mgmt_util이 실행 중일 때 `Command:`로 변경됩니다.

명령이 실패할 경우(예: `Daemon socket connection error` 메시지를 반환) 구성 파일을 업데이트해 보십시오.

key_mgmt_util의 기본적인 사용법

key_mgmt_util 도구의 기본적인 사용법은 다음 주제를 참조하십시오.

주제

- HSM에 로그인
- HSM에서 로그아웃
- key_mgmt_util 중지

HSM에 로그인

loginHSM 명령을 사용하여 HSM에 로그인합니다. 다음 명령은 `example_user`이라는 CU(Crypto User)로 로그인합니다. 출력은 클러스터의 세 HSM 모두에 대해 성공적인 로그인을 나타냅니다.

```
Command:  loginHSM -u CU -s example_user -p <PASSWORD>
Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS
```

```
Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

다음은 loginHSM 명령의 구문을 보여 줍니다.

```
Command:  loginHSM -u <USER TYPE> -s <USERNAME> -p <PASSWORD>
```

HSM에서 로그아웃

logoutHSM 명령을 사용하여 HSM에서 로그아웃합니다.

```
Command:  logoutHSM
Cfm3LogoutHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

key_mgmt_util 중지

exit 명령을 사용하여 key_mgmt_util을 중지합니다.

```
Command:  exit
```

AWS CloudHSM 클라이언트 설치 및 구성 (Linux)

AWS CloudHSM 클러스터의 HSM과 상호 작용하려면 Linux용 AWS CloudHSM 클라이언트 소프트웨어가 필요합니다. 이전에 생성한 Linux EC2 클라이언트 인스턴스에 설치해야 합니다. Windows를 사용하는 경우에도 클라이언트를 설치할 수 있습니다. 자세한 정보는 AWS CloudHSM 클라이언트 설치 및 구성 (Windows)을 참조하세요.

Tasks

- AWS CloudHSM 클라이언트 및 명령줄 도구를 설치합니다.
- 클라이언트 구성 편집

AWS CloudHSM 클라이언트 및 명령줄 도구를 설치합니다.

클라이언트 인스턴스에 연결하고 다음 명령을 실행하여 AWS CloudHSM 클라이언트 및 명령줄 도구를 다운로드하고 설치합니다.

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-latest.el6.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

```
sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

```
sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb
```

클라이언트 구성 편집

AWS CloudHSM 클라이언트를 사용하여 클러스터에 연결하려면 먼저 클라이언트 구성을 편집해야 합니다.

클라이언트 구성을 편집하려면

1. 발급 인증서 복사 - 클러스터 인증서에 서명하는 데 사용한 인증서를 클라이언트 인스턴스의 /opt/cloudhsm/etc/customerCA.crt 위치에 복사합니다. 인증서를 이 위치로 복사하려면 클라이언트 인스턴스에 대해 인스턴스 루트 사용자 권한이 필요합니다.
2. 다음 configure 명령을 사용하여 클러스터에 있는 HSM의 IP 주소를 지정하여 AWS CloudHSM 클라이언트 및 명령줄 도구의 구성 파일을 업데이트하십시오. HSM의 IP 주소를 가져오려면 AWS CloudHSM 콘솔에서 클러스터를 보거나 명령을 실행하십시오. describe-clusters AWS CLI 명령의 출력에서 HSM의 IP 주소는 EniIp 필드의 값입니다. HSM이 두 개 이상인 경우 임의 HSM의 IP 주소를 선택합니다.

   ```
   sudo /opt/cloudhsm/bin/configure -a <IP address>

   Updating server config in /opt/cloudhsm/etc/cloudhsm_client.cfg
   Updating server config in /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

3. 클러스터 활성화로 이동합니다.

AWS CloudHSM 클라이언트 설치 및 구성 (Windows)

Windows의 AWS CloudHSM 클러스터에서 HSM을 사용하려면 Windows용 AWS CloudHSM 클라이언트 소프트웨어가 필요합니다. 이전에 생성한 Windows 서버 인스턴스에 설치해야 합니다.

최신 Windows 클라이언트 및 명령줄 도구를 설치(또는 업데이트)하려면

1. Windows 서버 인스턴스에 연결합니다.
2. 다운로드 페이지에서 최신 버전 (AWSCloudHSMClient-latest.msi) 을 다운로드합니다.
3. 다운로드 위치로 이동하여 관리자 권한으로 설치 프로그램 (AWSCloudHSMClient-latest.msi) 을 실행합니다.
4. 설치 프로그램 지침을 따르고 설치 프로그램이 완료되면 닫기를 선택합니다.
5. 자체 서명된 발급 인증서 복사 - 클러스터 인증서에 서명하는 데 사용된 인증서를 C:\ProgramData\Amazon\CloudHSM 폴더로 복사합니다.
6. 다음 명령을 실행하여 구성 파일을 업데이트합니다. 업데이트하려면 재구성 중에 클라이언트를 중지했다 시작해야 합니다.

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ .\configure.exe -a <HSM IP address>
   ```

7. 클러스터 활성화로 이동합니다.

참고:

- 클라이언트를 업데이트하려는 경우, 이전 설치의 기존 구성 파일을 덮어쓰지 않습니다.
- Windows용 AWS CloudHSM 클라이언트 설치 프로그램은 암호화 API: 차세대 (CNG) 및 KSP (키 저장소 공급자) 를 자동으로 등록합니다. 클라이언트를 제거하려면 설치 프로그램을 다시 실행하고 제거 지침을 따르십시오.
- Linux를 사용하는 경우, Linux 클라이언트를 설치할 수 있습니다. 자세한 내용은 AWS CloudHSM 클라이언트 설치 및 구성 (Linux) 단원을 참조하십시오.

key_mgmt_util 명령 참조

key_mgmt_util 명령줄 도구는 키 및 속성의 생성, 삭제, 찾기 등 클러스터의 HSM에서 키를 관리하는 데 사용할 수 있습니다. 이 도구에는 여러 명령이 포함되며, 이 주제에서 각 명령을 자세히 설명합니다.

빠른 시작은 key_mgmt_util 시작하기 섹션을 참조하십시오. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오. 클러스터에서 HSM 및 사용자를 관리하기 위한 명령을 포함하는 cloudhsm_mgmt_util 명령줄 도구에 대한 자세한 내용은 CloudHSM 관리 유틸리티(CMU) 섹션을 참조하십시오.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

key_mgmt_util 명령을 모두 나열하려면 다음과 같이 입력합니다.

```
Command: help
```

특정 key_mgmt_util 명령에 대한 도움말을 보려면 다음과 같이 입력합니다.

```
Command: <command-name> -h
```

key_mgmt_util 세션을 종료하려면 다음과 같이 입력합니다.

```
Command: exit
```

다음 주제에서는 key_mgmt_util의 명령에 대해 설명합니다.

> **Note**
>
> key_mgmt_util과 cloudhsm_mgmt_util의 일부 명령은 이름이 같습니다. 하지만 일반적으로 명령은 구문, 출력 및 기능이 조금씩 다릅니다.

Command	설명
aesWrapUnwrap	파일에서 키의 내용을 암호화 또는 암호 해독합니다.
deleteKey	HSM에서 키를 삭제합니다.
Error2String	key_mgmt_util 16진수 오류 코드에 해당하는 오류를 가져옵니다.
exit	key_mgmt_util을 종료합니다.
exportPrivateKey	HSM의 프라이빗 키 사본을 디스크의 파일로 내보냅니다.
exportPubKey	퍼블릭 키 사본을 HSM에서 파일로 내보냅니다.
exSymKey	대칭 키의 일반 텍스트 사본을 HSM에서 파일로 내보냅니다.
extractMaskedObject	HSM의 키를 마스킹 처리된 객체 파일로 추출합니다.
findKey	키 속성 값을 기준으로 키를 검색합니다.
findSingleKey	키가 클러스터의 모든 HSM에 존재하는지 확인합니다.
GenDSA KeyPair	HSM에서 디지털 서명 알고리즘(DSA) 키 페어를 생성합니다.
제네C KeyPair	HSM에서 타원 곡선 암호(ECC) 키 페어를 생성합니다.

Command	설명
GenRSA KeyPair	HSM에서 RSA 비대칭 키 페어를 생성합니다.
genSymKey	HSM에서 대칭 키 페어를 생성합니다.
getAttribute	AWS CloudHSM 키의 속성 값을 가져와 파일에 기록합니다.
getCaviumPriv키	프라이빗 키의 가짜 PEM 형식 버전을 만들어 파일로 내보냅니다.
getCert	HSM의 파티션 인증서를 검색하여 파일에 저장합니다.
getKeyInfo	키를 사용할 수 있는 사용자의 HSM 사용자 ID를 가져옵니다. 키가 쿼럼 제어되는 경우 쿼럼의 사용자 수를 가져옵니다.
help	key_mgmt_util에서 사용 가능한 명령에 대한 도움말 정보를 표시합니다.
importPrivateKey	프라이빗 키를 HSM으로 가져옵니다.
importPubKey	퍼블릭 키를 HSM으로 가져옵니다.
imSymKey	대칭 키의 평문 사본을 파일에서 HSM으로 가져옵니다.
insertMaskedObject	디스크에 있는 파일의 마스킹 처리된 객체를 해당 객체의 원본 클러스터와 관련된 클러스터에 포함된 HSM에 삽입합니다. 관련 클러스터는 원본 클러스터의 백업에서 생성된 모든 클러스터입니다.
???	지정된 파일에 실제 프라이빗 키가 포함되었는지 가짜 PEM 키가 포함되었는지 확인합니다.

Command	설명
listAttributes	AWS CloudHSM 키의 속성과 이를 나타내는 상수를 나열합니다.
listUsers	HSM의 사용자, 해당 사용자 유형/ID 및 기타 속성을 가져옵니다.
loginHSM 및 logoutHSM	클러스터의 HSM에 로그인 및 로그아웃합니다.
setAttribute	세션 키를 영구 키로 변환합니다.
sign	선택한 프라이빗 키를 사용하여 파일의 서명을 생성합니다.
unWrapKey	래핑(암호화)된 키를 파일에서 HSM으로 가져옵니다.
verify	지정된 키를 사용하여 지정된 파일에 서명했는지 여부를 확인합니다.
wrapKey	키의 암호화된 복사본을 HSM에서 파일로 내보냅니다.

aesWrapUnwrap

aesWrapUnwrap 명령은 디스크에 있는 파일의 내용을 암호화 또는 암호화 해제합니다. 이 명령은 암호화 키를 래핑/언래핑하도록 설계되었지만, 이 명령을 4KB(4,096바이트) 미만의 데이터를 포함하는 임의의 파일에 사용할 수 있습니다.

aesWrapUnwrap은 AES 키 래핑을 사용합니다. 이 방식은 HSM의 AES 키를 래핑 또는 언래핑 키로 사용합니다. 그런 다음 결과를 디스크의 다른 파일에 기록합니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
aesWrapUnwrap -h
```

```
aesWrapUnwrap -m <wrap-unwrap mode>
              -f <file-to-wrap-unwrap>
              -w <wrapping-key-handle>
              [-i <wrapping-IV>]
              [-out <output-file>]
```

예제

다음 예제는 aesWrapUnwrap을 사용하여 파일에서 암호화 키를 암호화 및 암호화 해제하는 방법을 보여 줍니다.

Example : 암호화 키 래핑

이 명령은 HSM에서 일반 aesWrapUnwrap 텍스트로 내보낸 트리플 DES 대칭 키를 파일로 래핑하는 데 사용됩니다. `3DES.key` 유사한 명령을 사용하여 파일에 저장된 임의의 키를 래핑할 수 있습니다.

이 명령은 `-m` 파라미터에서 `1` 값을 사용하여 래핑 모드를 표시합니다. 이 명령은 `-w` 파라미터를 사용하여 HSM의 AES 키(키 핸들 6)를 래핑 키로 지정합니다. 이 명령은 래핑된 키를 `3DES.key.wrapped` 파일에 기록합니다.

출력은 이 명령이 성공했으며 작업이 기본값 IV를 사용(권장)한 것을 보여 줍니다.

```
 Command:  aesWrapUnwrap -f 3DES.key -w 6 -m 1 -out 3DES.key.wrapped

        Warning: IV (-i) is missing.
                 0xA6A6A6A6A6A6A6A6 is considered as default IV
result data:
49 49 E2 D0 11 C1 97 22
17 43 BD E3 4E F4 12 75
8D C1 34 CF 26 10 3A 8D
6D 0A 7B D5 D3 E8 4D C2
79 09 08 61 94 68 51 B7

result written to file 3DES.key.wrapped

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

Example : 암호화 키 언래핑

이 예제는 aesWrapUnwrap을 사용하여 파일의 래핑된(암호화된) 키를 언래핑(암호화 해제)하는 방법을 보여 줍니다. 키를 HSM으로 가져오기 전에 이러한 작업을 수행해야 할 수 있습니다. 예를 들어

imSymKey명령을 사용하여 암호화된 키를 가져오려고 하면 암호화된 키가 해당 유형의 일반 텍스트 키에 필요한 형식이 아니기 때문에 오류가 반환됩니다.

이 명령은 3DES.key.wrapped 파일의 키를 언래핑하고 평문을 3DES.key.unwrapped 파일에 기록합니다. 이 명령은 -m 파라미터에서 0 값을 사용하여 언래핑 모드를 표시합니다. 이 명령은 -w 파라미터를 사용하여 HSM의 AES 키(키 핸들 6)를 래핑 키로 지정합니다. 이 명령은 래핑된 키를 3DES.key.unwrapped 파일에 기록합니다.

```
 Command:  aesWrapUnwrap -m 0 -f 3DES.key.wrapped -w 6 -out 3DES.key.unwrapped

        Warning: IV (-i) is missing.
                 0xA6A6A6A6A6A6A6A6 is considered as default IV
result data:
14 90 D7 AD D6 E4 F5 FA
A1 95 6F 24 89 79 F3 EE
37 21 E6 54 1F 3B 8D 62

result written to file 3DES.key.unwrapped

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-m

모드를 지정합니다. 파일 내용을 래핑(암호화)하려면 1을 입력하고, 파일 내용을 언래핑(암호 해독)하려면 0을 입력합니다.

필수 여부: 예

-f

래핑할 파일을 지정합니다. 4KB(4,096바이트) 미만의 데이터를 포함하는 파일을 입력합니다. 이 작업은 암호화 키를 래핑 및 언래핑하도록 설계되었습니다.

필수 여부: 예

-w

래핑 키를 지정합니다. HSM에서 AES 키 또는 키의 키 핸들을 입력합니다. 이 파라미터는 필수 사항입니다. 키 핸들을 찾으려면 findKey 명령을 사용하십시오.

래핑 키를 만들려면 AES 키 (유형 31) 를 생성하는 데 사용합니다 genSymKey.

필수 여부: 예

-i

알고리즘에 대체 초기 값(IV)을 지정합니다. 대체 값이 필요한 특별한 조건이 아니면 기본값을 사용합니다.

기본값: `0xA6A6A6A6A6A6A6A6`. 기본값은 AES 키 래핑 알고리즘 사양에 정의되어 있습니다.

필수 여부: 아니요

-out

래핑 또는 언래핑된 키를 포함하는 출력 파일에 대해 대체 이름을 지정합니다. 기본값은 로컬 디렉터리의 `wrapped_key`(래핑 작업) 및 `unwrapped_key`(언래핑 작업)입니다.

파일이 존재하는 경우, aesWrapUnwrap은 경고 없이 파일을 덮어씁니다. 명령이 실패할 경우, aesWrapUnwrap이 내용이 없는 출력 파일을 생성합니다.

기본값: 래핑의 경우: `wrapped_key`. 언래핑의 경우: `unwrapped_key`.

필수 여부: 아니요

관련 주제

- exSymKey
- imSymKey
- unWrapKey
- wrapKey

deleteKey

key_mgmt_util의 deleteKey 명령은 HSM에서 키를 삭제합니다. 키는 한 번에 하나만 삭제할 수 있습니다. 키 페어에서 키 하나를 삭제해도 해당 페어의 다른 키에는 아무 영향을 미치지 않습니다.

키 소유자만 키를 삭제할 수 있습니다. 키를 공유하는 사용자는 해당 키를 암호화 작업에 사용할 수 있지만 삭제할 수는 없습니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 암호화 사용자(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
deleteKey -h

deleteKey -k
```

예제

다음 예제에서는 deleteKey를 사용하여 HSM에서 키를 삭제하는 방법을 보여 줍니다.

Example : 키 삭제하기

이 명령은 키 핸들이 6인 키를 삭제합니다. deleteKey 명령이 성공하면 클러스터의 각 HSM에서 성공 메시지가 반환됩니다.

```
Command: deleteKey -k 6

        Cfm3DeleteKey returned: 0x00 : HSM Return: SUCCESS

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : 키 삭제하기(실패)

지정된 키 핸들이 있는 키가 없어 deleteKey 명령이 실패하면 잘못된 객체 핸들 오류 메시지가 반환됩니다.

```
Command: deleteKey -k 252126

        Cfm3FindKey returned: 0xa8 : HSM Error: Invalid object handle is passed to this
 operation

        Cluster Error Status
        Node id 1 and err state 0x000000a8 : HSM Error: Invalid object handle is passed
 to this operation
```

```
        Node id 2 and err state 0x000000a8 : HSM Error: Invalid object handle is passed
 to this operation
```

현재 사용자가 키 소유자가 아니기 때문에 명령이 실패할 경우 액세스 거부됨 오류가 반환됩니다.

```
Command:  deleteKey -k 262152

Cfm3DeleteKey returned: 0xc6 : HSM Error: Key Access is denied.
```

파라미터

-h

명령에 대한 명령줄 도움말을 표시합니다.

필수 여부: 예

-k

삭제할 키의 키 핸들을 지정합니다. HSM에서 키의 키 핸들을 찾으려면 findKey를 사용하십시오.

필수: 예

관련 주제

- findKey

Error2String

key_mgmt_util의 Error2String 헬퍼 명령은 key_mgmt_util 16진수 오류 코드에 해당하는 오류를 반환합니다. 명령 및 스크립트를 문제 해결할 때 이 명령을 사용할 수 있습니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
Error2String -h

Error2String -r <response-code>
```

예제

다음 예제는 Error2String을 사용하여 key_mgmt_util 오류 코드의 오류 문자열을 가져오는 방법을 보여 줍니다.

Example : 오류 설명 가져오기

이 명령은 0xdb 오류 코드에 대한 오류 설명을 가져옵니다. 이 설명은 사용자가 잘못된 사용자 유형이기 때문에 key_mgmt_util에 대한 로그인 시도가 실패했다고 설명합니다. CU(Crypto User)만 key_mgmt_util에 로그인할 수 있습니다.

```
        Command:  Error2String -r 0xdb

        Error Code db maps to HSM Error: Invalid User Type.
```

Example : 오류 코드 찾기

이 예제는 key_mgmt_util 오류의 오류 코드를 찾을 수 있는 위치를 보여 줍니다. 오류 코드 0xc6은 문자열 Cfm3*command-name* returned: 다음에 표시됩니다.

이 예시에서는 현재 사용자 (사용자 4) 가 암호화 작업에 키를 getKeyInfo사용할 수 있음을 나타냅니다. 그럼에도 사용자가 deleteKey를 사용하여 키를 삭제하려고 시도할 경우 명령이 오류 코드 0xc6을 반환합니다.

```
        Command:  deleteKey -k 262162

        Cfm3DeleteKey returned: 0xc6 : HSM Error: Key Access is denied

        Cluster Error Status

        Command:  getKeyInfo -k 262162

        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 1 user(s):

                4
```

0xc6 오류가 보고될 경우 이와 같은 Error2String 명령을 사용하여 오류를 조회할 수 있습니다. 이 경우에는 현재 사용자는 키를 공유하는 것이고 다른 사용자가 소유자이기 때문에 deleteKey 명령이 실패한 것입니다. 키 소유자만 키를 삭제할 권한이 있습니다.

```
        Command:  Error2String -r 0xa8

        Error Code c6 maps to HSM Error: Key Access is denied
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-r

16진수 오류 코드를 지정합니다. 0x 16진수 표시기가 필요합니다.

필수 여부: 예

exit

key_mgmt_util의 exit 명령은 key_mgmt_util을 종료합니다. 성공적으로 종료하면 표준 명령줄로 돌아갑니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작해야 합니다.

구문

```
exit
```

파라미터

이 명령에 대한 파라미터는 없습니다.

관련 주제

- key_mgmt_util 시작

exportPrivateKey

key_mgmt_util의 exportPrivateKey 명령은 HSM의 비대칭 프라이빗 키를 파일로 내보냅니다. HSM에서는 일반 텍스트로 키를 직접 내보낼 수 없습니다. 이 명령은 사용자가 지정한 AES 래핑 키를 사용하여 프라이빗 키를 래핑하고, 래핑된 바이트를 해독하며, 일반 텍스트 프라이빗 키를 파일에 복사합니다.

exportPrivateKey 명령은 HSM에서 키를 제거하거나 키 속성을 변경하거나 향후 암호화 작업에서 키를 사용하는 것을 방지하지 않습니다. 동일한 키를 여러 번 내보낼 수 있습니다.

OBJ_ATTR_EXTRACTABLE 속성 값이 1인 프라이빗 키만 내보낼 수 있습니다. OBJ_ATTR_WRAP 및 OBJ_ATTR_DECRYPT 속성 값 1이 있는 AES 래핑 키를 지정해야 합니다. 키의 속성을 찾으려면 getAttribute 명령을 사용합니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 HSM에 CU(crypto user)로 로그인해야 합니다.

구문

```
exportPrivateKey -h

exportPrivateKey -k <private-key-handle
                 -w <wrapping-key-handle>
                 -out <key-file>
                 [-m <wrapping-mechanism>]
                 [-wk <wrapping-key-file>]
```

예제

이 예제에서는 exportPrivateKey를 사용하여 HSM에서 프라이빗 키를 내보내는 방법을 보여 줍니다.

Example : 프라이빗 키 내보내기

이 명령은 핸들이 16인 래핑 키를 사용하여 핸들이 15인 프라이빗 키를 exportKey.pem이라는 PEM 파일로 내보냅니다. 명령이 성공하면 exportPrivateKey가 성공 메시지를 반환합니다.

```
Command: exportPrivateKey -k 15 -w 16 -out exportKey.pem

Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

```
PEM formatted private key is written to exportKey.pem
```

파라미터

이 명령은 다음 파라미터를 사용합니다.

-h

명령에 대한 명령줄 도움말을 표시합니다.

필수 여부: 예

-k

내보낼 프라이빗 키의 키 핸들을 지정합니다.

필수 여부: 예

-w

래핑 키의 키 핸들을 지정합니다. 이 파라미터는 필수 사항입니다. 키 핸들을 찾으려면 findKey 명령을 사용합니다.

키를 래핑 키로 사용할 수 있는지 여부를 확인하려면 getAttribute를 사용하여 OBJ_ATTR_WRAP 속성(262)의 값을 가져옵니다. 래핑 키를 생성하려면 genSymKey를 사용하여 AES 키(유형 31)를 생성합니다.

-wk 파라미터를 사용하여 외부 언래핑 키를 지정할 경우, 내보내기 시 -w 래핑 키가 키를 래핑하는 데 사용되고 언래핑에는 사용되지 않습니다.

필수 여부: 예

-out

내보낸 프라이빗 키를 쓸 파일 이름을 지정합니다.

필수 여부: 예

-m

내보내고 있는 프라이빗 키를 래핑할 래핑 메커니즘을 지정합니다. 유일하게 유효한 값은 NIST_AES_WRAP mechanism.을 나타내는 4입니다.

기본값: 4(NIST_AES_WRAP)

필수 여부: 아니요

-wk

내보내고 있는 키를 언래핑하는 데 사용할 키를 지정합니다. 평문 AES 키를 포함하는 파일의 경로 및 이름을 입력합니다.

이 파라미터를 포함할 경우, exportPrivateKey는 -w 파일의 키를 사용하여 내보낼 키를 래핑하고 -wk 파라미터로 지정된 키를 사용하여 언래핑합니다.

기본값: -w 파라미터에 지정된 래핑 키를 사용하여 래핑과 언래핑을 모두 합니다.

필수 여부: 아니요

관련 주제

- importPrivateKey
- wrapKey
- unWrapKey
- genSymKey

exportPubKey

key_mgmt_util의 exportPubKey 명령은 HSM의 퍼블릭 키를 파일로 내보냅니다. 이 명령을 사용하여 HSM에 생성한 퍼블릭 키를 내보낼 수 있습니다. 또한 이 명령을 사용하여 HSM으로 가져온 퍼블릭 키(예: importPubKey 명령으로 가져온 키)를 내보낼 수도 있습니다.

exportPubKey 작업은 키 구성 요소를 지정한 파일에 복사합니다. 그러나 HSM에서 키를 제거하거나, 키 속성을 변경하거나, 향후 암호화 작업에서 키 사용을 못하게 하지는 않습니다. 동일한 키를 여러 번 내보낼 수 있습니다.

OBJ_ATTR_EXTRACTABLE 값이 1인 퍼블릭 키만 내보낼 수 있습니다. 키의 속성을 찾으려면 getAttribute 명령을 사용합니다.

key_mgmt_util 명령을 실행하려면 먼저 key_mgmt_util을 시작하고 HSM에 CU(crypto user)로 로그인해야 합니다.

구문

```
exportPubKey -h

exportPubKey -k <public-key-handle
```

```
            -out <key-file>
```

예제

이 예제에서는 exportPubKey를 사용하여 HSM에서 퍼블릭 키를 내보내는 방법을 보여 줍니다.

Example : 퍼블릭 키 내보내기

이 명령은 핸들이 10인 퍼블릭 키를 public.pem이라는 파일로 내보냅니다. 명령이 성공하면 exportPubKey가 성공 메시지를 반환합니다.

```
Command: exportPubKey -k 10 -out public.pem

PEM formatted public key is written to public.pem

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

파라미터

이 명령은 다음 파라미터를 사용합니다.

-h

명령에 대한 명령줄 도움말을 표시합니다.

필수 여부: 예

-k

내보낼 퍼블릭 키의 키 핸들을 지정합니다.

필수 여부: 예

-out

내보낸 퍼블릭 키를 쓸 파일 이름을 지정합니다.

필수: 예

관련 주제

- importPubKey
- 키 생성

exSymKey

key_mgmt_util 도구의 exSymKey 명령은 HSM에서 대칭 키의 일반 텍스트 복사본을 내보내 디스크의 파일에 저장합니다. 키의 암호화(래핑)된 사본을 내보내려면 wrapKey 명령을 사용합니다. 내보내는 키와 같은 일반 텍스트 키를 가져오려면 를 사용하십시오. exSymKey imSymKey

가져오기 절차 도중, exSymKey는 지정된 AES 키(래핑 키)를 사용하여 내보낼 키를 래핑(암호화)한 후 언래핑(암호화 해제)합니다. 하지만 내보내기 작업의 결과는 디스크에 저장된 평문(언래핑된) 키입니다.

키 소유자, 즉 키를 생성한 CU 사용자만 키를 내보낼 수 있습니다. 키를 공유하는 사용자는 해당 키를 암호화 작업에 사용할 수 있지만 내보낼 수는 없습니다.

exSymKey 작업은 키 구성 요소를 지정한 파일에 복사하지만, HSM에서 키를 제거하지 않고 키 속성을 변경하거나 암호화 작업에서 키를 사용할 수 없게 만듭니다. 동일한 키를 여러 번 내보낼 수 있습니다.

exSymKey는 대칭 키만 내보냅니다. 퍼블릭 키를 내보내려면 를 사용하십시오. exportPubKey 개인 키를 내보내려면 를 사용하십시오 exportPrivateKey.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 암호화 사용자(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
exSymKey -h

exSymKey -k <key-to-export>
         -w <wrapping-key>
         -out <key-file>
         [-m 4]
         [-wk <unwrapping-key-file> ]
```

예제

다음 예제는 exSymKey를 사용하여 HSM에서 소유하는 대칭 키를 내보내는 방법을 보여 줍니다.

Example : 3DES 대칭 키 내보내기

이 명령은 Triple DES(3DES) 대칭 키(키 핸들 7)를 내보냅니다. 이 명령은 HSM의 기존 AES 키(키 핸들 6)를 래핑 키로 사용합니다. 그런 다음 3DES 키의 평문을 3DES.key 파일에 기록합니다.

출력은 키 7(3DES 키)가 성공적으로 래핑되고 3DES.key 파일에 기록된 것을 보여 줍니다.

> **⚠ Warning**
>
> 출력이 "래핑된 대칭 키"가 출력 파일에 기록되었음을 보여 주지만, 출력 파일에는 평문(언래핑된) 키가 포함되어 있습니다.

```
        Command: exSymKey -k 7 -w 6 -out 3DES.key

       Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS


Wrapped Symmetric Key written to file "3DES.key"
```

Example : 세션 전용 래핑 키를 사용하여 내보내기

이 예제는 해당 세션에서만 존재하는 키를 래핑 키로 사용하는 방법을 보여 줍니다. 내보낼 키가 래핑되었다가 즉시 언래핑되고 평문으로 전달되므로 래핑 키를 유지할 필요가 없습니다.

이 일련의 명령은 HSM에서 키 핸들이 8인 AES 키를 내보냅니다. 이 명령은 전용으로 생성된 AES 세션 키를 사용합니다.

첫 번째 명령은 256비트 AES 키를 생성하는 genSymKey데 사용됩니다. 이 명령은 `-sess` 파라미터를 사용하여 현재 세션에만 존재하는 키를 생성합니다.

출력은 HSM이 키 262168을 생성했음을 보여 줍니다.

```
        Command:  genSymKey -t 31 -s 32 -l AES-wrapping-key -sess

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 262168

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

다음으로, 예제는 내보낼 키인 키 8이 추출 가능한 대칭 키인지 확인합니다. 또한 래핑 키(키 262168)가 해당 세션에서만 존재하는 AES 키인지 확인합니다. findKey 명령을 사용할 수 있지만, 이 예제는 두 키의 속성을 모두 내보낸 다음 `grep`을 사용하여 파일에서 관련 속성 값을 찾습니다.

이러한 명령은 getAttribute에서 512의 -a(모두) 값을 사용하여 키 8 및 262168의 모든 속성을 가져옵니다. 키 속성에 대한 자세한 내용은 the section called “키 속성 참조” 섹션을 참조하십시오.

```
getAttribute -o 8 -a 512 -out attributes/attr_8
getAttribute -o 262168 -a 512 -out attributes/attr_262168
```

다음 명령은 grep을 사용하여 내보낼 키(키 8)의 속성과 세션 전용 래핑 키(키 262168)를 확인합니다.

```
    // Verify that the key to be exported is a symmetric key.
    $  grep -A 1 "OBJ_ATTR_CLASS" attributes/attr_8
    OBJ_ATTR_CLASS
    0x04

    // Verify that the key to be exported is extractable.
    $  grep -A 1 "OBJ_ATTR_KEY_TYPE" attributes/attr_8
    OBJ_ATTR_EXTRACTABLE
    0x00000001

    // Verify that the wrapping key is an AES key
    $  grep -A 1 "OBJ_ATTR_KEY_TYPE" attributes/attr_262168
    OBJ_ATTR_KEY_TYPE
    0x1f

    // Verify that the wrapping key is a session key
    $  grep -A 1 "OBJ_ATTR_TOKEN" attributes/attr_262168
    OBJ_ATTR_TOKEN
    0x00

    // Verify that the wrapping key can be used for wrapping
     $  grep -A 1 "OBJ_ATTR_WRAP" attributes/attr_262168
    OBJ_ATTR_WRAP
    0x00000001
```

마지막으로, exSymKey 명령을 통해 세션 키(키 262168)를 래핑 키로 사용해 키 8을 내보냅니다.

세션이 끝나면 키 262168은 더 이상 존재하지 않습니다.

```
        Command:  exSymKey -k 8 -w 262168 -out aes256_H8.key

        Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS
```

```
Wrapped Symmetric Key written to file "aes256_H8.key"
```

Example : 외부 언래핑 키 사용

이 예제는 the HSM 외부 언래핑 키를 사용하여 HSM에서 키를 내보내는 방법을 보여 줍니다.

HSM에서 키를 내보낼 때 래핑 키로 사용할 HSM의 AES 키를 지정합니다. 기본적으로 래핑 키는 내보낼 키를 래핑 및 언래핑하는 데 사용합니다. 하지만 -wk 파라미터를 사용하여 언래핑에 파일에 있는 외부 키를 사용하도록 exSymKey에게 알릴 수 있습니다. 그러면 -w 파라미터로 지정된 키가 대상 키를 래핑하고 -wk 파라미터로 지정된 파일의 키가 언래핑합니다.

래핑 키는 대칭 키인 AES 키여야 하므로, HSM의 래핑 키와 디스크의 언래핑 키는 키 구성 요소가 같아야 합니다. 이렇게 하려면 내보내기 작업 전에 HSM으로 래핑 키를 가져오거나 HSM에서 래핑 키를 내보내야 합니다.

이 예제에서는 키를 HSM 외부에서 생성하여 HSM으로 가져옵니다. 이 예제는 키의 내부 사본을 사용하여 가져오는 대칭 키를 래핑하고 파일에 저장된 키의 사본을 사용하여 언래핑합니다.

첫 번째 명령은 OpenSSL을 사용하여 256비트 AES 키를 생성합니다. 이 명령은 aes256-forImport.key 파일에 키를 저장합니다. OpenSSL 명령은 출력을 반환하지 않지만, 여러 명령을 사용하여 성공 여부를 확인할 수 있습니다. 이 예제는 wc(wordcount) 도구를 사용하여 파일이 32바이트의 데이터를 포함하는지 확인합니다.

```
$  openssl rand -out keys/aes256-forImport.key 32

$ wc keys/aes256-forImport.key
 0  2 32 keys/aes256-forImport.key
```

이 명령은 명령을 사용하여 파일에서 HSM으로 AES 키를 가져옵니다. imSymKeyaes256-forImport.key 명령이 완료되면 키가 HSM(키 핸들 262167) 및 aes256-forImport.key 파일에 존재합니다.

```
Command:  imSymKey -f keys/aes256-forImport.key -t 31 -l aes256-imported -w 6

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS
```

```
        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Unwrapped.  Key Handle: 262167

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

이 명령은 내보내기 작업에 키를 사용합니다. 이 명령은 exSymKey를 사용하여 192비트 AES 키인 키 21을 내보냅니다. 키를 래핑하기 위해 이 명령은 HSM으로 가져온 사본인 키 262167을 사용합니다. 키를 언래핑하기 위해 `aes256-forImport.key` 파일에서 동일한 키 구성 요소를 사용합니다. 명령이 완료되면 키 21가 `aes192_h21.key` 파일로 내보내집니다.

```
        Command:  exSymKey -k 21 -w 262167 -out aes192_H21.key -wk aes256-forImport.key

        Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

Wrapped Symmetric Key written to file "aes192_H21.key"
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-k

내보낼 키의 키 핸들을 지정합니다. 이 파라미터는 필수 사항입니다. 소유하는 대칭 키의 키 핸들을 입력합니다. 이 파라미터는 필수 사항입니다. 키 핸들을 찾으려면 findKey 명령을 사용하십시오.

키를 내보낼 수 있는지 확인하려면 getAttribute 명령을 사용하여 상수 354로 표시되는 `OBJ_ATTR_EXTRACTABLE` 속성의 값을 가져옵니다. 또한 본인이 소유한 키만 내보낼 수 있습니다. 키 소유자를 찾으려면 getKeyInfo명령을 사용합니다.

필수 여부: 예

-w

래핑 키의 키 핸들을 지정합니다. 이 파라미터는 필수 사항입니다. 키 핸들을 찾으려면 findKey 명령을 사용하십시오.

래핑 키는 내보낼 키를 암호화(래핑)한 후 암호 해독(언래핑)하는 데 사용되는 HSM의 키입니다. AES 키만 래핑 키로 사용할 수 있습니다.

임의의 AES 키(모든 크기)를 래핑 키로 사용할 수 있습니다. 래핑 키는 대상 키를 래핑했다 즉시 언래핑하므로 세션 전용 AES 키를 래핑 키로 사용할 수 있습니다. 키를 래핑 키로 사용할 수 있는지 판단하려면 getAttribute를 사용하여 262라는 상수로 표시되는 OBJ_ATTR_WRAP 속성의 값을 가져옵니다. 래핑 키를 만들려면 genSymKey를 사용하여 AES 키를 생성합니다 (31 입력).

-wk 파라미터를 사용하여 외부 언래핑 키를 지정할 경우, 내보내기 시 -w 래핑 키가 키를 래핑하는 데 사용되고 언래핑에는 사용되지 않습니다.

> **Note**
>
> 키 4는 지원되지 않는 내부 키를 나타냅니다. 래핑 키로 생성하고 관리하는 AES 키를 사용하는 것이 좋습니다.

필수 여부: 예

-out

출력 파일의 경로 및 이름을 지정합니다. 명령이 성공하면 이 파일이 내보낸 키를 평문으로 포함합니다. 파일이 이미 존재하는 경우, 이 명령은 경고 없이 파일에 덮어씁니다.

필수 여부: 예

-m

래핑 메커니즘을 지정합니다. 유일하게 유효한 값은 NIST_AES_WRAP 메커니즘을 나타내는 4입니다.

필수 여부: 아니요

기본값: 4

-wk

지정된 파일의 AES 키를 사용하여 내보내는 키를 언래핑합니다. 평문 AES 키를 포함하는 파일의 경로 및 이름을 입력합니다.

이 파라미터를 포함할 경우, exSymKey는 -w 파라미터로 지정된 HSM의 키를 사용하여 내보낼 키를 래핑하고 -wk 파일의 키를 사용하여 언래핑합니다. -w 및 -wk 파라미터 값은 동일한 평문 키로 확인되어야 합니다.

필수 여부: 아니요

기본값: 래핑 키 또는 HSM을 사용하여 언래핑

관련 주제

- genSymKey
- imSymKey
- wrapKey

extractMaskedObject

key_mgmt_util의 extractMaskedObject 명령은 HSM에서 키를 추출하여 마스킹된 개체로 파일에 저장합니다. 마스크된 개체는 복제된 개체로, insertMaskedObject 명령을 사용하여 원래 클러스터에 다시 삽입한 후에만 사용할 수 있습니다. 마스킹 처리된 객체는 이 객체가 생성된 클러스터나 해당 클러스터의 복제본에만 삽입할 수 있습니다. 여기에는 리전 간에 백업을 복사하고 해당 백업을 사용하여 새 클러스터를 만들어 생성된 클러스터의 복제 버전이 포함됩니다.

마스킹 처리된 객체는 추출할 수 없는 키를 포함하여(즉, OBJ_ATTR_EXTRACTABLE 값이 0인 키) 키를 오프로드하고 동기화하는 효율적인 방법입니다. 이렇게 하면 구성 파일을 업데이트할 필요 없이 여러 지역의 관련 클러스터 간에 키를 안전하게 동기화할 수 있습니다. AWS CloudHSM

> **Important**
>
> 삽입 시 마스킹 처리된 객체가 해독되고 원래 키의 키 핸들과 다른 키 핸들이 제공됩니다. 마스킹 처리된 객체에는 원래 키와 관련된 메타데이터가 모두 포함됩니다(예: 속성, 소유권 및 공유 정보, 쿼럼 설정 등). 애플리케이션의 클러스터 간에 키를 동기화해야 하는 경우 대신 cloudhsm_mgmt_util의 SyncKey를 사용하십시오.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 HSM에 로그인해야 합니다. extractMaskedObject 명령은 키를 소유한 CU나 모든 CO가 사용할 수 있습니다.

구문

```
extractMaskedObject -h

extractMaskedObject -o <object-handle>
```

```
                    -out <object-file>
```

예제

이 예제에서는 extractMaskedObject를 사용하여 HSM에서 키를 마스킹 처리된 객체로 추출하는 방법을 보여 줍니다.

Example : 마스킹 처리된 객체 추출

이 명령은 핸들이 524295인 키의 HSM에서 마스킹 처리된 객체를 추출하고 이를 maskedObj라는 파일로 저장합니다. 명령이 성공하면 extractMaskedObject가 성공 메시지를 반환합니다.

```
Command: extractMaskedObject -o 524295 -out maskedObj

Object was masked and written to file "maskedObj"

        Cfm3ExtractMaskedObject returned: 0x00 : HSM Return: SUCCESS
```

파라미터

이 명령은 다음 파라미터를 사용합니다.

-h

명령에 대한 명령줄 도움말을 표시합니다.

필수 여부: 예

-o

마스킹 처리된 객체로 추출할 키 핸들을 지정합니다.

필수 여부: 예

-out

마스킹 처리된 객체를 저장할 파일 이름을 지정합니다.

필수: 예

관련 주제

- insertMaskedObject
- syncKey

- 리전 간 백업 복사
- 이전 백업에서 AWS CloudHSM 클러스터 생성

findKey

key_mgmt_util에서 findKey 명령을 사용하여 키 속성 값을 기준으로 키를 검색합니다. 키가 설정한 모든 기준과 일치하면 findKey가 키 핸들을 반환합니다. 파라미터가 없으면 findKey는 HSM에서 사용할 수 있는 모든 키의 키 핸들을 반환합니다. 특정 키의 속성 값을 찾으려면 getAttribute를 사용하십시오.

모든 key_mgmt_util 명령과 같이 findKey도 사용자별 명령입니다. 따라서 현재 사용자가 암호화 작업에서 사용할 수 있는 키만 반환합니다. 여기에는 현재 사용자가 소유한 키 및 현재 사용자와 공유된 키가 포함됩니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
findKey -h

findKey [-c <key class>]
        [-t <key type>]
        [-l <key label>]
        [-id <key ID>]
        [-sess (0 | 1)]
        [-u <user-ids>]
        [-m <modulus>]
        [-kcv <key_check_value>]
```

예제

이 예제에서는 findKey를 사용하여 HSM에서 키를 찾고 식별하는 방법을 보여 줍니다.

Example : 모든 키 찾기

이 명령은 HSM에서 현재 사용자의 모든 키를 찾습니다. 출력에는 사용자가 소유 및 공유하는 키와 HSM의 모든 퍼블릭 키가 포함됩니다.

특정 키 핸들이 있는 키의 속성을 가져오려면 getAttribute를 사용합니다. 현재 사용자가 특정 키를 소유하고 있는지 공유하는지 확인하려면 또는 in cloudhsm_mgmt_util을 사용하십시오. getKeyInfofindAllKeys

```
Command: findKey

Total number of keys present 13

 number of keys matched from start index 0::12
6, 7, 524296, 9, 262154, 262155, 262156, 262157, 262158, 262159, 262160, 262161, 262162

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example : 유형, 사용자, 세션 기준으로 키 찾기

이 명령은 현재 사용자와 사용자 3이 사용할 수 있는 영구 AES 키를 찾습니다. (사용자 3은 현재 사용자가 볼 수 없는 다른 키를 사용할 수 있습니다.)

```
Command: findKey -t 31 -sess 0 -u 3
```

Example : 클래스 및 라벨 기준으로 키 찾기

이 명령은 `2018-sept` 라벨이 있는 현재 사용자의 모든 퍼블릭 키를 찾습니다.

```
Command: findKey -c 2 -l 2018-sept
```

Example : 모듈러스를 기준으로 RSA 키 찾기

이 명령은 `m4.txt` 파일에서 모듈러스를 사용하여 생성된 현재 사용자의 RSA 키(유형 0)를 찾습니다.

```
Command: findKey -t 0 -m m4.txt
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-t

지정된 유형의 키를 찾습니다. 키 클래스를 나타내는 상수를 입력합니다. 예를 들어 3DES 키를 찾으려면 `-t 21`을 입력합니다.

유효한 값:

- 0: RSA
- 1: DSA
- 3: EC
- 16: GENERIC_SECRET
- 18: RC4
- 21: Triple DES(3DES)
- 31: AES

필수 여부: 아니요

-c

지정된 클래스에서 키를 찾습니다. 키 클래스를 나타내는 상수를 입력합니다. 예를 들어 퍼블릭 키를 찾으려면 `-c 2`를 입력합니다.

각 키 유형에 유효한 값:

- 2: 퍼블릭. 이 클래스는 퍼블릭-프라이빗 키 페어의 퍼블릭 키를 포함합니다.
- 3: 프라이빗. 이 클래스는 퍼블릭-프라이빗 키 페어의 프라이빗 키를 포함합니다.
- 4: 보안. 이 클래스는 모든 대칭 키를 포함합니다.

필수 여부: 아니요

-l

지정된 라벨을 가진 키를 찾습니다. 정확한 라벨을 입력합니다. `--l` 값에서는 와일드카드 문자나 정규식을 사용할 수 없습니다.

필수 여부: 아니요

-id

지정된 ID를 가진 키를 찾습니다. 정확한 ID 문자열을 입력합니다. `-id` 값에서는 와일드카드 문자나 정규식을 사용할 수 없습니다.

필수 여부: 아니요

-sess

세션 상태를 기준으로 키를 찾습니다. 현재 세션에서만 유효한 키를 찾으려면 1을 입력합니다. 영구 키를 찾으려면 0을 입력합니다.

필수 여부: 아니요

-u

지정된 사용자와 현재 사용자가 공유하는 키를 찾습니다. `-u 3` 또는 `-u 4,7`과 같이 쉼표로 구분된 HSM 사용자 ID 목록을 입력합니다. HSM에 있는 사용자의 ID를 찾으려면 listUsers를 사용합니다.

한 사용자 ID를 지정하면 findKey가 해당 사용자의 키를 반환합니다. 여러 사용자 ID를 지정하면 findKey가 지정된 모든 사용자가 사용할 수 있는 키를 반환합니다.

findKey는 현재 사용자가 사용할 수 있는 키만 반환하므로 `-u` 결과는 항상 현재 사용자의 키와 동일하거나 현재 사용자 키의 하위 집합입니다. 암호화폐 책임자 (CO) 는 cloudhsm_mgmt_util을 사용하여 사용자가 소유하거나 다른 사용자와 공유하는 모든 키를 가져올 수 있습니다. findAllKeys

필수 여부: 아니요

-m

지정된 파일에서 RSA 모듈러스를 사용하여 생성된 키를 찾습니다. 모듈러스를 저장하는 파일의 경로를 입력합니다.

-m은 일치시킬 RSA 모듈러스가 포함된 바이너리 파일을 지정합니다(선택 사항).

필수 여부: 아니요

-kcv

지정된 키 확인 값을 가진 키를 찾습니다.

키 확인 값(KCV)은 HSM이 키를 가져오거나 생성할 때 생성된 키의 3바이트 해시 또는 체크섬입니다. 또한 키를 내보낸 후와 같이 HSM 밖에서 KCV를 계산할 수 있습니다. 그 다음에는 KCV 값을 비교하여 키의 자격 증명 및 무결성을 확인할 수 있습니다. 키의 KCV를 확인하려면 getAttribute를 사용합니다.

AWS CloudHSM 다음과 같은 표준 방법을 사용하여 키 검사 값을 생성합니다.

- 대칭 키: 키를 사용하여 0 블록을 암호화한 결과의 처음 3바이트.
- 비대칭 키 페어: 퍼블릭 키 SHA-1 해시의 처음 3바이트.

- HMAC 키: HMAC 키의 KVC는 현재 지원되지 않습니다.

필수 여부: 아니요

출력

findKey 출력은 일치하는 키 및 키 핸들의 총 수를 나열합니다.

```
        Command:  findKey
Total number of keys present 10

 number of keys matched from start index 0::9
6, 7, 8, 9, 10, 11, 262156, 262157, 262158, 262159

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

관련 주제

- findSingleKey
- getKeyInfo
- getAttribute
- findAllKeyscloudhsm_mgmt_util에서
- 키 속성 참조

findSingleKey

key_mgmt_util 도구의 findSingleKey 명령은 클러스터의 모든 HSM에 키가 존재하는지 확인합니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 암호화 사용자(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
findSingleKey -h
```

```
findSingleKey -k <key-handle>
```

예

Example

이 명령은 키 252136이 클러스터의 3개 HSM 모두에 존재하는지 확인합니다.

```
Command: findSingleKey -k 252136
Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-k

HSM에서 한 키의 키 핸들을 지정합니다. 이 파라미터는 필수 사항입니다.

키 핸들을 찾으려면 findKey 명령을 사용하십시오.

필수: 예

관련 주제

- findKey
- getKeyInfo
- getAttribute

A 세대 KeyPair

key_mgmt_util 도구의 genDSAKeyPair 명령은 HSM에 DSA(디지털 서명 알고리즘) 키 쌍을 생성합니다. 모듈러스 길이를 지정해야 합니다. 이 명령은 모듈러스 값을 생성합니다. 또한 ID를 할당하고, 다른

HSM 사용자와 키를 공유하며, 추출할 수 없는 키를 생성하고, 세션이 끝날 때 만료되는 키를 생성할 수 있습니다. 명령이 성공하면 HSM이 퍼블릭 및 프라이빗 키에 할당하는 키 핸들이 반환됩니다. 키 핸들을 사용하여 다른 명령에 대한 키를 식별할 수 있습니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

> **Tip**
>
> 생성한 키의 속성(예: 형식, 길이, 레이블, ID)을 찾으려면 getAttribute를 사용합니다. 특정 사용자의 키를 찾으려면 를 사용하십시오. getKeyInfo 속성 값에 따라 키를 찾으려면 findKey를 사용합니다.

구문

```
genDSAKeyPair -h

genDSAKeyPair -m <modulus length>
              -l <label>
              [-id <key ID>]
              [-min_srv <minimum number of servers>]
              [-m_value <0..8>]
              [-nex]
              [-sess]
              [-timeout <number of seconds> ]
              [-u <user-ids>]
              [-attest]
```

예제

이 예제에서는 genDSAKeyPair를 사용하여 DSA 키 페어를 생성하는 방법을 보여 줍니다.

Example : DSA 키 페어 생성

이 명령은 라벨이 DSA인 DSA 키 페어를 생성합니다. 출력은 퍼블릭 키의 키 핸들이 19이고 프라이빗 키의 핸들이 21임을 보여 줍니다.

```
Command: genDSAKeyPair -m 2048 -l DSA

        Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS
```

```
        Cfm3GenerateKeyPair:    public key handle: 19    private key handle: 21

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : 세션 전용 DSA 키 페어 생성

이 명령은 현재 세션에서만 유효한 DSA 키 페어를 생성합니다. 이 명령은 필수 (고유하지 않은) 라벨 외에 DSA_temp_pair의 고유한 ID를 할당합니다. 세션 전용 토큰 서명 및 확인을 위해서는 이와 같은 키 페어를 생성하는 것이 좋습니다. 출력은 퍼블릭 키의 키 핸들이 12이고 프라이빗 키의 핸들이 14임을 보여 줍니다.

```
Command: genDSAKeyPair -m 2048 -l DSA-temp -id DSA_temp_pair -sess

        Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 12    private key handle: 14

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

키 쌍이 세션에만 존재하는지 확인하려면 1(참) 값과 함께 findKey의 -sess 매개 변수를 사용합니다.

```
  Command: findKey -sess 1

  Total number of keys present 2

 number of keys matched from start index 0::1
12, 14

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example : 추출할 수 없는 공유 DSA 키 페어 생성

이 명령은 DSA 키 페어를 생성합니다. 프라이빗 키는 세 명의 다른 사용자와 공유되며, HSM에서 내보낼 수 없습니다. 퍼블릭 키는 모든 사용자가 사용할 수 있으며, 항상 추출이 가능합니다.

```
        Command:  genDSAKeyPair -m 2048 -l DSA -id DSA_shared_pair -nex -u 3,5,6
```

```
Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 11    private key handle: 19

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : 쿼럼 제어 키 페어 생성

이 명령은 라벨이 DSA-mV2인 DSA 키 페어를 생성합니다. 이 명령은 -u 파라미터를 사용하여 프라이빗 키를 사용자 4 및 6과 공유합니다. 이 명령은 -m_value 파라미터를 사용하여 프라이빗 키를 사용하는 암호화 작업에 대한 최소 2개의 승인을 쿼럼에 요청합니다. 또 -attest 파라미터를 사용하여 키 페어가 생성되는 펌웨어의 무결성을 확인합니다.

출력은 이 명령이 키 핸들이 12인 퍼블릭 키와 키 핸들이 17인 프라이빗 키를 생성하며, 클러스터 펌웨어에서의 증명 확인에 통과했음을 보여 줍니다.

```
Command:  genDSAKeyPair -m 2048 -l DSA-mV2 -m_value 2 -u 4,6 -attest

Cfm3GenerateKeyPair: returned: 0x00 : HSM Return: SUCCESS

Cfm3GenerateKeyPair:    public key handle: 12    private key handle: 17

Attestation Check : [PASS]

Cluster Error Status
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

이 명령은 개인 키 (키 핸들17) getKeyInfo에 사용됩니다. 출력은 키를 현재 사용자(사용자 3)가 소유하고 있고 사용자 4 및 6(그 외에는 없음)과 공유하고 있음을 확인합니다. 또한 쿼럼 인증이 활성화되어 있고 쿼럼 크기가 2라는 것도 보여 줍니다.

```
Command:  getKeyInfo -k 17

Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

Owned by user 3
```

```
        also, shared to following 2 user(s):

                 4
                 6
         2 Users need to approve to use/manage this key
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-m

모듈러스 길이를 비트 단위로 지정합니다. 유일한 유효 값은 2048입니다.

필수 여부: 예

-l

키 페어에 대해 사용자 정의 레이블을 지정합니다. 문자열을 입력합니다. 같은 레이블이 페어의 두 키에 적용됩니다. label에 허용되는 최대 크기는 127자입니다.

키 식별을 지원하는 임의의 문구를 사용할 수 있습니다. 레이블이 고유할 필요는 없으므로 이를 그룹에 사용하여 키를 분류할 수 있습니다.

필수 여부: 예

-id

키 페어에 대해 사용자 정의 식별자를 지정합니다. 클러스터에 고유한 문자열을 입력합니다. 기본값은 빈 문자열입니다. 사용자가 지정하는 ID는 페어의 두 키에 모두 적용됩니다.

기본값: ID 값이 없음.

필수 여부: 아니요

-min_srv

-timeout 파라미터의 값이 만료되기 전에 키가 동기화되는 HSM의 최소 개수를 지정합니다. 할당된 시간에 지정된 서버 개수에 키가 동기화되지 않으면 키가 생성되지 않습니다.

AWS CloudHSM 클러스터의 모든 HSM에 모든 키를 자동으로 동기화합니다. 이 과정을 더 빠르게 진행하려면 `min_srv`의 값을 클러스터의 HSM 개수보다 적게 설정하고 낮은 제한 시간 값을 설정합니다. 그러나 일부 요청은 키를 생성하지 않을 수 있음을 주의하십시오.

기본값: 1

필수 여부: 아니요

-m_value

페어의 프라이빗 키를 사용하는 모든 암호화 작업을 승인해야 하는 사용자의 수를 지정합니다. `0`에서 8까지의 값을 입력합니다.

이 파라미터는 프라이빗 키에 대해 쿼럼 인증 요구 사항을 설정합니다. 기본값 `0`은 키에 대한 쿼럼 인증 기능을 비활성화합니다. 쿼럼 인증이 활성화되면 지정된 사용자 수는 토큰에 서명하여 그 프라이빗 키를 사용하는 암호화 작업과 그 프라이빗 키를 공유 또는 공유 해제하는 모든 작업을 승인해야 합니다.

키를 찾으려면 `m_value` 를 사용하십시오. getKeyInfo

이 파라미터는 명령의 `-u` 파라미터가 `m_value` 요구 사항을 충족할 만큼 충분히 많은 사용자와 키 페어를 공유할 때만 유효합니다.

기본값: 0

필수 여부: 아니요

-nex

프라이빗 키를 추출할 수 없도록 합니다. 생성된 프라이빗 키는 HSM에서 내보내기를 수행할 수 없습니다. 퍼블릭 키는 항상 추출할 수 있습니다.

기본값: 키 페어의 퍼블릭 및 프라이빗 키를 모두 추출할 수 있습니다.

필수 여부: 아니요

-sess

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

세션 키를 영구적인 (토큰) 키로 변경하려면 setAttribute를 사용합니다.

기본값: 키는 영구적입니다.

필수 여부: 아니요

-timeout

명령이 `min_srv` 파라미터에서 지정한 HSM의 수에 키가 동기화하기를 기다리는 시간(단위: 초)을 지정합니다.

이 파라미터는 명령에서 `min_srv` 파라미터도 사용되는 경우에만 유효합니다.

기본값: 제한 시간 없음. 명령은 무기한 기다리다가 최소 서버 개수에 키가 동기화될 때만 복귀합니다.

필수 여부: 아니요

-u

지정된 사용자와 페어의 프라이빗 키를 공유합니다. 이 파라미터는 다른 HSM CU(Crypto User)에게 암호화 작업에 프라이빗 키를 사용할 수 있는 권한을 부여합니다. 퍼블릭 키는 공유하지 않고도 모든 사용자와 공유할 수 있습니다.

`u 5,6`과 같이 쉼표로 구분된 HSM 사용자 ID 목록을 입력합니다. 현재 사용자의 HSM 사용자 ID는 포함하지 마십시오. HSM에서 CU의 HSM 사용자 ID를 찾으려면 listUsers를 사용하십시오. 기존 키를 공유 및 공유 해제하려면 cloudhsm_mgmt_util에서 shareKey를 사용합니다.

기본값: 현재 사용자만 프라이빗 키를 사용할 수 있습니다.

필수 여부: 아니요

-attest

클러스터가 실행되는 펌웨어가 변조되지 않았는지 확인하는 무결성 점검을 실행합니다.

기본값: 증명 점검은 없음.

필수 여부: 아니요

관련 주제

- GenRSA KeyPair

- genSymKey
- 제네C KeyPair

GenECC KeyPair

key_mgmt_util 도구의 genECCKeyPair 명령은 HSM에서 타원 곡선 암호(ECC) 키 페어를 생성합니다. genECCKeyPair 명령을 실행할 때 타원 곡선 식별자와 키 페어의 레이블을 지정해야 합니다. 또한 다른 CU 사용자와 프라이빗 키를 공유할 수 있으며 추출할 수 없는 키, 쿼럼 제어 키, 세션이 끝날 때 만료되는 키를 생성할 수 있습니다. 명령이 성공하면 HSM이 퍼블릭 및 프라이빗 ECC 키에 할당하는 키 핸들이 반환됩니다. 키 핸들을 사용하여 다른 명령에 대한 키를 식별할 수 있습니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

> **Tip**
>
> 생성한 키의 속성(예: 형식, 길이, 레이블, ID)을 찾으려면 getAttribute를 사용합니다. 특정 사용자의 키를 찾으려면 를 사용하십시오. getKeyInfo 속성 값에 따라 키를 찾으려면 findKey를 사용합니다.

구문

```
genECCKeyPair -h

genECCKeyPair -i <EC curve id>
              -l <label>
              [-id <key ID>]
              [-min_srv <minimum number of servers>]
              [-m_value <0..8>]
              [-nex]
              [-sess]
              [-timeout <number of seconds> ]
              [-u <user-ids>]
              [-attest]
```

예제

다음 예는 genECCKeyPair를 사용하여 HSM에서 ECC 키 페어를 생성하는 방법을 보여줍니다.

Example : ECC 키 페어 생성 및 검사

이 명령은 NID_secp384r1 타원 곡선과 ecc14 레이블을 사용하여 ECC 키 페어를 생성합니다. 출력은 프라이빗 키의 키 핸들이 262177이고 퍼블릭 키의 키 핸들이 262179임을 보여 줍니다. 레이블은 퍼블릭 및 프라이빗 키 모두에 적용됩니다.

```
Command: genECCKeyPair -i 14 -l ecc14

        Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 262179    private key handle: 262177

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

키를 생성한 후 해당 속성을 검사할 수 있습니다. getAttribute를 사용하여 새 ECC 프라이빗 키의 모든 속성(상수 512로 표시)을 attr_262177 파일에 기록합니다.

```
Command: getAttribute -o 262177 -a 512 -out attr_262177
got all attributes of size 529 attr cnt 19
Attributes dumped into attr_262177

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

그런 다음 cat 명령을 사용하여 attr_262177 속성 파일의 내용을 봅니다. 출력은 키가 서명에는 사용할 수 있지만 암호화, 암호 해독, 래핑, 언래핑 또는 확인에는 사용할 수 없는 타원 곡선 프라이빗 키임을 보여 줍니다. 이 키는 영구적이며 내보낼 수 있습니다.

```
$  cat attr_262177

OBJ_ATTR_CLASS
0x03
OBJ_ATTR_KEY_TYPE
0x03
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
OBJ_ATTR_ENCRYPT
0x00
```

```
OBJ_ATTR_DECRYPT
0x00
OBJ_ATTR_WRAP
0x00
OBJ_ATTR_UNWRAP
0x00
OBJ_ATTR_SIGN
0x01
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
ecc2
OBJ_ATTR_ID

OBJ_ATTR_VALUE_LEN
0x0000008a
OBJ_ATTR_KCV
0xbbb32a
OBJ_ATTR_MODULUS
044a0f9d01d10f7437d9fa20995f0cc742552e5ba16d3d7e9a65a33e20ad3e569e68eb62477a9960a87911e6121d112
OBJ_ATTR_MODULUS_BITS
0x0000019f
```

Example 잘못된 EEC 곡선 사용

이 명령은 NID_X9_62_prime192v1 곡선을 사용하여 ECC 키 페어를 생성하려고 시도합니다. 이 타원 곡선은 FIPS 모드 HSM에 유효하지 않으므로 명령이 실패합니다. 메시지는 클러스터에 서버가 없다고 보고하지만, 일반적으로 이는 클러스터의 HSM에 문제가 있음을 나타내는 것은 아닙니다.

```
Command:  genECCKeyPair -i 1 -l ecc1

       Cfm3GenerateKeyPair returned: 0xb3 : HSM Error: This operation violates the
 current configured/FIPS policies

       Cluster Error Status
       Node id 0 and err state 0x30000085 : HSM CLUSTER ERROR: Server in cluster is
 unavailable
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-i

타원 곡선의 식별자를 지정합니다. 식별자를 입력합니다.

유효한 값:

- 2: NID_X9_62_prime256v1
- 14: NID_secp384r1
- 16: NID_secp256k1

필수 여부: 예

-l

키 페어에 대해 사용자 정의 레이블을 지정합니다. 문자열을 입력합니다. 같은 레이블이 페어의 두 키에 적용됩니다. `label`에 허용되는 최대 크기는 127자입니다.

키 식별을 지원하는 임의의 문구를 사용할 수 있습니다. 레이블이 고유할 필요는 없으므로 이를 그룹에 사용하여 키를 분류할 수 있습니다.

필수 여부: 예

-id

키 페어에 대해 사용자 정의 식별자를 지정합니다. 클러스터에 고유한 문자열을 입력합니다. 기본값은 빈 문자열입니다. 사용자가 지정하는 ID는 페어의 두 키에 모두 적용됩니다.

기본값: ID 값이 없음.

필수 여부: 아니요

-min_srv

`-timeout` 파라미터의 값이 만료되기 전에 키가 동기화되는 HSM의 최소 개수를 지정합니다. 할당된 시간에 지정된 서버 개수에 키가 동기화되지 않으면 키가 생성되지 않습니다.

AWS CloudHSM 클러스터의 모든 HSM에 모든 키를 자동으로 동기화합니다. 이 과정을 더 빠르게 진행하려면 `min_srv`의 값을 클러스터의 HSM 개수보다 적게 설정하고 낮은 제한 시간 값을 설정합니다. 그러나 일부 요청은 키를 생성하지 않을 수 있음을 주의하십시오.

기본값: 1

필수 여부: 아니요

-m_value

페어의 프라이빗 키를 사용하는 모든 암호화 작업을 승인해야 하는 사용자의 수를 지정합니다. `0`에서 8까지의 값을 입력합니다.

이 파라미터는 프라이빗 키에 대해 쿼럼 인증 요구 사항을 설정합니다. 기본값 `0`은 키에 대한 쿼럼 인증 기능을 비활성화합니다. 쿼럼 인증이 활성화되면 지정된 사용자 수는 토큰에 서명하여 그 프라이빗 키를 사용하는 암호화 작업과 그 프라이빗 키를 공유 또는 공유 해제하는 모든 작업을 승인해야 합니다.

키를 찾으려면 `m_value` 를 사용하십시오. getKeyInfo

이 파라미터는 명령의 `-u` 파라미터가 `m_value` 요구 사항을 충족할 만큼 충분히 많은 사용자와 키 페어를 공유할 때만 유효합니다.

기본값: 0

필수 여부: 아니요

-nex

프라이빗 키를 추출할 수 없도록 합니다. 생성된 프라이빗 키는 HSM에서 내보내기를 수행할 수 없습니다. 퍼블릭 키는 항상 추출할 수 있습니다.

기본값: 키 페어의 퍼블릭 및 프라이빗 키를 모두 추출할 수 있습니다.

필수 여부: 아니요

-sess

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

세션 키를 영구적인 (토큰) 키로 변경하려면 setAttribute를 사용합니다.

기본값: 키는 영구적입니다.

필수 여부: 아니요

-timeout

명령이 `min_srv` 파라미터에서 지정한 HSM의 수에 키가 동기화하기를 기다리는 시간(단위: 초)을 지정합니다.

이 파라미터는 명령에서 `min_srv` 파라미터도 사용되는 경우에만 유효합니다.

기본값: 제한 시간 없음. 명령은 무기한 기다리다가 최소 서버 개수에 키가 동기화될 때만 복귀합니다.

필수 여부: 아니요

-u

지정된 사용자와 페어의 프라이빗 키를 공유합니다. 이 파라미터는 다른 HSM CU(Crypto User)에게 암호화 작업에 프라이빗 키를 사용할 수 있는 권한을 부여합니다. 퍼블릭 키는 공유하지 않고도 모든 사용자와 공유할 수 있습니다.

`u 5,6`과 같이 쉼표로 구분된 HSM 사용자 ID 목록을 입력합니다. 현재 사용자의 HSM 사용자 ID는 포함하지 마십시오. HSM에서 CU의 HSM 사용자 ID를 찾으려면 listUsers를 사용하십시오. 기존 키를 공유 및 공유 해제하려면 cloudhsm_mgmt_util에서 shareKey를 사용합니다.

기본값: 현재 사용자만 프라이빗 키를 사용할 수 있습니다.

필수 여부: 아니요

-attest

클러스터가 실행되는 펌웨어가 변조되지 않았는지 확인하는 무결성 점검을 실행합니다.

기본값: 증명 점검은 없음.

필수 여부: 아니요

관련 주제

- genSymKey

- 제네-RSA KeyPair
- 젠다 A KeyPair

GenRSA KeyPair

key_mgmt_util 도구의 genRSAKeyPair 명령은 RSA 비대칭 키 페어를 생성합니다. 사용자는 키 유형, 모듈러스 길이 및 퍼블릭 지수를 지정합니다. 이 명령은 지정된 길이의 모듈러스를 생성하고 키 페어를 생성합니다. ID를 할당하고, 다른 HSM 사용자와 키를 공유하고, 추출할 수 없는 키를 생성하고, 세션이 끝날 때 만료되는 키를 생성할 수 있습니다. 명령이 성공하면 HSM이 키에 할당하는 키 핸들이 반환됩니다. 키 핸들을 사용하여 다른 명령에 대한 키를 식별할 수 있습니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

> **Tip**
>
> 생성한 키의 속성(예: 형식, 길이, 레이블, ID)을 찾으려면 getAttribute를 사용합니다. 특정 사용자의 키를 찾으려면 를 사용하십시오. getKeyInfo 속성 값에 따라 키를 찾으려면 findKey를 사용합니다.

구문

```
genRSAKeyPair -h

genRSAKeyPair -m <modulus length>
              -e <public exponent>
              -l <label>
              [-id <key ID>]
              [-min_srv <minimum number of servers>]
              [-m_value <0..8>]
              [-nex]
              [-sess]
              [-timeout <number of seconds> ]
              [-u <user-ids>]
              [-attest]
```

예제

다음 예제는 genRSAKeyPair를 사용하여 HSM에서 비대칭 키 페어를 생성하는 방법을 보여 줍니다.

Example : RSA 키 페어 생성 및 검사

이 명령은 모듈러스가 2048비트이고 지수가 65537인 RSA 키 페어를 생성합니다. 출력은 퍼블릭 키 핸들이 2100177이고 프라이빗 키 핸들이 2100426임을 보여 줍니다.

```
Command: genRSAKeyPair -m 2048 -e 65537 -l rsa_test

Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 2100177    private key handle:
 2100426

        Cluster Status:
        Node id 0 status: 0x00000000 : HSM Return: SUCCESS
        Node id 1 status: 0x00000000 : HSM Return: SUCCESS
```

다음 명령은 getAttribute를 사용하여 방금 생성한 퍼블릭 키의 속성을 가져옵니다. 이 명령은 attr_2100177 파일에 출력을 기록합니다. 이어서 cat 명령이 속성 파일의 내용을 가져옵니다. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

결과 16진수 값은 이 키가 퍼블릭 키(OBJ_ATTR_CLASS 0x02)이고 유형이 RSA(OBJ_ATTR_KEY_TYPE 0x00)임을 확인해 줍니다. 이 퍼블릭 키를 사용하여 암호화(OBJ_ATTR_ENCRYPT 0x01)는 할 수 있지만 암호 해독(OBJ_ATTR_DECRYPT 0x00)은 할 수 없습니다. 결과는 키 길이(512, 0x200), 모듈러스, 모듈러스 길이(2048, 0x800), 퍼블릭 지수(65537, 0x10001)도 포함합니다.

```
Command:  getAttribute -o 2100177 -a 512 -out attr_2100177

Attribute size: 801, count: 26
Written to: attr_2100177 file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS

$  cat attr_2100177
OBJ_ATTR_CLASS
0x02
OBJ_ATTR_KEY_TYPE
0x00
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
```

```
OBJ_ATTR_ENCRYPT
0x01
OBJ_ATTR_DECRYPT
0x00
OBJ_ATTR_WRAP
0x01
OBJ_ATTR_UNWRAP
0x00
OBJ_ATTR_SIGN
0x00
OBJ_ATTR_VERIFY
0x01
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x00
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
rsa_test
OBJ_ATTR_ID

OBJ_ATTR_VALUE_LEN
0x00000200
OBJ_ATTR_KCV
0xc51c18
OBJ_ATTR_MODULUS
0xbb9301cc362c1d9724eb93da8adab0364296bde7124a241087d9436b9be57e4f7780040df03c2c
1c0fe6e3b61aa83c205280119452868f66541bbbffacbbe787b8284fc81deaeef2b8ec0ba25a077d
6983c77a1de7b17cbe8e15b203868704c6452c2810344a7f2736012424cf0703cf15a37183a1d2d0
97240829f8f90b063dd3a41171402b162578d581980976653935431da0c1260bfe756d85dca63857
d9f27a541676cb9c7def0ef6a2a89c9b9304bcac16fdf8183c0a555421f9ad5dfeb534cf26b65873
970cdf1a07484f1c128b53e10209cc6f7ac308669112968c81a5de408e7f644fe58b1a9ae1286fec
b3e4203294a96fae06f8f0db7982cb5d7f
OBJ_ATTR_MODULUS_BITS
0x00000800
OBJ_ATTR_PUBLIC_EXPONENT
0x010001
OBJ_ATTR_TRUSTED
0x00
OBJ_ATTR_WRAP_WITH_TRUSTED
0x00
OBJ_ATTR_DESTROYABLE
0x01
```

```
OBJ_ATTR_DERIVE
0x00
OBJ_ATTR_ALWAYS_SENSITIVE
0x00
OBJ_ATTR_NEVER_EXTRACTABLE
0x00
```

Example : 공유 RSA 키 페어 생성

이 명령은 RSA 키 페어를 생성하고 프라이빗 키를 HSM의 다른 CU인 사용자 4와 공유합니다. 이 명령은 m_value 파라미터를 사용하여 페어의 프라이빗 키를 암호화 작업 사용하기 위해 최소 2개의 승인을 요구합니다. m_value 파라미터를 사용할 때 명령에 -u도 사용해야 합니다. 그래야 m_value가 총 사용자 수(-u 값 + 소유자)를 초과하지 않습니다.

```
Command:  genRSAKeyPair -m 2048 -e 65537 -l rsa_mofn -id rsa_mv2 -u 4 -m_value 2

       Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

       Cfm3GenerateKeyPair:    public key handle: 27    private key handle: 28

       Cluster Error Status
       Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
       Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-m

모듈러스 길이를 비트 단위로 지정합니다. 최소값은 2048입니다.

필수 여부: 예

-e

퍼블릭 지수를 지정합니다. 값은 65537 이상의 홀수여야 합니다.

필수 여부: 예

-l

키 페어에 대해 사용자 정의 레이블을 지정합니다. 문자열을 입력합니다. 같은 레이블이 페어의 두 키에 적용됩니다. `label`에 허용되는 최대 크기는 127자입니다.

키 식별을 지원하는 임의의 문구를 사용할 수 있습니다. 레이블이 고유할 필요는 없으므로 이를 그룹에 사용하여 키를 분류할 수 있습니다.

필수 여부: 예

-id

키 페어에 대해 사용자 정의 식별자를 지정합니다. 클러스터에 고유한 문자열을 입력합니다. 기본값은 빈 문자열입니다. 사용자가 지정하는 ID는 페어의 두 키에 모두 적용됩니다.

기본값: ID 값이 없음.

필수 여부: 아니요

-min_srv

`-timeout` 파라미터의 값이 만료되기 전에 키가 동기화되는 HSM의 최소 개수를 지정합니다. 할당된 시간에 지정된 서버 개수에 키가 동기화되지 않으면 키가 생성되지 않습니다.

AWS CloudHSM 클러스터의 모든 HSM에 모든 키를 자동으로 동기화합니다. 이 과정을 더 빠르게 진행하려면 `min_srv`의 값을 클러스터의 HSM 개수보다 적게 설정하고 낮은 제한 시간 값을 설정합니다. 그러나 일부 요청은 키를 생성하지 않을 수 있음을 주의하십시오.

기본값: 1

필수 여부: 아니요

-m_value

페어의 프라이빗 키를 사용하는 모든 암호화 작업을 승인해야 하는 사용자의 수를 지정합니다. `0`에서 8까지의 값을 입력합니다.

이 파라미터는 프라이빗 키에 대해 쿼럼 인증 요구 사항을 설정합니다. 기본값 `0`은 키에 대한 쿼럼 인증 기능을 비활성화합니다. 쿼럼 인증이 활성화되면 지정된 사용자 수는 토큰에 서명하여 그 프라이빗 키를 사용하는 암호화 작업과 그 프라이빗 키를 공유 또는 공유 해제하는 모든 작업을 승인해야 합니다.

키를 찾으려면 `m_value` 를 사용하십시오. getKeyInfo

이 파라미터는 명령의 -u 파라미터가 m_value 요구 사항을 충족할 만큼 충분히 많은 사용자와 키 페어를 공유할 때만 유효합니다.

기본값: 0

필수 여부: 아니요

-nex

프라이빗 키를 추출할 수 없도록 합니다. 생성된 프라이빗 키는 HSM에서 내보내기를 수행할 수 없습니다. 퍼블릭 키는 항상 추출할 수 있습니다.

기본값: 키 페어의 퍼블릭 및 프라이빗 키를 모두 추출할 수 있습니다.

필수 여부: 아니요

-sess

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

세션 키를 영구적인 (토큰) 키로 변경하려면 setAttribute를 사용합니다.

기본값: 키는 영구적입니다.

필수 여부: 아니요

-timeout

명령이 min_srv 파라미터에서 지정한 HSM의 수에 키가 동기화하기를 기다리는 시간(단위: 초)을 지정합니다.

이 파라미터는 명령에서 min_srv 파라미터도 사용되는 경우에만 유효합니다.

기본값: 제한 시간 없음. 명령은 무기한 기다리다가 최소 서버 개수에 키가 동기화될 때만 복귀합니다.

필수 여부: 아니요

-u

지정된 사용자와 페어의 프라이빗 키를 공유합니다. 이 파라미터는 다른 HSM CU(Crypto User)에게 암호화 작업에 프라이빗 키를 사용할 수 있는 권한을 부여합니다. 퍼블릭 키는 공유하지 않고도 모든 사용자와 공유할 수 있습니다.

u 5,6과 같이 쉼표로 구분된 HSM 사용자 ID 목록을 입력합니다. 현재 사용자의 HSM 사용자 ID는 포함하지 마십시오. HSM에서 CU의 HSM 사용자 ID를 찾으려면 listUsers를 사용하십시오. 기존 키를 공유 및 공유 해제하려면 cloudhsm_mgmt_util에서 shareKey를 사용합니다.

기본값: 현재 사용자만 프라이빗 키를 사용할 수 있습니다.

필수 여부: 아니요

-attest

클러스터가 실행되는 펌웨어가 변조되지 않았는지 확인하는 무결성 점검을 실행합니다.

기본값: 증명 점검은 없음.

필수 여부: 아니요

관련 주제

- genSymKey
- GenDSA KeyPair
- 제네C KeyPair

genSymKey

key_mgmt_util 도구의 genSymKey 명령은 HSM에서 대칭 키를 생성합니다. 키 유형 및 크기를 지정하고, ID 및 레이블을 할당하고, 다른 HSM 사용자와 키를 공유할 수 있습니다. 또한 추출할 수 없는 키와 세션이 끝날 때 만료되는 키를 생성할 수도 있습니다. 명령이 성공하면 HSM이 키에 할당하는 키 핸들이 반환됩니다. 키 핸들을 사용하여 다른 명령에 대한 키를 식별할 수 있습니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
genSymKey -h

genSymKey -t <key-type>
          -s <key-size>
          -l <label>
          [-id <key-ID>]
```

```
        [-min_srv <minimum-number-of-servers>]
        [-m_value <0..8>]
        [-nex]
        [-sess]
        [-timeout <number-of-seconds> ]
        [-u <user-ids>]
        [-attest]
```

예

다음 예제는 genSymKey를 사용하여 HSM에서 대칭 키를 생성하는 방법을 보여 줍니다.

> **Tip**
>
> HMAC 작업을 위해 이러한 예제에서 만든 키를 사용하려면 키를 생성한 후 OBJ_ATTR_SIGN 및 OBJ_ATTR_VERIFY를 TRUE으로 설정해야 합니다. 이러한 값을 설정하려면 CloudHSM 관리 유틸리티(CMU)에서 setAttribute을 사용합니다. 자세한 내용은 setAttribute를 참조하십시오.

Example : AES 키 생성

이 명령은 레이블이 aes256인 256비트 AES 키를 생성합니다. 출력은 새 키의 키 핸들이 6임을 보여 줍니다.

```
Command: genSymKey -t 31 -s 32 -l aes256

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 6

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : 세션 키 생성

이 명령은 현재 세션에서만 유효한 추출할 수 없는 192비트 AES 키를 생성합니다. 키를 래핑(그런 다음 즉시 언래핑)하려면 이와 같은 키를 생성할 수 있습니다.

```
Command: genSymKey -t 31 -s 24 -l tmpAES -id wrap01 -nex -sess
```

Example : 빠르게 반환

이 명령은 라벨이 IT_test_key인 일반 512비트 키를 생성합니다. 이 명령은 키가 클러스터의 모든 HSM과 동기화될 때까지 기다리지 않습니다. 대신, 어느 한 HSM에서 키가 생성되는 시점(-min_srv 1) 또는 1초(-timeout 1) 중 먼저 도래하는 시점에 키를 반환합니다. 키가 시간 제한이 경과할 때까지 지정된 최소 수의 HSM에 동기화되지 않으면 키가 생성되지 않습니다. 다음 예제의 for 루프와 같이, 여러 개의 키를 생성하려면 스크립트에서 이러한 명령을 사용할 수 있습니다.

```
Command: genSymKey -t 16 -s 512 -l IT_test_key -min_srv 1 -timeout 1

$  for i in {1..30};
     do /opt/cloudhsm/bin/key_mgmt_util singlecmd loginHSM -u CU -s example_user -p
 example_pwd genSymKey -l aes -t 31 -s 32 -min_srv 1 -timeout 1;
 done;
```

Example : 쿼럼 인증 일반 키 생성

이 명령은 라벨이 generic-mV2인 2,048비트 일반 보안 키를 생성합니다. 이 명령은 -u 파라미터를 사용하여 키를 다른 CU 사용자 6과 공유합니다. 이 명령은 -m_value 파라미터를 사용하여 키를 사용하는 암호화 작업에 대한 최소 2개의 승인을 쿼럼에 요청합니다. 또 -attest 파라미터를 사용하여 키가 생성되는 펌웨어의 무결성을 확인합니다.

출력은 이 명령이 키 핸들이 9인 키를 생성하고 클러스터 펌웨어에서의 증명 점검에 통과했음을 보여줍니다.

```
                Command:  genSymKey -t 16 -s 2048 -l generic-mV2 -m_value 2 -u 6 -
attest

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 9

        Attestation Check : [PASS]

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : 키 생성 및 검사

명령은 레이블이 3DES_shared이고 ID가 IT-02인 Triple DES 키를 생성합니다. 이 키는 현재 사용자, 사용자 4 및 사용자 5가 사용할 수 있습니다. ID가 클러스터에서 고유하지 않거나 현재 사용자가 사용자 4 또는 5일 경우 명령이 실패합니다.

출력은 새 키의 키 핸들이 7임을 보여줍니다.

```
Command: genSymKey -t 21 -s 24 -l 3DES_shared -id IT-02 -u 4,5

       Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 7

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

새 3DES 키를 현재 사용자가 소유하고 사용자 4 및 5와 공유하는지 확인하려면 getKeyInfo를 사용합니다. 이 명령은 새 키에 할당된 핸들(Key Handle: 7)을 사용합니다.

출력은 이 키를 사용자 3이 소유하고 사용자 4 및 5와 공유하고 있음을 보여 줍니다.

```
Command:  getKeyInfo -k 7

        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

        Owned by user 3

        also, shared to following 2 user(s):

                4, 5
```

키의 다른 속성을 확인하려면 getAttribute를 사용합니다. 첫 번째 명령은 getAttribute를 사용하여 키 핸들 7(-o 7)의 모든 속성(-a 512)을 가져옵니다. 이 명령은 결과를 attr_7 파일에 기록합니다. 두 번째 명령은 cat을 사용하여 attr_7 파일의 내용을 가져옵니다.

이 명령은 키 7이 레이블이 3DES_shared(OBJ_ATTR_LABEL 3DES_shared)이고 ID가 IT_02(OBJ_ATTR_ID IT-02)인 192비트(OBJ_ATTR_VALUE_LEN 0x00000018 또는 24바이트) 3DES(OBJ_ATTR_KEY_TYPE 0x15) 대칭 키(OBJ_ATTR_CLASS 0x04)임을 확인해 줍니다. 키는 영구적이며(OBJ_ATTR_TOKEN 0x01) 추출 가능하며(OBJ_ATTR_EXTRACTABLE 0x01) 암호화, 암호 해독 및 래핑에 사용할 수 있습니다.

> **Tip**
>
> 생성한 키의 속성(예: 형식, 길이, 레이블, ID)을 찾으려면 getAttribute를 사용합니다. 특정 사용자의 키를 찾으려면 를 사용하십시오. getKeyInfo 속성 값에 따라 키를 찾으려면 findKey를 사용합니다.

키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

```
Command:  getAttribute -o 7 -a 512 -out attr_7

got all attributes of size 444 attr cnt 17
Attributes dumped into attr_7 file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS


$  cat attr_7

OBJ_ATTR_CLASS
0x04
OBJ_ATTR_KEY_TYPE
0x15
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
OBJ_ATTR_ENCRYPT
0x01
OBJ_ATTR_DECRYPT
0x01
OBJ_ATTR_WRAP
0x00
OBJ_ATTR_UNWRAP
0x00
OBJ_ATTR_SIGN
0x00
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
```

```
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
3DES_shared
OBJ_ATTR_ID
IT-02
OBJ_ATTR_VALUE_LEN
0x00000018
OBJ_ATTR_KCV
0x59a46e
```

> **Tip**
>
> HMAC 작업을 위해 이러한 예제에서 만든 키를 사용하려면 키를 생성한 후 `OBJ_ATTR_SIGN` 및 `OBJ_ATTR_VERIFY`를 `TRUE`으로 설정해야 합니다. 이러한 값을 설정하려면 CMU에서 setAttribute을 사용하십시오. 자세한 내용은 setAttribute를 참조하십시오.

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-t

대칭 키의 유형을 지정합니다. 키 유형을 나타내는 상수를 입력합니다. 예를 들어 AES 키를 생성하려면 `-t 31`을 입력합니다.

유효한 값:

- 16: GENERIC_SECRET 일반 보안 키는 AES 키 요구 사항과 같은 특정 표준을 준수하지 않는 바이트 어레이입니다.
- 18: RC4 RC4 키는 FIPS 모드 HSM에서 유효하지 않습니다.
- 21: Triple DES(3DES). NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에는 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단 섹션을 참조하십시오.
- 31: AES

필수 여부: 예

-s

키 크기를 바이트 단위로 지정합니다. 예를 들어 192비트 키를 생성하려면 24를 입력합니다.

각 키 유형에 유효한 값:

- AES: 16(128비트), 24(192비트), 32(256비트)
- 3DES: 24(192비트)
- 일반 보안: <3584(28672비트)

필수 여부: 예

-l

키에 대해 사용자 정의 레이블을 지정합니다. 문자열을 입력합니다.

키 식별을 지원하는 임의의 문구를 사용할 수 있습니다. 레이블이 고유할 필요는 없으므로 이를 그룹에 사용하여 키를 분류할 수 있습니다.

필수 여부: 예

-attest

클러스터가 실행되는 펌웨어가 변조되지 않았는지 확인하는 무결성 점검을 실행합니다.

기본값: 증명 점검은 없음.

필수 여부: 아니요

-id

키에 대해 사용자 정의 식별자를 지정합니다. 클러스터에 고유한 문자열을 입력합니다. 기본값은 빈 문자열입니다.

기본값: ID 값이 없음.

필수 여부: 아니요

-min_srv

`-timeout` 파라미터의 값이 만료되기 전에 키가 동기화되는 HSM의 최소 개수를 지정합니다. 할당된 시간에 지정된 서버 개수에 키가 동기화되지 않으면 키가 생성되지 않습니다.

AWS CloudHSM 클러스터의 모든 HSM에 모든 키를 자동으로 동기화합니다. 이 과정을 더 빠르게 진행하려면 `min_srv`의 값을 클러스터의 HSM 개수보다 적게 설정하고 낮은 제한 시간 값을 설정합니다. 그러나 일부 요청은 키를 생성하지 않을 수 있음을 주의하십시오.

기본값: 1

필수 여부: 아니요

-m_value

키를 사용하는 모든 암호화 작업을 승인해야 하는 사용자의 수를 지정합니다. `0`에서 `8`까지의 값을 입력합니다.

이 파라미터는 키에 대해 쿼럼 인증 요구 사항을 설정합니다. 기본값 `0`은 키에 대한 쿼럼 인증 기능을 비활성화합니다. 쿼럼 인증이 활성화되면 지정된 사용자 수는 토큰에 서명하여 그 키를 사용하는 암호화 작업과 그 키를 공유 또는 공유 해제하는 모든 작업을 승인해야 합니다.

키를 찾으려면 `m_value` 를 사용하십시오. getKeyInfo

이 파라미터는 명령의 `-u` 파라미터가 `m_value` 요구 사항을 충족할 만큼 충분히 많은 사용자와 키를 공유할 때만 유효합니다.

기본값: 0

필수 여부: 아니요

-nex

키를 추출할 수 없도록 합니다. 생성된 키는 HSM에서 내보내기를 수행할 수 없습니다.

기본값: 키를 추출할 수 있습니다.

필수 여부: 아니요

-sess

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

세션 키를 영구적인 (토큰) 키로 변경하려면 setAttribute를 사용합니다.

기본값: 키는 영구적입니다.

필수 여부: 아니요

-timeout

명령이 `min_srv` 파라미터에서 지정한 HSM의 수에 키가 동기화하기를 기다리는 시간(단위: 초)을 지정합니다.

이 파라미터는 명령에서 `min_srv` 파라미터도 사용되는 경우에만 유효합니다.

기본값: 제한 시간 없음. 명령은 무기한 기다리다가 최소 서버 개수에 키가 동기화될 때만 복귀합니다.

필수 여부: 아니요

-u

지정된 사용자와 키를 공유합니다. 이 파라미터는 다른 HSM CU(Crypto User)에게 암호화 작업에 이 키를 사용할 수 있는 권한을 부여합니다.

`u 5,6`과 같이 쉼표로 구분된 HSM 사용자 ID 목록을 입력합니다. 현재 사용자의 HSM 사용자 ID는 포함하지 마십시오. HSM에서 CU의 HSM 사용자 ID를 찾으려면 listUsers를 사용하십시오. 기존 키를 공유 및 공유 해제하려면 cloudhsm_mgmt_util에서 shareKey를 사용합니다.

기본값: 현재 사용자만 키를 사용할 수 있습니다.

필수 여부: 아니요

관련 주제

- exSymKey
- GenRSA KeyPair
- 젠다 A KeyPair
- 제네C KeyPair
- setAttribute

getAttribute

key_mgmt_util의 getAttribute 명령은 키의 속성 값 중 하나 또는 전체를 파일에 기록합니다. AWS CloudHSM 지정한 속성이 해당 키 유형에 없는 경우(예: AES 키의 모듈러스), getAttribute가 오류를 반환합니다.

키 속성은 키의 특성입니다. 여기에는 키 유형, 클래스, 레이블, ID 같은 특성과 암호화, 복호화, 래핑, 서명, 확인 등 키를 사용하여 수행할 수 있는 작업을 나타내는 값이 포함됩니다.

사용자가 소유 및 공유하는 키에 대해서만 getAttribute를 사용할 수 있습니다. 이 명령 또는 cloudhsm_mgmt_util에서 getAttribute 명령을 실행할 수 있습니다. 이 명령은 클러스터의 모든 HSM에서 키의 속성 값 하나를 가져와 이를 stdout 또는 파일에 씁니다.

속성과 각 속성을 나타내는 상수의 목록을 가져오려면 listAttributes 명령을 사용합니다. 기존 키의 속성 값을 변경하려면 key_mgmt_util에서 setAttribute를 사용하고 cloudhsm_mgmt_util에서 setAttribute를 사용합니다. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
getAttribute -h

getAttribute -o <key handle>
             -a <attribute constant>
             -out <file>
```

예제

다음 예제에서는 getAttribute를 사용하여 HSM에서 키의 속성을 가져오는 방법을 보여 줍니다.

Example : 키 유형 가져오기

이 예제에서는 AES, 3DES 또는 일반 키(예: RSA 또는 타원 곡선 키 페어)의 유형을 가져옵니다.

첫 번째 명령은 키 속성과 각 속성을 나타내는 상수를 가져오는 listAttributes를 실행합니다. 출력은 키 유형의 상수가 256임을 보여 줍니다. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

```
Command: listAttributes

Description
===========
The following are all of the possible attribute values for getAttributes.

      OBJ_ATTR_CLASS                  = 0
```

```
      OBJ_ATTR_TOKEN                  = 1
      OBJ_ATTR_PRIVATE                = 2
      OBJ_ATTR_LABEL                  = 3
      OBJ_ATTR_KEY_TYPE               = 256
      OBJ_ATTR_ID                     = 258
      OBJ_ATTR_SENSITIVE              = 259
      OBJ_ATTR_ENCRYPT                = 260
      OBJ_ATTR_DECRYPT                = 261
      OBJ_ATTR_WRAP                   = 262
      OBJ_ATTR_UNWRAP                 = 263
      OBJ_ATTR_SIGN                   = 264
      OBJ_ATTR_VERIFY                 = 266
      OBJ_ATTR_LOCAL                  = 355
      OBJ_ATTR_MODULUS                = 288
      OBJ_ATTR_MODULUS_BITS           = 289
      OBJ_ATTR_PUBLIC_EXPONENT        = 290
      OBJ_ATTR_VALUE_LEN              = 353
      OBJ_ATTR_EXTRACTABLE            = 354
      OBJ_ATTR_KCV                    = 371
```

두 번째 명령은 getAttribute를 실행합니다. 이 명령은 키 핸들 524296의 키 유형(속성 256)을 요청하여 `attribute.txt` 파일에 기록합니다.

```
Command: getAttribute -o 524296 -a 256 -out attribute.txt
Attributes dumped into attribute.txt file
```

마지막 명령은 키 파일의 콘텐츠를 가져옵니다. 출력은 키 유형이 `0x15` 또는 21, 즉 Triple DES(3DES) 키라는 것을 보여 줍니다. 클래스 및 유형 값의 정의는 키 속성 참조 섹션을 참조하십시오.

```
$  cat attribute.txt
OBJ_ATTR_KEY_TYPE
0x00000015
```

Example : 키의 모든 속성 가져오기

이 명령은 키 핸들이 6인 키의 모든 속성을 가져와 `attr_6` 파일에 기록합니다. 이 명령은 모든 속성을 나타내는 속성 값 512를 사용합니다.

```
Command: getAttribute -o 6 -a 512 -out attr_6

got all attributes of size 444 attr cnt 17
```

```
Attributes dumped into attribute.txt file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS>
```

이 명령은 모든 속성 값을 포함하는 샘플 속성 파일의 내용을 보여 줍니다. 값 가운데, 키가 ID는 test_01이고 레이블은 aes256인 256비트 AES 키라는 것을 보고합니다. 이 키는 추출 가능하고 지속적입니다. 즉 세션 전용 키가 아닙니다. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

```
$  cat attribute.txt

OBJ_ATTR_CLASS
0x04
OBJ_ATTR_KEY_TYPE
0x15
OBJ_ATTR_TOKEN
0x01
OBJ_ATTR_PRIVATE
0x01
OBJ_ATTR_ENCRYPT
0x01
OBJ_ATTR_DECRYPT
0x01
OBJ_ATTR_WRAP
0x01
OBJ_ATTR_UNWRAP
0x01
OBJ_ATTR_SIGN
0x00
OBJ_ATTR_VERIFY
0x00
OBJ_ATTR_LOCAL
0x01
OBJ_ATTR_SENSITIVE
0x01
OBJ_ATTR_EXTRACTABLE
0x01
OBJ_ATTR_LABEL
aes256
OBJ_ATTR_ID
test_01
OBJ_ATTR_VALUE_LEN
0x00000020
```

```
OBJ_ATTR_KCV
0x1a4b31
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-o

대상 키의 키 핸들을 지정합니다. 각 명령에서 키를 하나만 지정할 수 있습니다. 키의 키 핸들을 가져오려면 findKey를 사용합니다.

또한 사용자가 지정된 키를 소유 또는 공유해야 합니다. 키 사용자를 찾으려면 를 사용하십시오. getKeyInfo

필수 여부: 예

-a

속성을 식별합니다. 모든 속성을 나타내는 상수, 즉 모든 속성을 나타내는 512를 입력합니다. 예를 들어 키 유형을 가져오려면 OBJ_ATTR_KEY_TYPE 속성의 상수인 256을 입력합니다.

속성과 해당 상수를 나열하려면 listAttributes를 사용합니다. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

필수 여부: 예

-out

지정된 파일에 출력을 기록합니다. 파일 경로를 입력합니다. 출력을 stdout에 기록할 수는 없습니다.

지정된 파일이 존재하면 getAttribute가 경고 없이 파일을 덮어씁니다.

필수: 예

관련 주제

- cloudhsm_mgmt_util의 getAttribute

- listAttributes
- setAttribute
- findKey
- 키 속성 참조

getCaviumPriv키

getCaviumPrivkey_mgmt_util의 키 명령은 HSM에서 가짜 PEM 형식으로 개인 키를 내보냅니다. 실제 프라이빗 키 자료를 포함하지 않지만 대신에 HSM의 프라이빗 키를 참조하는 가짜 PEM 파일을 사용하여 웹 서버에서 AWS CloudHSM으로 오프로드하는 SSL/TLS를 설정할 수 있습니다. 자세한 내용은 Linux에서의 SSL/TLS 오프로드를 참조하십시오.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
getCaviumPrivKey -h

getCaviumPrivKey -k <private-key-handle
                 -out <fake-PEM-file>
```

예제

이 예제에서는 getCaviumPrivKey를 사용하여 프라이빗 키를 가짜 PEM 형식으로 내보내는 방법을 보여 줍니다.

Example : 가짜 PEM 파일 내보내기

이 명령은 핸들이 15인 프라이빗 키의 가짜 PEM 버전을 만들어 내보내고 이를 `cavKey.pem`이라는 파일에 저장합니다. 명령이 성공하면 exportPrivateKey가 성공 메시지를 반환합니다.

```
Command: getCaviumPrivKey -k 15 -out cavKey.pem

Private Key Handle is written to cavKey.pem in fake PEM format

        getCaviumPrivKey returned: 0x00 : HSM Return: SUCCESS
```

파라미터

이 명령은 다음 파라미터를 사용합니다.

-h

명령에 대한 명령줄 도움말을 표시합니다.

필수 여부: 예

-k

가짜 PEM 형식으로 내보낼 프라이빗 키의 키 핸들을 지정합니다.

필수 여부: 예

-out

가짜 PEM 키를 쓸 파일 이름을 지정합니다.

필수: 예

관련 주제

- importPrivateKey
- Linux에서의 SSL/TLS 오프로드

getCert

key_mgmt_util의 getCert 명령은 HSM의 파티션 인증서를 검색하여 파일에 저장합니다. 명령을 실행할 때 검색할 인증서의 유형을 지정합니다. 그러려면 다음의 파라미터 단원에서 설명하는 대로 해당하는 정수 중 하나를 사용합니다. 이 인증서의 각 역할을 알아보려면 HSM 자격 증명 확인 섹션을 참조하십시오.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
getCert -h

getCert -f <file-name>
```

```
        -t <certificate-type>
```

예

이 예제에서는 getCert를 사용하여 클러스터의 고객 루트 인증서를 검색하여 파일로 저장하는 방법을 보여줍니다.

Example : 고객 루트 인증서 검색

이 명령은 정수 4로 표시되는 고객 루트 인증서를 내보내고 `userRoot.crt`라는 파일에 저장합니다. 명령이 성공하면 getCert가 성공 메시지를 반환합니다.

```
Command: getCert -f userRoot.crt -s 4

Cfm3GetCert() returned 0 :HSM Return: SUCCESS
```

파라미터

이 명령은 다음 파라미터를 사용합니다.

`-h`

명령에 대한 명령줄 도움말을 표시합니다.

필수 여부: 예

`-f`

검색한 인증서가 저장될 파일의 이름을 지정합니다.

필수 여부: 예

-s

검색할 파티션 인증서의 유형을 지정하는 정수입니다. 정수 및 해당 인증서 유형은 다음과 같습니다.

- 1 – 제조업체 루트 인증서
- 2 – 제조업체 하드웨어 인증서
- 4 – 고객 루트 인증서
- 8 – 클러스터 인증서(고객 루트 인증서로 서명됨)
- 16 – 클러스터 인증서(제조업체 루트 인증서에 묶임)

필수: 예

관련 주제

- HSM 자격 증명 확인
- getCert(cloudhsm_mgmt_util에서)

getKeyInfo

key_mgmt_util의 getKeyInfo 명령은 키를 공유하는 소유자 및 암호화 사용자(Crypto User)를 포함하여 키를 사용할 수 있는 사용자의 HSM 사용자 ID를 반환합니다. 키에서 쿼럼 인증이 활성화되면 getKeyInfo는 키를 사용하는 암호화 작업을 승인해야 하는 사용자의 수도 반환합니다. getKeyInfo는 소유한 키 및 공유된 키에서만 실행할 수 있습니다.

퍼블릭 키에서 getKeyInfo를 실행하면 HSM의 모든 사용자가 퍼블릭 키를 사용할 수 있더라도 getKeyInfo는 키 소유자만 반환합니다. HSM에 있는 사용자의 HSM 사용자 ID를 찾으려면 listUsers를 사용하십시오. 특정 사용자의 키를 찾으려면 findKey -u를 사용합니다.

생성하는 키는 본인의 소유입니다. 키를 생성하면 다른 사용자와 공유할 수 있습니다. 그런 다음 기존 키를 공유하거나 공유 해제하려면 cloudhsm_mgmt_util에서 shareKey를 사용합니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
getKeyInfo -h

getKeyInfo -k <key-handle>
```

예제

이러한 예제에서는 getKeyInfo를 사용하여 키의 사용자에 대한 정보를 가져오는 방법을 보여 줍니다.

Example : 대칭 키의 사용자 가져오기

이 명령은 키 핸들이 9인 AES(대칭) 키를 사용할 수 있는 사용자를 가져옵니다. 출력은 사용자 3이 키를 소유하고 사용자 4와 공유함을 보여 줍니다.

```
Command:  getKeyInfo -k 9

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS
```

```
       Owned by user 3

       also, shared to following 1 user(s):

                4
```

Example : 비대칭 키 페어의 사용자 가져오기

이러한 명령은 getKeyInfo를 사용하여 RSA(비대칭) 키 페어에서 키를 사용할 수 있는 사용자를 가져옵니다. 퍼블릭 키에는 키 핸들 21이 있습니다. 프라이빗 키에는 키 핸들 20이 있습니다.

프라이빗 키(20)에서 getKeyInfo를 실행하면 키 소유자(3)와 키가 공유되는 CU(Crypto User) 4 및 5가 반환됩니다.

```
Command:  getKeyInfo -k 20

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 2 user(s):

                4
                5
```

퍼블릭 키(21)에서 getKeyInfo를 실행하면 키 소유자(3)만 반환됩니다.

```
Command:  getKeyInfo -k 21

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3
```

사용자 4가 퍼블릭 키(및 HSM의 모든 퍼블릭 키)를 사용할 수 있는지 확인하려면 -u에서 findKey의 파라미터를 사용하십시오.

출력은 사용자 4가 키 페어의 퍼블릭 키(21)와 프라이빗 키(20)를 모두 사용할 수 있음을 보여 줍니다. 사용자 4는 다른 모든 퍼블릭 키 및 생성되거나 공유된 어떤 프라이빗 키도 사용할 수 있습니다.

```
Command:  findKey -u 4
```

```
Total number of keys present 8

 number of keys matched from start index 0::7
11, 12, 262159, 262161, 262162, 19, 20, 21

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

Example : 키의 쿼럼 인증 값(m_value) 가져오기

이 예제는 키의 `m_value`, 즉 키를 사용하는 암호화 작업을 승인해야 하는 쿼럼 내 사용자 수를 가져오는 방법을 보여 줍니다.

키에서 쿼럼 인증이 활성화되면 사용자들의 쿼럼은 해당 키를 사용하는 모든 암호화 작업을 승인해야 합니다. 쿼럼 인증을 활성화하고 쿼럼 크기를 설정하려면 키를 생성할 때 `-m_value` 파라미터를 사용합니다.

이 명령은 KeyPairGenRSA를 사용하여 사용자 4와 공유되는 RSA 키 쌍을 생성합니다. 이 명령은 `m_value` 파라미터를 사용하여 키 페어의 프라이빗 키에서 쿼럼 인증을 활성화하고 쿼럼 크기를 2명의 사용자로 설정합니다. 사용자의 수는 필요한 승인을 제공할 수 있을 만큼 커야 합니다.

출력은 이 명령이 퍼블릭 키 27과 프라이빗 키 28을 생성했음을 보여 줍니다.

```
Command:  genRSAKeyPair -m 2048 -e 195193 -l rsa_mofn -id rsa_mv2 -u 4 -m_value 2

        Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

        Cfm3GenerateKeyPair:    public key handle: 27    private key handle: 28

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

이 명령은 getKeyInfo를 사용하여 프라이빗 키의 사용자에 대한 정보를 가져옵니다. 출력은 이 키를 사용자 3이 소유하고 사용자 4와 공유하고 있음을 보여 줍니다. 또한 2명의 사용자로 이루어진 쿼럼이 해당 키를 사용하는 모든 암호화 작업을 승인해야 함을 보여 줍니다.

```
Command:  getKeyInfo -k 28

        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS
```

```
        Owned by user 3

        also, shared to following 1 user(s):

                 4
         2 Users need to approve to use/manage this key
```

파라미터

-h

명령에 대한 명령줄 도움말을 표시합니다.

필수 여부: 예

-k

HSM에서 한 키의 키 핸들을 지정합니다. 소유하거나 공유하는 키의 키 핸들을 입력합니다. 이 파라미터는 필수 사항입니다.

키 핸들을 찾으려면 findKey 명령을 사용하십시오.

필수: 예

관련 주제

- getKeyInfocloudhsm_mgmt_util에서
- listUsers
- findKey
- findAllKeyscloudhsm_mgmt_util에서

도움

key_mgmt_util에서 help 명령은 사용 가능한 모든 key_mgmt_util 명령에 대한 정보를 표시합니다.

help를 실행하기 전에 key_mgmt_util을 시작해야 합니다.

구문

```
help
```

예

이 예제에서는 help 명령의 출력을 보여 줍니다.

Example

```
Command:  help

Help Commands Available:

Syntax: <command> -h


   Command               Description
   =======               ===========

   exit                   Exits this application
   help                   Displays this information

        Configuration and Admin Commands
   getHSMInfo             Gets the HSM Information
   getPartitionInfo       Gets the Partition Information
   listUsers              Lists all users of a partition
   loginStatus            Gets the Login Information
   loginHSM               Login to the HSM
   logoutHSM              Logout from the HSM

        M of N commands
   getToken               Initiate an MxN service and get Token
   delToken               delete Token(s)
   approveToken           Approves an MxN service
   listTokens             List all Tokens in the current partition

        Key Generation Commands

        Asymmetric Keys:
   genRSAKeyPair          Generates an RSA Key Pair
   genDSAKeyPair          Generates a DSA Key Pair
   genECCKeyPair          Generates an ECC Key Pair

        Symmetric Keys:
   genPBEKey              Generates a PBE DES3 key
   genSymKey              Generates a Symmetric keys
```

```
     Key Import/Export Commands
createPublicKey        Creates an RSA public key
importPubKey           Imports RSA/DSA/EC Public key
exportPubKey           Exports RSA/DSA/EC Public key
importPrivateKey       Imports RSA/DSA/EC private key
exportPrivateKey       Exports RSA/DSA/EC private key
imSymKey               Imports a Symmetric key
exSymKey               Exports a Symmetric key
wrapKey                Wraps a key from from HSM using the specified handle
unWrapKey              UnWraps a key into HSM using the specified handle

     Key Management Commands
deleteKey              Delete Key
setAttribute           Sets an attribute of an object
getKeyInfo             Get Key Info about shared users/sessions
findKey                Find Key
findSingleKey          Find single Key
getAttribute           Reads an attribute from an object

     Certificate Setup Commands
getCert                Gets Partition Certificates stored on HSM

     Key Transfer Commands
insertMaskedObject     Inserts a masked object
extractMaskedObject    Extracts a masked object

     Management Crypto Commands
sign                   Generates a signature
verify                 Verifies a signature
aesWrapUnwrap          Does NIST AES Wrap/Unwrap

     Helper Commands
Error2String           Converts Error codes to Strings
                       save key handle in fake PEM format
getCaviumPrivKey       Saves an RSA private key handle
                       in fake PEM format
IsValidKeyHandlefile   Checks if private key file has
                       an HSM key handle or a real key
listAttributes         List all attributes for getAttributes
listECCCurveIds        List HSM supported ECC CurveIds
```

파라미터

이 명령에 대한 파라미터는 없습니다.

관련 주제

- loginHSM 및 logoutHSM

importPrivateKey

key_mgmt_util의 importPrivateKey 명령은 파일에서 HSM으로 비대칭 프라이빗 키를 가져옵니다. HSM에서는 일반 텍스트로 키를 직접 가져올 수 없습니다. 명령은 사용자가 지정한 AES 래핑 키를 사용하여 프라이빗 키를 암호화하고 HSM 내에서 키 래핑을 해제합니다. 키를 인증서와 연결하려는 경우 이 항목을 참조하십시오. AWS CloudHSM

> **Note**
>
> 대칭 키나 프라이빗 키를 사용하여 암호로 보호된 PEM 키를 가져올 수 없습니다.

OBJ_ATTR_UNWRAP과 OBJ_ATTR_ENCRYPT 속성 값 1이 있는 AES 래핑 키를 지정해야 합니다. 키의 속성을 찾으려면 getAttribute 명령을 사용합니다.

> **Note**
>
> 이 명령은 가져온 키를 내보낼 수 없도록 표시하는 옵션을 제공하지 않습니다.

key_mgmt_util 명령을 실행하려면 먼저 key_mgmt_util을 시작하고 HSM에 암호화 사용자(crypto user)로 로그인해야 합니다.

구문

```
importPrivateKey -h

importPrivateKey -l <label>
                 -f <key-file>
                 -w <wrapping-key-handle>
                 [-sess]
                 [-id <key-id>]
                 [-m_value <0...8>]
                 [min_srv <minimum-number-of-servers>]
                 [-timeout <number-of-seconds>]
                 [-u <user-ids>]
```

```
                [-wk <wrapping-key-file>]
                [-attest]
```

예제

이 예제에서는 importPrivateKey를 사용하여 프라이빗 키를 HSM으로 가져오는 방법을 보여 줍니다.

Example : 프라이빗 키 가져오기

이 명령은 `rsa2048.key`라는 파일에서 레이블이 `rsa2048-imported`인 프라이빗 키와 핸들이 524299인 래핑 키를 가져옵니다. 명령이 성공하면 importPrivateKey가 가져온 키의 키 핸들과 성공 메시지를 반환합니다.

```
Command: importPrivateKey -f rsa2048.key -l rsa2048-imported -w 524299

BER encoded key length is 1216

Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

Private Key Unwrapped.  Key Handle: 524301

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

파라미터

이 명령은 다음 파라미터를 사용합니다.

-h

명령에 대한 명령줄 도움말을 표시합니다.

필수 여부: 예

-l

사용자 정의 프라이빗 키 레이블을 지정합니다.

필수 여부: 예

-f

가져올 키의 파일 이름을 지정합니다.

필수 여부: 예

-w

래핑 키의 키 핸들을 지정합니다. 이 파라미터는 필수 사항입니다. 키 핸들을 찾으려면 findKey 명령을 사용합니다.

키를 래핑 키로 사용할 수 있는지 여부를 확인하려면 getAttribute를 사용하여 OBJ_ATTR_WRAP 속성(262)의 값을 가져옵니다. 래핑 키를 생성하려면 genSymKey를 사용하여 AES 키(유형 31)를 생성합니다.

-wk 파라미터를 사용하여 외부 언래핑 키를 지정할 경우, 가져오기 시 -w 래핑 키가 키를 래핑하는 데 사용되고 언래핑에는 사용되지 않습니다.

필수 여부: 예

-sess

가져온 키를 세션 키로 지정합니다.

기본값: 가져온 키는 클러스터에 지속적(토큰) 키로 보관됩니다.

필수 여부: 아니요

-id

가져올 키의 ID를 지정합니다.

기본값: ID 값이 없음.

필수 여부: 아니요

-m_value

가져온 키를 사용하는 모든 암호화 작업을 승인해야 하는 사용자의 수를 지정합니다. **0**에서 **8**까지의 값을 입력합니다.

이 파라미터는 명령의 -u 파라미터가 m_value 요구 사항을 충족할 만큼 충분히 많은 사용자와 키를 공유할 때만 유효합니다.

기본값: 0

필수 여부: 아니요

-min_srv

-timeout 파라미터의 값이 만료되기 전에 가져온 키가 동기화되는 HSM의 최소 개수를 지정합니다. 할당된 시간에 지정된 서버 개수에 키가 동기화되지 않으면 키가 생성되지 않습니다.

AWS CloudHSM 클러스터의 모든 HSM에 모든 키를 자동으로 동기화합니다. 이 과정을 더 빠르게 진행하려면 min_srv의 값을 클러스터의 HSM 개수보다 적게 설정하고 낮은 제한 시간 값을 설정합니다. 그러나 일부 요청은 키를 생성하지 않을 수 있음을 주의하십시오.

기본값: 1

필수 여부: 아니요

-timout

min-serv 파라미터가 포함된 경우 키가 HSM 간에 동기화될 때까지 기다려야 하는 시간(초)을 지정합니다. 숫자를 지정하지 않으면 폴링이 영원히 계속됩니다.

기본값: 제한 없음

필수 여부: 아니요

-u

가져온 프라이빗 키를 공유할 사용자 목록을 지정합니다. 이 파라미터는 다른 HSM CU(Crypto User)에게 암호화 작업에 가져온 키를 사용할 수 있는 권한을 부여합니다.

-u 5,6과 같이 쉼표로 구분된 HSM 사용자 ID 목록을 입력합니다. 현재 사용자의 HSM 사용자 ID는 포함하지 마십시오. HSM에서 CU의 HSM 사용자 ID를 찾으려면 listUsers를 사용하십시오.

기본값: 현재 사용자만 가져온 키를 사용할 수 있습니다.

필수 여부: 아니요

-wk

내보내고 있는 키를 래핑하는 데 사용할 키를 지정합니다. 평문 AES 키를 포함하는 파일의 경로 및 이름을 입력합니다.

이 파라미터를 포함하면 importPrivateKey가 -wk 파일의 키를 사용하여 가져올 키를 래핑합니다. 또한 -w 파라미터로 지정된 키를 사용하여 언래핑합니다.

기본값: -w 파라미터에 지정된 래핑 키를 사용하여 래핑과 언래핑을 모두 합니다.

필수 여부: 아니요

-attest

펌웨어 응답에서 증명 점검을 수행하여 클러스터가 실행되는 펌웨어가 손상되지 않았는지 확인합니다.

필수 여부: 아니요

관련 주제

- wrapKey
- unWrapKey
- genSymKey
- exportPrivateKey

importPubKey

key_mgmt_util에서 importPubKey 명령은 PEM 형식 퍼블릭 키를 HSM으로 가져옵니다. 이 명령을 사용하여 HSM 외부에서 생성된 퍼블릭 키를 가져올 수 있습니다. 명령을 사용하여 HSM에서 내보낸 키(예: 명령으로 내보낸 키) 를 가져올 수도 있습니다. exportPubKey

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 암호화 사용자(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
importPubKey -h

importPubKey -l <label>
             -f <key-file>
             [-sess]
             [-id <key-id>]
             [min_srv <minimum-number-of-servers>]
             [-timeout <number-of-seconds>]
```

예제

이 예제에서는 importPubKey를 사용하여 퍼블릭 키를 HSM으로 가져오는 방법을 보여 줍니다.

Example : 퍼블릭 키 가져오기

이 명령은 public.pem이라는 파일에서 레이블이 importedPublicKey인 퍼블릭 키를 가져옵니다. 명령이 성공하면 importPubKey가 가져온 키의 키 핸들과 성공 메시지를 반환합니다.

```
Command: importPubKey -l importedPublicKey -f public.pem

Cfm3CreatePublicKey returned: 0x00 : HSM Return: SUCCESS

Public Key Handle: 262230

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

파라미터

이 명령은 다음 파라미터를 사용합니다.

-h

명령에 대한 명령줄 도움말을 표시합니다.

필수 여부: 예

-l

사용자 정의 퍼블릭 키 레이블을 지정합니다.

필수 여부: 예

-f

가져올 키의 파일 이름을 지정합니다.

필수 여부: 예

-sess

가져온 키를 세션 키로 지정합니다.

기본값: 가져온 키는 클러스터에 지속적(토큰) 키로 보관됩니다.

필수 여부: 아니요

-id

가져올 키의 ID를 지정합니다.

기본값: ID 값이 없음.

필수 여부: 아니요

-min_srv

`-timeout` 파라미터의 값이 만료되기 전에 가져온 키가 동기화되는 HSM의 최소 개수를 지정합니다. 할당된 시간에 지정된 서버 개수에 키가 동기화되지 않으면 키가 생성되지 않습니다.

AWS CloudHSM 모든 키를 클러스터의 모든 HSM과 자동으로 동기화합니다. 이 과정을 더 빠르게 진행하려면 `min_srv`의 값을 클러스터의 HSM 개수보다 적게 설정하고 낮은 제한 시간 값을 설정합니다. 그러나 일부 요청은 키를 생성하지 않을 수 있음을 주의하십시오.

기본값: 1

필수 여부: 아니요

-timeout

`min-serv` 파라미터가 포함된 경우 키가 HSM 간에 동기화될 때까지 기다려야 하는 시간(초)을 지정합니다. 숫자를 지정하지 않으면 폴링이 영원히 계속됩니다.

기본값: 제한 없음

필수 여부: 아니요

관련 주제

- exportPubKey
- 키 생성

imSymKey

key_mgmt_util 도구의 imSymKey 명령은 파일에서 HSM으로 대칭 키의 일반 텍스트 복사본을 가져옵니다. 이를 사용하여 HSM 외부의 모든 방법으로 생성한 키와 HSM에서 내보낸 키 (예: exSymKey, 명령이 파일에 쓰는 키) 를 가져올 수 있습니다.

가져오기 프로세스 중에 imSymKey는 사용자가 선택하는 AES 키(래핑 키)를 사용하여 가져올 키를 래핑(암호화)한 후 언래핑(암호 해독)합니다. 하지만 imSymKey는 일반 텍스트 키를 포함하는 파일에서만 유효합니다. 암호화된 키를 내보내고 가져오려면 unWrapKeyWrapKey 및 명령을 사용합니다.

또한 imSymKey 명령은 대칭 키만 가져옵니다. 퍼블릭 키를 가져오려면 를 사용하십시오. importPubKey 프라이빗 키를 가져오려면 importPrivateKey또는 WrapKey를 사용하십시오.

> **Note**
>
> 대칭 키나 프라이빗 키를 사용하여 암호로 보호된 PEM 키를 가져올 수 없습니다.

가져온 키는 HSM에서 생성된 키와 거의 비슷하게 동작합니다. 하지만 OBJ_ATTR_LOCAL 속성 값은 0입니다. 이는 객체가 로컬에서 생성되지 않았음을 의미합니다. 대칭 키를 가져올 때 다음 명령을 사용하여 공유할 수 있습니다. 키를 가져온 후 cloudhsm_mgmt_util의 shareKey 명령을 사용하여 키를 공유할 수 있습니다.

```
imSymKey -l aesShared -t 31 -f kms.key -w 3296 -u 5
```

키를 가져온 후, 키 파일을 마킹 또는 삭제해야 합니다. 이 명령은 동일한 키 자료를 여러 번 가져오도록 허용합니다. 키 핸들은 고유하지만 키 자료는 동일한 키가 여러 개 있을 경우 키 자료의 사용을 추적하기 어렵고 암호화 한도를 초과하는 것을 방지할 수 없습니다.

key_mgmt_util 명령을 실행하려면 먼저 key_mgmt_util을 시작하고 HSM에 암호화 사용자(crypto user)로 로그인해야 합니다.

구문

```
imSymKey -h

imSymKey -f <key-file>
         -w <wrapping-key-handle>
         -t <key-type>
         -l <label>
         [-id <key-ID>]
         [-sess]
         [-wk <wrapping-key-file> ]
         [-attest]
         [-min_srv <minimum-number-of-servers>]
         [-timeout <number-of-seconds> ]
         [-u <user-ids>]
```

예제

다음 예제에서는 imSymKey를 사용하여 대칭 키를 HSM으로 가져오는 방법을 보여줍니다.

Example : AES 대칭 키 가져오기

이 예제에서는 imSymKey를 사용하여 AES 대칭 키를 HSM으로 가져옵니다.

첫 번째 명령은 OpenSSL을 사용하여 무작위 256비트 AES 대칭 키를 생성합니다. 이 명령은 `aes256.key` 파일에 키를 저장합니다.

```
$  openssl rand -out aes256-forImport.key 32
```

두 번째 명령은 imSymKey를 사용하여 AES 키를 `aes256.key` 파일에서 HSM으로 가져옵니다. 이 명령은 키 20(HSM의 AES 키)을 래핑 키로 사용하며 `imported` 레이블을 지정합니다. ID와 달리, 레이블은 클러스터에서 고유할 필요는 없습니다. `-t`(유형) 파라미터 값은 AES를 나타내는 `31`입니다.

출력은 파일의 키를 래핑 및 언래핑하고 HSM으로 가져와 키 핸들 262180을 할당한 것을 보여 줍니다.

```
Command:  imSymKey -f aes256.key -w 20 -t 31 -l imported

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Unwrapped.  Key Handle: 262180

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

다음 명령은 getAttribute를 사용하여 새로 가져온 키의 OBJ_ATTR_LOCAL 속성(attribute 355)을 가져와 `attr_262180` 파일에 기록합니다.

```
Command:  getAttribute -o 262180 -a 355 -out attributes/attr_262180
Attributes dumped into attributes/attr_262180_imported file
```

```
        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

속성 파일을 검사할 때 OBJ_ATTR_LOCAL 속성 값이 0인 것을 알 수 있습니다. 이는 키 자료가 HSM에서 생성되지 않았음을 나타냅니다.

```
$  cat attributes/attr_262180_local
OBJ_ATTR_LOCAL
0x00000000
```

Example : 클러스터 간 대칭 키 이동하기

이 예제에서는 클러스터 간에 일반 텍스트 AES 키를 exSymKey사용하고 이동하는 imSymKey 방법을 보여줍니다. 이러한 프로세스를 사용하여 양쪽 클러스터의 HSM에 존재하는 AES 래핑을 생성할 수 있습니다. 공유 래핑 키가 준비되면 unWrapKeyWrapKey를 사용하여 클러스터 간에 암호화된 키를 이동할 수 있습니다.

이 작업을 수행하는 CU 사용자는 양쪽 클러스터의 HSM에 로그인할 수 있는 권한이 있어야 합니다.

첫 번째 명령은 32비트 AES 키인 키 14를 클러스터 1에서 파일로 내보내는 exSymKey데 사용합니다. aes.key 이 명령은 키 6, 즉 클러스터 1의 HSM에 대한 AES 키를 래핑 키로 사용합니다.

```
Command: exSymKey -k 14 -w 6 -out aes.key

        Cfm3WrapKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapHostKey returned: 0x00 : HSM Return: SUCCESS


Wrapped Symmetric Key written to file "aes.key"
```

그런 다음 사용자가 클러스터 2의 key_mgmt_util에 로그인하고 imSymKey 명령을 실행하여 aes.key 파일의 키를 클러스터 2의 HSM으로 가져옵니다. 이 명령은 키 252152, 즉 클러스터 2의 HSM에 대한 AES 키를 래핑 키로 사용합니다.

WARP를 imSymKey 사용하는 래핑 키는 대상 키를 래핑한 후 즉시 래핑을 해제하므로 여러 클러스터의 래핑 키가 동일할 필요는 없습니다. exSymKey

출력은 성공적으로 키를 클러스터 2로 가져와 키 핸들 21을 할당한 것을 보여줍니다.

```
Command:  imSymKey -f aes.key -w 262152 -t 31 -l xcluster
```

```
        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Unwrapped.  Key Handle: 21

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

클러스터 1의 키 14와 클러스터 2의 키 21이 동일한 키 자료를 가지는 것을 증명하려면 각 키의 키 확인 값(KCV)을 가져옵니다. KCV 값이 동일한 경우 키 자료가 동일한 것입니다.

다음 명령은 클러스터 1에서 getAttribute를 사용하여 키 14의 KCV 속성(속성 371) 값을 `attr_14_kcv` 파일에 기록합니다. 그런 다음 cat 명령을 사용하여 `attr_14_kcv` 파일의 내용을 가져옵니다.

```
Command:  getAttribute -o 14 -a 371 -out attr_14_kcv
Attributes dumped into attr_14_kcv file

$  cat attr_14_kcv
OBJ_ATTR_KCV
0xc33cbd
```

이 비슷한 명령은 클러스터 2에서 setAttribute를 사용하여 키 21의 KCV 속성(속성 371) 값을 `attr_21_kcv` 파일에 기록합니다. 그런 다음 cat 명령을 사용하여 `attr_21_kcv` 파일의 내용을 가져옵니다.

```
Command:  getAttribute -o 21 -a 371 -out attr_21_kcv
Attributes dumped into attr_21_kcv file

$  cat attr_21_kcv
OBJ_ATTR_KCV
0xc33cbd
```

출력은 두 키의 KCV 값이 같다는 것을 보여 줍니다. 즉, 키 자료가 동일함을 증명합니다.

같은 키 자료가 양쪽 클러스터의 HSM에 존재하므로 이제 평문 키를 노출시키지 않고 클러스터 사이에서 암호화된 키를 공유할 수 있습니다. 예를 들어 `wrapKey` 명령을 래핑 키 14와 함께 사용하여 암호화

된 키를 클러스터 1에서 내보낸 후, unWrapKey를 래핑 키 21과 함께 암호화된 키를 클러스터 2로 가져올 수 있습니다.

Example : 세션 키 가져오기

이 명령은 imSymKey의 -sess 파라미터를 사용하여 현재 세션에서만 유효한 192비트 Triple DES 키를 가져옵니다.

이 명령은 -f 파라미터를 사용하여 가져올 키를 포함하는 파일을 지정하고, -t 파라미터를 사용하여 키 유형을 지정하고, -w 파라미터를 사용하여 래핑 키를 지정합니다. 이 명령은 -l 파라미터를 사용하여 키를 범주화하는 레이블을 지정하고 -id 파라미터를 사용하여 키의 고유한 표시 ID를 생성합니다. 또한 -attest 파라미터를 사용하여 키를 가져오는 펌웨어를 확인합니다.

출력은 성공적으로 키를 래핑 및 언래핑하고 HSM으로 가져와 키 핸들 37을 할당한 것을 보여 줍니다. 또한 증명 점검을 통과했습니다. 이는 펌웨어가 변조되지 않았음을 나타냅니다.

```
Command:  imSymKey -f 3des192.key -w 6 -t 21 -l temp -id test01 -sess -attest

        Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS

        Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Unwrapped.  Key Handle: 37

        Attestation Check : [PASS]

        Cluster Error Status
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

다음에는 getAttribute 또는 findKey 명령을 사용하여 새로 가져온 키의 속성을 확인할 수 있습니다. 다음 명령은 findKey를 사용하여 키 37에 명령에서 지정한 유형, 레이블 및 ID가 있으며 이 키가 세션 키임을 확인합니다. 출력의 5행에 표시된 대로, findKey는 모든 속성과 일치하는 유일한 키가 키 37임을 보고합니다.

```
Command:  findKey -t 21 -l temp -id test01 -sess 1
Total number of keys present 1

 number of keys matched from start index 0::0
37
```

```
        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

파라미터

-attest

클러스터가 실행되는 펌웨어가 변조되지 않았는지 확인하는 무결성 점검을 실행합니다.

기본값: 증명 점검은 없음.

필수 여부: 아니요

-f

가져올 키를 포함하는 파일을 지정합니다.

파일은 지정된 길이의 AES 또는 Triple DES 키의 평문 사본을 포함해야 합니다. RC4 및 DES 키는 FIPS 모드 HSM에서 유효하지 않습니다.

- AES: 16, 24 또는 32바이트
- Triple DES(3DES): 24바이트

필수 여부: 예

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-id

키에 대해 사용자 정의 식별자를 지정합니다. 클러스터에 고유한 문자열을 입력합니다. 기본값은 빈 문자열입니다.

기본값: ID 값이 없음.

필수 여부: 아니요

-l

키에 대해 사용자 정의 레이블을 지정합니다. 문자열을 입력합니다.

키 식별을 지원하는 임의의 문구를 사용할 수 있습니다. 레이블이 고유할 필요는 없으므로 이를 그룹에 사용하여 키를 분류할 수 있습니다.

필수 여부: 예

-min_srv

`-timeout` 파라미터의 값이 만료되기 전에 키가 동기화되는 HSM의 최소 개수를 지정합니다. 할당된 시간에 지정된 서버 개수에 키가 동기화되지 않으면 키가 생성되지 않습니다.

AWS CloudHSM 클러스터의 모든 HSM에 모든 키를 자동으로 동기화합니다. 이 과정을 더 빠르게 진행하려면 `min_srv`의 값을 클러스터의 HSM 개수보다 적게 설정하고 낮은 제한 시간 값을 설정합니다. 그러나 일부 요청은 키를 생성하지 않을 수 있음을 주의하십시오.

기본값: 1

필수 여부: 아니요

-sess

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

세션 키를 영구적인 (토큰) 키로 변경하려면 setAttribute를 사용합니다.

기본값: 키는 영구적입니다.

필수 여부: 아니요

-timeout

명령이 `min_srv` 파라미터에서 지정한 HSM의 수에 키가 동기화하기를 기다리는 시간(단위: 초)을 지정합니다.

이 파라미터는 명령에서 `min_srv` 파라미터도 사용되는 경우에만 유효합니다.

기본값: 제한 시간 없음. 명령은 무기한 기다리다가 최소 서버 개수에 키가 동기화될 때만 복귀합니다.

필수 여부: 아니요

-t

대칭 키의 유형을 지정합니다. 키 유형을 나타내는 상수를 입력합니다. 예를 들어 AES 키를 생성하려면 `-t 31`을 입력합니다.

유효한 값:

- 21: Triple DES(3DES).
- 31: AES

필수 여부: 예

-u

가져오는 키를 지정된 사용자와 공유합니다. 이 파라미터는 다른 HSM CU(Crypto User)에게 암호화 작업에 이 키를 사용할 수 있는 권한을 부여합니다.

ID 한 개를 입력하거나, `-u 5,6`과 같이 쉼표로 구분된 HSM 사용자 ID 목록을 입력합니다. 현재 사용자의 HSM 사용자 ID는 포함하지 마십시오. ID를 찾으려면 cloudhsm_mgmt_util 명령줄 도구에서 listUsers 명령을 사용하거나, key_mgmt_util 명령줄 도구에서 listUsers 명령을 사용합니다.

필수 여부: 아니요

-w

래핑 키의 키 핸들을 지정합니다. 이 파라미터는 필수 사항입니다. 키 핸들을 찾으려면 findKey 명령을 사용하십시오.

래핑 키는 가져오기 절차 도중 키를 암호화("래핑")한 후 암호 해독("언래핑")하는 데 사용되는 HSM의 키입니다. AES 키만 래핑 키로 사용할 수 있습니다.

임의의 AES 키(모든 크기)를 래핑 키로 사용할 수 있습니다. 래핑 키는 대상 키를 래핑했다 즉시 언래핑하므로 세션 전용 AES 키를 래핑 키로 사용할 수 있습니다. 키를 래핑 키로 사용할 수 있는지 여부를 확인하려면 getAttribute를 사용하여 `OBJ_ATTR_WRAP` 속성(262)의 값을 가져옵니다. 래핑 키를 생성하려면 AES 키 (유형 31) 를 생성하는 genSymKey데 사용합니다.

`-wk` 파라미터를 사용하여 외부 래핑 키를 지정할 경우, `-w` 래핑 키가 가져오는 키를 언래핑하는 데 사용되고 래핑에는 사용되지 않습니다.

> **Note**
>
> 키 4는 지원되지 않는 내부 키입니다. 래핑 키로 생성하고 관리하는 AES 키를 사용하는 것이 좋습니다.

필수 여부: 예

-wk

지정된 파일의 AES 키를 사용하여 가져오는 키를 래핑합니다. 평문 AES 키를 포함하는 파일의 경로 및 이름을 입력합니다.

이 파라미터를 포함할 경우, imSymKey는 -wk 파일의 키를 사용하여 가져올 키를 래핑하고, -w 파라미터로 지정된 HSM의 키를 사용하여 언래핑합니다. -w 및 -wk 파라미터 값은 동일한 평문 키로 확인되어야 합니다.

기본값: 래핑 키 또는 HSM을 사용하여 언래핑

필수 여부: 아니요

관련 주제

- genSymKey
- exSymKey
- wrapKey
- unWrapKey
- exportPrivateKey
- exportPubKey

insertMaskedObject

key_mgmt_util의 insertMaskedObject 명령은 파일의 마스킹된 개체를 지정된 HSM에 삽입합니다. 마스킹된 개체는 extractMaskedObject 명령을 사용하여 HSM에서 추출한 복제된 개체입니다. 이 객체는 원래 클러스터에 다시 삽입한 후에만 사용할 수 있습니다. 마스킹 처리된 객체는 이 객체가 생성된 클러스터나 해당 클러스터의 복제본에만 삽입할 수 있습니다. 여기에는 리전 간에 백업을 복사하고 해당 백업을 사용하여 새 클러스터를 만들어 생성된 원래 클러스터의 복제 버전이 포함됩니다.

마스킹 처리된 객체는 추출할 수 없는 키를 포함하여(즉, OBJ_ATTR_EXTRACTABLE 값이 0인 키) 키를 오프로드하고 동기화하는 효율적인 방법입니다. 이렇게 하면 구성 파일을 업데이트할 필요 없이 여러 지역의 관련 클러스터 간에 키를 안전하게 동기화할 수 있습니다. AWS CloudHSM

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 암호화 사용자(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
insertMaskedObject -h

insertMaskedObject -f <filename>
                   [-min_srv <minimum-number-of-servers>]
                   [-timeout <number-of-seconds>]
```

예제

이 예제는 insertMaskedObject를 사용하여 마스킹 처리된 객체 파일을 HSM에 삽입하는 방법을 보여 줍니다.

Example : 마스킹 처리된 객체 삽입

이 명령은 `maskedObj`라는 파일에서 HSM으로 마스킹 처리된 객체를 삽입합니다. 명령이 성공하면 insertMaskedObject가 마스킹 처리된 객체에서 해독된 키의 키 핸들과 성공 메시지를 반환합니다.

```
Command: insertMaskedObject -f maskedObj

Cfm3InsertMaskedObject returned: 0x00 : HSM Return: SUCCESS
        New Key Handle: 262433

        Cluster Error Status
        Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

파라미터

이 명령은 다음 파라미터를 사용합니다.

`-h`

명령에 대한 명령줄 도움말을 표시합니다.

필수 여부: 예

`-f`

삽입할 마스킹 처리된 객체의 파일 이름을 지정합니다.

필수 여부: 예

-min_srv

-timeout 파라미터의 값이 만료되기 전에 삽입된 마스킹 처리된 객체가 동기화되는 서버의 최소 개수를 지정합니다. 할당된 시간에 지정된 서버 개수에 객체가 동기화되지 않으면 객체가 삽입되지 않습니다.

기본값: 1

필수 여부: 아니요

-timeout

min-serv 파라미터가 포함된 경우 키가 서버 간에 동기화될 때까지 기다려야 하는 시간(초)을 지정합니다. 숫자를 지정하지 않으면 폴링이 영원히 계속됩니다.

기본값: 제한 없음

필수 여부: 아니요

관련 주제

- extractMaskedObject
- syncKey
- 리전 간 백업 복사
- 이전 백업에서 AWS CloudHSM 클러스터 생성

IsValidKeyHandlefile

키 파일에 실제 개인 키가 포함되어 있는지 또는 가짜 RSA PEM 키가 포함되어 있는지 확인하는 데 key_mgmt_util의 IsValidKeyHandlefile 명령이 사용됩니다. 가짜 PEM 파일에는 실제 프라이빗 키 자료가 포함되지 않지만 대신에 HSM의 프라이빗 키를 참조합니다. 파일을 사용하여 웹 서버에서 AWS CloudHSM으로 오프로드하는 SSL/TLS를 설정할 수 있습니다. 자세한 내용은 Linux에서의 SSL/TLS 오프로드를 참조하십시오.

> **Note**
>
> IsValidKeyHandlefile은(는) RSA 키에만 작동합니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
IsValidKeyHandlefile -h

IsValidKeyHandlefile -f <rsa-private-key-file>
```

예제

다음 예제에서는 IsValidKeyHandlefile을 사용하여 지정된 키 파일에 실제 키 자료가 포함되었는지 가짜 PEM 키 자료가 포함되었는지 확인합니다.

Example : 실제 프라이빗 키 확인

이 명령은 `privateKey.pem`이라는 파일에 실제 키 자료가 포함되어 있음을 확인합니다.

```
Command: IsValidKeyHandlefile -f privateKey.pem

Input key file has real private key
```

Example : 가짜 PEM 키 무효화

이 명령은 `caviumKey.pem`이라는 파일에 키 핸들 15에서 만든 가짜 PEM 키 자료가 포함되어 있음을 확인합니다.

```
Command: IsValidKeyHandlefile -f caviumKey.pem

Input file has invalid key handle: 15
```

파라미터

이 명령은 다음 파라미터를 사용합니다.

-h

명령에 대한 명령줄 도움말을 표시합니다.

필수 여부: 예

-f

유효한 키 자료가 있는지 확인할 RSA 개인 키 파일을 지정합니다.

필수: 예

관련 주제

- getCaviumPriv키
- Linux에서의 SSL/TLS 오프로드

listAttributes

listAttributeskey_mgmt_util의 명령은 키의 속성과 이를 나타내는 상수를 나열합니다. AWS CloudHSM 이러한 상수를 사용하여 getAttribute 및 setAttribute 명령에서 속성을 식별합니다. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

이 명령에는 파라미터가 없습니다.

```
listAttributes
```

예

이 명령은 key_mgmt_util에서 가져오고 변경할 수 있는 키 속성과 속성을 나타내는 상수를 나열합니다. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

key_mgmt_util의 getAttribute 명령에서 모든 속성을 나타내려면 512를 사용합니다.

```
Command: listAttributes

Following are the possible attribute values for getAttributes:

      OBJ_ATTR_CLASS                  = 0
      OBJ_ATTR_TOKEN                  = 1
      OBJ_ATTR_PRIVATE                = 2
      OBJ_ATTR_LABEL                  = 3
```

```
      OBJ_ATTR_KEY_TYPE               = 256
      OBJ_ATTR_ENCRYPT                = 260
      OBJ_ATTR_DECRYPT                = 261
      OBJ_ATTR_WRAP                   = 262
      OBJ_ATTR_UNWRAP                 = 263
      OBJ_ATTR_SIGN                   = 264
      OBJ_ATTR_VERIFY                 = 266
      OBJ_ATTR_LOCAL                  = 355
      OBJ_ATTR_MODULUS                = 288
      OBJ_ATTR_MODULUS_BITS           = 289
      OBJ_ATTR_PUBLIC_EXPONENT        = 290
      OBJ_ATTR_VALUE_LEN              = 353
      OBJ_ATTR_EXTRACTABLE            = 354
      OBJ_ATTR_KCV                    = 371
```

관련 주제

- cloudhsm_mgmt_util의 listAttributes
- getAttribute
- setAttribute
- 키 속성 참조

listUsers

key_mgmt_util에서 listUsers 명령은 HSM의 사용자와 함께 사용자 유형 및 기타 속성을 가져옵니다.

key_mgmt_util에서 listUsers는 일관적이지 않더라도 클러스터의 모든 HSM을 나타내는 출력을 반환합니다. 각 HSM의 사용자에 대한 정보를 가져오려면 cloudhsm_mgmt_util에서 listUsers 명령을 사용합니다.

key_mgmt_util 및 의 사용자 명령은 암호화 사용자 (CU) 가 실행할 수 있는 권한을 가진 읽기 전용 명령입니다. listUsers getKeyInfo 나머지 사용자 관리 명령은 cloudhsm_mgmt_util의 일부입니다. 이들 명령은 사용자 관리 권한을 가진 CO(Crypto Officer)가 실행합니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
listUsers
```

```
listUsers -h
```

예

이 명령은 클러스터에 있는 각 HSM의 사용자를 나열하고 그 속성을 표시합니다. 이 User ID 속성을 사용하여 FindKey, GetAttribute 등과 같은 다른 명령에서 사용자를 식별할 수 있습니다. getKeyInfo

```
Command:  listUsers

        Number Of Users found 4

        Index       User ID     User Type       User Name           MofnPubKey
 LoginFailureCnt          2FA
        1                1      PCO             admin                    NO
    0               NO
        2                2      AU              app_user                 NO
    0               NO
        3                3      CU              alice                    YES
    0               NO
        4                4      CU              bob                      NO
    0               NO
        5                5      CU              trent                    YES
    0               NO

        Cfm3ListUsers returned: 0x00 : HSM Return: SUCCESS
```

출력에는 다음의 사용자 속성이 포함됩니다.

- User ID: key_mgmt_util 및 cloudhsm_mgmt_util 명령에서 사용자를 식별합니다.
- User type: 사용자가 HSM에서 수행할 수 있는 작업을 결정합니다.
- User Name: 해당 사용자의 사용자 정의 표시 이름을 표시합니다.
- MofnPubKey: 사용자가 쿼럼 인증 토큰에 서명하기 위해 키 쌍을 등록했는지 여부를 나타냅니다.
- LoginFailureCnt: 사용자가 로그인에 실패한 횟수를 나타냅니다.
- 2FA: 사용자가 멀티 팩터 인증을 활성화했는지 여부를 나타냅니다.

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수: 예

관련 주제

- cloudhsm_mgmt_util의 listUsers
- findKey
- getAttribute
- getKeyInfo

loginHSM 및 logoutHSM

key_mgmt_util에서 loginHSM 및 logoutHSM 명령을 사용하여 클러스터의 HSM에 로그인하거나 로그아웃할 수 있습니다. HSM에 로그인하면 key_mgmt_util을 사용하여 퍼블릭 및 프라이빗 키 생성, 동기화, 래핑을 포함한 다양한 키 관리 작업을 수행할 수 있습니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작해야 합니다. key_mgmt_util로 키를 관리하려면 HSM에 CU(crypto user)로 로그인해야 합니다.

> **Note**
>
> 잘못된 로그인 시도 횟수가 5회를 초과하면 계정이 잠깁니다. 2018년 2월 이전에 클러스터를 만든 경우 로그인 시도가 20회 실패한 후에 계정이 잠깁니다. 계정 잠금을 해제하려면 CO(암호화 관리자)가 cloudhsm_mgmt_util의 changePswd 명령을 사용하여 암호를 재설정해야 합니다.
>
> 클러스터에 HSM이 둘 이상인 경우 계정이 잠기기 전에 잘못된 로그인 시도가 추가로 허용될 수 있습니다. 이는 CloudHSM 클라이언트가 다양한 HSM 간에 로드 균형을 조정하기 때문입니다. 따라서 매번 동일한 HSM에서 로그인 시도가 시작되지 않을 수 있습니다. 이 기능을 테스트하는 경우 활성화된 HSM이 하나 뿐인 클러스터에서 수행하는 것이 좋습니다.

구문

```
loginHSM -h

loginHSM -u <user type>
         { -p | -hpswd } <password>
         -s <username>
```

예

이 예제에서는 `loginHSM` 및 `logoutHSM` 명령을 사용하여 클러스터의 HSM에 로그인 및 로그아웃하는 방법을 보여 줍니다.

Example : HSM에 로그인

이 명령은 사용자 이름 `example_user`와 암호 `aws`를 사용하여 CU(crypto user)로 HSM에 로그인합니다. 출력은 클러스터의 모든 HSM에 로그인했음을 보여줍니다.

```
Command:  loginHSM -u CU -s example_user -p aws

Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : 숨겨진 암호로 로그인

이 명령은 시스템이 암호를 숨기도록 지정한다는 점을 제외하면 위의 예와 동일합니다.

```
Command:  loginHSM -u CU -s example_user -hpswd
```

시스템이 암호를 묻는 메시지를 표시합니다. 암호를 입력하면 시스템은 암호를 숨기며 출력에는 명령이 성공했고 HSM에 연결되었다는 메시지가 표시됩니다.

```
Enter password:

Cfm3LoginHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS

Command:
```

Example : HSM에서 로그아웃

이 명령은 HSM에서 로그아웃합니다. 출력은 클러스터의 모든 HSM에서 로그아웃했음을 보여줍니다.

```
Command: logoutHSM

Cfm3LogoutHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

파라미터

-h

이 명령에 대한 도움말을 표시합니다.

-u

로그인 사용자 유형을 지정합니다. key_mgmt_util을 사용하려면 CU로 로그인해야 합니다.

필수 여부: 예

-s

로그인 사용자 이름을 지정합니다.

필수 여부: 예

{ -p | -hpswd }

-p로 로그인 암호를 지정합니다. 암호를 입력하면 일반 텍스트로 암호가 나타납니다. 암호를 숨기려면 -p 대신 선택적 -hpswd 파라미터를 사용하고 프롬프트를 따릅니다.

필수: 예

관련 주제

- exit

setAttribute

key_mgmt_util에서 setAttribute 명령은 현재 세션에서만 유효한 키를 삭제할 때까지 존재하는 영구 키로 변환합니다. 이 변환은 키의 토큰 속성(OBJ_ATTR_TOKEN) 값을 false(0)에서 true(1)로 변경함으로써 이루어집니다. 본인이 소유한 키의 속성만 변경할 수 있습니다.

cloudhsm_mgmt_util에서 setAttribute 명령을 사용하여 label, wrap, unwrap, encrypt 및 decrypt 속성도 변경할 수 있습니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
setAttribute -h

setAttribute -o <object handle>
             -a 1
```

예

이 예제는 세션 키를 영구 키로 변환하는 방법을 보여 줍니다.

첫 번째 명령은 의 -sess 매개 변수를 사용하여 현재 세션에서만 유효한 192비트 AES 키를 genSymKey만듭니다. 출력은 새 세션 키의 키 핸들이 262154임을 보여 줍니다.

```
Command: genSymKey -t 31 -s 24 -l tmpAES -sess

        Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

        Symmetric Key Created.  Key Handle: 262154

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

이 명령은 findKey를 사용하여 현재 세션에서 세션 키를 찾습니다. 출력은 키 262154가 세션 키임을 확인합니다.

```
Command:  findKey -sess 1

Total number of keys present 1

 number of keys matched from start index 0::0
262154

        Cluster Error Status
```

```
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

이 명령은 setAttribute를 사용하여 키 262154를 세션 키에서 영구 키로 변환합니다. 이를 위해 이 명령은 키의 토큰 속성(OBJ_ATTR_TOKEN) 값을 0(false)에서 1(true)로 변경합니다. 키 속성 해석에 대한 도움말은 키 속성 참조 섹션을 참조하십시오.

이 명령은 -o 파라미터를 사용하여 키 핸들(262154)을 지정하고, -a 파라미터를 사용하여 토큰 속성(1)을 나타내는 상수를 지정합니다. 명령을 실행하면 토큰 속성의 값을 묻는 메시지가 표시됩니다. 유일한 유효 값은 1(true)로서 영구 키의 값입니다.

```
Command: setAttribute -o 262154 -a 1
         This attribute is defined as a boolean value.
          Enter the boolean attribute value (0 or 1):1

        Cfm3SetAttribute returned: 0x00 : HSM Return: SUCCESS

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

이 명령은 키 262154가 이제 영구 키임을 확인하기 위해 findKey를 사용하여 세션 키(-sess 1)와 영구 키(-sess 0)를 찾습니다. 이번에는 이 명령이 세션 키를 찾지 못하지만 영구 키 목록에서 262154를 반환합니다.

```
Command: findKey -sess 1

Total number of keys present 0

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS



Command: findKey -sess 0
```

```
Total number of keys present 5

 number of keys matched from start index 0::4
6, 7, 524296, 9, 262154

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS

        Cfm3FindKey returned: 0x00 : HSM Return: SUCCESS
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-o

대상 키의 키 핸들을 지정합니다. 각 명령에서 키를 하나만 지정할 수 있습니다. 키의 키 핸들을 가져오려면 findKey를 사용합니다.

필수 여부: 예

-a

변경하려는 속성을 나타내는 상수를 지정합니다. 유일한 유효 값은 1로서 토큰 속성 OBJ_ATTR_TOKEN을 나타냅니다.

속성과 해당 정수 값을 가져오려면 listAttributes를 사용합니다.

필수: 예

관련 주제

- cloudhsm_mgmt_util setAttribute
- getAttribute
- listAttributes
- 키 속성 참조

sign

key_mgmt_util의 sign 명령은 선택한 프라이빗 키를 사용하여 파일의 서명을 생성합니다.

sign을 사용하려면 먼저 HSM에 프라이빗 키가 있어야 합니다. genSymKey, genRSAKeyPair 또는 genECCKeyPair 명령을 사용하여 프라이빗 키를 생성할 수 있습니다. 또한 importPrivateKey 명령을 사용하여 프라이빗 키를 가져올 수도 있습니다. 자세한 내용은 키 생성을 참조하십시오.

sign 명령은 정수로 표시된 사용자 지정 서명 메커니즘을 사용하여 메시지 파일에 서명합니다. 가능한 서명 메커니즘 목록은 파라미터를 참조하십시오.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
sign -h

sign -f <file name>
     -k <private key handle>
     -m <signature mechanism>
     -out <signed file name>
```

예

이 예에서는 sign을 사용하여 파일에 서명하는 방법을 보여 줍니다.

Example : 파일 서명

이 명령은 핸들이 266309인 프라이빗 키를 사용하여 messageFile이라는 파일에 서명합니다. SHA256_RSA_PKCS(1) 서명 메커니즘을 사용하고 서명된 결과 파일을 signedFile로 저장합니다.

```
Command: sign -f messageFile -k 266309 -m 1 -out signedFile

Cfm3Sign returned: 0x00 : HSM Return: SUCCESS

signature is written to file signedFile

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
```

```
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

파라미터

이 명령은 다음 파라미터를 사용합니다.

-f

서명할 파일의 이름입니다.

필수 여부: 예

-k

서명에 사용할 프라이빗 키의 핸들입니다.

필수 여부: 예

-m

서명에 사용할 서명 메커니즘을 나타내는 정수입니다. 가능한 메커니즘은 다음 정수에 해당합니다.

서명 메커니즘	해당 정수
SHA1_RSA_PKCS	0
SHA256_RSA_PKCS	1
SHA384_RSA_PKCS	2
SHA512_RSA_PKCS	3
SHA224_RSA_PKCS	4
SHA1_RSA_PKCS_PSS	5
SHA256_RSA_PKCS_PSS	6
SHA384_RSA_PKCS_PSS	7
SHA512_RSA_PKCS_PSS	8
SHA224_RSA_PKCS_PSS	9

서명 메커니즘	해당 정수
ECDSA_SHA1	15
ECDSA_SHA224	16
ECDSA_SHA256	17
ECDSA_SHA384	18
ECDSA_SHA512	19

필수 여부: 예

-out

서명된 파일을 저장할 파일 이름입니다.

필수: 예

관련 주제

- verify
- importPrivateKey
- GenRSA KeyPair
- 제네C KeyPair
- genSymKey
- 키 생성

unWrapKey

key_mgmt_util 도구의 unWrapKey 명령은 래핑(암호화)된 대칭 또는 프라이빗 키를 파일에서 HSM으로 가져옵니다. key_mgmt_util의 WrapKey 명령으로 래핑된 암호화된 키를 가져오도록 설계되었지만 다른 도구로 래핑된 키를 언래핑하는 데에도 사용할 수 있습니다. 하지만 이런 상황에서는 PKCS#11 또는 JCE 소프트웨어 라이브러리를 사용하여 키를 언래핑하는 것이 좋습니다.

가져온 AWS CloudHSM키는 에서 생성한 키와 동일하게 작동합니다. 하지만 OBJ_ATTR_LOCAL 속성 값은 0입니다. 이는 객체가 로컬에서 생성되지 않았음을 의미합니다.

키를 가져온 후 키 파일을 마킹 또는 삭제해야 합니다. 이 명령은 동일한 키 자료를 여러 번 가져오도록 허용합니다. 여러 개의 키가 고유한 키 핸들이고 동일한 키 자료라는 결과는 키 자료의 사용을 추적하기 어렵게 하고 암호화 한도를 초과하는 것을 방지합니다.

key_mgmt_util 명령을 실행하려면 먼저 key_mgmt_util을 시작하고 HSM에 암호화 사용자(crypto user)로 로그인해야 합니다.

구문

```
unWrapKey -h

unWrapKey -f <key-file-name>
          -w <wrapping-key-handle>
          [-sess]
          [-min_srv <minimum-number-of-HSMs>]
          [-timeout <number-of-seconds>]
          [-aad <additional authenticated data filename>]
          [-tag_size <tag size>]
          [-iv_file <IV file>]
          [-attest]
          [-m <wrapping-mechanism>]
          [-t <hash-type>]
          [-nex]
          [-u <user id list>]
          [-m_value <number of users needed for approval>]
          [-noheader]
          [-l <key-label>]
          [-id <key-id>]
          [-kt <key-type>]
          [-kc <key-class]
          [-i <unwrapping-IV>]
```

예

이 예제에서는 unWrapKey를 사용하여 파일에서 HSM으로 래핑된 키를 가져오는 방법을 보여줍니다. 첫 번째 예시에서는 wrapKey key_mgmt_util 명령으로 래핑되어 헤더가 있는 키를 언래핑합니다. 두 번째 예시에서는 key_mgmt_util 외부에서 래핑되어 헤더가 없는 키를 언래핑합니다.

Example : 키 언래핑하기(헤더 있음)

이 명령은 3DES 대칭 키의 래핑된 복사본을 HSM으로 가져옵니다. 키는 레이블이 6인 AES 키로 언래핑됩니다. 3DES 키를 래핑하는 데 사용한 것과 암호화 방식에 있어 동일합니다. 출력은 파일의 키를 언래핑하고 가져왔으며 가져온 키의 핸들이 29임을 보여줍니다.

```
        Command:  unWrapKey -f 3DES.key -w 6 -m 4

        Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS

        Key Unwrapped.  Key Handle: 29

        Cluster Error Status
        Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
        Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

Example : 키 언래핑하기(헤더 없음)

이 명령은 3DES 대칭 키의 래핑된 복사본을 HSM으로 가져옵니다. 키는 레이블이 6인 AES 키로 언래핑됩니다. 3DES 키를 래핑하는 데 사용한 것과 암호화 방식에 있어 동일합니다. 이 3DES 키는 key_mgmt_util로 래핑되지 않았으므로 함께 제공되는 필수 파라미터인 키 레이블(unwrapped3DES), 키 클래스(4) 및 키 유형(21)과 함께 noheader 파라미터가 지정됩니다. 출력은 파일의 키를 언래핑하고 가져왔으며 가져온 키의 핸들이 8임을 보여줍니다.

```
Command: unWrapKey -f 3DES.key -w 6 -noheader -l unwrapped3DES -kc 4 -kt 21 -m 4

      Cfm3CreateUnwrapTemplate2 returned: 0x00 : HSM Return: SUCCESS
       Cfm2UnWrapWithTemplate3 returned: 0x00 : HSM Return: SUCCESS

       Key Unwrapped. Key Handle: 8

       Cluster Error Status
       Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
       Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-f

래핑된 키를 포함하는 파일의 경로 및 이름입니다.

필수 여부: 예

-w

래핑 키를 지정합니다. HSM에서 AES 키 또는 RSA 키의 키 핸들을 입력합니다. 이 파라미터는 필수 사항입니다. 키 핸들을 찾으려면 findKey 명령을 사용하십시오.

래핑 키를 생성하려면 AES 키 (유형 31) 를 생성하거나 KeyPairGenRSA를 사용하여 genSymKeyRSA 키 쌍 (유형 0) 을 생성합니다. RSA 키 페어를 사용하는 경우 키 중 하나로 키를 래핑하고 다른 키로 언래핑해야 합니다. 키를 래핑 키로 사용할 수 있는지 확인하려면 getAttribute를 사용하여 262라는 상수로 표시되는 OBJ_ATTR_WRAP 속성의 값을 가져옵니다.

필수 여부: 예

-sess

현재 세션에만 존재하는 키를 생성합니다. 세션이 종료된 후에는 키를 복구할 수 없습니다.

다른 키를 암호화한 후 다시 재빨리 암호를 해독하는 래핑 키와 같이 키가 일시적으로 필요한 경우 이 파라미터를 사용합니다. 세션 종료 후에 암호를 해독해야 할 수 있는 데이터를 암호화하는 데 세션 키를 사용해서는 안 됩니다.

세션 키를 영구적인 (토큰) 키로 변경하려면 setAttribute를 사용합니다.

기본값: 키는 영구적입니다.

필수 여부: 아니요

-min_srv

-timeout 파라미터의 값이 만료되기 전에 키가 동기화되는 HSM의 최소 개수를 지정합니다. 할당된 시간에 지정된 서버 개수에 키가 동기화되지 않으면 키가 생성되지 않습니다.

AWS CloudHSM 모든 키를 클러스터의 모든 HSM과 자동으로 동기화합니다. 이 과정을 더 빠르게 진행하려면 min_srv의 값을 클러스터의 HSM 개수보다 적게 설정하고 낮은 제한 시간 값을 설정합니다. 그러나 일부 요청은 키를 생성하지 않을 수 있음을 주의하십시오.

기본값: 1

필수 여부: 아니요

-timeout

명령이 `min_srv` 파라미터에서 지정한 HSM의 수에 키가 동기화하기를 기다리는 시간(단위: 초)을 지정합니다.

이 파라미터는 명령에서 `min_srv` 파라미터도 사용되는 경우에만 유효합니다.

기본값: 제한 시간 없음. 명령은 무기한 기다리다가 최소 서버 개수에 키가 동기화될 때만 복귀합니다.

필수 여부: 아니요

-attest

클러스터가 실행되는 펌웨어가 변조되지 않았는지 확인하는 무결성 점검을 실행합니다.

기본값: 증명 점검은 없음.

필수 여부: 아니요

-nex

키를 추출할 수 없도록 합니다. 생성된 키는 HSM에서 내보내기를 수행할 수 없습니다.

기본값: 키를 추출할 수 있습니다.

필수 여부: 아니요

-m

래핑 메커니즘을 나타내는 값입니다. CloudHSM은 다음과 같은 메커니즘을 지원합니다.

메커니즘	값
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7

메커니즘	값
RSA_OAEP(최대 데이터 크기는 이 단원 뒷부분의 참고 참조)	8
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS (최대 데이터 크기는 이 섹션 뒷부분의 참고 사항을 참조하십시오). 예정된 변경 사항은 1 아래 참고를 참조하세요.	12

필수 여부: 예

> **Note**
>
> RSA_OAEP래핑 메커니즘을 사용할 때 래핑할 수 있는 최대 키 크기는 다음과 같이 RSA 키의 모듈러스와 지정된 해시의 길이에 따라 결정됩니다. 최대 키 크기 = modulusLengthIn 바이트- (2* Bytes) -2. hashLengthIn
> RSA_PKCS 래핑 메커니즘을 사용하는 경우 래핑할 수 있는 최대 키 크기는 다음과 같이 RSA 키의 모듈러스에 의해 결정됩니다. 최대 키 크기 = (modulusLengthIn바이트 -11).

-t

해시 알고리즘	값
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224(RSA_AES 및 RSA_OAEP 메커니즘에 대해 유효함)	6

필수 여부: 아니요

-noheader

key_mgmt_util 외부에서 래핑된 키를 언래핑하는 경우 이 파라미터 및 연관된 다른 모든 파라미터를 지정해야 합니다.

필수 여부: 아니요

> **Note**
>
> 이 파라미터를 지정할 경우 반드시 다음의 `-noheader` 파라미터도 지정해야 합니다.

- -l

  언래핑된 키에 추가할 레이블을 지정합니다.

  필수 여부: 예

- -kc

  언래핑할 키의 클래스를 지정합니다. 사용할 수 있는 값은 다음과 같습니다.

  3 = 퍼블릭-프라이빗 키 페어에서 프라이빗 키

  4 = 보안(대칭) 키

  필수 여부: 예

- -kt

  언래핑할 키의 유형을 지정합니다. 사용할 수 있는 값은 다음과 같습니다.

  0 = RSA

  1 = DSA

  3 = ECC

  16 = GENERIC_SECRET

  21 = DES3

  31 = AES

필수 여부: 예

선택적으로 다음 -noheader 파라미터를 지정할 수도 있습니다.

- -id

  언래핑된 키에 추가할 ID입니다.

  필수 여부: 아니요

- -i

  사용할 언래핑 IV(초기화 벡터)입니다.

  필수 여부: 아니요

[1] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에서는 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단를 참조하세요.

관련 주제

- wrapKey
- exSymKey
- imSymKey

verify

key_mgmt_util의 verify 명령은 파일이 지정된 키로 서명되었는지 여부를 확인합니다. 그렇게 하기 위해 verify 명령은 서명된 파일을 소스 파일과 비교하고 지정된 퍼블릭 키와 서명 메커니즘에 기반하여 두 파일이 암호적으로 관련되어 있는지 분석합니다. 작업을 통해 파일에 로그인할 수 있습니다. AWS CloudHSM sign

서명 메커니즘은 파라미터 단원에 나열된 정수로 표시됩니다.

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 CU(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
verify -h
```

```
verify -f <message-file>
       -s <signature-file>
       -k <public-key-handle>
       -m <signature-mechanism>
```

예

다음 예에서는 verify를 사용하여 지정된 파일에 서명할 때 특정 퍼블릭 키가 사용되었는지 여부를 확인하는 방법을 보여 줍니다.

Example : 파일 서명 확인

이 명령은 hardwareCertSigned 서명된 파일을 생성하기 위해 SHA256_RSA_PKCS 서명 메커니즘을 사용하여 퍼블릭 키 262276으로 hardwarCert.crt라는 파일에 서명했는지 여부를 확인하려고 시도합니다. 지정된 파라미터가 참 서명 관계를 나타내므로, 이 명령은 성공 메시지를 반환합니다.

```
Command: verify -f hardwareCert.crt -s hardwareCertSigned -k 262276 -m 1

Signature verification successful

Cfm3Verify returned: 0x00 : HSM Return: SUCCESS
```

Example : 거짓 서명 관계 입증

이 명령은 userCertSigned 서명된 파일을 생성하기 위해 SHA256_RSA_PKCS 서명 메커니즘을 사용하여 퍼블릭 키 262276으로 hardwareCert.crt라는 파일에 서명했는지 여부를 확인합니다. 지정된 파라미터가 참 서명 관계를 구성하지 않으므로, 이 명령은 오류 메시지를 반환합니다.

```
Command: verify -f hardwarecert.crt -s usercertsigned -k 262276 -m 1
Cfm3Verify returned: 0x1b

CSP Error: ERR_BAD_PKCS_DATA
```

파라미터

이 명령은 다음 파라미터를 사용합니다.

-f

원본 메시지 파일의 이름입니다.

필수 여부: 예

-s

서명된 파일의 이름입니다.

필수 항목 여부: 예

-k

파일에 서명할 때 사용하려고 하는 퍼블릭 키의 핸들입니다.

필수 여부: 예

-m

파일에 서명할 때 사용되는 제안된 서명 메커니즘을 나타내는 정수입니다. 가능한 메커니즘은 다음 정수에 해당합니다.

서명 메커니즘	해당 정수
SHA1_RSA_PKCS	0
SHA256_RSA_PKCS	1
SHA384_RSA_PKCS	2
SHA512_RSA_PKCS	3
SHA224_RSA_PKCS	4
SHA1_RSA_PKCS_PSS	5
SHA256_RSA_PKCS_PSS	6
SHA384_RSA_PKCS_PSS	7
SHA512_RSA_PKCS_PSS	8
SHA224_RSA_PKCS_PSS	9
ECDSA_SHA1	15
ECDSA_SHA224	16

서명 메커니즘	해당 정수
ECDSA_SHA256	17
ECDSA_SHA384	18
ECDSA_SHA512	19

필수: 예

관련 주제

- sign
- getCert
- 키 생성

wrapKey

key_mgmt_util에서 wrapKey 명령은 대칭 또는 프라이빗 키의 암호화된 복사본을 HSM에서 파일로 내보냅니다. wrapKey를 실행할 때 내보낼 키, 내보내려는 키를 암호화(래핑)할 HSM의 키, 출력 파일을 지정합니다.

wrapKey 명령은 지정한 파일에 암호화된 키를 작성하지만, HSM에서 키를 제거하거나 암호화 작업에서 사용하지 못하게 할 수는 없습니다. 동일한 키를 여러 번 내보낼 수 있습니다.

키 소유자, 즉 키를 생성한 CU(Crypto User)만 키를 내보낼 수 있습니다. 키를 공유하는 사용자는 해당 키를 암호화 작업에 사용할 수 있지만 내보낼 수는 없습니다.

암호화된 키를 HSM으로 다시 가져오려면 를 사용하십시오. unWrapKey HSM에서 일반 텍스트 키를 내보내려면 필요에 따라 또는 를 사용하십시오 exSymKey. exportPrivateKey 이 aesWrapUnwrap명령은 암호화하는 키를 복호화 (언래핑) 할 수 없습니다. wrapKey

key_mgmt_util 명령을 실행하기 전에 key_mgmt_util을 시작하고 암호화 사용자(Crypto User)로 HSM에 로그인해야 합니다.

구문

```
wrapKey -h
```

```
wrapKey -k <exported-key-handle>
        -w <wrapping-key-handle>
        -out <output-file>
        [-m <wrapping-mechanism>]
        [-aad <additional authenticated data filename>]
        [-t <hash-type>]
        [-noheader]
        [-i <wrapping IV>]
        [-iv_file <IV file>]
        [-tag_size <num_tag_bytes>>]
```

예

Example

이 명령은 192비트 Triple DES(3DES) 대칭 키(키 핸들 7)를 내보냅니다. 이 명령은 HSM의 256 비트 AES 키(키 핸들 14)를 사용하여 키 7을 래핑합니다. 그런 다음 암호화된 3DES 키를 `3DES-encrypted.key` 파일에 씁니다.

출력은 키 7(3DES 키)가 성공적으로 래핑되고 지정된 파일에 기록된 것을 보여 줍니다. 암호화된 키는 길이가 307바이트입니다.

```
        Command:  wrapKey -k 7 -w 14 -out 3DES-encrypted.key -m 4

        Key Wrapped.

        Wrapped Key written to file "3DES-encrypted.key length 307

        Cfm2WrapKey returned: 0x00 : HSM Return: SUCCESS
```

파라미터

-h

명령에 대한 도움말을 표시합니다.

필수 여부: 예

-k

내보낼 키의 키 핸들. 소유하는 대칭 또는 프라이빗 키의 키 핸들을 입력합니다. 키 핸들을 찾으려면 findKey 명령을 사용하십시오.

키를 내보낼 수 있는지 확인하려면 getAttribute 명령을 사용하여 상수 354로 표시되는 OBJ_ATTR_EXTRACTABLE 속성의 값을 가져옵니다. 키 속성 해석에 대한 도움말은 키 속성 참조을 참조하십시오.

본인이 소유한 키만 내보낼 수 있습니다. 키 소유자를 찾으려면 명령을 사용합니다. getKeyInfo

필수 여부: 예

-w

래핑 키를 지정합니다. HSM에서 AES 키 또는 RSA 키의 키 핸들을 입력합니다. 이 파라미터는 필수 사항입니다. 키 핸들을 찾으려면 findKey 명령을 사용하십시오.

래핑 키를 생성하려면 AES 키 (유형 31) 를 생성하거나 KeyPairGenRSA를 사용하여 genSymKeyRSA 키 쌍 (유형 0) 을 생성합니다. RSA 키 페어를 사용하는 경우 키 중 하나로 키를 래핑하고 다른 키로 언래핑해야 합니다. 키를 래핑 키로 사용할 수 있는지 확인하려면 getAttribute를 사용하여 262라는 상수로 표시되는 OBJ_ATTR_WRAP 속성의 값을 가져옵니다.

필수 여부: 예

-out

출력 파일의 경로 및 이름입니다. 명령이 성공하면 이 파일이 내보낸 키의 암호화된 사본을 포함합니다. 파일이 이미 존재하는 경우, 이 명령은 경고 없이 파일에 덮어씁니다.

필수 여부: 예

-m

래핑 메커니즘을 나타내는 값입니다. CloudHSM은 다음과 같은 메커니즘을 지원합니다.

메커니즘	값
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP(최대 데이터 크기는 이 단원 뒷부분의 참고 참조)	8

메커니즘	값
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS (최대 데이터 크기는 이 섹션 뒷부분의 참고 사항을 참조하십시오). 예정된 변경 사항은 1 아래 참고를 참조하세요.	12

필수 여부: 예

> **Note**
>
> RSA_OAEP래핑 메커니즘을 사용할 때 래핑할 수 있는 최대 키 크기는 다음과 같이 RSA 키의 모듈러스와 지정된 해시의 길이에 따라 결정됩니다. 최대 키 크기 = (바이트-2* 바이트-2). modulusLengthIn hashLengthIn
> RSA_PKCS 래핑 메커니즘을 사용하는 경우 래핑할 수 있는 최대 키 크기는 다음과 같이 RSA 키의 모듈러스에 의해 결정됩니다. 최대 키 크기 = (modulusLengthIn바이트 -11).

-t

해시 알고리즘을 나타내는 값입니다. CloudHSM은 다음 알고리즘을 지원합니다.

해시 알고리즘	값
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224(RSA_AES 및 RSA_OAEP 메커니즘에 대해 유효함)	6

필수 여부: 아니요

-aad

AAD를 포함하는 파일 이름입니다.

> **Note**
>
> AES_GCM 및 CLOUDHSM_AES_GCM 메커니즘에만 유효합니다.

필수 여부: 아니요

-noheader

CloudHSM 관련 키 속성을 지정하는 헤더를 생략합니다. key_mgmt_util 외부의 도구를 사용하여 키의 래핑을 해제하려는 경우에만 이 파라미터를 사용하십시오.

필수 여부: 아니요

-i

초기화 벡터(IV)(16진수 값) 입니다.

> **Note**
>
> CLOUDHSM_AES_KEY_WRAP 및 NIST_AES_WRAP 메커니즘에 대한 -noheader 파라미터와 함께 전달된 경우에만 유효합니다.

필수 여부: 아니요

-iv_file

응답으로 얻은 IV 값을 쓰려는 파일입니다.

> **Note**
>
> AES_GCM 메커니즘에 대한 -noheader 파라미터와 함께 전달된 경우에만 유효합니다.

필수 여부: 아니요

-tag_size

래핑된 blob와 함께 저장할 태그의 크기입니다.

> **Note**
>
> AES_GCM 및 CLOUDHSM_AES_GCM 메커니즘에 대한 -noheader 파라미터와 함께 전달된 경우에만 유효합니다. 최소 태그 크기는 8입니다.

필수 여부: 아니요

[1] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에서는 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단를 참조하세요.

관련 주제

- exSymKey
- imSymKey
- unWrapKey

키 속성 참조

key_mgmt_util 명령은 HSM의 키 속성을 표현하기 위해 상수를 사용합니다. 이 주제는 속성을 식별하고, 명령에서 속성을 나타내는 상수를 찾고, 그 값을 이해하는 데 도움이 됩니다.

키를 만들 때 키의 속성을 설정합니다. 키가 지속적인지 아니면 세션에만 존재하는지를 나타내는 토큰 속성을 변경하려면 key_mgmt_util의 setAttribute 명령을 사용합니다. 레이블, 래핑, 언래핑, 암호화 또는 해독 속성을 변경하려면 cloudhsm_mgmt_util의 setAttribute 명령을 사용합니다.

속성과 해당 상수의 목록을 가져오려면 listAttributes를 사용합니다. 키의 속성 값을 가져오려면 getAttribute를 사용합니다.

다음 표에는 키 속성, 속성의 상수 및 유효한 값이 나와 있습니다.

속성	상수	값
OBJ_ATTR_ALL	512	모든 속성을 나타냅니다.

속성	상수	값
OBJ_ATTR_ALWAYS_SENSITIVE	357	0: False 1: True.
OBJ_ATTR_CLASS	0	2: 퍼블릭-프라이빗 키 쌍의 퍼블릭 키 3: 퍼블릭-프라이빗 키 쌍의 프라이빗 키 4: 보안(대칭) 키
OBJ_ATTR_DECRYPT	261	0: False 1: True. 키를 사용하여 데이터 암호를 해독할 수 있습니다.
OBJ_ATTR_DERIVE	268	0: False 1: True. 이 함수는 키를 도출합니다.
OBJ_ATTR_DESTROYABLE	370	0: False 1: True.
OBJ_ATTR_ENCRYPT	260	0: False 1: True. 키를 사용하여 데이터를 암호화할 수 있습니다.
OBJ_ATTR_EXTRACTABLE	354	0: False 1: True. 키를 HSM에서 내보낼 수 있습니다.
OBJ_ATTR_ID	258	사용자 정의 문자열. 클러스터에서 고유해야 합니다. 기본값은 빈 문자열입니다.

속성	상수	값
OBJ_ATTR_KCV	371	키의 키 확인 값입니다. 자세한 내용은 추가 세부 정보를 참조하십시오.
OBJ_ATTR_KEY_TYPE	256	0: RSA 1: DSA 3: EC 16: 일반 보안 18: RC4 21: Triple DES(3DES) 31: AES
OBJ_ATTR_LABEL	3	사용자 정의 문자열. 클러스터에서 고유할 필요는 없습니다.
OBJ_ATTR_LOCAL	355	0: False. HSM으로 가져온 키입니다. 1: True.
OBJ_ATTR_MODULUS	288	RSA 키 페어를 생성하는 데 사용된 모듈러스입니다. EC 키의 경우 이 값은 ANSI X9.62 ECPoint 값 “Q”의 DER 인코딩을 16진수 형식으로 나타냅니다. 다른 키 유형의 경우, 이 속성이 존재하지 않습니다.

속성	상수	값
OBJ_ATTR_MODULUS_BITS	289	RSA 키 페어를 생성하는 데 사용된 모듈러스의 길이입니다. EC 키의 경우 이는 키를 생성하는 데 사용된 타원 곡선의 ID를 나타냅니다. 다른 키 유형의 경우, 이 속성이 존재하지 않습니다.
OBJ_ATTR_NEVER_EXTRACTABLE	356	0: False 1: True. HSM에서 키를 내보낼 수 없습니다.
OBJ_ATTR_PUBLIC_EXPONENT	290	RSA 키 페어를 생성하는 데 사용된 퍼블릭 지수입니다. 다른 키 유형의 경우, 이 속성이 존재하지 않습니다.
OBJ_ATTR_PRIVATE	2	0: False 1: True. 이 속성은 인증되지 않은 사용자가 키 속성을 목록에 추가할 수 있는지 여부를 나타냅니다. CloudHSM PKCS#11 공급자가 현재 퍼블릭 세션을 지원하지 않기 때문에 모든 키(퍼블릭-프라이빗 키 페어의 퍼블릭 키를 포함)는 속성이 1로 설정되어 있습니다.
OBJ_ATTR_SENSITIVE	259	0: False. 퍼블릭-프라이빗 키 쌍의 퍼블릭 키. 1: True.

속성	상수	값
OBJ_ATTR_SIGN	264	0: False 1: True. 키를 서명(프라이빗 키)에 사용할 수 있습니다.
OBJ_ATTR_TOKEN	1	0: False 세션 키 1: True. 영구 키
OBJ_ATTR_TRUSTED	134	0: False 1: True.
OBJ_ATTR_UNWRAP	263	0: False 1: True. 키를 사용하여 키 암호를 해독할 수 있습니다.
OBJ_ATTR_UNWRAP_TEMPLATE	1073742354	값은 이 래핑 키를 사용하여 언래핑된 모든 키에 적용되는 속성 템플릿을 사용해야 합니다.
OBJ_ATTR_VALUE_LEN	353	키 길이(바이트)
OBJ_ATTR_VERIFY	266	0: False 1: True. 키를 확인(퍼블릭 키)에 사용할 수 있습니다.
OBJ_ATTR_WRAP	262	0: False 1: True. 키를 사용하여 키를 암호화할 수 있습니다.
OBJ_ATTR_WRAP_TEMPLATE	1073742353	값은 이 래핑 키를 사용하여 래핑된 키와 일치하도록 속성 템플릿을 사용해야 합니다.

속성	상수	값
OBJ_ATTR_WRAP_WITH_TRUSTED	528	0: False 1: True.

추가 세부 정보

키 확인 값 (KCV)

키 확인 값(KCV)은 HSM이 키를 가져오거나 생성할 때 생성된 키의 3바이트 해시 또는 체크섬입니다. 또한 키를 내보낸 후와 같이 HSM 밖에서 KCV를 계산할 수 있습니다. 그 다음에는 KCV 값을 비교하여 키의 자격 증명 및 무결성을 확인할 수 있습니다. 키의 KCV를 확인하려면 getAttribute를 사용합니다.

AWS CloudHSM 다음과 같은 표준 방법을 사용하여 키 검사 값을 생성합니다.

- 대칭 키: 키를 사용하여 0 블록을 암호화한 결과의 처음 3바이트.
- 비대칭 키 페어: 퍼블릭 키 SHA-1 해시의 처음 3바이트.
- HMAC 키: HMAC 키의 KVC는 현재 지원되지 않습니다.

AWS CloudHSM 클라이언트 SDK

클라이언트 SDK를 사용하면 플랫폼 또는 언어 기반 애플리케이션에서 하드웨어 보안 모듈(HSM) 으로 암호화 작업을 오프로드할 수 있습니다.

AWS CloudHSM 두 가지 주요 버전을 제공하며 클라이언트 SDK 5는 최신 버전입니다. 클라이언트 SDK 3(이전 시리즈) 에 비해 다양한 이점을 제공합니다. 자세한 내용은 클라이언트 SDK의 이점 5를 참조하세요. 지원되는 플랫폼에 대한 자세한 내용은 Client SDK 5 지원 플랫폼 섹션을 참조하세요.

Client SDK 3 사용에 대한 자세한 내용은 이전 클라이언트 SDK (클라이언트 SDK 3) 단원을 참조하십시오.

the section called “PKCS #11 라이브러리”

PKCS #11 는 하드웨어 보안 모듈(HSM) 에서 암호화 작업을 수행하기 위한 표준입니다. AWS CloudHSM PKCS #11 버전 2.40과 호환되는 PKCS #11 라이브러리 구현을 제공합니다.

the section called “OpenSSL Dynamic Engine”

AWS CloudHSM OpenSSL 동적 엔진을 사용하면 OpenSSL API를 통해 암호화 작업을 CloudHSM 클러스터로 오프로드할 수 있습니다.

the section called “JCE 공급자”

AWS CloudHSM JCE 공급자는 자바 암호화 아키텍처 (JCA) 를 준수합니다. 공급자를 통해 HSM에서 암호화 작업을 수행할 수 있습니다.

the section called “KSP 및 CNG 공급자”

Windows용 AWS CloudHSM 클라이언트에는 CNG 및 KSP 공급자가 포함됩니다. 현재는 클라이언트 SDK 3만 CNG 및 KSP 공급자를 지원합니다.

Client SDK 5 지원 플랫폼

기본 지원은 클라이언트 AWS CloudHSM SDK의 각 버전마다 다릅니다. SDK의 구성 요소에 대한 플랫폼 지원은 일반적으로 기본 지원과 일치하지만 항상 그런 것은 아닙니다. 특정 구성 요소에 대한 플랫폼 지원을 확인하려면 먼저 원하는 플랫폼이 SDK의 기본 섹션에 표시되는지 확인한 다음 구성 요소 섹션에서 제외 사항이나 기타 관련 정보가 있는지 확인하세요.

AWS CloudHSM 64비트 운영 체제만 지원합니다.

플랫폼 지원은 시간이 지남에 따라 변경됩니다. 이전 버전의 CloudHSM 클라이언트 SDK는 여기에 나열된 모든 운영 체제를 지원하지 않을 수 있습니다. 릴리스 노트를 사용하여 이전 버전의 CloudHSM 클라이언트 SDK에 대한 운영 체제 지원을 확인할 수 있습니다. 자세한 내용은 AWS CloudHSM 클라이언트 SDK용 다운로드 단원을 참조하십시오.

이전 Client SDK가 지원되는 플랫폼은 Client SDK 3 지원 플랫폼 단원을 참조하십시오.

클라이언트 SDK 5에는 클라이언트 대몬이 필요하지 않습니다.

Client SDK 5에 대한 리눅스 지원

지원하는 플랫폼	X86_64 아키텍처	ARM 아키텍처
Amazon Linux 2	예	예
Amazon Linux 2023	예	예
센토스 7 (7.8+)	예	아니요
레드햇 엔터프라이즈 리눅스 7 (7.8+)	예	아니요
레드햇 엔터프라이즈 리눅스 8 (8.3+)	예	아니요
레드햇 엔터프라이즈 리눅스 9 (9.2+)	예	예
Ubuntu 20.04 LTS	예	아니요
Ubuntu 22.04 LTS	예	예

참고: SDK 5.4.2는 CentOS 8 플랫폼 지원을 제공하는 마지막 릴리스입니다. 자세한 내용은 CentOS 웹 사이트를 참조하세요.

Client SDK 5에 대한 윈도우 지원

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Client SDK 5에 대한 서버리스 지원

- Lambda
- Docker/ECS

Client SDK 5의 HSM 호환성

hsm1.medium	hsm2m.medium
클라이언트 SDK 버전 5.0.0 이상과 호환됩니다.	클라이언트 SDK 버전 5.12.0 이상과 호환됩니다.

구성 요소 지원

CloudHSM CLI

CloudHSM CLI는 관리자가 클러스터의 사용자를 관리하는 데 도움이 되는 명령줄 도구입니다. 자세한 정보는 CloudHSM 명령줄 인터페이스(CLI)을 참조하세요.

PKCS #11 라이브러리

PKCS #11 라이브러리는 Linux 및 Windows Client SDK 5 기본 지원과 일치하는 크로스 플랫폼 구성 요소입니다. 자세한 정보는 the section called “Client SDK 5에 대한 리눅스 지원” 및 the section called “Client SDK 5에 대한 윈도우 지원” 단원을 참조하십시오.

OpenSSL Dynamic Engine

OpenSSL 다이내믹 엔진은 OpenSSL 1.0.2, 1.1.1 또는 3.x가 필요한 리눅스 전용 구성 요소입니다.

JCE 공급자

JCE 제공자는 지원되는 모든 플랫폼에서 OpenJDK 8, OpenJDK 11, OpenJDK 17 및 OpenJDK 21과 호환되는 자바 SDK입니다.

Client SDK 5의 이점

Client SDK 3에 비해 Client SDK 5는 관리가 더 쉽고, 뛰어난 구성 기능과 향상된 안정성을 제공합니다. Client SDK 5는 Client SDK 3에 몇 가지 추가 주요 이점도 제공합니다.

서버리스 아키텍처용으로 설계됨

Client SDK 5에는 클라이언트 데몬이 필요하지 않으므로 더 이상 백그라운드 서비스를 관리할 필요가 없습니다. 이는 다음과 같은 몇 가지 중요한 측면에서 사용자에게 도움이 됩니다.

- 애플리케이션 시작 프로세스를 간소화합니다. CloudHSM을 시작하기 위해 해야 할 일은 애플리케이션을 실행하기 전에 SDK를 구성하는 것뿐입니다.
- 지속적으로 실행되는 프로세스가 필요하지 않으므로 Lambda 및 Elastic Container Service(ECS)와 같은 서버리스 구성 요소와의 통합이 더 쉬워집니다.

더 나은 타사 통합 및 더 쉬운 이식성

Client SDK 5는 JCE 사양을 밀접하게 따르며 다양한 JCE 공급자 간의 더 쉬운 이식성과 더 나은 타사 통합을 제공합니다

사용자 경험 및 구성 가능성 개선

Client SDK 5는 로그 메시지 가독성을 향상시키고 보다 명확한 예외 및 오류 처리 메커니즘을 제공하므로 사용자가 셀프 서비스 분류를 훨씬 쉽게 할 수 있습니다. 또한 SDK 5는 구성 도구 페이지에 나열된 다양한 구성을 제공합니다.

더 광범위한 플랫폼 지원

Client SDK 5는 최신 운영 플랫폼에 대한 추가 지원을 제공합니다. 이것은 ARM 기술에 대한 지원과 JCE, PKCS #11 및 OpenSSL에 대한 더 많은 지원이 포함됩니다. 자세한 내용은 지원된 플랫폼을 참조하십시오.

추가 기능 및 메커니즘

Client SDK 5에는 Client SDK 3에서 사용할 수 없는 추가 기능 및 메커니즘이 포함되어 있으며, Client SDK 5에는 앞으로 더 많은 메커니즘이 추가될 예정입니다.

클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션

클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하는 방법에 대한 자세한 지침은 각 개별 클라이언트 SDK의 마이그레이션 지침을 참조하십시오.

- PKCS #11 라이브러리를 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하십시오.

- OpenSSL 동적 엔진을 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하십시오.
- JCE 제공자를 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션
- 클라이언트 SDK 3 CMU 및 KMU에서 클라이언트 SDK 5 CloudHSM CLI로 마이그레이션

CloudHSM CLI에서 지원하지 않는 기능이나 사용 사례에 대해서는 지원팀에 문의하세요.

> **Note**
>
> 클라이언트 SDK 5 PKCS #11 라이브러리는 이제 Windows 플랫폼에서 지원됩니다. CNG 및 KSP 공급자가 대체할 수 있고 대체자로 간주되어야 하는 대부분의 사용 사례를 처리할 수 있습니다. KSP는 현재 클라이언트 SDK 3에서만 사용할 수 있습니다.

PKCS #11 라이브러리

PKCS #11 는 하드웨어 보안 모듈(HSM) 에서 암호화 작업을 수행하기 위한 표준입니다. AWS CloudHSM PKCS #11 버전 2.40과 호환되는 PKCS #11 라이브러리 구현을 제공합니다.

부트스트래핑에 대한 자세한 내용은 클러스터에 연결 단원을 참조하십시오. 문제 PKCS#11 SDK의 알려진 문제 해결에 대한 내용은 을 참조하십시오.

Client SDK 3 사용에 대한 자세한 내용은 이전 클라이언트 SDK (클라이언트 SDK 3) 단원을 참조하십시오.

주제

- 클라이언트 SDK 5용 PKCS #11 라이브러리 설치
- PKCS #11 라이브러리에 인증하십시오.
- PKCS #11 라이브러리에 지원되는 키 유형
- PKCS #11 라이브러리에 지원되는 메커니즘
- PKCS #11 라이브러리에 지원되는 API 작업
- PKCS #11 라이브러리에 지원되는 주요 속성
- PKCS #11 라이브러리의 코드 샘플
- PKCS #11 라이브러리를 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하십시오.
- PKCS #11의 고급 구성

클라이언트 SDK 5용 PKCS #11 라이브러리 설치

이 항목에서는 git 클라이언트 SDK 5 버전 시리즈용 PKCS #11 라이브러리의 최신 버전을 설치하기 위한 지침을 제공합니다. 클라이언트 SDK 또는 PKCS #11 라이브러리에 대한 자세한 내용은 클라이언트 SDK 및 PKCS #11 라이브러리 사용을 참조하십시오.

설치

클라이언트 SDK 5를 사용하면 클라이언트 데몬을 설치하거나 실행할 필요가 없습니다.

클라이언트 SDK 5를 사용하여 단일 HSM 클러스터를 실행하려면 먼저 `disable_key_availability_check`을 `True`로 설정하여 클라이언트 키 내구성 설정을 관리해야 합니다. 자세한 내용은 키 동기화 및 클라이언트 SDK 5 구성 도구를 참조하십시오.

클라이언트 SDK 5의 PKCS #11 라이브러리에 대한 자세한 내용은 PKCS #11 라이브러리를 참조하십시오.

> **Note**
>
> 클라이언트 SDK 5를 사용하여 단일 HSM 클러스터를 실행하려면 먼저 `disable_key_availability_check`을 `True`로 설정하여 클라이언트 키 내구성 설정을 관리해야 합니다. 자세한 내용은 키 동기화 및 클라이언트 SDK 5 구성 도구를 참조하십시오.

PKCS #11 라이브러리를 설치하고 구성하려면

1. 다음 명령을 사용하여 PKCS #11 라이브러리를 다운로드하고 설치합니다.

   Amazon Linux 2

   Amazon Linux 2용 PKCS #11 라이브러리를 X86_64 아키텍처에 설치합니다.

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-pkcs11-latest.el7.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-pkcs11-latest.el7.x86_64.rpm
   ```

   Amazon Linux 2용 PKCS #11 라이브러리를 ARM64 아키텍처에 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-pkcs11-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.aarch64.rpm
```

Amazon Linux 2023

X86_64 아키텍처에 아마존 리눅스 2023용 PKCS #11 라이브러리를 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-pkcs11-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.amzn2023.x86_64.rpm
```

ARM64 아키텍처에 아마존 리눅스 2023용 PKCS #11 라이브러리를 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-pkcs11-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.amzn2023.aarch64.rpm
```

CentOS 7 (7.8+)

CentOS 7.8+용 PKCS #11 라이브러리를 X86_64 아키텍처에 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

RHEL 7 (7.8+)

X86_64 아키텍처에 RHEL 7용 PKCS #11 라이브러리를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

X86_64 아키텍처에 RHEL 8용 PKCS #11 라이브러리를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el8.x86_64.rpm
```

RHEL 9 (9.2+)

X86_64 아키텍처에 RHEL 9용 PKCS #11 라이브러리를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-pkcs11-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el9.x86_64.rpm
```

ARM64 아키텍처에 RHEL 9용 PKCS #11 라이브러리를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-pkcs11-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-pkcs11-latest.el9.aarch64.rpm
```

Ubuntu 20.04 LTS

Ubuntu 20.04 LTS용 PKCS #11 라이브러리를 X86_64 아키텍처에 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-pkcs11_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u20.04_amd64.deb
```

Ubuntu 22.04 LTS

Ubuntu 22.04 LTS용 PKCS #11 라이브러리를 X86_64 아키텍처에 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-pkcs11_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u22.04_amd64.deb
```

ARM64 아키텍처에 우분투 22.04 LTS용 PKCS #11 라이브러리를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-pkcs11_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-pkcs11_latest_u22.04_arm64.deb
```

Windows Server 2016

Windows Server 2016용 PKCS #11 라이브러리를 X86_64 아키텍처에 설치합니다.

1. 클라이언트 SDK 5용 PKCS #11 라이브러리를 다운로드합니다.
2. Windows 관리자 권한으로 PKCS #11 라이브러리 설치 프로그램 (11-latest.msi) 을 실행합니다. AWSCloudHSMPKCS

Windows Server 2019

Windows Server 2019용 PKCS #11 라이브러리를 X86_64 아키텍처에 설치합니다.

1. 클라이언트 SDK 5용 PKCS #11 라이브러리를 다운로드합니다.
2. Windows 관리 권한으로 PKCS #11 라이브러리 설치 프로그램 (AWSCloudHSMPKCS11-latest.msi) 을 실행합니다.

2. 구성 도구를 사용하여 발급 인증서의 위치를 지정합니다. 지침은 인증서의 위치를 지정합니다. 섹션을 참조하십시오.
3. 클러스터에 연결하려면 Client SDK 부트스트랩합니다. 섹션을 참조하십시오.
4. 다음 위치에서 PKCS #11 라이브러리 파일을 찾을 수 있습니다.

- Linux 바이너리, 구성 스크립트 및 로그 파일:

```
/opt/cloudhsm
```

Windows 바이너리:

```
C:\ProgramFiles\Amazon\CloudHSM
```

Windows 구성 스크립트 및 로그 파일:

```
C:\ProgramData\Amazon\CloudHSM
```

PKCS #11 라이브러리에 인증하십시오.

PKCS #11 라이브러리를 사용하면 애플리케이션이 HSM에서 특정 CU(Crypto User)로 실행됩니다. 애플리케이션은 CU가 소유하고 공유하는 키만 보고 관리할 수 있습니다. HSM에서 기존 CU를 사용하거나, 애플리케이션을 위한 CU를 새로 만들 수 있습니다. CU 관리에 대한 자세한 내용은 CloudHSM CLI를 사용한 HSM 사용자 관리 및 CloudHSM 관리 유틸리티(CMU)를 사용한 HSM 사용자 관리를 참조하십시오.

CU를 PKCS #11 라이브러리에 지정하려면 PKCS #11 C_Login 함수의 핀 파라미터를 사용하십시오. 의 AWS CloudHSM경우 핀 매개변수의 형식은 다음과 같습니다.

```
<CU_user_name>:<password>
```

예를 들어, 다음 명령은 PKCS #11 라이브러리 핀을 사용자 이름 `CryptoUser` 및 `CUPassword123!` 암호를 사용하여 CU에 설정합니다.

```
CryptoUser:CUPassword123!
```

PKCS #11 라이브러리에 지원되는 키 유형

PKCS #11 라이브러리는 다음과 같은 키 유형을 지원합니다.

키 유형	설명
AES	128, 192 및 256비트 AES 키를 생성합니다.
트리플 DES(3DES, DESede)	192비트 트리플 DES 키를 생성합니다. 예정된 변경 사항은 1 아래 참고를 참조하세요.
EC	secp224r1(P-224), secp256r1(P-256), secp256k1(Blockchain), secp384r1(P-384) 및 secp521r1(P-521) 곡선을 사용하여 키를 생성합니다.
GENERIC_SECRET	1~800바이트의 일반 암호를 생성합니다.
RSA	2048비트~4096비트의 RSA 키를 생성합니다(256비트 증가).

[1] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에서는 이 기능이 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단를 참조하세요.

PKCS #11 라이브러리에 지원되는 메커니즘

PKCS #11 라이브러리는 PKCS #11 사양 버전 2.40을 준수합니다. PKCS#11을 사용하여 암호화 기능을 호출하려면 주어진 메커니즘을 가진 함수를 호출하십시오. 다음 섹션에서는 AWS CloudHSM에서 지원하는 기능과 메커니즘의 조합을 요약합니다.

PKCS #11용 소프트웨어 라이브러리는 다음과 같은 알고리즘을 지원합니다.

- 암호화 및 복호화 - AES-CBC, AES-CTR, AES-ECB, AES-GCM, DES3-CBC, DES3-ECB, RSA-OAEP 및 RSA-PKCS
- 서명 및 확인 – RSA, HMAC 및 ECDSA(해싱 사용 및 사용 안 함)
- 해시/다이제스트 – SHA1, SHA224, SHA256, SHA384 및 SHA512
- 키 랩 - AES 키 랩 [1], AES-GCM, RSA-AES 및 RSA-OAEP

주제

- 키 및 키 페어 함수 생성

- 서명 및 인증 기능
- 서명, 복구 및 복구 기능 확인
- 다이제스트 기능
- 암호화 및 해독 기능
- 키 함수 유도
- 랩 및 언랩 해제 기능
- 각 메커니즘의 최대 데이터 크기
- 메커니즘 주석

키 및 키 페어 함수 생성

PKCS #11 라이브러리용 AWS CloudHSM 소프트웨어 라이브러리를 사용하면 키 생성 및 키 쌍 기능에 다음 메커니즘을 사용할 수 있습니다.

- `CKM_RSA_PKCS_KEY_PAIR_GEN`
- `CKM_RSA_X9_31_KEY_PAIR_GEN` - 이 메커니즘은 기능적으로 `CKM_RSA_PKCS_KEY_PAIR_GEN` 메커니즘과 동일하지만 p 및 q 생성에 대해 더 강력한 보장을 제공합니다.
- `CKM_EC_KEY_PAIR_GEN`
- `CKM_GENERIC_SECRET_KEY_GEN`
- `CKM_AES_KEY_GEN`
- `CKM_DES3_KEY_GEN`— 5 각주에 예정된 변경 사항이 나열되어 있습니다.

서명 및 인증 기능

PKCS #11 라이브러리용 AWS CloudHSM 소프트웨어 라이브러리에서는 서명 및 확인 기능에 다음과 같은 메커니즘을 사용할 수 있습니다. Client SDK 5를 사용하면 데이터가 소프트웨어에서 로컬로 해시됩니다. 즉, SDK로 해시할 수 있는 데이터 크기에는 제한이 없습니다.

Client SDK 5를 사용하면 RSA 및 ECDSA 해싱이 로컬에서 수행되므로 데이터 제한이 없습니다. HMAC에는 데이터 제한이 있습니다. 자세한 내용은 2 각주를 참조하십시오.

RSA

- `CKM_RSA_X_509`

- CKM_RSA_PKCS – 단일 부분 작업에만 해당됩니다.
- CKM_RSA_PKCS_PSS – 단일 부분 작업에만 해당됩니다.
- CKM_SHA1_RSA_PKCS
- CKM_SHA224_RSA_PKCS
- CKM_SHA256_RSA_PKCS
- CKM_SHA384_RSA_PKCS
- CKM_SHA512_RSA_PKCS
- CKM_SHA512_RSA_PKCS
- CKM_SHA1_RSA_PKCS_PSS
- CKM_SHA224_RSA_PKCS_PSS
- CKM_SHA256_RSA_PKCS_PSS
- CKM_SHA384_RSA_PKCS_PSS
- CKM_SHA512_RSA_PKCS_PSS

ECDSA

- CKM_ECDSA – 단일 부분 작업에만 해당됩니다.
- CKM_ECDSA_SHA1
- CKM_ECDSA_SHA224
- CKM_ECDSA_SHA256
- CKM_ECDSA_SHA384
- CKM_ECDSA_SHA512

HMAC

- CKM_SHA_1_HMAC[2]
- CKM_SHA224_HMAC[2]
- CKM_SHA256_HMAC[2]
- CKM_SHA384_HMAC[2]
- CKM_SHA512_HMAC[2]

CMAC

- CKM_AES_CMAC

서명, 복구 및 복구 기능 확인

Client SDK 5는 서명 복구 및 복구 확인 기능을 지원하지 않습니다.

다이제스트 기능

PKCS #11 라이브러리용 AWS CloudHSM 소프트웨어 라이브러리에서는 다이제스트 함수에 다음과 같은 메커니즘을 사용할 수 있습니다. Client SDK 5를 사용하면 데이터가 소프트웨어에서 로컬로 해시됩니다. 즉, SDK로 해시할 수 있는 데이터 크기에는 제한이 없습니다.

- CKM_SHA_1
- CKM_SHA224
- CKM_SHA256
- CKM_SHA384
- CKM_SHA512

암호화 및 해독 기능

PKCS #11 라이브러리용 AWS CloudHSM 소프트웨어 라이브러리에서는 다음과 같은 암호화 및 복호화 함수 메커니즘을 사용할 수 있습니다.

- CKM_RSA_X_509
- CKM_RSA_PKCS – 단일 부분 작업에만 해당됩니다. 5 각주에 예정된 변경 사항이 나열되어 있습니다.
- CKM_RSA_PKCS_OAEP – 단일 부분 작업에만 해당됩니다.
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_DES3_CBC— 5 각주에 예정된 변경 사항이 나열되어 있습니다.

- CKM_DES3_ECB— 5 각주에 예정된 변경 사항이 나열되어 있습니다.
- CKM_DES3_CBC_PAD— 5 각주에 예정된 변경 사항이 나열되어 있습니다.
- CKM_AES_GCM [1, 2]
- CKM_CLOUDHSM_AES_GCM[3]

키 함수 유도

PKCS #11 라이브러리용 AWS CloudHSM 소프트웨어 라이브러리를 사용하면 다음과 같은 파생 함수 메커니즘을 사용할 수 있습니다.

- CKM_SP800_108_COUNTER_KDF

랩 및 언랩 해제 기능

PKCS #11 라이브러리용 AWS CloudHSM 소프트웨어 라이브러리를 사용하면 Wrap 및 Unwrap 함수에 다음과 같은 메커니즘을 사용할 수 있습니다.

AES 키 래핑에 대한 추가 정보는 AES 키 래핑을 참조하십시오.

- CKM_RSA_PKCS – 단일 부분 작업에만 해당됩니다. 5 각주에 예정된 변경 사항이 나열되어 있습니다.
- CKM_RSA_PKCS_OAEP[4]
- CKM_AES_GCM[1, 3]
- CKM_CLOUDHSM_AES_GCM[3]
- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD[3]
- CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD[3]
- CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD[3]

각 메커니즘의 최대 데이터 크기

다음 표에는 각 메커니즘에 설정된 최대 데이터 크기가 나와 있습니다.

최대 데이터 세트 크기

메커니즘	최대 데이터 크기(바이트)
CKM_SHA_1_HMAC	16288
CKM_SHA224_HMAC	16256
CKM_SHA256_HMAC	16288
CKM_SHA384_HMAC	16224
CKM_SHA512_HMAC	16224
CKM_AES_CBC	16272
CKM_AES_GCM	16224
CKM_CLOUDHSM_AES_GCM	16224
CKM_DES3_CBC	16280

메커니즘 주석

- [1] AES-GCM 암호화를 수행할 때 HSM은 애플리케이션의 IV(초기화 벡터) 데이터를 허용하지 않습니다. 생성되는 IV를 사용해야 합니다. HSM이 제공하는 12바이트 IV는 사용자가 제공하는 CK_GCM_PARAMS 파라미터 구조의 pIV 요소가 가리키는 메모리 참조에 기록됩니다. 사용자의 혼동을 피하기 위해 버전 1.1.1 이상인 PKCS#11 SDK는 AES-GCM 암호화가 초기화될 때 pIV가 초기화된 버퍼를 가리키는지 확인합니다.
- [2] 다음 메커니즘 중 하나를 사용하여 데이터에 대해 작업을 수행할 때 데이터 버퍼가 최대 데이터 크기를 초과하면 작업 결과 오류가 발생합니다. 이러한 메커니즘의 경우 모든 데이터 처리는 HSM 내에서 이루어져야 합니다. 각 메커니즘의 최대 데이터 크기 세트에 대한 자세한 내용은 각 메커니즘의 최대 데이터 크기 단원을 참조하십시오.
- [3] 공급업체가 정의한 메커니즘. CloudHSM 공급자 정의 메커니즘을 사용하려면 컴파일하는 동안 PKCS #11 애플리케이션에 /opt/cloudhsm/include/pkcs11t.h가 포함되어 있어야 합니다.

  CKM_CLOUDHSM_AES_GCM: 이 독점 메커니즘은 CKM_AES_GCM 표준에 대한 프로그래밍 방식으로 안전한 대안입니다. 이 메커니즘은 암호 초기화 중에 제공되는 CK_GCM_PARAMS 구조로 다시 암호 텍스트를 쓰는 대신 암호 텍스트 앞에 HSM에 의해 생성된 IV를 추가합니다. 이 메커니즘은

C_Encrypt, C_WrapKey, C_Decrypt 및 C_UnwrapKey 함수와 함께 사용할 수 있습니다. 이 메커니즘을 사용할 때는 CK_GCM_PARAMS 구문의 piV 변수를 NULL로 설정해야 합니다. C_Decrypt 및 C_UnwrapKey와 함께 이 메커니즘을 사용할 경우, IV는 언래핑되는 암호화 텍스트 앞에 추가될 것으로 예상됩니다.

CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD: PKCS #5 패딩을 사용한 AES 키 랩.

CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD: 제로 패딩을 사용한 AES 키 래핑.

- [4] 다음 CK_MECHANISM_TYPE 및 CK_RSA_PKCS_MGF_TYPE은 CKM_RSA_PKCS_OAEP에 대해 CK_RSA_PKCS_OAEP_PARAMS로 지원됩니다.
  - CKG_MGF1_SHA1을 사용하는 CKM_SHA_1
  - CKG_MGF1_SHA224를 사용하는 CKM_SHA224
  - CKG_MGF1_SHA256를 사용하는 CKM_SHA256
  - CKM_MGF1_SHA384를 사용하는 CKM_SHA384
  - CKM_MGF1_SHA512를 사용하는 CKM_SHA512
- [5] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에서는 이 기능이 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단를 참조하세요.

PKCS #11 라이브러리에 지원되는 API 작업

PKCS #11 라이브러리는 다음과 같은 PKCS #11 API 작업을 지원합니다.

- C_CloseAllSessions
- C_CloseSession
- C_CreateObject
- C_Decrypt
- C_DecryptFinal
- C_DecryptInit
- C_DecryptUpdate
- C_DeriveKey
- C_DestroyObject
- C_Digest
- C_DigestFinal

- C_DigestInit
- C_DigestUpdate
- C_Encrypt
- C_EncryptFinal
- C_EncryptInit
- C_EncryptUpdate
- C_Finalize
- C_FindObjects
- C_FindObjectsFinal
- C_FindObjectsInit
- C_GenerateKey
- C_GenerateKeyPair
- C_GenerateRandom
- C_GetAttributeValue
- C_GetFunctionList
- C_GetInfo
- C_GetMechanismInfo
- C_GetMechanismList
- C_GetSessionInfo
- C_GetSlotInfo
- C_GetSlotList
- C_GetTokenInfo
- C_Initialize
- C_Login
- C_Logout
- C_OpenSession
- C_Sign
- C_SignFinal
- C_SignInit

- C_SignUpdate
- C_UnWrapKey
- C_Verify
- C_VerifyFinal
- C_VerifyInit
- C_VerifyUpdate
- C_WrapKey

PKCS #11 라이브러리에 지원되는 주요 속성

키 객체는 퍼블릭 키, 프라이빗 키 또는 비밀 키일 수 있습니다. 키 객체에 대해 허용되는 작업은 속성을 통해 지정됩니다. 속성은 키 객체가 생성될 때 정의됩니다. PKCS #11 라이브러리를 사용하는 경우 PKCS #11 표준에 지정된 대로 기본값을 할당합니다.

AWS CloudHSM PKCS #11 사양에 나열된 모든 속성을 지원하지는 않습니다. 우리는 우리가 지원하는 모든 속성에 대한 사양을 준수합니다. 이러한 속성은 각 표에 나열되어 있습니다.

객체를 생성, 수정 또는 복사하는 C_CreateObject, C_GenerateKey, C_GenerateKeyPair, C_UnwrapKey, C_DeriveKey 등의 암호화 함수는 속성 템플릿을 파라미터 중 하나로 사용합니다. 객체 생성 중 속성 템플릿 전달에 대한 자세한 내용은 PKCS #11을 통해 키 생성 라이브러리의 예를 참조하십시오.

PKCS #11 라이브러리 속성 테이블 해석

PKCS #11 라이브러리 테이블에는 키 유형별로 다른 속성 목록이 포함되어 있습니다. 에서 특정 암호화 함수를 사용할 때 특정 키 유형에 대해 지정된 속성이 지원되는지 여부를 나타냅니다. AWS CloudHSM

범례:

- ✓는 CloudHSM이 특정 키 유형에 대해 해당 속성을 지원함을 나타냅니다.
- ✖는 CloudHSM이 특정 키 유형에 대해 해당 속성을 지원하지 않음을 나타냅니다.
- R은 속성 값이 특정 키 유형에 대해 읽기 전용으로 설정됨을 나타냅니다.
- S는 중요하므로 GetAttributeValue로 속성을 읽을 수 없음을 나타냅니다.
- 기본값 열의 빈 셀은 속성에 할당된 특정 기본값이 없음을 나타냅니다.

GenerateKeyPair

속성	키 유형				기본 값
	EC 프라이빗	EC 퍼블릭	RSA 프라이빗	RSA 퍼블릭	
CKA_CLASS	✓	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	✓	
CKA_LABEL	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	True
CKA_TOKEN	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✖	✓	✖	✓	False
CKA_DECRYPT	✓	✖	✓	✖	False
CKA_DERIVE	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	True

속성	키 유형				기본 값
CKA_DESTROYABLE	✓	✓	✓	✓	True
CKA_SIGN	✓	✗	✓	✗	False
CKA_SIGN_RECOVER	✗	✗	✗	✗	
CKA_VERIFY	✗	✓	✗	✓	False
CKA_VERIFY_RECOVER	✗	✗	✗	✗	
CKA_WRAP	✗	✓	✗	✓	False
CKA_WRAP_TEMPLATE	✗	✓	✗	✓	
CKA_TRUSTED	✗	✓	✗	✓	False
CKA_WRAP_WITH_TRUSTED	✓	✗	✓	✗	False
CKA_UNWRAP	✓	✗	✓	✗	False
CKA_UNWRAP_TEMPLATE	✓	✗	✓	✗	
CKA_SENSITIVE	✓[1]	✗	✓[1]	✗	True

속성	키 유형				기본 값
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	
CKA_EXTRACTABLE	✓	✖	✓	✖	True
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	
CKA_MODULUS	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✓[2]	
CKA_PRIME_1	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	

속성	키 유형				기본 값
CKA_PUBLIC_EXPONENT	✖	✖	✖	✓[2]	
CKA_EC_PARAMS	✖	✓[2]	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	

GenerateKey

속성	키 유형			기본 값
	AES	DES3	일반 보안	
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	False

속성	키 유형			기본 값
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✓	✓	✖	False
CKA_DECRYPT	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	True
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	True
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	False
CKA_WRAP_TEMPLATE	✓	✓	✖	
CKA_TRUSTED	✓	✓	✖	False

속성	키 유형			기본 값
CKA_WRAP_WITH_TRUSTED	✓	✓	✓	False
CKA_UNWRAP	✓	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✓	✖	
CKA_SENSITIVE	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	✖	✖	✖	
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	

속성	키 유형			기본 값
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

CreateObject

속성	키 유형							기본 값
	EC 프라이빗	EC 퍼블릭	RSA 프라이빗	RSA 퍼블릭	AES	DES3	일반 보안	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	R	R	False
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖	False
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True

속성	키 유형							기본 값
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	✖	✖	False
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖	False
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖	
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓	False
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓	True

속성	키 유형							기본 값
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	R	R	R	
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	R	R	R	
CKA_MODULUS	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✓	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✓[2]	✖	✖	✖	✖	

속성	키 유형							기본 값
CKA_PUBLIC_EXPONENT	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_EC_PARAMS	✓[2]	✓[2]	✖	✖	✖	✖	✖	
CKA_EC_POINT	✖	✓[2]	✖	✖	✖	✖	✖	
CKA_VALUE	✓[2]	✖	✖	✖	✓[2]	✓[2]	✓[2]	
CKA_VALUE_LEN	✖	✖	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	R	R	

UnwrapKey

속성	키 유형					기본 값
	EC 프라이빗	RSA 프라이빗	AES	DES3	일반 보안	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	

속성	키 유형					기본 값
CKA_ID	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	False
CKA_TOKEN	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✖	✖	✓	✓	✖	False
CKA_DECRYPT	✖	✓	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	False
CKA_VERIFY	✖	✖	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	

속성	키 유형					기본 값
CKA_WRAP	✖	✖	✓	✓	✖	False
CKA_UNWRAP	✖	✓	✓	✓	✖	False
CKA_SENSITIVE	✓	✓	✓	✓	✓	True
CKA_EXTRACTABLE	✓	✓	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	
CKA_MODULUS	✖	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	✖	

속성	키 유형					기본 값
CKA_EXPONENT_2	✖	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	

DeriveKey

속성	키 유형			기본 값
	AES	DES3	일반 보안	
CKA_CLASS	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	

속성	키 유형			기본 값
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✓	✓	✖	False
CKA_DECRYPT	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓[1]	✓[1]	✓[1]	True
CKA_SIGN	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	

속성	키 유형			기본 값
CKA_WRAP	✓	✓	✖	False
CKA_UNWRAP	✓	✓	✖	False
CKA_SENSITIVE	R	R	R	True
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	

속성	키 유형			기본 값
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

속성	키 유형						
	EC 프라이빗	EC 퍼블릭	RSA 프라이빗	RSA 퍼블릭	AES	DES3	일반 보안
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓

속성	키 유형						
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓
CKA_SIGN_RECOVER	✖	✖	✓	✖	✖	✖	✖

속성	키 유형						
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓
CKA_VERIFY_RECOVER	✖	✖	✖	✓	✖	✖	✖
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✓
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_NEVER_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓

속성	키 유형						
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	R	R
CKA_MODULUS	✖	✖	✓	✓	✖	✖	✖
CKA_MODULUS_BITS	✖	✖	✖	✓	✖	✖	✖
CKA_PRIME_1	✖	✖	S	✖	✖	✖	✖
CKA_PRIME_2	✖	✖	S	✖	✖	✖	✖
CKA_COEFFICIENT	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_1	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_2	✖	✖	S	✖	✖	✖	✖
CKA_PRIVATE_EXPONENT	✖	✖	S	✖	✖	✖	✖
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖

속성	키 유형						
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
CKA_VALUE	S	✖	✖	✖	✓	✓	✓
CKA_VALUE_LEN	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	✖

속성 주석

- [1] 이 속성은 펌웨어에서 부분적으로 지원되며 명시적으로 기본값으로만 설정되어야 합니다.
- [2] 필수 속성.

속성 수정

객체의 일부 속성은 객체가 생성된 후 수정할 수 있지만, 일부 속성은 수정할 수 없습니다. 속성을 수정하려면 cloudhsm_mgmt_util에서 setAttribute 명령을 사용하십시오. cloudhsm_mgmt_util에서 listAttribute 명령을 사용하여 속성 및 이를 나타내는 상수 목록을 파생시킬 수도 있습니다.

다음 목록은 객체 생성 후 수정할 수 있는 속성을 보여 줍니다.

- CKA_LABEL
- CKA_TOKEN

> **Note**
>
> 세션 키를 토큰 키로 변경하는 경우에만 수정이 허용됩니다. 속성 값을 변경하려면 key_mgmt_util에서 setAttribute 명령을 사용하십시오.

- CKA_ENCRYPT
- CKA_DECRYPT

- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

> **Note**
>
> 이 속성은 키 파생을 지원합니다. 모든 퍼블릭 키에 대해 False여야 하고 True로 설정할 수 없습니다. 보안 및 EC 프라이빗 키의 경우 True 또는 False로 설정할 수 있습니다.

- CKA_TRUSTED

> **Note**
>
> CO(Crypto Officer)만 이 속성을 True 또는 False로 설정할 수 있습니다.

- CKA_WRAP_WITH_TRUSTED

> **Note**
>
> 이 속성을 내보낼 수 있는 데이터 키에 적용하여 이 키를 CKA_TRUSTED로 표시된 키로만 래핑할 수 있도록 지정합니다. CKA_WRAP_WITH_TRUSTED는 true로 설정하면 속성이 읽기 전용이 되며 속성을 변경하거나 제거할 수 없습니다.

오류 코드 해석

템플릿에서 특정 키로 지원되지 않는 속성을 지정하면 오류가 발생합니다. 다음 표에 사양을 위반하면 발생되는 오류 코드가 나와 있습니다.

오류 코드	설명
CKR_TEMPLATE_INCONSISTENT	어떤 속성이 PKCS #11 사양을 준수하지만 CloudHSM에서 지원되지 않는 경우, 속성 템플

오류 코드	설명
	릿에서 해당 속성을 지정하면 이 오류가 발생합니다.
CKR_ATTRIBUTE_TYPE_INVALID	PKCS #11 사양을 준수하지만 CloudHSM에서 지원되지 않는 속성의 값을 검색하면 이 오류가 발생합니다.
CKR_ATTRIBUTE_INCOMPLETE	속성 템플릿에서 필수 속성을 지정하지 않으면 이 오류가 발생합니다.
CKR_ATTRIBUTE_READ_ONLY	속성 템플릿에서 읽기 전용 속성을 지정하면 이 오류가 발생합니다.

PKCS #11 라이브러리의 코드 샘플

위의 코드 샘플은 PKCS #11 라이브러리를 사용하여 기본 작업을 수행하는 방법을 GitHub 보여줍니다.

사전 조건

샘플을 실행하기 전에 다음 단계를 수행하여 환경을 설정하십시오.

- 클라이언트 SDK 5용 PKCS #11 라이브러리를 설치하고 구성합니다.
- CU(Cryptographic User) 설정 애플리케이션은 이 HSM 계정을 사용하여 HSM에서 코드 샘플을 실행합니다.

코드 샘플

PKCS #11 AWS CloudHSM 소프트웨어 라이브러리의 코드 샘플은 에서 사용할 수 있습니다. GitHub 이 리포지토리에는 암호화, 암호 해독, 서명 및 확인 등 PKCS #11을 사용하여 일반적인 작업을 수행하는 방법에 대한 예제가 포함되어 있습니다.

- 키 생성(AES, RSA, EC)
- 키 속성 나열
- AES GCM을 사용하여 데이터 암호화 및 암호화 해제
- AES_CTR을 사용하여 데이터 암호화 및 복호화

- 3DES를 사용하여 데이터 암호화 및 복호화
- RSA를 사용하여 데이터 서명 및 확인
- HMAC KDF를 사용하여 키 추출
- PKCS #5 패딩을 사용하는 AES로 키 래핑 및 언래핑
- 패딩을 사용하지 않는 AES로 키 래핑 및 언래핑
- 제로 패딩을 사용하는 AES로 키 래핑 및 언래핑
- AES-GCM을 사용하여 키 래핑 및 언래핑
- RSA를 사용하여 키 래핑 및 언래핑

PKCS #11 라이브러리를 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하십시오.

이 주제를 사용하여 PKCS #11 라이브러리를 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하십시오. 마이그레이션의 이점은 을 참조하십시오. Client SDK 5의 이점

에서 AWS CloudHSM고객 애플리케이션은 AWS CloudHSM 클라이언트 소프트웨어 개발 키트 (SDK)를 사용하여 암호화 작업을 수행합니다. 클라이언트 SDK 5는 계속해서 새로운 기능과 플랫폼 지원이 추가되는 기본 SDK입니다.

모든 공급자에 대한 마이그레이션 지침을 검토하려면 을 참조하십시오. 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션

주요 변경 사항을 해결하여 대비하십시오.

이러한 주요 변경 사항을 검토하고 그에 따라 개발 환경에서 애플리케이션을 업데이트하십시오.

랩 메커니즘이 변경되었습니다.

클라이언트 SDK 3 메커니즘	상응하는 클라이언트 SDK 5 메커니즘
CKM_AES_KEY_WRAP	CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
CKM_AES_KEY_WRAP_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD

클라이언트 SDK 3 메커니즘	상응하는 클라이언트 SDK 5 메커니즘
CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD

ECDH

클라이언트 SDK 3에서는 ECDH를 사용하고 KDF를 지정할 수 있습니다. 이 기능은 현재 클라이언트 SDK 5에서 사용할 수 없습니다. 애플리케이션에 이 기능이 필요한 경우 지원팀에 문의하세요.

이제 키 핸들은 세션별로 다릅니다.

Client SDK 5에서 키 핸들을 성공적으로 사용하려면 애플리케이션을 실행할 때마다 키 핸들을 가져와야 합니다. 여러 세션에서 동일한 키 핸들을 사용하는 기존 애플리케이션이 있는 경우 애플리케이션을 실행할 때마다 키 핸들을 가져오도록 코드를 수정해야 합니다. 키 핸들 검색에 대한 자세한 내용은 이 AWS CloudHSM PKCS #11 예제를 참조하십시오. 이 변경은 PKCS #11 2.40 사양을 준수합니다.

클라이언트 SDK 5로 마이그레이션

이 섹션의 지침에 따라 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하십시오.

> **Note**
>
> 아마존 리눅스, 우분투 16.04, 우분투 18.04, CentOS 6, CentOS 8 및 RHEL 6은 현재 클라이언트 SDK 5에서 지원되지 않습니다. 현재 클라이언트 SDK 3과 함께 이러한 플랫폼 중 하나를 사용하고 있다면 클라이언트 SDK 5로 마이그레이션할 때 다른 플랫폼을 선택해야 합니다.

1. 클라이언트 SDK 3용 PKCS #11 라이브러리를 제거합니다.

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-pkcs11
   ```

CentOS 7

```
$ sudo yum remove cloudhsm-pkcs11
```

RHEL 7

```
$ sudo yum remove cloudhsm-pkcs11
```

RHEL 8

```
$ sudo yum remove cloudhsm-pkcs11
```

2. 클라이언트 SDK용 클라이언트 데몬 제거 3.

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client
```

CentOS 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 8

```
$ sudo yum remove cloudhsm-client
```

> **Note**
>
> 사용자 지정 구성을 다시 활성화해야 합니다.

3. 의 단계에 따라 클라이언트 SDK PKCS #11 라이브러리를 설치합니다. 클라이언트 SDK 5용 PKCS #11 라이브러리 설치

4. 클라이언트 SDK 5에는 새로운 구성 파일 형식과 명령줄 부트스트래핑 도구가 도입되었습니다. 클라이언트 SDK 5 PKCS #11 라이브러리를 부트스트랩하려면 아래 사용 설명서에 나와 있는 지침을 따르십시오. Client SDK 부트스트랩합니다.
5. 개발 환경에서 애플리케이션을 테스트하십시오. 최종 마이그레이션 전에 기존 코드를 업데이트하여 주요 변경 사항을 해결하세요.

관련 주제

- 에 대한 모범 사례 AWS CloudHSM

PKCS #11의 고급 구성

AWS CloudHSM PKCS #11 공급자에는 다음과 같은 고급 구성이 포함되며, 이는 대부분의 고객이 사용하는 일반 구성에는 포함되지 않습니다. 이러한 구성은 추가 기능을 제공합니다.

- PKCS #11을 사용하여 여러 슬롯에 연결
- PKCS #11 구성 다시 시도

PKCS #11를 사용하여 여러 슬롯에 연결

클라이언트 SDK 5 PKCS #11 라이브러리의 단일 슬롯은 AWS CloudHSM의 클러스터에 대한 단일 연결을 나타냅니다. 클라이언트 SDK 5를 사용하면 여러 슬롯이 단일 PKCS#11 애플리케이션에서 여러 CloudHSM 클러스터에 사용자를 연결할 수 있도록 PKCS11 라이브러리를 구성할 수 있습니다.

이 주제의 지침에 따라 애플리케이션이 다중 슬롯 기능을 사용하여 여러 클러스터에 연결되도록 할 수 있습니다.

주제

- 다중 슬롯 필수 조건
- 다중 슬롯 기능을 위한 PKCS #11 라이브러리 구성
- configure-pkcs11 add-cluster
- configure-pkcs11 remove-cluster

다중 슬롯 필수 조건

- 연결하려는 두 개 이상의 AWS CloudHSM 클러스터와 해당 클러스터 인증서

- 위의 모든 클러스터에 연결하도록 보안 그룹이 올바르게 구성된 EC2 인스턴스. 클러스터 및 클라이언트 인스턴스를 설정하는 방법에 대한 자세한 내용은 시작하기를 참조하십시오 AWS CloudHSM.
- 다중 슬롯 기능을 설정하려면 PKCS #11 라이브러리를 이미 다운로드하여 설치했어야 합니다. 아직 수행하지 않은 경우 ???의 지침을 참조하십시오.

다중 슬롯 기능을 위한 PKCS #11 라이브러리 구성

다중 슬롯 기능을 위해 PKCS #11 라이브러리를 구성하려면 다음 단계를 따르십시오.

1. 다중 슬롯 기능을 사용하여 연결하려는 클러스터를 식별합니다.
2. ???의 지침에 따라 이러한 클러스터를 PKCS #11 구성에 추가합니다.
3. 다음 번에 PKCS #11 애플리케이션을 실행하면 다중 슬롯 기능을 사용할 수 있게 됩니다.

configure-pkcs11 add-cluster

PKCS #11을 사용하여 여러 슬롯에 연결할 때 configure-pkcs11 add-cluster 명령을 사용하여 구성에 클러스터를 추가합니다.

구문

```
configure-pkcs11 add-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [--region <REGION>]
        [--endpoint <ENDPOINT>]
        [--hsm-ca-cert <HSM CA CERTIFICATE FILE>]
        [--server-client-cert-file <CLIENT CERTIFICATE FILE>]
        [--server-client-key-file <CLIENT KEY FILE>]
        [-h, --help]
```

예

`cluster-id` 파라미터를 사용하여 클러스터를 추가합니다.

Example

`cluster-id` 파라미터와 함께 configure-pkcs11 add-cluster을 사용하여 구성에 클러스터(ID `cluster-1234567`)를 추가합니다.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 add-cluster --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-pkcs11.exe add-cluster --cluster-
id cluster-1234567
```

> **Tip**
>
> `cluster-id` 파라미터와 함께 configure-pkcs11 add-cluster을 사용해도 클러스터가 추가되지 않는 경우, 추가되는 클러스터를 식별하기 위해 `--region` 및 `--endpoint` 파라미터도 필요한 이 명령의 더 긴 버전에 대한 다음 예를 참조하십시오. 예를 들어 클러스터의 리전이 AWS CLI 기본값으로 구성된 리전과 다른 경우 `--region` 파라미터를 사용하여 올바른 리전을 사용해야 합니다. 또한 호출에 사용할 AWS CloudHSM API 엔드포인트를 지정할 수 있습니다. 이는 기본 DNS 호스트 이름을 사용하지 않는 VPC 인터페이스 엔드포인트를 사용하는 등 다양한 네트워크 설정에 필요할 수 있습니다. AWS CloudHSM

`cluster-id`, **`endpoint`**, **`region`** 파라미터를 사용하여 클러스터를 추가합니다.

Example

`cluster-id`, `endpoint`, `region` 파라미터와 함께 configure-pkcs11 add-cluster을 사용하여 구성에 클러스터(ID `cluster-1234567`)를 추가합니다.

Linux

```
        $ sudo /opt/cloudhsm/bin/configure-pkcs11 add-cluster --cluster-
id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

Windows

```
        C:\Program Files\Amazon\CloudHSM\> .\configure-pkcs11.exe add-cluster --
cluster-id cluster-1234567--region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

`--cluster-id`, `--region`, `--endpoint` 파라미터에 대한 자세한 내용은 the section called “파라미터” 섹션을 참조하십시오.

파라미터

--cluster-id *<Cluster ID>*

클러스터 ID와 연결된 클러스터의 모든 HSM Elastic Network 인터페이스(ENI) IP 주소를 찾기 위해 `DescribeClusters`를 호출합니다. 시스템은 ENI IP 주소를 구성 파일에 추가합니다. AWS CloudHSM

> **Note**
>
> 퍼블릭 인터넷에 액세스할 수 없는 VPC 내 EC2 인스턴스의 `--cluster-id` 파라미터를 사용하는 경우 연결할 인터페이스 VPC 엔드포인트를 만들어야 합니다. AWS CloudHSM VPC 엔드포인트에 대한 자세한 내용은 ??? 섹션을 참조하십시오.

필수 여부: 예

--endpoint *<Endpoint>*

AWS CloudHSM 호출에 사용되는 API 엔드포인트를 지정합니다. `DescribeClusters` 이 옵션은 `--cluster-id`와 조합하여 설정해야 합니다.

필수 여부: 아니요

-- hsm-ca-cert <HsmCA Certificate Filepath>

HSM CA 인증서의 파일 경로를 지정합니다.

필수 여부: 아니요

--region *<Region>*

클러스터의 지역을 지정합니다. 이 옵션은 `--cluster-id`와 조합하여 설정해야 합니다.

`--region` 파라미터를 제공하지 않으면 시스템은 `AWS_DEFAULT_REGION` 또는 `AWS_REGION` 환경 변수를 읽으려고 시도하여 리전을 선택합니다. 이러한 변수가 설정되지 않은 경우, 시스템은 사용자가 `AWS_CONFIG_FILE` 환경 변수에 다른 파일을 지정하지 않는 한 AWS config 파일(일반적으로 `~/.aws/config`)의 프로필과 연결된 리전을 확인합니다. 위 항목 중 아무 것도 설정되지 않은 경우 시스템은 `us-east-1` 리전을 기본값으로 사용합니다.

필수 여부: 아니요

-- server-client-cert-file <Client Certificate Filepath>

TLS 클라이언트-서버 상호 인증에 사용되는 클라이언트 인증서의 경로입니다.

클라이언트 SDK 5에 포함된 기본 키 및 SSL/TLS 인증서를 사용하지 않으려는 경우에만 이 옵션을 사용하세요. 이 옵션은 `--server-client-key-file`와 조합하여 설정해야 합니다.

필수 여부: 아니요

-- server-client-key-file <Client Key Filepath>

TLS 클라이언트-서버 상호 인증에 사용되는 클라이언트 키의 경로.

클라이언트 SDK 5에 포함된 기본 키 및 SSL/TLS 인증서를 사용하지 않으려는 경우에만 이 옵션을 사용하세요. 이 옵션은 `--server-client-cert-file`와 조합하여 설정해야 합니다.

필수 여부: 아니요

configure-pkcs11 remove-cluster

PKCS#11을 사용하여 여러 슬롯에 연결할 때 configure-pkcs11 remove-cluster 명령을 사용하여 사용 가능한 PKCS #11 슬롯에서 클러스터를 제거합니다.

구문

```
configure-pkcs11 remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

예

`cluster-id` 파라미터를 사용하여 클러스터를 삭제합니다.

Example

`cluster-id` 파라미터와 함께 configure-pkcs11 remove-cluster을 사용하여 구성에서 클러스터(ID `cluster-1234567`)를 삭제합니다.

Linux

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 remove-cluster --cluster-
id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-pkcs11.exe remove-cluster --cluster-
id cluster-1234567
```

`--cluster-id` 파라미터에 대한 자세한 내용은 the section called “파라미터”을 참조하세요.

파라미터

--cluster-id ***<Cluster ID>***

구성에서 제거할 클러스터의 ID

필수 여부: 예

PKCS #11 재시도 명령

Client SDK 5.8.0 이상에는 HSM 제한 작업을 클라이언트 측에서 재시도하는 자동 재시도 전략이 내장되어 있습니다. HSM이 이전 작업을 수행하느라 너무 바빠서 더 많은 요청을 받을 수 없어 작업을 제한하는 경우 Client SDK는 제한이 발생한 작업을 최대 3회까지 재시도하고 기하급수적으로 백오프합니다. 이 자동 재시도 전략은 끄기 모드와 표준 모드 중 하나로 설정할 수 있습니다.

- 끄기 모드: Client SDK는 HSM에서 병목 현상이 발생한 작업에 대해 재시도 전략을 수행하지 않습니다.
- 표준 모드: Client SDK 5.8.0 이상의 기본 모드입니다. 이 모드에서는 Client SDK가 기하급수적으로 백오프를 수행하여 병목 현상이 발생한 작업을 자동으로 재시도합니다.

자세한 내용은 HSM 스로틀링 단원을 참조하십시오.

재시도 명령을 끄기 모드로 설정합니다.

Linux

Linux용 Client SDK 5의 재시도 명령을 off로 설정하려면

- 다음 명령을 사용하여 재시도 구성을 off 모드로 설정할 수 있습니다.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --default-retry-mode off
```

Windows

Client SDK 5 on Windows에 대한 재시도 명령을 off로 설정하려면

- 다음 명령을 사용하여 재시도 구성을 off 모드로 설정할 수 있습니다.

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-pkcs11.exe --default-retry-
mode off
```

OpenSSL Dynamic Engine

AWS CloudHSM OpenSSL 동적 엔진을 사용하면 OpenSSL API를 통해 암호화 작업을 CloudHSM 클러스터로 오프로드할 수 있습니다.

AWS CloudHSM 에서 읽을 수 있는 OpenSSL 동적 엔진을 제공합니다. Linux에서의 SSL/TLS 오프로드 AWS CloudHSM OpenSSL과 함께 사용하는 방법에 대한 예는 이 AWS 보안 블로그를 참조하십시오. SDK에 대한 플랫폼 지원에 대한 자세한 내용은 the section called “지원하는 플랫폼” 섹션을 참조하세요. 문제 해결에 대해서는 을 참조하십시오OpenSSL Dynamic Engine의 알려진 문제.

Client SDK 3 사용에 대한 자세한 내용은 이전 클라이언트 SDK (클라이언트 SDK 3) 단원을 참조하십시오.

자세한 내용은 아래 주제를 참조하십시오.

주제

- OpenSSL Dynamic Engine 설치
- OpenSSL 다이내믹 엔진 키 유형
- OpenSSL 다이내믹 엔진 메커니즘
- OpenSSL 동적 엔진을 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하십시오.
- OpenSSL용 고급 구성

OpenSSL Dynamic Engine 설치

> **Note**
>
> 클라이언트 SDK 5를 사용하여 단일 HSM 클러스터를 실행하려면 먼저 `disable_key_availability_check`을 `True`로 설정하여 클라이언트 키 내구성 설정을 관리해야 합니다. 자세한 내용은 키 동기화 및 클라이언트 SDK 5 구성 도구를 참조하십시오.

OpenSSL Dynamic Engine을 설치 및 구성하려면

1. 다음 명령을 사용하여 OpenSSL 엔진을 다운로드하고 설치합니다.

   Amazon Linux 2

   x86_64 아키텍처에 아마존 리눅스 2용 OpenSSL 다이내믹 엔진을 설치합니다.

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-dyn-latest.el7.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-dyn-latest.el7.x86_64.rpm
   ```

   ARM64 아키텍처에 아마존 리눅스 2용 OpenSSL 다이내믹 엔진을 설치합니다.

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-dyn-latest.el7.aarch64.rpm
   ```

```
$ sudo yum install ./cloudhsm-dyn-latest.el7.aarch64.rpm
```

Amazon Linux 2023

x86_64 아키텍처에 아마존 리눅스 2023용 OpenSSL 다이내믹 엔진을 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-dyn-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.amzn2023.x86_64.rpm
```

ARM64 아키텍처에 아마존 리눅스 2023용 OpenSSL 다이내믹 엔진을 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/
cloudhsm-dyn-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.amzn2023.aarch64.rpm
```

CentOS 7 (7.8+)

x86_64 아키텍처에 CentOS 7용 OpenSSL 동적 엔진을 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-
dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el7.x86_64.rpm
```

RHEL 7 (7.8+)

x86_64 아키텍처에 RHEL 7용 OpenSSL 동적 엔진을 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-
dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

x86_64 아키텍처에 RHEL 8용 OpenSSL 동적 엔진을 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-dyn-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el8.x86_64.rpm
```

RHEL 9 (9.2+)

x86_64 아키텍처에 RHEL 9용 OpenSSL 동적 엔진을 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-dyn-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el9.x86_64.rpm
```

ARM64 아키텍처에 RHEL 9용 OpenSSL 동적 엔진을 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-dyn-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-dyn-latest.el9.aarch64.rpm
```

Ubuntu 20.04 LTS

x86_64 아키텍처에 우분투 20.04 LTS용 OpenSSL 다이내믹 엔진을 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/cloudhsm-dyn_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u20.04_amd64.deb
```

Ubuntu 22.04 LTS

x86_64 아키텍처에 우분투 22.04 LTS용 OpenSSL 다이내믹 엔진을 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-dyn_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u22.04_amd64.deb
```

ARM64 아키텍처에 우분투 22.04 LTS용 OpenSSL 다이내믹 엔진을 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-dyn_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-dyn_latest_u22.04_arm64.deb
```

`/opt/cloudhsm/lib/libcloudhsm_openssl_engine.so`에서 Dynamic Engine용 공유 라이브러리를 설치했습니다.

2. 부트스트래핑 클라이언트 SDK 5. 부트스트래핑에 대한 자세한 내용은 Client SDK 부트스트랩합니다.을 참조하십시오.
3. 암호화 사용자(CU) 의 자격 증명으로 환경 변수를 설정합니다. CU 생성에 대한 자세한 정보는 CMU를 사용하여 사용자 관리 섹션을 참조하세요.

```
$ export CLOUDHSM_PIN=<HSM user name>:<password>
```

> **Note**
>
> 클라이언트 SDK 5는 CU의 자격 증명을 저장하기 위한 `CLOUDHSM_PIN` 환경 변수를 도입합니다. 클라이언트 SDK 3에서는 CU 자격 증명을 `n3fips_password` 환경 변수에 저장합니다. 클라이언트 SDK 5는 두 환경 변수를 모두 지원하지만 `CLOUDHSM_PIN`을 사용하는 것이 좋습니다.

4. 설치된 OpenSSL Dynamic Engine을 클러스터에 연결합니다. 자세한 내용은 클러스터에 연결 단원을 참조하세요.
5. 클라이언트 SDK 부트스트래핑 5. 자세한 정보는 the section called “Client SDK 부트스트랩합니다.”을 참조하세요.

클라이언트 SDK 5용 OpenSSL Dynamic Engine 확인

다음 명령을 사용하여 OpenSSL Dynamic Engine 설치를 확인합니다.

```
$ openssl engine -t cloudhsm
```

다음 출력은 구성을 확인합니다.

```
(cloudhsm) CloudHSM OpenSSL Engine
     [ available ]
```

OpenSSL 다이내믹 엔진 키 유형

AWS CloudHSM OpenSSL 동적 엔진은 다음 키 유형을 지원합니다.

키 유형	설명
EC	P-256, P-384 및 secp256k1 키 유형에 대한 ECDSA 서명/검증. OpenSSL 엔진과 상호 운용 가능한 EC 키를 생성하려면 을 참조하세요. 키 생성-파일
RSA	2048, 3072 및 4096비트 키에 대한 RSA 키 생성. RSA 서명/검증. 검증은 OpenSSL 소프트웨어로 오프로드됩니다.

OpenSSL 다이내믹 엔진 메커니즘

AWS CloudHSM OpenSSL 동적 엔진 메커니즘을 사용하는 방법을 알아보십시오.

서명 및 인증 기능

AWS CloudHSM OpenSSL 동적 엔진에서는 서명 및 확인 기능에 다음과 같은 메커니즘을 사용할 수 있습니다.

Client SDK 5를 사용하면 데이터가 소프트웨어에서 로컬로 해시됩니다. 즉, 해시할 수 있는 데이터 크기에는 제한이 없습니다.

RSA 서명 유형

- SHA1withRSA
- SHA224(RSA 포함)
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA

ECDSA 서명 유형

- SHA1(ECDSA 포함)
- SHA224(ECDSA 포함)
- SHA256(ECDSA 포함)
- SHA384(ECDSA 포함)
- SHA512(ECDSA 포함)

OpenSSL 동적 엔진을 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하십시오.

이 주제를 사용하여 OpenSSL 동적 엔진을 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션할 수 있습니다. 마이그레이션의 이점은 을 참조하십시오. Client SDK 5의 이점

에서 AWS CloudHSM고객 애플리케이션은 AWS CloudHSM 클라이언트 소프트웨어 개발 키트 (SDK)를 사용하여 암호화 작업을 수행합니다. 클라이언트 SDK 5는 계속해서 새로운 기능과 플랫폼 지원이 추가되는 기본 SDK입니다.

> **Note**
>
> OpenSSL 동적 엔진을 사용하는 클라이언트 SDK 5에서는 현재 난수 생성이 지원되지 않습니다.

모든 공급자에 대한 마이그레이션 지침을 검토하려면 을 참조하십시오. 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션

클라이언트 SDK 5로 마이그레이션

이 섹션의 지침에 따라 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하십시오.

> **Note**
>
> 아마존 리눅스, 우분투 16.04, 우분투 18.04, CentOS 6, CentOS 8 및 RHEL 6은 현재 클라이언트 SDK 5에서 지원되지 않습니다. 현재 클라이언트 SDK 3과 함께 이러한 플랫폼 중 하나를 사용하고 있다면 클라이언트 SDK 5로 마이그레이션할 때 다른 플랫폼을 선택해야 합니다.

1. 클라이언트 SDK용 OpenSSL 동적 엔진 제거 3.

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

2. 클라이언트 SDK용 클라이언트 데몬 제거 3.

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-client
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-client
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-client
   ```

RHEL 8

```
$ sudo yum remove cloudhsm-client
```

> **Note**
>
> 사용자 지정 구성을 다시 활성화해야 합니다.

3. 의 단계에 따라 클라이언트 SDK OpenSSL 동적 엔진을 설치합니다. OpenSSL Dynamic Engine 설치
4. 클라이언트 SDK 5에는 새로운 구성 파일 형식과 명령줄 부트스트래핑 도구가 도입되었습니다. 클라이언트 SDK 5 OpenSSL 동적 엔진을 부트스트랩하려면 아래 사용 설명서에 나와 있는 지침을 따르십시오. Client SDK 부트스트랩합니다.
5. 개발 환경에서 애플리케이션을 테스트하십시오. 최종 마이그레이션 전에 기존 코드를 업데이트하여 주요 변경 사항을 해결하세요.

관련 주제

- 에 대한 모범 사례 AWS CloudHSM

OpenSSL용 고급 구성

AWS CloudHSM OpenSSL 공급자에는 대부분의 고객이 사용하는 일반 구성의 일부가 아닌 다음과 같은 고급 구성이 포함되어 있습니다. 이러한 구성은 추가 기능을 제공합니다.

- OpenSSL용 재시도 명령

OpenSSL용 재시도 명령

Client SDK 5.8.0 이상에는 HSM 제한 작업을 클라이언트 측에서 재시도하는 자동 재시도 전략이 내장되어 있습니다. HSM이 이전 작업을 수행하느라 너무 바빠서 더 많은 요청을 받을 수 없어 작업을 제한하는 경우 Client SDK는 제한이 발생한 작업을 최대 3회까지 재시도하고 기하급수적으로 백오프합니다. 이 자동 재시도 전략은 끄기 모드와 표준 모드 중 하나로 설정할 수 있습니다.

- 끄기 모드: Client SDK는 HSM에서 병목 현상이 발생한 작업에 대해 재시도 전략을 수행하지 않습니다.
- 표준 모드: Client SDK 5.8.0 이상의 기본 모드입니다. 이 모드에서는 Client SDK가 기하급수적으로 백오프를 수행하여 병목 현상이 발생한 작업을 자동으로 재시도합니다.

자세한 내용은 HSM 스로틀링 단원을 참조하십시오.

재시도 명령을 끄기 모드로 설정합니다.

Linux

Linux용 Client SDK 5의 재시도 명령을 off로 설정하려면

- 다음 명령을 사용하여 Retry 명령을 off 모드로 설정할 수 있습니다.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --default-retry-mode off
```

Windows

Client SDK 5 on Windows에 대한 재시도 명령을 off로 설정하려면

- 다음 명령을 사용하여 Retry 명령을 off 모드로 설정할 수 있습니다.

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-dyn.exe --default-retry-mode
 off
```

JCE 공급자

AWS CloudHSM JCE 제공자는 JCE (Java 암호화 확장) 제공자 프레임워크를 기반으로 구축된 제공자 구현입니다. JCE를 사용하면 자바 개발자 키트(Java Development Kit)를 사용하여 암호화 작업을 수행할 수 있습니다. 이 안내서에서는 AWS CloudHSM JCE 제공자를 JCE 제공자라고도 합니다. JCE 공급자와 JDK를 사용하여 암호화 작업을 HSM으로 오프로드하십시오. 문제 해결에 대한 내용은 을 참조하십시오. JCE SDK의 알려진 문제

Client SDK 3 사용에 대한 자세한 내용은 이전 클라이언트 SDK (클라이언트 SDK 3) 단원을 참조하십시오.

주제

클라이언트 SDK 5용 AWS CloudHSM JCE 공급자 설치 및 사용

JCE 제공자는 OpenJDK 8, OpenJDK 11, OpenJDK 17 및 OpenJDK 21과 호환됩니다. OpenJDK 웹사이트에서 둘 다 다운로드할 수 있습니다.

> **Note**
>
> 클라이언트 SDK 5를 사용하여 단일 HSM 클러스터를 실행하려면 먼저 `disable_key_availability_check`을 `True`로 설정하여 클라이언트 키 내구성 설정을 관리해야 합니다. 자세한 내용은 키 동기화 및 클라이언트 SDK 5 구성 도구를 참조하십시오.

주제

JCE 공급자 설치하기

1. 다음 명령을 사용하여 JCE 공급자를 다운로드하고 설치합니다.

   Amazon Linux 2

   x86_64 아키텍처에 아마존 리눅스 2용 JCE 공급자를 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el7.x86_64.rpm
```

ARM64 아키텍처에 아마존 리눅스 2용 JCE 공급자를 설치합니다.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el7.aarch64.rpm
```

Amazon Linux 2023

x86_64 아키텍처에 아마존 리눅스 2023용 JCE 공급자를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-jce-latest.amzn2023.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.amzn2023.x86_64.rpm
```

ARM64 아키텍처에 아마존 리눅스 2023용 JCE 공급자를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Amzn2023/cloudhsm-jce-latest.amzn2023.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.amzn2023.aarch64.rpm
```

CentOS 7 (7.8+)

x86_64 아키텍처에 CentOS 7용 JCE 공급자를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el7.x86_64.rpm
```

RHEL 7 (7.8+)

x86_64 아키텍처에 RHEL 7용 JCE 공급자를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el7.x86_64.rpm
```

RHEL 8 (8.3+)

x86_64 아키텍처에 RHEL 8용 JCE 공급자를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el8.x86_64.rpm
```

RHEL 9 (9.2+)

x86_64 아키텍처에 RHEL 9 (9.2+) 용 JCE 공급자를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-jce-latest.el9.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el9.x86_64.rpm
```

ARM64 아키텍처에 RHEL 9 (9.2+) 용 JCE 공급자를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL9/cloudhsm-jce-latest.el9.aarch64.rpm
```

```
$ sudo yum install ./cloudhsm-jce-latest.el9.aarch64.rpm
```

Ubuntu 20.04 LTS

x86_64 아키텍처에 우분투 20.04 LTS용 JCE 공급자를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Focal/
cloudhsm-jce_latest_u20.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u20.04_amd64.deb
```

Ubuntu 22.04 LTS

x86_64 아키텍처에 우분투 22.04 LTS용 JCE 공급자를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-jce_latest_u22.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u22.04_amd64.deb
```

ARM64 아키텍처에 우분투 22.04 LTS용 JCE 공급자를 설치하십시오.

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Jammy/
cloudhsm-jce_latest_u22.04_arm64.deb
```

```
$ sudo apt install ./cloudhsm-jce_latest_u22.04_arm64.deb
```

Windows Server 2016

x86_64 아키텍처에 Windows Server 2016용 JCE 공급자를 설치하고 관리자 권한으로 열고 다음 PowerShell 명령을 실행합니다.

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/
Windows/AWSCloudHSMJCE-latest.msi -Outfile C:\AWSCloudHSMJCE-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMJCE-
latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

Windows Server 2019

x86_64 아키텍처에 Windows Server 2019용 JCE 공급자를 설치하고 관리자 PowerShell 권한으로 열고 다음 명령을 실행합니다.

```
PS C:\> wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/
Windows/AWSCloudHSMJCE-latest.msi -Outfile C:\AWSCloudHSMJCE-latest.msi
```

```
PS C:\> Start-Process msiexec.exe -ArgumentList '/i C:\AWSCloudHSMJCE-
latest.msi /quiet /norestart /log C:\client-install.txt' -Wait
```

2. 부트스트래핑 클라이언트 SDK 5. 부트스트래핑에 대한 자세한 내용은 Client SDK 부트스트랩합니다.을 참조하십시오.
3. 다음 JCE 공급자 파일을 찾으십시오.

Linux

- `/opt/cloudhsm/java/cloudhsm-version.jar`
- `/opt/cloudhsm/bin/configure-jce`
- `/opt/cloudhsm/bin/jce-info`

Windows

- `C:\Program Files\Amazon\CloudHSM\java\cloudhsm-version.jar>`
- `C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe`
- `C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe`

JCE 공급자에게 자격 증명을 제공하십시오.

Java 애플리케이션이 HSM을 사용하려면 먼저 HSM이 애플리케이션을 인증해야 합니다. HSM은 명시적 로그인 또는 암시적 로그인 메서드를 사용하여 인증한다.

명시적 로그인 - 이 메서드를 사용하면 애플리케이션에서 직접 AWS CloudHSM 자격 증명을 제공할 수 있습니다. `AuthProvider`의 메서드를 사용하며, 여기서 CU 사용자 이름과 비밀번호를 핀 패턴으로 전달합니다. 자세한 내용은 HSM 코드 예로 로그인을 참조하십시오.

암시적 로그인 - 이 메서드를 사용하면 새 속성 파일, 시스템 속성 또는 환경 변수로 AWS CloudHSM 자격 증명을 설정할 수 있습니다.

- 시스템 속성 – 애플리케이션을 실행할 때 시스템 속성을 통해 자격 증명을 설정합니다. 다음 예제와 같이 이 작업을 두 가지 방법으로 수행할 수 있습니다.

  Linux

  ```
  $ java -DHSM_USER=<HSM user name> -DHSM_PASSWORD=<password>
  ```

  ```
  System.setProperty("HSM_USER","<HSM user name>");
  System.setProperty("HSM_PASSWORD","<password>");
  ```

  Windows

  ```
  PS C:\> java -DHSM_USER=<HSM user name> -DHSM_PASSWORD=<password>
  ```

  ```
  System.setProperty("HSM_USER","<HSM user name>");
  System.setProperty("HSM_PASSWORD","<password>");
  ```

- 환경 변수 – 자격 증명을 환경 변수로 설정합니다.

  Linux

  ```
  $ export HSM_USER=<HSM user name>
  $ export HSM_PASSWORD=<password>
  ```

  Windows

  ```
  PS C:\> $Env:HSM_USER="<HSM user name>"
  PS C:\> $Env:HSM_PASSWORD="<password>"
  ```

애플리케이션이 자격 증명을 제공하지 않거나 HSM이 세션을 인증하기 전에 작업을 시도하면 자격 증명을 사용할 수 없기도 합니다. 이러한 경우 Java용 CloudHSM 소프트웨어 라이브러리가 다음 순서로 자격 증명을 검색합니다.

1. 시스템 속성
2. 환경 변수

JCE 공급자의 키 관리 기본 사항

JCE 공급자의 키 관리에 대한 기본 사항은 키 가져오기, 키 내보내기, 핸들별 키 로드 또는 키 삭제를 포함합니다. 키 관리에 대한 자세한 내용은 Manage keys 코드 예제를 참조하십시오.

또한 코드 샘플에서 JCE 제공자 코드 예시를 더 확인할 수 있습니다.

JCE 제공자가 지원하는 키 유형

Java용 AWS CloudHSM 소프트웨어 라이브러리를 사용하면 다음과 같은 키 유형을 생성할 수 있습니다.

키 유형	설명
AES	128, 192 및 256비트 AES 키를 생성합니다.
트리플 DES(3DES, DESede)	192비트 트리플 DES 키 생성 예정된 변경 사항은 각주를 1 참조하십시오.
EC	EC 키 쌍 생성하기 — NIST curves secp224r1 (P-224), secp256r1 (P-256), secp256k1 (블록체인), secp384r1 (P-384) 및 secp521r1 (P-521).
GENERIC_SECRET	1~800바이트의 일반 보안을 생성합니다.
HMAC	SHA1, SHA224, SHA256, SHA384, SHA512에 대한 해시 지원.
RSA	2048비트~4096비트의 RSA 키를 생성합니다 (256비트 증가).

[1] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에서는 이 설정이 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단를 참조하세요.

JCE 제공자를 위한 지원 메커니즘

이 항목에서는 클라이언트 SDK 5를 사용하는 JCE 공급자가 지원하는 메커니즘에 대한 정보를 제공합니다. 에서 지원하는 JCA (Java 암호화 아키텍처) 인터페이스 및 엔진 클래스에 대한 자세한 내용은 다음 항목을 참조하십시오. AWS CloudHSM

주제

키 및 키 페어 함수 생성

Java용 AWS CloudHSM 소프트웨어 라이브러리에서는 키 및 키 쌍 생성 함수를 위해 다음 작업을 사용할 수 있습니다.

- `RSA`
- `EC`
- `AES`
- `DESede (Triple DES)`[1] 참고 참조
- `GenericSecret`

암호 함수

Java용 AWS CloudHSM 소프트웨어 라이브러리는 다음과 같은 알고리즘, 모드 및 패딩 조합을 지원합니다.

알고리즘	Mode	패딩	참고
AES	CBC	AES/CBC/NoPadding AES/CBC/PKCS5Padding	Cipher.ENCRYPT_MODE 와 Cipher.DECRYPT_MODE 를 구현합니다. Cipher.UNWRAP_MODE for AES/CBC NoPadding 구현
AES	ECB	AES/ECB/PKCS5Padding AES/ECB/NoPadding	Cipher.ENCRYPT_MODE 와 Cipher.DECRYPT_MODE 를 구현합니다.
AES	CTR	AES/CTR/NoPadding	Cipher.ENCRYPT_MODE 와 Cipher.DECRYPT_MODE 를 구현합니다.
AES	GCM	AES/GCM/NoPadding	Cipher.WRAP_MODE , Cipher.UNWRAP_MODE , Cipher.ENCRYPT_MODE , Cipher.DECRYPT_MODE 를 구현합니다. AES-GCM 암호화를 수행할 때 HSM은 요청에서 초기화 벡터

알고리즘	Mode	패딩	참고
			(IV)를 무시하고 HSM이 생성하는 IV를 사용합니다. 작업이 완료되면 `Cipher.getIV()` 를 호출하여 IV를 가져와야 합니다.

AESWrap	ECB	`AESWrap/ECB/NoPadding`
		
		`AESWrap/ECB/PKCS5Padding`
		
		`AESWrap/ECB/ZeroPadding`
			`Cipher.WRAP_MODE` 와 `Cipher.UNWRAP_MODE` 를 구현합니다.

DESede (Triple DES)	CBC	`DESede/CBC/PKCS5Padding`
		
		`DESede/CBC/NoPadding`
			`Cipher.ENCRYPT_MODE` 와 `Cipher.DECRYPT_MODE` 를 구현합니다. 예정된 변경 사항은 1 아래 참고를 참조하세요.

DESede (Triple DES)	ECB	`DESede/ECB/NoPadding`
		
		`DESede/ECB/PKCS5Padding`
			`Cipher.ENCRYPT_MODE` 와 `Cipher.DECRYPT_MODE` 를 구현합니다. 예정된 변경 사항은 1 아래 참고를 참조하세요.

알고리즘	Mode	패딩	참고
RSA	ECB	RSA/ECB/PKCS1Padding 참조 1 RSA/ECB/OAEPPadding RSA/ECB/OAEPWithSHA-1ANDMGF1Padding RSA/ECB/OAEPWithSHA-224ANDMGF1Padding RSA/ECB/OAEPWithSHA-256ANDMGF1Padding RSA/ECB/OAEPWithSHA-384ANDMGF1Padding RSA/ECB/OAEPWithSHA-512ANDMGF1Padding	참고 Cipher.WRAP_MODE , Cipher.UNWRAP_MODE , Cipher.ENCRYPT_MODE , Cipher.DECRYPT_MODE 를 구현합니다.

알고리즘	Mode	패딩	참고
RSA	ECB	RSA/ECB/NoPadding	Cipher.ENCRYPT_MODE 와 Cipher.DECRYPT_MODE 를 구현합니다.
RSAAESWrap	ECB	RSAAESWrap/ECB/OAEPPadding RSAAESWrap/ECB/OAEPWithSHA-1ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-224ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-256ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-384ANDMGF1Padding RSAAESWrap/ECB/OAEPWithSHA-512ANDMGF1Padding	Cipher.WRAP_MODE 와 Cipher.UNWRAP_MODE 를 구현합니다.

서명 및 인증 기능

Java용 AWS CloudHSM 소프트웨어 라이브러리는 다음 유형의 서명 및 검증을 지원합니다. 클라이언트 SDK 5와 해싱 기능이 있는 서명 알고리즘을 사용하면 데이터가 서명/검증을 위해 HSM으로 전송되기 전에 소프트웨어에서 로컬로 해시됩니다. 즉, SDK로 해시할 수 있는 데이터 크기에는 제한이 없습니다.

RSA 서명 유형

- NONEwithRSA
- RSASSA-PSS
- SHA1withRSA
- SHA1withRSA/PSS
- SHA1withRSAandMGF1
- SHA224withRSA
- SHA224withRSAandMGF1
- SHA224withRSA/PSS
- SHA256withRSA
- SHA256withRSAandMGF1
- SHA256withRSA/PSS
- SHA384withRSA
- SHA384withRSAandMGF1
- SHA384withRSA/PSS
- SHA512withRSA
- SHA512withRSAandMGF1
- SHA512withRSA/PSS

ECDSA 서명 유형

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA

- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

다이제스트 기능

Java용 AWS CloudHSM 소프트웨어 라이브러리는 다음 메시지 다이제스트를 지원합니다. Client SDK 5를 사용하면 데이터가 소프트웨어에서 로컬로 해시됩니다. 즉, SDK로 해시할 수 있는 데이터 크기에는 제한이 없습니다.

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

해시 기반 메시지 인증 코드(HMAC) 함수

Java용 AWS CloudHSM 소프트웨어 라이브러리는 다음 HMAC 알고리즘을 지원합니다.

- HmacSHA1(최대 데이터 크기(바이트): 16288)
- HmacSHA224(최대 데이터 크기(바이트): 16256)
- HmacSHA256(최대 데이터 크기(바이트): 16288)
- HmacSHA384(최대 데이터 크기(바이트): 16224)
- HmacSHA512(최대 데이터 크기(바이트): 16224)

암호 기반 메시지 인증 코드(CMAC) 함수

CMAC(암호 기반 메시지 인증 코드) 는 블록 암호와 비밀 키를 사용하여 메시지 인증 코드(MAC) 를 생성합니다. MAC에서는 해싱 방법이 아닌 블록 대칭 키 방법을 사용한다는 점에서 HMAC과 다릅니다.

Java용 AWS CloudHSM 소프트웨어 라이브러리는 다음 CMAC 알고리즘을 지원합니다.

- AESCMAC

키 팩토리를 사용하여 키를 키 사양으로 변환합니다.

키 팩토리를 사용하여 키를 키 사양으로 변환할 수 있습니다. AWS CloudHSM JCE에는 두 가지 유형의 키 팩토리가 있습니다.

SecretKeyFactory: 대칭 키를 가져오거나 파생하는 데 사용됩니다. 를 사용하여 SecretKeyFactory 지원되는 키 또는 대칭 키를 가져오거나 파생할 수 KeySpec 있도록 지원되는 키를 전달할 수 있습니다. AWS CloudHSM지원되는 사양은 다음과 같습니다. KeyFactory

- SecretKeyFactoryFor의 `generateSecret` 메서드는 다음과 같은 KeySpec클래스가 지원됩니다.
  - KeyAttributesMap추가 속성이 있는 키 바이트를 CloudHSM 키로 가져오는 데 사용할 수 있습니다. 예제는 여기에서 확인할 수 있습니다.
  - SecretKeySpec대칭 키 사양을 CloudHSM 키로 가져오는 데 사용할 수 있습니다.
  - AesCmacKdfParameterSpec다른 CloudHSM AES 키를 사용하여 대칭 키를 도출하는 데 사용할 수 있습니다.

> **Note**
>
> SecretKeyFactory의 `translateKey` 메서드는 키 인터페이스를 구현하는 모든 키를 사용합니다.

KeyFactory: 비대칭 키를 가져오는 데 사용됩니다. 를 사용하여 KeyFactory 지원되는 키 또는 비대칭 키 가져오기가 지원되는 KeySpec 키를 전달할 수 있습니다. AWS CloudHSM자세한 정보는 다음 리소스를 참조하세요.

- KeyFactoryFor의 `generatePublic` 메서드에는 다음과 같은 KeySpec클래스가 지원됩니다.
- 다음을 포함하여 RSA 및 EC 모두를 위한 KeyAttributesMap CloudHSM KeyTypes
  - RSA 및 EC 퍼블릭 모두를 위한 KeyAttributesMap CloudHSM. KeyTypes 예제는 여기에서 확인할 수 있습니다.
  - RSA 및 EC 퍼블릭 키 EncodedKeySpec 모두에 사용할 수 있는 X509
  - RSA 퍼블릭 키용 RSA PublicKeySpec
  - EC 퍼블릭 PublicKeySpec 키용 EC
- KeyFactoryFor의 `generatePrivate` 메서드에는 다음 KeySpec클래스가 지원됩니다.
- 다음을 포함하여 RSA 및 EC 모두를 위한 KeyAttributesMap CloudHSM KeyTypes

- RSA 및 EC 퍼블릭 모두를 위한 KeyAttributesMap CloudHSM. KeyTypes 예제는 여기에서 확인할 수 있습니다.
- EC 및 EncodedKeySpec RSA 프라이빗 키 모두에 사용할 수 있는 PKCS8
- RSA 프라이빗 키를 PrivateCrtKeySpec 위한 RSA
- EC 프라이빗 PrivateKeySpec 키용 EC

KeyFactoryFor의 `translateKey` 메서드는 키 인터페이스를 구현하는 모든 키를 받아들입니다.

메커니즘 주석

[1] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에서는 이 기능이 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단를 참조하세요.

지원되는 Java 키 속성

이 항목에서는 클라이언트 SDK 5에 지원되는 Java 키 속성에 대한 정보를 제공합니다. 이 항목에서는 JCE 공급자가 독점 확장을 사용하여 주요 속성을 설정하는 방법에 대해 설명합니다. 이 확장을 사용하여 다음 작업 중에 지원되는 키 속성 및 해당 값을 설정할 수 있습니다:

- 키 생성
- 키 가져오기

키 속성 사용 방법에 대한 예는 the section called “코드 샘플”을 참조하십시오.

주제

- 속성 이해
- 지원되는 속성
- 키에 대한 속성 설정

속성 이해

키 속성을 사용하여 퍼블릭 키, 개인 키 또는 보안 키를 포함하여 키 객체에 허용되는 작업을 지정합니다. 키 객체 생성 작업 중에 키 속성과 값을 정의합니다.

Java Cryptography Extension(JCE)은 키 속성에 대한 값을 설정하는 방법을 지정하지 않으므로 대부분의 작업이 기본적으로 허용되었습니다. 이와 반대로 PKCS# 11 표준은 더 제한적인 기본값을 가진

포괄적인 속성 집합을 정의합니다. JCE 제공자 3.1부터는 일반적으로 사용되는 속성에 대해 보다 제한적인 값을 설정할 수 있는 전용 확장을 AWS CloudHSM 제공합니다.

지원되는 속성

다음 표에 나열된 속성에 대한 값을 설정할 수 있습니다. 제한적으로 만들려는 속성의 값만 설정하는 것이 좋습니다. 값을 지정하지 않는 경우 는 아래 표에 지정된 기본값을 AWS CloudHSM 사용합니다. 기본값 열의 빈 셀은 속성에 할당된 특정 기본값이 없음을 나타냅니다.

속성	기본 값			참고
	대칭 키	키 페어의 공개 키	키 페어의 개인 키	
DECRYPT	TRUE		TRUE	True는 키를 사용하여 버퍼의 암호를 해독할 수 있음을 나타냅니다. 일반적으로 WRAP가 true로 설정된 키에 대해 이를 FALSE로 설정합니다.
DERIVE				키를 사용하여 다른 키를 파생할 수 있습니다.
ENCRYPT	TRUE	TRUE		True는 키를 사용하여 버퍼를 암호화할 수 있음을 나타냅니다.
EXTRACTABLE	TRUE		TRUE	True는 이 키를 HSM에서 내보낼 수 있음을 나타냅니다.

속성	기본 값			참고
	대칭 키	키 페어의 공개 키	키 페어의 개인 키	
ID				키를 식별하는 데 사용되는 사용자 정의 값입니다.
KEY_TYPE				키 유형(AES, DESede, 일반 암호, EC 또는 RSA)을 식별하는 데 사용됩니다.
LABEL				HSM의 키를 편리하게 식별할 수 있는 사용자 정의 문자열입니다. 모범 사례를 따르려면 나중에 쉽게 찾을 수 있도록 각 키에 고유한 레이블을 사용하십시오.
LOCAL				HSM에서 생성한 키를 나타냅니다.
OBJECT_CLASS				키 (SecretKey, PublicKey 또는 PrivateKey) 의 객체 클래스를 식별하는 데 사용됩니다.

속성	기본 값			참고
	대칭 키	키 페어의 공개 키	키 페어의 개인 키	
PRIVATE	TRUE	TRUE	TRUE	True는 사용자가 인증될 때까지 사용자가 키에 액세스할 수 없음을 나타냅니다. 명확히 하자면, 이 속성이 FALSE로 설정되어 있더라도 사용자는 AWS CloudHSM 인증되기 전까지는 어떤 키에도 액세스할 수 없습니다.
SIGN	TRUE		TRUE	True는 키를 사용하여 메시지 다이제스트에 서명할 수 있음을 나타냅니다. 일반적으로 아카이브 완료된 개인 키와 퍼블릭 키의 경우 FALSE로 설정됩니다.

속성	기본 값			참고
	대칭 키	키 페어의 공개 키	키 페어의 개인 키	
SIZE				키 크기를 정의하는 속성입니다. 지원되는 키 크기에 대한 자세한 내용은 Client SDK 5 지원 메커니즘을 참조하십시오.
TOKEN	FALSE	FALSE	FALSE	클러스터의 모든 HSM에 복제되고 백업에 포함된 영구 키입니다. TOKEN = FALSE는 HSM에 대한 연결이 끊어지거나 로그아웃될 때 자동으로 지워지는 임시 키를 의미합니다.
UNWRAP	TRUE		TRUE	True는 키를 사용하여 다른 키를 언래핑(가져오기)할 수 있음을 나타냅니다.

속성	기본 값			참고
	대칭 키	키 페어의 공개 키	키 페어의 개인 키	
VERIFY	TRUE	TRUE		True는 키를 사용하여 서명을 확인할 수 있음을 나타냅니다. 일반적으로 개인 키의 경우 FALSE로 설정됩니다.
WRAP	TRUE	TRUE		True는 키를 사용하여 다른 키를 래핑할 수 있음을 나타냅니다. 일반적으로 개인 키의 경우 FALSE로 설정됩니다.

속성	기본 값			참고
	대칭 키	키 페어의 공개 키	키 페어의 개인 키	
WRAP_WITH_TRUSTED	FALSE		FALSE	True는 TRUSTED 속성이 true로 설정된 키로만 키를 래핑하고 래핑을 해제할 수 있음을 나타냅니다. 키가 true로 WRAP_WITH_TRUSTED 설정되면 해당 속성은 읽기 전용이며 false로 설정할 수 없습니다. 신뢰 래핑에 대해 알아보려면 신뢰할 수 있는 키를 사용하여 키 래핑 해제 제어를 참조하세요.

> **Note**
>
> PKCS #11 라이브러리의 속성에 대한 광범위한 지원을 받을 수 있습니다. 자세한 내용은 지원되는 PKCS #11 속성을 참조하십시오.

키에 대한 속성 설정

KeyAttributesMap은 Java Map과 유사한 객체로, 키 객체에 대한 속성 값을 설정하는 데 사용할 수 있습니다. KeyAttributesMap 함수에 대한 메서드는 Java Map 조작에 사용되는 메서드와 비슷합니다.

속성에 대한 사용자 지정 값을 설정하기 위해 다음 두 가지 옵션이 제공됩니다.

- 다음 표에 나열된 메서드 사용
- 이 문서 뒷부분에 설명된 빌더 패턴 사용

속성 맵 객체는 속성을 설정하기 위한 다음과 같은 메서드를 지원합니다:

Operation	반환 값	**KeyAttributesMap** 메서드
기존 키에 대한 키 속성 값 가져오기	객체(값 포함) 또는 null	get(keyAttribute)
키 속성 하나의 값 채우기	키 속성과 연결된 이전 값 또는 키 속성에 대한 매핑이 없는 경우에는 null	put(keyAttribute, 값)
여러 키 속성에 대한 값 채우기	N/A	PuTall () keyAttributesMap
속성 맵에서 키-값 페어 제거	키 속성과 연결된 이전 값 또는 키 속성에 대한 매핑이 없는 경우에는 null	remove(keyAttribute)

> **Note**
>
> 명시적으로 지정하지 않은 속성은 the section called “지원되는 속성”의 이전 표에 나열된 기본 값으로 설정됩니다.

키 페어에 대한 속성 설정

Java 클래스 `KeyPairAttributesMap`을 사용하여 키 페어에 대한 키 속성을 처리합니다. `KeyPairAttributesMap`은 두 개의 `KeyAttributesMap` 객체를 캡슐화합니다. 하나는 퍼블릭 키용이고 다른 하나는 개인 키용입니다.

퍼블릭 키와 개인 키에 대해 별도로 개별 속성을 설정하려면 키의 해당하는 `KeyAttributes` 맵 객체에 `put()` 메서드를 사용하면 됩니다. `getPublic()` 메서드를 사용하여 공개 키에 대한 속성 맵을 검색하고 `getPrivate()`을 사용하여 개인 키에 대한 속성 맵을 검색합니다. 키 페어 속성 맵에 `putAll()`을 인수로 사용하여 퍼블릭 및 개인 키 페어에 대한 여러 키 속성 값을 함께 채웁니다.

Java용 AWS CloudHSM 소프트웨어 라이브러리의 코드 샘플

이 항목에서는 클라이언트 SDK 5의 Java 코드 샘플에 대한 리소스와 정보를 제공합니다.

사전 조건

샘플을 실행하기 전에 사용자 환경을 설정해야 합니다.

- Java 암호화 확장(Java Cryptographic Extension) 공급자를 설치하고 구성합니다.
- 유효한 HSM 사용자 이름과 암호를 설정합니다. 이러한 작업을 수행하기 위해서는 CU(Cryptographic User) 권한이면 충분합니다. 각 예제에서 애플리케이션은 이러한 자격 증명을 사용하여 HSM에 로그인합니다.
- JCE 공급자에게 자격 증명을 제공하는 방법을 결정하십시오.

코드 샘플

다음 코드 샘플은 AWS CloudHSM JCE 공급자를 사용하여 기본 태스크를 수행하는 방법을 보여줍니다. 에서 더 많은 코드 샘플을 사용할 수 있습니다. GitHub

- HSM에 로그인
- 키 관리
- 대칭 키 생성하기
- 비대칭 키 생성하기
- AES GCM을 사용하여 암호화 및 해독
- AES-CTR을 사용하여 암호화 및 해독
- DESede-ECB로 암호화 및 해독[1] (1 참고 참조)
- RSA 키로 서명 및 확인
- EC 키를 사용한 서명 및 확인
- 지원되는 키 속성 사용
- CloudHSM 키 스토어 사용

[1] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에는 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단를 참조하세요.

AWS CloudHSM JCE 프로바이더 자바독스

JCE 공급자 Javadocs를 사용하여 AWS CloudHSM JCE SDK에 정의된 Java 유형 및 메서드에 대한 사용 정보를 얻을 수 있습니다. 최신 Javadocs를 AWS CloudHSM다운로드하려면 다운로드 페이지의 최신 릴리스 섹션을 참조하십시오.

Javadocs를 통합 개발 환경(IDE)으로 가져오거나 웹 브라우저에서 볼 수 있습니다.

AWS CloudHSM KeyStore 자바 클래스 사용

이 AWS CloudHSM `KeyStore` 클래스는 특수 목적의 PKCS12 키 스토어를 제공합니다. 이 키 스토어는 키 데이터와 함께 인증서를 저장하고 AWS CloudHSM에 저장된 키 데이터와 상호 연관시킬 수 있습니다. 이 AWS CloudHSM `KeyStore` 클래스는 JCE (Java 암호화 확장) 의 SPI (`KeyStore`서비스 공급자 인터페이스) 를 구현합니다. 사용에 `KeyStore` 대한 자세한 내용은 클래스를 참조하십시오. KeyStore

> **Note**
>
> 인증서는 공개 정보이며 암호화 키의 저장 용량을 극대화하기 위해 HSM에 인증서를 저장하는 AWS CloudHSM 것을 지원하지 않습니다.

적절한 키 스토어 선택

AWS CloudHSM 자바 암호화 확장 (JCE) 공급자는 특수 목적의 AWS CloudHSM을 제공합니다. KeyStore 이 AWS CloudHSM `KeyStore` 클래스는 주요 작업을 HSM으로 오프로드하고, 인증서를 로컬로 저장하고, 인증서 기반 작업을 수행할 수 있도록 지원합니다.

다음과 같이 특수 목적의 CloudHSM을 로드합니다. KeyStore

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

초기화 중 AWS CloudHSM KeyStore

JCE AWS CloudHSM KeyStore 제공자에 로그인하는 것과 같은 방법으로 로그인합니다. 환경 변수 또는 시스템 속성 파일을 사용할 수 있으며, KeyStore CloudHSM을 사용하기 전에 로그인해야 합니다. JCE 공급자를 사용하여 HSM에 로그인하는 예는 HSM에 로그인을 참조하십시오.

원하는 경우, 암호를 지정하여 키 스토어 데이터를 보유하는 로컬 PKCS12 파일을 암호화할 수 있습니다. AWS CloudHSM 키 저장소를 생성할 때 암호를 설정하고 load, set 및 get 메서드를 사용할 때 암호를 입력합니다.

다음과 같이 새 CloudHSM 객체를 인스턴스화합니다. KeyStore

```
ks.load(null, null);
```

`store` 메서드를 사용하여 파일에 키 스토어 데이터를 씁니다. 이 시점부터 다음과 같이 소스 파일 및 암호와 함께 `load` 메서드를 사용하여 기존 키 스토어를 로드할 수 있습니다.

```
ks.load(inputStream, password);
```

사용 AWS CloudHSM KeyStore

AWS CloudHSM KeyStore JCE 클래스 KeyStore 사양을 준수하며 다음 기능을 제공합니다.

- `load`

  지정된 입력 스트림에서 키 스토어를 로드합니다. 키 스토어를 저장할 때 암호가 설정된 경우, 로드가 성공하려면 이와 동일한 암호가 제공되어야 합니다. 새로운 빈 키 스토어를 초기화하려면 두 파라미터를 모두 null로 설정합니다.

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  ks.load(inputStream, password);
  ```

- `aliases`

  지정된 키 스토어 인스턴스에 있는 모든 항목의 별칭 이름의 열거를 반환합니다. 결과는 PKCS12 파일에 로컬로 저장된 객체와 HSM에 있는 객체를 포함합니다.

  샘플 코드:

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();) {
      String label = entry.nextElement();
      System.out.println(label);
  }
  ```

- `containsalias`

키 스토어가 지정된 별칭이 있는 객체에 최소 하나 이상 액세스할 수 있는 경우 true를 반환합니다. 키 스토어는 PKCS12 파일에 로컬로 저장된 객체와 HSM에 상주하는 객체를 확인합니다.

- `deleteEntry`

  로컬 PKCS12 파일에서 인증서 항목을 삭제합니다. 를 사용하여 HSM에 저장된 주요 데이터를 삭제할 수 없습니다. AWS CloudHSM KeyStore Destroyable 인터페이스의 `destroy` 방법을 사용하여 키를 삭제할 수 있습니다.

  ```
  ((Destroyable) key).destroy();
  ```

- `getCertificate`

  사용 가능한 경우 별칭과 연결된 인증서를 반환합니다. 별칭이 없거나 인증서가 아닌 객체를 참조하는 경우, 함수는 NULL을 반환합니다.

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  Certificate cert = ks.getCertificate(alias);
  ```

- `getCertificateAlias`

  데이터가 지정된 인증서와 일치하는 첫 번째 키 스토어 항목의 이름(별칭)을 반환합니다.

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  String alias = ks.getCertificateAlias(cert);
  ```

- `getCertificateChain`

  지정된 별칭과 연결된 인증서 체인을 반환합니다. 별칭이 없거나 인증서가 아닌 객체를 참조하는 경우, 함수는 NULL을 반환합니다.

- `getCreationDate`

  지정된 별칭에 의해 식별된 항목의 생성 날짜를 반환합니다. 생성 날짜를 사용할 수 없는 경우 함수는 인증서가 유효해지는 날짜를 반환합니다.

- `getKey`

  GetKey HSM으로 전달되고 지정된 레이블에 해당하는 키 객체를 반환합니다. HSM을 `getKey` 직접 쿼리하므로 HSM에서 키를 생성했는지 여부에 관계없이 HSM의 모든 키에 사용할 수 있습니다. KeyStore

```
Key key = ks.getKey(keyLabel, null);
```

- `isCertificateEntry`

  지정된 별칭이 있는 항목이 인증서 항목을 나타내는지 확인합니다.

- `isKeyEntry`

  지정된 별칭이 있는 항목이 키 항목을 나타내는지 확인합니다. 이 작업은 PKCS12 파일과 HSM에서 모두 별칭을 검색합니다.

- `setCertificateEntry`

  지정된 인증서를 지정된 별칭에 할당합니다. 지정된 별칭이 키 또는 인증서를 식별하는 데 이미 사용 중인 경우 `KeyStoreException`이 발생합니다. JCE 코드를 사용하여 키 개체를 가져온 다음 KeyStore `SetKeyEntry` 메서드를 사용하여 인증서를 키에 연결할 수 있습니다.

- `byte[]` 키가 있는 `setKeyEntry`

  이 API는 현재 클라이언트 SDK 5에서 지원되지 않습니다.

- `Key` 객체가 있는 `setKeyEntry`

  지정된 키를 지정된 별칭에 할당하고 HSM 내부에 저장합니다. 키가 HSM 내에 아직 없는 경우 추출 가능한 세션 키로서 HSM에 가져옵니다.

  `Key` 객체가 `PrivateKey` 유형인 경우 해당 인증서 체인이 함께 제공되어야 합니다.

  별칭이 이미 존재하는 경우 `SetKeyEntry` 호출은 `KeyStoreException`을 발생시키고 키를 덮어쓰지 못하게 합니다. 키를 덮어써야 하는 경우 해당 목적으로 KMU 또는 JCE를 사용하십시오.

- `engineSize`

  키 스토어의 항목 수를 반환합니다.

- `store`

  키 스토어를 지정된 출력 스트림에 PKCS12 파일로 저장하고 지정된 암호로 보호합니다. 또한, 모든 로드된 키(`setKey` 호출을 사용하여 설정됨)를 유지합니다.

JCE 제공자를 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션

이 주제를 사용하여 JCE 제공자를 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션할 수 있습니다. 마이그레이션의 이점은 을 참조하십시오. Client SDK 5의 이점

에서 AWS CloudHSM고객 애플리케이션은 AWS CloudHSM 클라이언트 소프트웨어 개발 키트 (SDK)를 사용하여 암호화 작업을 수행합니다. 클라이언트 SDK 5는 계속해서 새로운 기능과 플랫폼 지원이 추가되는 기본 SDK입니다.

클라이언트 SDK 3 JCE 공급자는 표준 JCE 사양에 포함되지 않은 사용자 지정 클래스와 API를 사용합니다. JCE 공급자용 클라이언트 SDK 5는 JCE 사양을 준수하며 특정 영역에서 클라이언트 SDK 3과 이전 버전으로는 호환되지 않습니다. 고객 애플리케이션을 클라이언트 SDK 5로 마이그레이션하는 과정에서 변경이 필요할 수 있습니다. 이 섹션에서는 성공적인 마이그레이션에 필요한 변경 사항을 간략하게 설명합니다.

모든 공급자에 대한 마이그레이션 지침을 검토하려면 을 참조하십시오클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션.

주제

주요 변경 사항을 해결하여 대비하십시오.

이러한 주요 변경 사항을 검토하고 그에 따라 개발 환경에서 애플리케이션을 업데이트하십시오.

Provider 클래스와 이름이 변경되었습니다.

변경된 내용	클라이언트 SDK 3에 포함된 내용	클라이언트 SDK 5의 내용	예
제공자 클래스 및 이름	클라이언트 SDK 3의 JCE 제공자 클래스가 `CaviumProvider` 호출되며 제공자 이름을 가집니다. `Cavium`	클라이언트 SDK 5에서는 공급자 클래스가 `CloudHsmProvider` 호출되며 공급자 이름을 가집니다. `CloudHSM`	`CloudHsmProvider` 객체를 초기화하는 방법의 예는 AWS CloudHSM GitHub 샘플 리포지토

변경된 내용	클라이언트 SDK 3에 포함된 내용	클라이언트 SDK 5의 내용	예
			리에서 확인할 수 있습니다.

명시적 로그인은 변경되었지만 암시적 로그인은 변경되지 않았습니다.

변경된 사항	클라이언트 SDK 3에 포함된 내용	클라이언트 SDK 5의 내용	예
명시적 로그인	클라이언트 SDK 3은 `LoginManager` 클래스를 사용하여 명시적 로그인을 수행합니다. [1]	클라이언트 SDK 5에서 `CloudHSM` 공급자는 명시적 로그인을 구현합니다. `AuthProvider` `AuthProvider` 표준 Java 클래스이며 Java의 관용적 방법을 따라 제공자에 로그인합니다. Client SDK 5의 향상된 로그인 상태 관리를 통해 애플리케이션은 더 이상 재연결 중에 로그인을 모니터링하고 수행할 필요가 없습니다. [2]	클라이언트 SDK 5에서 명시적 로그인을 사용하는 방법에 대한 예는 AWS GitHub CloudHSM LoginRunner 샘플 리포지토리의 샘플을 참조하십시오.
암시적 로그인	암시적 로그인에는 변경이 필요하지 않습니다. 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션할 때 동일한 속성 파일 및 모든 환경 변수가 암시적 로그인에 계속 적용됩니다.		클라이언트 SDK 5에서 암시적 로그인을 사용하는 방법에 대한 예제는 샘플 저장소의 샘플을 참조하십시오. LoginRunner AWS CloudHSM GitHub

- [1] 클라이언트 SDK 3 코드 스니펫:

```
LoginManager lm = LoginManager.getInstance();

lm.login(partition, user, pass);
```

- [2] 클라이언트 SDK 5 코드 스니펫:

```
// Construct or get the existing provider object
AuthProvider provider = new CloudHsmProvider();

// Call login method on the CloudHsmProvider object
// Here loginHandler is a CallbackHandler
provider.login(null, loginHandler);
```

Client SDK 5에서 명시적 로그인을 사용하는 방법에 대한 예제는 LoginRunner 샘플 저장소의 샘플을 참조하십시오. AWS CloudHSM GitHub

키 생성이 변경되었습니다.

무엇이 변경되었나요?	클라이언트 SDK 3에 포함된 내용	클라이언트 SDK 5의 내용	예
키 생성	클라이언트 SDK 3에서는 키 생성 매개변수를 지정하는 데 `Cavium[Key-type]AlgorithmParameterSpec` 사용됩니다. 코드 스니펫은 각주를 참조하십시오. 1	클라이언트 SDK 5에서는 키 생성 속성을 지정하는 데 `KeyAttributesMap` 사용됩니다. 코드 스니펫은 각주를 참조하십시오. 2	대칭 키를 생성하는 `KeyAttributesMap` 데 사용하는 방법에 대한 예는 AWS CloudHSM SymmetricKeys Github 샘플 리포지토리의 샘플을 참조하십시오.
키 페어 생성	클라이언트 SDK 3에서는 `Cavium[Key-type]Algorithmparameterspec` 키	클라이언트 SDK 5에서는 `KeyPairAttributesMap` 이러한 매개변수를 지정하는 데 사용됩니다.	비대칭 키를 생성하는 `KeyAttributesMap` 데 사용하는 방법에 대한 예제는 AsymmetricKeys

무엇이 변경되었나요?	클라이언트 SDK 3에 포함된 내용	클라이언트 SDK 5의 내용	예
	페어 생성 파라미터를 지정하는 데 사용됩니다. 코드 스니펫은 각주를 참조하십시오. 3	코드 스니펫은 각주를 참조하십시오. 4	샘플 저장소의 샘플을 참조하십시오. AWS CloudHSM GitHub

- [1] 클라이언트 SDK 3 키 생성 코드 스니펫:

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
CaviumAESKeyGenParameterSpec aesSpec = new CaviumAESKeyGenParameterSpec(
keySizeInBits,
keyLabel,
isExtractable,
isPersistent);
keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [2] 클라이언트 SDK 5 키 생성 코드 스니펫:

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES",
CloudHsmProvider.PROVIDER_NAME);

final KeyAttributesMap aesSpec = new KeyAttributesMap();
aesSpec.put(KeyAttribute.LABEL, keyLabel);
aesSpec.put(KeyAttribute.SIZE, keySizeInBits);
aesSpec.put(KeyAttribute.EXTRACTABLE, isExtractable);
aesSpec.put(KeyAttribute.TOKEN, isPersistent);

keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [3] 클라이언트 SDK 3 키 페어 생성 코드 스니펫:

```
KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("rsa", "Cavium");
CaviumRSAKeyGenParameterSpec spec = new CaviumRSAKeyGenParameterSpec(
keySizeInBits,
new BigInteger("65537"),
label + ":public",
label + ":private",
```

```
isExtractable,
isPersistent);

keyPairGen.initialize(spec);

keyPairGen.generateKeyPair();
```

- [4] 클라이언트 SDK 5 키 페어 생성 코드 스니펫:

```
KeyPairGenerator keyPairGen =
KeyPairGenerator.getInstance("RSA", providerName);

// Set attributes for RSA public key
final KeyAttributesMap publicKeyAttrsMap = new KeyAttributesMap();
publicKeyAttrsMap.putAll(additionalPublicKeyAttributes);
publicKeyAttrsMap.put(KeyAttribute.LABEL, label + ":Public");
publicKeyAttrsMap.put(KeyAttribute.MODULUS_BITS, keySizeInBits);
publicKeyAttrsMap.put(KeyAttribute.PUBLIC_EXPONENT,
new BigInteger("65537").toByteArray());

// Set attributes for RSA private key
final KeyAttributesMap privateKeyAttrsMap = new KeyAttributesMap();
privateKeyAttrsMap.putAll(additionalPrivateKeyAttributes);
privateKeyAttrsMap.put(KeyAttribute.LABEL, label + ":Private");

// Create KeyPairAttributesMap and use that to initialize the
// keyPair generator
KeyPairAttributesMap keyPairSpec =
new KeyPairAttributesMapBuilder()
.withPublic(publicKeyAttrsMap)
.withPrivate(privateKeyAttrsMap)
.build();

keyPairGen.initialize(keyPairSpec);
keyPairGen.generateKeyPair();
```

키 찾기, 삭제, 참조가 변경되었습니다.

이미 생성된 키를 찾으려면 를 AWS CloudHSM 사용해야 합니다. KeyStore 클라이언트 SDK 3에는 다음과 같은 두 가지 KeyStore 유형이 있습니다. `Cavium` `CloudHSM` 클라이언트 SDK 5에는 다음 한 가지 KeyStore 유형만 있습니다. `CloudHSM`

에서 로 Cavium KeyStore CloudHSM KeyStore 이동하려면 KeyStore 유형을 변경해야 합니다. 또한 클라이언트 SDK 3은 키 핸들을 사용하여 키를 참조하는 반면, 클라이언트 SDK 5는 키 레이블을 사용합니다. 그에 따른 동작 변경은 다음과 같습니다.

변경된 사항	클라이언트 SDK 3에 포함된 내용	클라이언트 SDK 5의 내용	예
주요 레퍼런스	Client SDK 3에서는 애플리케이션이 키 레이블 또는 키 핸들을 사용하여 HSM의 키를 참조합니다. 레이블을 KeyStore 사용하여 키를 찾거나 핸들을 사용하여 객체를 생성합니다CaviumKey .	Client SDK 5에서 애플리케이션은 를 사용하여 AWS CloudHSM KeyStore 자바 클래스 사용 레이블로 키를 찾을 수 있습니다. 핸들로 키를 찾으려면 AWS CloudHSM KeyStoreWithAttributes with AWS CloudHSM KeyRefereneSpec 를 사용하십시오.	
여러 항목 찾기	getEntrygetKey, 를 사용하여 키를 검색하거나 동일한 기준을 가진 항목이 여러 개 있는 getCertificate 시나리오에서 키를 검색하는 경우 찾은 첫 번째 항목만 반환됩니다. Cavium KeyStore	AWS CloudHSM KeyStore및 KeyStoreWithAttributes 를 사용하면 이와 동일한 시나리오에서 예외가 발생합니다. 이 문제를 해결하려면 CloudHSM CLI에서 키 세트-속성 명령을 사용하여 키에 고유한 레이블을 설정하는 것이 좋습니다. 또는 기준과 일치하는 모든 키를 반환하는 KeyStoreWithAttrib	

변경된 사항	클라이언트 SDK 3에 포함된 내용	클라이언트 SDK 5의 내용	예
		utes#getKeys 데 사용합니다.	
모든 키 찾기	클라이언트 SDK 3에서는 를 사용하여 HSM의 모든 키를 찾을 수 있습니다. Util.findAllKeys()	클라이언트 SDK 5를 사용하면 클래스를 사용하여 키를 더 간단하고 효율적으로 찾을 수 있습니다. KeyStoreWithAttributes 가능하면 키를 캐싱하여 지연 시간을 최소화하세요. 자세한 정보는 애플리케이션의 키를 효과적으로 관리을 참조하세요. HSM에서 모든 키를 검색해야 하는 경우 빈 KeyStoreWithAttributes#getKeys KeyAttributesMap 상태로 사용하십시오.	KeyStoreWithAttributes 클래스를 사용하여 키를 찾는 예제는 AWS CloudHSM Github 샘플 리포지토리에 있으며 코드 스니펫은 에 나와 있습니다. 1
키 삭제	클라이언트 SDK 3은 키를 삭제하는 데 사용합니다Util.deleteKey() .	Client SDK 5의 Key 객체는 이 Destroyable 인터페이스의 destroy() 방법을 사용하여 키를 삭제할 수 있는 인터페이스를 구현합니다.	키 삭제 기능을 보여주는 예제 코드는 CloudHSM Github 샘플 리포지토리에서 찾을 수 있습니다. 각 SDK의 샘플 스니펫은 에 나와 있습니다. 2

- [1] 스니펫은 다음과 같습니다.

```
KeyAttributesMap findSpec = new KeyAttributesMap();
findSpec.put(KeyAttribute.LABEL, label);
findSpec.put(KeyAttribute.KEY_TYPE, keyType);
KeyStoreWithAttributes keyStore = KeyStoreWithAttributes.getInstance("CloudHSM");

keyStore.load(null, null);
keyStore.getKey(findSpec);
```

- [2] 클라이언트 SDK 3에서 키 삭제:

```
Util.deleteKey(key);
```

  클라이언트 SDK 5에서 키 삭제:

```
((Destroyable) key).destroy();
```

암호 언랩 작업은 변경되었지만 다른 암호 작업은 변경되지 않았습니다.

> **Note**
>
> 암호 암호화/복호화/랩 작업은 변경할 필요가 없습니다.

언랩 작업을 수행하려면 클라이언트 SDK 3 `CaviumUnwrapParameterSpec` 클래스를 나열된 암호화 작업과 관련된 다음 클래스 중 하나로 바꿔야 합니다.

- `GCMUnwrapKeySpec`AES/GCM/NoPadding언래핑용
- `IvUnwrapKeySpec`용도 및 `AESWrap unwrap` AES/CBC/NoPadding `unwrap`
- `RSA OAEP unwrap`용 `OAEPUnwrapKeySpec`

예시 스니펫: `OAEPUnwrapkeySpec`

```
OAEPParameterSpec oaepParameterSpec =
new OAEPParameterSpec(
        "SHA-256",
        "MGF1",
        MGF1ParameterSpec.SHA256,
```

```
        PSpecified.DEFAULT);

KeyAttributesMap keyAttributesMap =
        new KeyAttributesMap(KeyAttributePermissiveProfile.KEY_CREATION);
keyAttributesMap.put(KeyAttribute.TOKEN, true);
keyAttributesMap.put(KeyAttribute.EXTRACTABLE, false);

OAEPUnwrapKeySpec spec = new OAEPUnwrapKeySpec(oaepParameterSpec,
        keyAttributesMap);

Cipher hsmCipher =
        Cipher.getInstance(
                "RSA/ECB/OAEPPadding",
                CloudHsmProvider.PROVIDER_NAME);
hsmCipher.init(Cipher.UNWRAP_MODE, key, spec);
```

서명 작업은 변경되지 않았습니다.

서명 작업에는 변경이 필요하지 않습니다.

클라이언트 SDK 5로 마이그레이션

이 섹션의 지침에 따라 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하십시오.

> **Note**
>
> 아마존 리눅스, 우분투 16.04, 우분투 18.04 센토스 6, 센토스 8, RHEL 6은 현재 클라이언트 SDK 5에서 지원되지 않습니다. 현재 클라이언트 SDK 3과 함께 이러한 플랫폼 중 하나를 사용하고 있다면 클라이언트 SDK 5로 마이그레이션할 때 다른 플랫폼을 선택해야 합니다.

1. 클라이언트 SDK용 JCE 공급자 제거 3.

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-jce
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-jce
   ```

RHEL 7

```
$ sudo yum remove cloudhsm-jce
```

RHEL 8

```
$ sudo yum remove cloudhsm-jce
```

2. 클라이언트 SDK용 클라이언트 데몬 제거 3.

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client
```

CentOS 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 8

```
$ sudo yum remove cloudhsm-client
```

> **Note**
>
> 사용자 지정 구성을 다시 활성화해야 합니다.

3. 의 단계에 따라 클라이언트 SDK JCE 공급자를 설치합니다. 클라이언트 SDK 5용 AWS CloudHSM JCE 공급자 설치 및 사용
4. 클라이언트 SDK 5에는 새로운 구성 파일 형식과 명령줄 부트스트래핑 도구가 도입되었습니다. 클라이언트 SDK 5 JCE 공급자를 부트스트랩하려면 아래 사용 설명서에 나와 있는 지침을 따르십시오. Client SDK 부트스트랩합니다.

5. 개발 환경에서 애플리케이션을 테스트하십시오. 최종 마이그레이션 전에 기존 코드를 업데이트하여 주요 변경 사항을 해결하세요.

관련 주제

- 에 대한 모범 사례 AWS CloudHSM

JCE용 고급 구성

AWS CloudHSM JCE 제공업체에는 대부분의 고객이 사용하는 일반 구성에 포함되지 않는 다음과 같은 고급 구성이 포함됩니다.

- 여러 클러스터에 연결
- JCE를 사용한 키 추출
- JCE 구성 재시도

JCE 공급자를 통해 여러 클러스터에 연결

이 구성을 사용하면 단일 클라이언트 인스턴스가 여러 클러스터와 통신할 수 있습니다. 단일 인스턴스가 단일 클러스터와만 통신하는 것과 비교하면 일부 사용 사례에서는 이 기능을 통해 비용을 절감할 수 있습니다. 이 `CloudHsmProvider` 클래스는 Java AWS CloudHSM Security의 공급자 클래스를 구현한 것입니다. 이 클래스의 각 인스턴스는 전체 AWS CloudHSM 클러스터에 대한 연결을 나타냅니다. 이 클래스를 인스턴스화하고 Java Security 공급자 목록에 추가하면 표준 JCE 클래스를 사용하여 이 클래스와 상호 작용할 수 있습니다.

다음 예제는 이 클래스를 인스턴스화하여 Java Security 공급자 목록에 추가합니다.

```
if (Security.getProvider(CloudHsmProvider.PROVIDER_NAME) == null) {
    Security.addProvider(new CloudHsmProvider());
}
```

`CloudHsmProvider` 구성

`CloudHsmProvider` 다음 두 가지 방법으로 구성할 수 있습니다:

1. 파일로 구성(기본 구성)

2. 코드로 구성

파일로 구성(기본 구성)

기본 생성자를 사용하여 CloudHsmProvider를 인스턴스화하면 기본적으로 Linux의 /opt/cloudhsm/etc/cloudhsm-jce.cfg 경로에서 구성 파일을 찾습니다. 이 구성 파일은 configure-jce를 사용하여 구성할 수 있습니다.

기본 생성자를 사용하여 생성한 객체는 기본 CloudHSM 공급자 이름 CloudHSM을 사용합니다. 공급자 이름은 JCE와 상호 작용하여 다양한 작업에 사용할 공급자를 알려주는 데 유용합니다. 암호화 작업에 CloudHSM 공급자 이름을 사용하는 예는 다음과 같습니다.

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHSM");
```

코드로 구성

Client SDK 버전 5.8.0부터 Java 코드를 사용하여 CloudHsmProvider를 구성할 수도 있습니다. 이 작업을 수행하는 방법은 CloudHsmProviderConfig 클래스의 객체를 사용하는 것입니다. CloudHsmProviderConfigBuilder를 사용하여 이 객체를 빌드할 수 있습니다.

CloudHsmProvider에는 다음 예제에서 볼 수 있듯이 CloudHsmProviderConfig 객체를 취하는 다른 생성자가 있습니다.

Example

```
CloudHsmProviderConfig config = CloudHsmProviderConfig.builder()
                                    .withCluster(
                                        CloudHsmCluster.builder()
                                            .withHsmCAFilePath(hsmCAFilePath)

 .withClusterUniqueIdentifier("CloudHsmCluster1")
        .withServer(CloudHsmServer.builder().withHostIP(hostName).build())
                        .build())
        .build();
CloudHsmProvider provider = new CloudHsmProvider(config);
```

이 예제에서 JCE 공급자의 이름은 CloudHsmCluster1이며, 이 이름은 애플리케이션이 JCE와 상호 작용하는 데 사용할 수 있는 이름입니다.

Example

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHsmCluster1");
```

또는 애플리케이션에서 위에서 생성한 공급자 객체를 사용하여 작업에 해당 공급자를 사용하도록 JCE에 알릴 수도 있습니다.

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider);
```

`withClusterUniqueIdentifier` 메서드에 고유 식별자를 지정하지 않은 경우 임의로 생성된 공급자 이름이 자동으로 생성됩니다. 임의로 생성된 이 식별자를 가져오기 위해 애플리케이션은 식별자를 가져오기 위해 `provider.getName()`을 호출할 수 있습니다.

여러 클러스터에 연결

위에서 설명한 것처럼 `CloudHsmProvider` 각각은 CloudHSM 클러스터에 대한 연결을 나타냅니다. 동일한 애플리케이션의 다른 `CloudHsmProvider` 클러스터와 통신하려는 경우 다른 클러스터의 구성을 사용하여 다른 객체를 생성하고 다음 예와 같이 공급자 객체 또는 공급자 이름을 사용하여 이 다른 클러스터와 상호 작용할 수 있습니다.

Example

```
CloudHsmProviderConfig config = CloudHsmProviderConfig.builder()
                                    .withCluster(
                                        CloudHsmCluster.builder()
                                            .withHsmCAFilePath(hsmCAFilePath)

 .withClusterUniqueIdentifier("CloudHsmCluster1")
        .withServer(CloudHsmServer.builder().withHostIP(hostName).build())
                        .build())
        .build();
CloudHsmProvider provider1 = new CloudHsmProvider(config);

if (Security.getProvider(provider1.getName()) == null) {
    Security.addProvider(provider1);
}

CloudHsmProviderConfig config2 = CloudHsmProviderConfig.builder()
                                    .withCluster(
                                        CloudHsmCluster.builder()
                                            .withHsmCAFilePath(hsmCAFilePath2)
```

```
 .withClusterUniqueIdentifier("CloudHsmCluster2")
        .withServer(CloudHsmServer.builder().withHostIP(hostName2).build())
                        .build())
        .build();
CloudHsmProvider provider2 = new CloudHsmProvider(config2);

if (Security.getProvider(provider2.getName()) == null) {
    Security.addProvider(provider2);
}
```

위의 두 공급자(두 클러스터 모두)를 구성한 후에는 공급자 객체 또는 공급자 이름을 사용하여 두 공급자와 상호 작용할 수 있습니다.

통신 방법을 보여주는 이 예제를 확장하여 NoPadding AES/GCM/ 작업에 다음 샘플을 사용할 수 있습니다. `cluster1`

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider1);
```

또한 동일한 애플리케이션에서 공급자 이름을 사용하여 두 번째 클러스터에서 “AES” 키 생성을 수행하는 경우 다음 샘플을 사용할 수도 있습니다.

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider2.getName());
```

JCE 재시도 명령

Client SDK 5.8.0 이상에는 HSM 제한 작업을 클라이언트 측에서 재시도하는 자동 재시도 전략이 내장되어 있습니다. HSM이 이전 작업을 수행하느라 너무 바빠서 더 많은 요청을 받을 수 없어 작업을 제한하는 경우 Client SDK는 제한이 발생한 작업을 최대 3회까지 재시도하고 기하급수적으로 백오프합니다. 이 자동 재시도 전략은 끄기 모드와 표준 모드 중 하나로 설정할 수 있습니다.

- 끄기 모드: Client SDK는 HSM에서 병목 현상이 발생한 작업에 대해 재시도 전략을 수행하지 않습니다.
- 표준 모드: Client SDK 5.8.0 이상의 기본 모드입니다. 이 모드에서는 Client SDK가 기하급수적으로 백오프를 수행하여 병목 현상이 발생한 작업을 자동으로 재시도합니다.

자세한 내용은 HSM 스로틀링 단원을 참조하십시오.

재시도 명령을 끄기 모드로 설정합니다.

Linux

Linux용 Client SDK 5의 재시도 명령을 off로 설정하려면

- 다음 명령을 사용하여 재시도 구성을 off 모드로 설정할 수 있습니다.

```
$ sudo /opt/cloudhsm/bin/configure-jce --default-retry-mode off
```

Windows

Client SDK 5 on Windows에 대한 재시도 명령을 off로 설정하려면

- 다음 명령을 사용하여 재시도 구성을 off 모드로 설정할 수 있습니다.

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-jce.exe --default-retry-mode
 off
```

JCE를 사용한 키 추출

JCE (Java 암호화 확장) 는 다양한 암호화 구현을 연결할 수 있는 아키텍처를 사용합니다. AWS CloudHSM 암호화 작업을 HSM으로 오프로드하는 JCE 공급자 하나를 제공합니다. 대부분의 다른 JCE 공급자가 AWS CloudHSM에 저장된 키를 사용하려면 HSM에서 일반 텍스트 형식의 키 바이트를 추출하여 시스템 메모리로 사용해야 합니다. HSM은 일반적으로 키를 래핑된 객체로만 추출하고 일반 텍스트로는 추출할 수 없습니다. 그러나 공급자 간 통합 사용 사례를 지원하기 위해 키 바이트를 쉽게 추출할 수 있는 옵트인 구성 옵션을 AWS CloudHSM 허용합니다.

> **Important**
>
> JCE는 AWS CloudHSM 공급자가 지정되거나 AWS CloudHSM 키 객체가 사용될 AWS CloudHSM 때마다 작업을 오프로드합니다. 작업이 HSM 내에서 수행될 것으로 예상되는 경우 키를 명확하게 추출할 필요가 없습니다. 일반 텍스트로 키를 추출하는 것은 타사 라이브러리 또는 JCE 공급자의 제한으로 인해 애플리케이션에서 키 래핑 및 래핑 해제와 같은 보안 메커니즘을 사용할 수 없는 경우에만 필요합니다.

AWS CloudHSM JCE 공급자는 기본적으로 외부 JCE 공급자와 연동되는 퍼블릭 키 추출을 허용합니다. 항상 허용되는 방법은 다음과 같습니다.

Class	메서드	형식(GETEncoded)
EcPublicKey	getEncoded()	X.509
	GetW ()	N/A
RSA PublicKey	getEncoded()	X.509
	getPublicExponent()	N/A
CloudHsmRsaPrivateCrtKey	getPublicExponent()	N/A

AWS CloudHSM JCE 제공자는 기본적으로 개인 또는 비밀 키에 대해 명확한 키 바이트 추출을 허용하지 않습니다. 사용 사례에 필요한 경우 다음과 같은 조건에서 프라이빗 또는 비밀 키에 대해 키 바이트를 선명하게 추출할 수 있습니다.

1. 프라이빗 및 비밀 키의 `EXTRACTABLE` 속성은 true로 설정됩니다.
   - 기본적으로 프라이빗 및 비밀 키의 `EXTRACTABLE` 속성은 true로 설정됩니다. `EXTRACTABLE`키는 HSM에서 내보낼 수 있는 키입니다. 지원되는 Java 속성에 대한 자세한 내용은 Client SDK 5를 참조하십시오.
2. 프라이빗 키와 비밀 키의 `WRAP_WITH_TRUSTED` 속성은 false로 설정됩니다.
   - `getEncoded`, `getPrivateExponent`, 그리고 `getS`는 명확하게 내보낼 수 없는 프라이빗 키와 함께 사용할 수 없습니다. `WRAP_WITH_TRUSTED`는 프라이빗 키를 HSM 밖으로 명확하게 내보낼 수 없습니다. 자세한 내용은 신뢰할 수 있는 키를 사용하여 키 래핑 해제 제어를 참조하십시오.

AWS CloudHSM JCE 제공자가 개인 키 암호를 추출할 수 있도록 허용 AWS CloudHSM

> **Important**
>
> 이 구성 변경을 통해 HSM 클러스터에서 모든 `EXTRACTABLE` 키 바이트를 안전하게 추출할 수 있습니다. 보안을 강화하려면 키 래핑 방법을 사용하여 HSM에서 키를 안전하게 추출하는 것을 고려해야 합니다. 이렇게 하면 HSM에서 실수로 키 바이트를 추출하는 것을 방지할 수 있습니다.

1. 다음 명령을 사용하여 프라이빗 또는 비밀 키를 JCE에서 추출할 수 있도록 하십시오.

   Linux

   ```
   $ /opt/cloudhsm/bin/configure-jce --enable-clear-key-extraction-in-software
   ```

   Windows

   ```
   C:\Program Files\Amazon\CloudHSM\> .\configure-jce.exe --enable-clear-key-extraction-in-software
   ```

2. 지우기 키 추출을 활성화하면 다음과 같은 방법으로 프라이빗 키를 메모리로 추출할 수 있습니다.

Class	메서드	형식(GETEncoded)
키	getEncoded()	RAW
EC PrivateKey	getEncoded()	PKCS #8
	GET ()	N/A
RSA PrivateCrtKey	getEncoded()	X.509
	getPrivateExponent()	N/A
	getPrimeP()	N/A
	getPrimeQ()	N/A
	getPrimeExponent(P)	N/A
	getPrimeExponentQ ()	N/A
	getCrtCoefficient()	N/A

기본 동작을 복원하고 JCE가 키를 정상적으로 내보내지 못하도록 하려면 다음 명령을 실행하십시오.

Linux

```
$ /opt/cloudhsm/bin/configure-jce --disable-clear-key-extraction-in-software
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\configure-jce.exe --disable-clear-key-
extraction-in-software
```

Microsoft Windows용 Cryptography API: Next Generation(CNG) 및 KSP(Key Storage Provider)

Windows용 AWS CloudHSM 클라이언트에는 CNG 및 KSP 공급자가 포함됩니다. 현재는 클라이언트 SDK 3만 CNG 및 KSP 공급자를 지원합니다.

KSP(Key Storage Provider)를 사용하여 키를 저장하고 검색할 수 있습니다. 예를 들어 Microsoft Active Directory Certificate Services(AD CS) 역할을 Windows 서버에 추가하고 인증 기관(CA)용 프라이빗 키를 새로 만들기로 한 경우, 키 저장을 관리할 KSP를 선택할 수 있습니다. AD CS 역할을 구성할 때 이 KSP를 선택할 수 있습니다. 자세한 내용은 Windows Server CA 생성 섹션을 참조하십시오.

CNG(Cryptography API: Next Generation)는 Microsoft Windows 운영 체제 전용 암호화 API입니다. 개발자들은 CNG를 통해 Windows 기반 애플리케이션을 보호하는 암호화 기법을 사용할 수 있습니다. CNG AWS CloudHSM 구현은 개괄적으로 다음과 같은 기능을 제공합니다.

- 암호화 프리미티브 - 기본 암호화 작업을 수행할 수 있습니다.
- 키 가져오기 및 내보내기 - 비대칭 키를 가져오고 내보낼 수 있습니다.
- 데이터 보호 API(CNG DPAPI) - 데이터를 쉽게 암호화하고 암호화를 해독할 수 있습니다.
- 키 저장 및 검색 - 비대칭 키 페어의 프라이빗 키를 안전하게 보관하고 격리할 수 있습니다.

주제

- Windows용 KSP 및 CNG 공급자 확인
- 윈도우 AWS CloudHSM 사전 요구 사항

- AWS CloudHSM 키를 인증서와 연결
- CNG 공급자용 코드 샘플

Windows용 KSP 및 CNG 공급자 확인

KSP 및 CNG 공급자는 Windows AWS CloudHSM 클라이언트를 설치할 때 설치됩니다. 클라이언트 설치(Windows)의 단계에 따라 클라이언트를 설치할 수 있습니다.

Windows AWS CloudHSM 클라이언트 구성 및 실행

Windows CloudHSM 클라이언트를 시작하려면 먼저 필수 조건의 조건을 충족해야 합니다. 그런 다음 공급자가 사용하는 구성 파일을 업데이트하고 아래 단계를 완료하여 클라이언트를 시작합니다. KSP 및 CNG 공급자를 처음 사용할 때와 클러스터에서 HSM을 추가하거나 제거한 후 이러한 단계를 수행해야 합니다. 이렇게 하면 AWS CloudHSM 클러스터의 모든 HSM에서 데이터를 동기화하고 일관성을 유지할 수 있습니다.

1단계: 클라이언트 중지 AWS CloudHSM

공급자가 사용하는 구성 파일을 업데이트하기 전에 AWS CloudHSM 클라이언트를 중지하십시오. 클라이언트가 이미 중지된 경우 중지 명령을 실행해도 아무 영향이 없습니다.

- Windows 클라이언트 1.1.2+의 경우:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Windows 클라이언트 1.1.1 이상의 경우:

  클라이언트를 시작한 명령 창에서 Ctrl + C를 사용합니다. AWS CloudHSM

2단계: 구성 파일 업데이트 AWS CloudHSM

이 단계에서는 구성 도구의 `-a` 파라미터를 사용하여 클러스터에 있는 HSM 중 하나의 탄력적 네트워크 인터페이스(ENI) IP 주소를 구성 파일에 추가합니다.

```
C:\Program Files\Amazon\CloudHSM configure.exe -a <HSM ENI IP>
```

클러스터에 있는 HSM의 ENI IP 주소를 가져오려면 AWS CloudHSM 콘솔로 이동하여 클러스터를 선택하고 원하는 클러스터를 선택합니다. DescribeClusters작업, describe-cluster 명령 또는 cmdlet을 사

용할 수도 있습니다. Get-HSM2Cluster PowerShell 오직 한 개의 ENI IP 주소만 입력하십시오. 어떤 ENI IP 주소를 사용해도 상관은 없습니다.

3단계: 클라이언트 시작 AWS CloudHSM

다음으로, AWS CloudHSM 클라이언트를 시작하거나 다시 시작합니다. AWS CloudHSM 클라이언트가 시작되면 구성 파일의 ENI IP 주소를 사용하여 클러스터를 쿼리합니다. 그런 다음 클러스터 정보 파일에 모든 HSM의 ENI IP 주소를 추가합니다.

- Windows 클라이언트 1.1.2+의 경우:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows 클라이언트 1.1.1 이상의 경우:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

KSP 및 CNG 공급자 확인

다음 명령 중 하나를 사용하여 시스템에 어떤 공급자가 설치되었는지 확인할 수 있습니다. 이 명령은 등록된 KSP 공급자와 CNG 공급자를 나열합니다. AWS CloudHSM 클라이언트를 실행할 필요는 없습니다.

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -enum
```

```
C:\Program Files\Amazon\CloudHSM>cng_config.exe -enum
```

Windows Server EC2 인스턴스에 KSP 및 CNG 공급자가 설치되어 있는지 확인하려면 목록에 다음 항목이 표시되어야 합니다.

```
Cavium CNG Provider
Cavium Key Storage Provider
```

CNG 공급자가 없으면 다음 명령을 실행합니다.

```
C:\Program Files\Amazon\CloudHSM>cng_config.exe -register
```

KSP 공급자가 없으면 다음 명령을 실행합니다.

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -register
```

윈도우 AWS CloudHSM 사전 요구 사항

Windows AWS CloudHSM 클라이언트를 시작하고 KSP 및 CNG 공급자를 사용하려면 먼저 시스템에서 HSM의 로그인 자격 증명을 설정해야 합니다. Windows 자격 증명 관리자 또는 시스템 환경 변수를 통해 자격 증명을 설정할 수 있습니다. 자격 증명 저장에는 Windows 자격 증명 관리자를 사용하는 것이 좋습니다. 이 옵션은 AWS CloudHSM 클라이언트 버전 2.0.4 이상에서 사용할 수 있습니다. 환경 변수를 사용하면 더 쉽게 설정할 수 있지만 Windows 자격 증명 관리자를 사용할 때보다 안전하지 않습니다.

Windows 자격 증명 관리자

`set_cloudhsm_credentials` 유틸리티 또는 Windows 자격 증명 관리자 인터페이스를 사용할 수 있습니다.

- **`set_cloudhsm_credentials`** 유틸리티 사용:

  `set_cloudhsm_credentials` 유틸리티는 Windows 설치 관리자에 포함되어 있습니다. 이 유틸리티를 사용하여 HSM 로그인 자격 증명을 Windows 자격 증명 관리자에 편리하게 전달할 수 있습니다. 소스에서 이 유틸리티를 컴파일하려는 경우 설치 관리자에 포함된 Python 코드를 사용할 수 있습니다.

  1. `C:\Program Files\Amazon\CloudHSM\tools\` 폴더로 이동합니다.
  2. CU 사용자 이름 및 암호 파라미터를 사용하여 `set_cloudhsm_credentials.exe` 파일을 실행합니다.

     ```
     set_cloudhsm_credentials.exe --username <CU USER> --password <CU PASSWORD>
     ```

- 자격 증명 관리자 인터페이스 사용:

  자격 증명 관리자 인터페이스를 사용하여 자격 증명을 수동으로 관리할 수 있습니다.

  1. 자격 증명 관리자를 열려면 작업 표시줄의 검색 상자에 `credential manager`를 입력하고 자격 증명 관리자를 선택합니다.
  2. Windows 자격 증명을 선택하여 Windows 자격 증명을 관리합니다.
  3. 일반 자격 증명 추가를 선택하고 다음과 같이 세부 정보를 채웁니다.
     - 인터넷 또는 네트워크 주소에 대상 이름을 `cloudhsm_client`로 입력합니다.

- 사용자 이름 및 암호에 CU 자격 증명을 입력합니다.
- 확인을 클릭합니다.

시스템 환경 변수

Windows 애플리케이션용 HSM 및 CU(Crypto User)를 식별하는 시스템 환경 변수를 설정할 수 있습니다. setx명령을 사용하여 시스템 환경 변수를 설정하거나, 프로그래밍 방식으로 또는 Windows 시스템 속성 제어판의 고급 탭에서 영구 시스템 환경 변수를 설정할 수 있습니다.

> **Warning**
>
> 시스템 환경 변수를 통해 자격 증명을 설정하면 사용자의 시스템에서 일반 텍스트로 암호를 사용할 수 있습니다. 이 문제를 해결하려면 Windows 자격 증명 관리자를 사용하십시오.

다음 시스템 환경 변수를 설정합니다.

`n3fips_password=`*`CU USERNAME`*`:`*`CU PASSWORD`*

HSM의 CU(Crypto User)를 식별하고 필요한 모든 로그인 정보를 제공합니다. 애플리케이션에서는 해당 CU로 인증하고 실행합니다. 애플리케이션은 이 CU의 권한을 가지며 CU가 소유하고 공유하는 키만 보고 관리할 수 있습니다. 새 CU를 생성하려면 createUser를 사용하십시오 기존 CU를 찾으려면 listUsers를 사용하십시오.

예:

```
setx /m n3fips_password test_user:password123
```

AWS CloudHSM 키를 인증서와 연결

Microsoft와 같은 타사 도구에서 AWS CloudHSM 키를 사용하려면 먼저 키의 SignTool메타데이터를 로컬 인증서 저장소로 가져와서 메타데이터를 인증서와 연결해야 합니다. 키의 메타데이터를 가져오려면 CloudHSM 버전 3.0 이상에 포함된 import_key.exe 유틸리티를 사용합니다. 다음 단계에서는 추가 정보와 샘플 출력을 제공합니다.

1단계: 인증서 가져오기

Windows에서는 인증서를 두 번 클릭하여 로컬 인증서 스토어로 가져올 수 있습니다.

그러나 두 번 클릭해도 작동하지 않는다면 Microsoft Certreq 도구를 사용하여 인증서를 Certificate Manager로 가져옵니다. 예:

```
certreq -accept certificatename
```

이 작업이 실패하고 Key not found 오류가 표시되는 경우 2단계를 진행합니다. 인증서가 키 스토어에 나타나면 작업이 완료된 것이므로 추가 작업이 필요하지 않습니다.

2단계: 인증서 식별 정보 수집

이전 단계가 성공하지 못한 경우 프라이빗 키를 인증서와 연결해야 합니다. 그러나 연결을 생성하려면 먼저 인증서의 고유 컨테이너 이름 및 일련번호를 찾아야 합니다. 필요한 인증서 정보를 표시하려면 certutil과 같은 유틸리티를 사용합니다. certutil의 다음 샘플 출력은 컨테이너 이름과 일련번호를 보여줍니다.

```
================ Certificate 1 ================ Serial Number:
   72000000047f7f7a9d41851b4e000000000004Issuer: CN=Enterprise-CANotBefore: 10/8/2019
 11:50
   AM NotAfter: 11/8/2020 12:00 PMSubject: CN=www.example.com, OU=Certificate
 Management,
   O=Information Technology, L=Seattle, S=Washington, C=USNon-root CertificateCert
   Hash(sha1): 7f d8 5c 00 27 bf 37 74 3d 71 5b 54 4e c0 94 20 45 75 bc 65No key
 provider
   information Simple container name: CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
 Unique
   container name: CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
```

3단계: AWS CloudHSM 개인 키를 인증서에 연결

키를 인증서에 연결하려면 먼저 AWS CloudHSM 클라이언트 데몬을 시작해야 합니다. 그런 다음 import_key.exe(CloudHSM 버전 3.0 이상에 포함되어 있음)를 사용하여 프라이빗 키를 인증서와 연결합니다. 인증서를 지정할 때 간단한 컨테이너 이름을 사용하십시오. 다음 예제에서는 명령과 응답을 보여줍니다. 이 작업은 키의 메타데이터만 복사하므로 키는 HSM에 남아 있습니다.

```
$> import_key.exe -RSA CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c

Successfully opened Microsoft Software Key Storage Provider : 0NCryptOpenKey failed :
 80090016
```

4단계: 인증서 스토어 업데이트

AWS CloudHSM 클라이언트 데몬이 아직 실행 중인지 확인하세요. 그런 다음 certutil 동사 -repairstore를 사용하여 인증서 일련번호를 업데이트합니다. 다음 샘플은 명령 및 출력을 보여줍니다. -repairstore 동사에 대한 자세한 내용은 Microsoft 설명서를 참조하십시오.

```
C:\Program Files\Amazon\CloudHSM>certutil -f -csp "Cavium Key Storage Provider"-
repairstore my "72000000047f7f7a9d41851b4e000000000004"
my "Personal"
================ Certificate 1 ================
Serial Number: 72000000047f7f7a9d41851b4e000000000004
Issuer: CN=Enterprise-CA
NotBefore: 10/8/2019 11:50 AM
NotAfter: 11/8/2020 12:00 PM
Subject: CN=www.example.com, OU=Certificate Management, O=Information Technology,
 L=Seattle, S=Washington, C=US
Non-root CertificateCert Hash(sha1): 7f d8 5c 00 27 bf 37 74 3d 71 5b 54 4e c0 94 20 45
 75 bc 65
SDK Version: 3.0
Key Container = CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c
Provider = Cavium Key Storage ProviderPrivate key is NOT exportableEncryption test
 passedCertUtil: -repairstore command completed successfully.
```

인증서 일련 번호를 업데이트한 후에는 Windows의 타사 서명 도구에서 이 인증서와 해당 AWS CloudHSM 개인 키를 사용할 수 있습니다.

CNG 공급자용 코드 샘플

⚠ ** 예제 코드 전용 - 프로덕션용으로 사용 불가 **
이 샘플 코드는 설명 목적으로만 제공됩니다. 이 코드를 프로덕션용으로 사용하지 마십시오.

다음 샘플은 Windows용 CloudHSM 클라이언트와 함께 설치된 CNG 공급자를 찾기 위해 시스템에 등록된 암호화 공급자를 열거하는 방법을 보여줍니다. 또한 비대칭 키 페어를 생성하는 방법과 키 페어를 사용하여 데이터에 서명하는 방법을 보여 줍니다.

> **Important**
>
> 이 예제를 실행하기 전에 사전 조건의 설명에 따라 HSM 보안 인증을 설정해야 합니다. 자세한 내용은 윈도우 AWS CloudHSM 사전 요구 사항 단원을 참조하십시오.

```
// CloudHsmCngExampleConsole.cpp : Console application that demonstrates CNG
 capabilities.
// This example contains the following functions.
//
//   VerifyProvider()          - Enumerate the registered providers and retrieve Cavium
 KSP and CNG providers.
//   GenerateKeyPair()         - Create an RSA key pair.
//   SignData()                - Sign and verify data.
//

#include "stdafx.h"
#include <Windows.h>

#ifndef NT_SUCCESS
#define NT_SUCCESS(Status) ((NTSTATUS)(Status) >= 0)
#endif

#define CAVIUM_CNG_PROVIDER L"Cavium CNG Provider"
#define CAVIUM_KEYSTORE_PROVIDER L"Cavium Key Storage Provider"

// Enumerate the registered providers and determine whether the Cavium CNG provider
// and the Cavium KSP provider exist.
//
bool VerifyProvider()
{
  NTSTATUS status;
  ULONG cbBuffer = 0;
  PCRYPT_PROVIDERS pBuffer = NULL;
  bool foundCng = false;
  bool foundKeystore = false;

  // Retrieve information about the registered providers.
  //   cbBuffer - the size, in bytes, of the buffer pointed to by pBuffer.
  //   pBuffer - pointer to a buffer that contains a CRYPT_PROVIDERS structure.
```

```
  status = BCryptEnumRegisteredProviders(&cbBuffer, &pBuffer);

  // If registered providers exist, enumerate them and determine whether the
  // Cavium CNG provider and Cavium KSP provider have been registered.
  if (NT_SUCCESS(status))
  {
    if (pBuffer != NULL)
    {
      for (ULONG i = 0; i < pBuffer->cProviders; i++)
      {
        // Determine whether the Cavium CNG provider exists.
        if (wcscmp(CAVIUM_CNG_PROVIDER, pBuffer->rgpszProviders[i]) == 0)
        {
          printf("Found %S\n", CAVIUM_CNG_PROVIDER);
          foundCng = true;
        }

        // Determine whether the Cavium KSP provider exists.
        else if (wcscmp(CAVIUM_KEYSTORE_PROVIDER, pBuffer->rgpszProviders[i]) == 0)
        {
          printf("Found %S\n", CAVIUM_KEYSTORE_PROVIDER);
          foundKeystore = true;
        }
      }
    }
  }
  else
  {
    printf("BCryptEnumRegisteredProviders failed with error code 0x%08x\n", status);
  }

  // Free memory allocated for the CRYPT_PROVIDERS structure.
  if (NULL != pBuffer)
  {
    BCryptFreeBuffer(pBuffer);
  }

  return foundCng == foundKeystore == true;
}

// Generate an asymmetric key pair. As used here, this example generates an RSA key
 pair
// and returns a handle. The handle is used in subsequent operations that use the key
 pair.
```

```
// The key material is not available.
//
// The key pair is used in the SignData function.
//
NTSTATUS GenerateKeyPair(BCRYPT_ALG_HANDLE hAlgorithm, BCRYPT_KEY_HANDLE *hKey)
{
  NTSTATUS status;

  // Generate the key pair.
  status = BCryptGenerateKeyPair(hAlgorithm, hKey, 2048, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptGenerateKeyPair failed with code 0x%08x\n", status);
    return status;
  }

  // Finalize the key pair. The public/private key pair cannot be used until this
  // function is called.
  status = BCryptFinalizeKeyPair(*hKey, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptFinalizeKeyPair failed with code 0x%08x\n", status);
    return status;
  }

  return status;
}

// Sign and verify data using the RSA key pair. The data in this function is hardcoded
// and is for example purposes only.
//
NTSTATUS SignData(BCRYPT_KEY_HANDLE hKey)
{
  NTSTATUS status;
  PBYTE sig;
  ULONG sigLen;
  ULONG resLen;
  BCRYPT_PKCS1_PADDING_INFO pInfo;

  // Hardcode the data to be signed (for demonstration purposes only).
  PBYTE message = (PBYTE)"d83e7716bed8a20343d8dc6845e57447";
  ULONG messageLen = strlen((char*)message);

  // Retrieve the size of the buffer needed for the signature.
```

```
  status = BCryptSignHash(hKey, NULL, message, messageLen, NULL, 0, &sigLen, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptSignHash failed with code 0x%08x\n", status);
    return status;
  }

  // Allocate a buffer for the signature.
  sig = (PBYTE)HeapAlloc(GetProcessHeap(), 0, sigLen);
  if (sig == NULL)
  {
    return -1;
  }

  // Use the SHA256 algorithm to create padding information.
  pInfo.pszAlgId = BCRYPT_SHA256_ALGORITHM;

  // Create a signature.
  status = BCryptSignHash(hKey, &pInfo, message, messageLen, sig, sigLen, &resLen,
 BCRYPT_PAD_PKCS1);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptSignHash failed with code 0x%08x\n", status);
    return status;
  }

  // Verify the signature.
  status = BCryptVerifySignature(hKey, &pInfo, message, messageLen, sig, sigLen,
 BCRYPT_PAD_PKCS1);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptVerifySignature failed with code 0x%08x\n", status);
    return status;
  }

  // Free the memory allocated for the signature.
  if (sig != NULL)
  {
    HeapFree(GetProcessHeap(), 0, sig);
    sig = NULL;
  }

  return 0;
}
```

```
// Main function.
//
int main()
{
  NTSTATUS status;
  BCRYPT_ALG_HANDLE hRsaAlg;
  BCRYPT_KEY_HANDLE hKey = NULL;

  // Enumerate the registered providers.
  printf("Searching for Cavium providers...\n");
  if (VerifyProvider() == false) {
    printf("Could not find the CNG and Keystore providers\n");
    return 1;
  }

  // Get the RSA algorithm provider from the Cavium CNG provider.
  printf("Opening RSA algorithm\n");
  status = BCryptOpenAlgorithmProvider(&hRsaAlg, BCRYPT_RSA_ALGORITHM,
 CAVIUM_CNG_PROVIDER, 0);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptOpenAlgorithmProvider RSA failed with code 0x%08x\n", status);
    return status;
  }

  // Generate an asymmetric key pair using the RSA algorithm.
  printf("Generating RSA Keypair\n");
  GenerateKeyPair(hRsaAlg, &hKey);
  if (hKey == NULL)
  {
    printf("Invalid key handle returned\n");
    return 0;
  }
  printf("Done!\n");

  // Sign and verify [hardcoded] data using the RSA key pair.
  printf("Sign/Verify data with key\n");
  SignData(hKey);
  printf("Done!\n");

  // Remove the key handle from memory.
  status = BCryptDestroyKey(hKey);
  if (!NT_SUCCESS(status))
```

```
  {
    printf("BCryptDestroyKey failed with code 0x%08x\n", status);
    return status;
  }

  // Close the RSA algorithm provider.
  status = BCryptCloseAlgorithmProvider(hRsaAlg, NULL);
  if (!NT_SUCCESS(status))
  {
    printf("BCryptCloseAlgorithmProvider RSA failed with code 0x%08x\n", status);
    return status;
  }

  return 0;
}
```

이전 클라이언트 SDK (클라이언트 SDK 3)

AWS CloudHSM 두 가지 주요 클라이언트 SDK 버전이 포함되어 있습니다.

- 클라이언트 SDK 5: 최신 기본 클라이언트 SDK입니다. 제공되는 이점 및 장점에 대한 자세한 내용은 Client SDK 5의 이점 섹션을 참조하세요.
- 클라이언트 SDK 3: 이전 클라이언트 SDK입니다. 여기에는 플랫폼 및 언어 기반 애플리케이션 호환성 및 관리 도구를 위한 전체 구성 요소 세트가 포함됩니다.

클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션하는 방법에 대한 지침은 을 참조하십시오. 클라이언트 SDK 3에서 클라이언트 SDK 5로 마이그레이션

클라이언트 SDK 3 설명서는 이 주제에 나열되어 있습니다.

다운로드하려면 다운로드 섹션을 참조하십시오.

클라이언트 SDK 버전을 확인하십시오.

Amazon Linux

다음 명령을 사용합니다.

```
rpm -qa | grep ^cloudhsm
```

Amazon Linux 2

다음 명령을 사용합니다.

```
rpm -qa | grep ^cloudhsm
```

CentOS 6

다음 명령을 사용합니다.

```
rpm -qa | grep ^cloudhsm
```

CentOS 7

다음 명령을 사용합니다.

```
rpm -qa | grep ^cloudhsm
```

CentOS 8

다음 명령을 사용합니다.

```
rpm -qa | grep ^cloudhsm
```

RHEL 6

다음 명령을 사용합니다.

```
rpm -qa | grep ^cloudhsm
```

RHEL 7

다음 명령을 사용합니다.

```
rpm -qa | grep ^cloudhsm
```

RHEL 8

다음 명령을 사용합니다.

```
rpm -qa | grep ^cloudhsm
```

Ubuntu 16.04 LTS

다음 명령을 사용합니다.

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 18.04 LTS

다음 명령을 사용합니다.

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 20.04 LTS

다음 명령을 사용합니다.

```
apt list --installed | grep ^cloudhsm
```

Windows Server

다음 명령을 사용합니다.

```
wmic product get name,version
```

클라이언트 SDK 구성 요소 비교

명령줄 도구 외에도 클라이언트 SDK 3에는 다양한 플랫폼 또는 언어 기반 애플리케이션에서 HSM으로 암호화 작업을 오프로드할 수 있는 구성 요소가 포함되어 있습니다. 클라이언트 SDK 5는 아직 CNG 및 KSP 공급자를 지원하지 않는다는 점을 제외하면 클라이언트 SDK 3과 동등합니다. 다음 표에서는 클라이언트 SDK 3과 클라이언트 SDK 5의 구성 요소 가용성을 비교합니다.

구성 요소	클라이언트 SDK 5	클라이언트 SDK 3
PKCS #11 라이브러리	예	예
JCE 공급자	예	예
OpenSSL Dynamic Engine	예	예
CNG 및 KSP 공급자		예

구성 요소	클라이언트 SDK 5	클라이언트 SDK 3
CloudHSM 관리 유틸리티 (CMU)[1]	예	예
키 관리 유틸리티(KMU)[1]	예	예
구성 도구	예	예

[1] CMU 및 KMU 구성 요소는 클라이언트 SDK 5와 함께 CloudHSM CLI에 포함되어 있습니다.

주제

- Client SDK 3 지원 플랫폼
- 리눅스용 클라이언트 SDK 3 업그레이드
- 클라이언트 SDK 3용 PKCS #11 라이브러리
- OpenSSL 다이내믹 엔진용 클라이언트 SDK 3 설치
- JCE 공급자용 클라이언트 SDK 3

Client SDK 3 지원 플랫폼

클라이언트 SDK 3에는 클라이언트 데몬이 필요하며 CloudHSM 관리 유틸리티(CMU), 키 관리 유틸리티(KMU), 구성 도구를 비롯한 명령줄 도구를 제공합니다.

기본 지원은 AWS CloudHSM 클라이언트 SDK의 각 버전마다 다릅니다. 일반적으로 SDK의 구성 요소에 대한 플랫폼 지원은 기본 지원과 일치하지만 항상 그런 것은 아닙니다. 특정 구성 요소에 대한 플랫폼 지원을 확인하려면 먼저 원하는 플랫폼이 SDK의 기본 섹션에 표시되는지 확인한 다음 구성 요소 섹션에서 제외 사항이나 기타 관련 정보가 있는지 확인하세요.

플랫폼 지원은 시간이 지남에 따라 변경됩니다. 이전 버전의 CloudHSM 클라이언트 SDK는 여기에 나열된 모든 운영 체제를 지원하지 않을 수 있습니다. 릴리스 노트를 사용하여 이전 버전의 CloudHSM 클라이언트 SDK에 대한 운영 체제 지원을 확인할 수 있습니다. 자세한 정보는 AWS CloudHSM 클라이언트 SDK용 다운로드을 참조하세요.

AWS CloudHSM 64비트 운영 체제만 지원합니다.

목차

- 윈도우 지원
- 클라이언트 SDK 3의 HSM 호환성
- 구성 요소 지원
  - PKCS #11 라이브러리
  - CloudHSM 관리 유틸리티(CMU)
  - 키 관리 유틸리티(KMU)
  - JCE 공급자
  - OpenSSL Dynamic Engine
  - CNG 및 KSP 공급자

Linux 지원

- Amazon Linux
- Amazon Linux 2
- CentOS 6.10+ [2]
- CentOS 7.3+
- CentOS 8 [1,4]
- Red Hat Enterprise Linux(RHEL) 6.10+ [2]
- Red Hat Enterprise Linux(RHEL) 7.3+
- Red Hat Enterprise Linux(RHEL) 8 [1]
- Ubuntu 16.04 LTS [3]
- Ubuntu 18.04 LTS [1]

[1] OpenSSL Dynamic Engine은 지원되지 않습니다. 자세한 내용은 OpenSSL 동적 엔진을 참조하십시오.

[2] 클라이언트 SDK 3.3.0 이상은 지원되지 않습니다.

[3] SDK 3.4는 Ubuntu 16.04에서 마지막으로 지원되는 릴리스입니다.

[4] SDK 3.4는 CentOS 8.3+에서 마지막으로 지원되는 릴리스입니다.

윈도우 지원

- Microsoft Windows Server 2012

- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

클라이언트 SDK 3의 HSM 호환성

hsm1.medium	hsm2m.medium
클라이언트 버전 SDK 3.1.0 이상과 호환됩니다.	지원하지 않음.

구성 요소 지원

PKCS #11 라이브러리

PKCS #11 라이브러리는 Linux 기본 지원과 일치하는 Linux 전용 구성 요소입니다. 자세한 정보는 the section called “Linux 지원”을 참조하세요.

CloudHSM 관리 유틸리티(CMU)

CloudHSM 관리 유틸리티 (CMU) 명령줄 도구는 암호화 담당자가 HSM에서 사용자를 관리하는 데 도움이 됩니다. 사용자를 생성, 삭제 및 나열하고 사용자 암호를 변경하는 도구가 여기에 포함됩니다. 자세한 정보는 CloudHSM 관리 유틸리티(CMU)을 참조하세요.

키 관리 유틸리티(KMU)

키 관리 유틸리티 (KMU) 는 암호화 사용자 (CU) 가 하드웨어 보안 모듈 (HSM) 의 키를 관리하는 데 도움이 되는 명령줄 도구입니다. 자세한 정보는 키 관리 유틸리티(KMU)을 참조하세요.

JCE 공급자

JCE 공급자는 Linux 기본 지원과 일치하는 Linux 전용 구성 요소입니다. 자세한 내용은 the section called “Linux 지원” 섹션을 참조하십시오.

- OpenJDK 1.8이 필요합니다.

OpenSSL Dynamic Engine

OpenSSL Dynamic Engine은 Linux 기본 지원과 일치하지 않는 Linux 전용 구성 요소입니다. 아래 제외 항목을 참조하십시오.

- OpenSSL 1.0.2[f+] 필요

지원되지 않는 플랫폼:

- CentOS 8
- Red Hat Enterprise Linux(RHEL) 8
- Ubuntu 18.04 LTS

이러한 플랫폼은 클라이언트 SDK 3용 OpenSSL Dynamic Engine과 호환되지 않는 OpenSSL 버전과 함께 제공됩니다. AWS CloudHSM 는 클라이언트 SDK 5용 OpenSSL Dynamic Engine과 함께 이러한 플랫폼을 지원합니다.

CNG 및 KSP 공급자

CNG 및 KSP 공급자는 Windows 기본 지원과 일치하는 Windows 전용 구성 요소입니다. 자세한 내용은 윈도우 지원 섹션을 참조하십시오.

리눅스용 클라이언트 SDK 3 업그레이드

AWS CloudHSM 클라이언트 SDK 3.1 이상에서는 클라이언트 데몬의 버전과 설치한 구성 요소가 일치해야 업그레이드할 수 있습니다. 모든 Linux 기반 시스템의 경우 단일 명령을 사용하여 동일한 버전의 PKCS #11 라이브러리, Java 암호화 확장(JCE) 공급자 또는 OpenSSL Dynamic Engine으로 클라이언트 데몬을 일괄 업그레이드해야 합니다. CNG 및 KSP 공급자의 바이너리가 이미 클라이언트 데몬 패키지에 포함되어 있으므로 이 요구 사항은 Windows 기반 시스템에는 적용되지 않습니다.

클라이언트 데몬 버전을 확인하려면

- Red Hat 기반 Linux 시스템(Amazon Linux 및 CentOS 포함)에서는 다음 명령을 사용합니다.

```
rpm -qa | grep ^cloudhsm
```

- Debian 기반 Linux 시스템에서는 다음 명령을 사용합니다.

```
apt list --installed | grep ^cloudhsm
```

- Windows 시스템에서 다음 명령을 사용합니다.

```
wmic product get name,version
```

주제

필수 조건

최신 버전의 AWS CloudHSM 클라이언트 데몬을 다운로드하고 구성 요소를 선택합니다.

> **Note**
>
> 모든 구성 요소를 설치할 필요는 없습니다. 설치한 모든 구성 요소에 대해 클라이언트 데몬 버전과 일치하도록 해당 구성 요소를 업그레이드해야 합니다.

최신 리눅스 클라이언트 데몬

Amazon Linux

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

CentOS 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

RHEL 7

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-latest.el7.x86_64.rpm
```

RHEL 8

```
sudo yum install wget
```

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client_latest_u18.04_amd64.deb
```

최신 PKCS #11 라이브러리

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-pkcs11_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

최신 OpenSSL Dynamic Engine

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-dyn_latest_amd64.deb
```

최신 JCE 공급자

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-jce-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-jce_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-jce_latest_u18.04_amd64.deb
```

1단계: 클라이언트 데몬 중지

다음 명령을 사용하여 클라이언트 데몬을 중지합니다.

Amazon Linux

```
$ sudo stop cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client stop
```

CentOS 7

```
$ sudo service cloudhsm-client stop
```

CentOS 8

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client stop
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client stop
```

2단계: 클라이언트 SDK 업그레이드

다음 명령은 클라이언트 데몬 및 구성 요소를 업그레이드하는 데 필요한 구문을 보여줍니다. 명령을 실행하기 전에 업그레이드하지 않으려는 구성 요소를 제거합니다.

Amazon Linux

```
$ sudo yum install ./cloudhsm-client-latest.el6.x86_64.rpm \
              <./cloudhsm-client-pkcs11-latest.el6.x86_64.rpm> \
              <./cloudhsm-client-dyn-latest.el6.x86_64.rpm> \
              <./cloudhsm-client-jce-latest.el6.x86_64.rpm>
```

Amazon Linux 2

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \
              <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \
```

```
                <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

CentOS 7

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \
                <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

CentOS 8

```
$ sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm \
                <./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el8.x86_64.rpm>
```

RHEL 7

```
$ sudo yum install ./cloudhsm-client-latest.el7.x86_64.rpm \
                <./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-dyn-latest.el7.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el7.x86_64.rpm>
```

RHEL 8

```
$ sudo yum install ./cloudhsm-client-latest.el8.x86_64.rpm \
                <./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm> \
                <./cloudhsm-client-jce-latest.el8.x86_64.rpm>
```

Ubuntu 16.04 LTS

```
$ sudo apt install ./cloudhsm-client_latest_amd64.deb \
                <cloudhsm-client-pkcs11_latest_amd64.deb> \
                <cloudhsm-client-dyn_latest_amd64.deb> \
                <cloudhsm-client-jce_latest_amd64.deb>
```

Ubuntu 18.04 LTS

```
$ sudo apt install ./cloudhsm-client_latest_u18.04_amd64.deb \
                <cloudhsm-client-pkcs11_latest_amd64.deb> \
                <cloudhsm-client-jce_latest_amd64.deb>
```

3단계: 클라이언트 데몬 시작

다음 명령을 사용하여 클라이언트 데몬을 시작합니다.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

클라이언트 SDK 3용 PKCS #11 라이브러리

PKCS #11은 하드웨어 보안 모듈(HSM)에서 암호화 작업을 수행하기 위한 표준입니다.

부트스트래핑에 대한 자세한 내용은 클러스터에 연결 단원을 참조하십시오.

주제

- PKCS #11 라이브러리용 Client SDK 3 설치
- PKCS #11 라이브러리 인증(클라이언트 SDK 3)
- 지원되는 키 유형(클라이언트 SDK 3)
- 지원되는 메커니즘(클라이언트 SDK 3)
- 지원되는 API 작업(클라이언트 SDK 3)
- 지원되는 키 속성(클라이언트 SDK 3)
- PKCS #11 라이브러리의 코드 샘플(클라이언트 SDK 3)

PKCS #11 라이브러리용 Client SDK 3 설치

Client SDK 3의 필요 조건

PKCS #11 라이브러리에는 클라이언트가 필요합니다. AWS CloudHSM

아직 AWS CloudHSM 클라이언트를 설치 및 구성하지 않았다면 의 단계를 따라 지금 실행하십시오. 클라이언트 설치(Linux) 클라이언트를 설치 및 구성한 후 다음 명령을 사용하여 시작합니다.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl cloudhsm-client start
```

CentOS 7

```
$ sudo systemctl cloudhsm-client start
```

CentOS 8

```
$ sudo systemctl cloudhsm-client start
```

RHEL 7

```
$ sudo systemctl cloudhsm-client start
```

RHEL 8

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Client SDK 3용 PKCS #11 라이브러리 설치

다음 명령은 PKCS #11 라이브러리를 다운로드하고 설치합니다.

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-pkcs11-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-pkcs11_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-pkcs11_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-pkcs11_latest_u18.04_amd64.deb
```

- PKCS #11 라이브러리를 설치한 EC2 인스턴스에 Client SDK 3의 다른 구성 요소가 설치되어 있지 않은 경우 Client SDK 3를 부트스트랩해야 합니다. Client SDK 3의 구성 요소를 포함하는 각 인스턴스에서 이 작업을 한 번만 수행하면 됩니다.
- 다음 위치에서 PKCS #11 라이브러리 파일을 찾을 수 있습니다.

  Linux 바이너리, 구성 스크립트, 인증서 및 로그 파일:

  ```
  /opt/cloudhsm/lib
  ```

PKCS #11 라이브러리 인증(클라이언트 SDK 3)

PKCS #11 라이브러리를 사용하면 애플리케이션이 HSM에서 특정 CU(Crypto User)로 실행됩니다. 애플리케이션은 CU가 소유하고 공유하는 키만 보고 관리할 수 있습니다. HSM에서 기존 CU를 사용하거나, CU를 새로 만들 수 있습니다. CU 관리에 대한 자세한 내용은 CloudHSM CLI를 사용하여 HSM 사용자 관리 및 CloudHSM 관리 유틸리티(CMU)를 사용하여 HSM 사용자 관리를 참조하십시오.

CU를 PKCS #11 라이브러리에 지정하려면 PKCS #11 C_Login 함수의 핀 파라미터를 사용하십시오. 의 AWS CloudHSM경우 핀 매개변수의 형식은 다음과 같습니다.

```
<CU_user_name>:<password>
```

예를 들어, 다음 명령은 PKCS #11 라이브러리 핀을 사용자 이름 `CryptoUser` 및 `CUPassword123!` 암호를 사용하여 CU에 설정합니다.

```
CryptoUser:CUPassword123!
```

지원되는 키 유형(클라이언트 SDK 3)

PKCS #11 라이브러리는 다음과 같은 키 유형을 지원합니다.

키 유형	설명
RSA	2048비트~4096비트의 RSA 키를 생성합니다(256비트 증가).
EC	secp224r1(P-224), secp256r1(P-256), secp256k1(Blockchain), secp384r1(P-384) 및 secp521r1(P-521) 곡선을 사용하여 키를 생성합니다.
AES	128, 192 및 256비트 AES 키를 생성합니다.
DES3(트리플 DES)	192비트 DES3 키를 생성합니다. 예정된 변경 사항은 [1] 아래 참고를 참조하세요.
GENERIC_SECRET	1~64바이트의 일반 보안을 생성합니다.

- [1] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에는 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단를 참조하세요.

지원되는 메커니즘(클라이언트 SDK 3)

PKCS #11용 소프트웨어 라이브러리는 다음과 같은 알고리즘을 지원합니다.

- 암호화 및 복호화 - AES-CBC, AES-CTR, AES-ECB, AES-GCM, DES3-CBC, DES3-ECB, RSA-OAEP 및 RSA-PKCS
- 서명 및 확인 – RSA, HMAC 및 ECDSA(해싱 사용 및 사용 안 함)
- 해시/다이제스트 – SHA1, SHA224, SHA256, SHA384 및 SHA512
- 키 랩 – AES 키 랩,[4] AES-GCM, RSA-AES 및 RSA-OAEP

- 주요 파생 – ECDH,[5] SP800-108 CTR KDF

PKCS #11 라이브러리 메커니즘-함수 표

PKCS #11 라이브러리는 PKCS #11 사양 버전 2.40을 준수합니다. PKCS#11을 사용하여 암호화 기능을 호출하려면 주어진 메커니즘을 가진 함수를 호출하십시오. 다음 표에는 AWS CloudHSM에서 지원하는 기능과 메커니즘의 조합이 요약되어 있습니다.

지원되는 PKCS #11 메커니즘-함수 표 해석

✓ 표시는 함수 메커니즘을 AWS CloudHSM 지원함을 나타냅니다. PKCS #11 사양에 나열된 가능한 함수를 모두 지원하지는 않습니다. ✖ 표시는 PKCS #11 표준에서 허용하지만 해당 기능에 대한 메커니즘을 아직 AWS CloudHSM 지원하지 않음을 나타냅니다. 빈 셀은 PKCS #11 표준이 주어진 함수에 대해 메커니즘을 지원하지 않는다는 의미입니다.

지원되는 PKCS #11 라이브러리 메커니즘 및 함수

메커니즘	함수						
	키 또는 키 페어 생성	서명 및 확인	SR 및 VR	다이제스트	암호화 및 암호 해독	키 추출	랩 앤 UnWrap
CKM_RSA_PKCS_KEY_PAIR_GEN	✓						
CKM_RSA_X9_31_KEY_PAIR_GEN	✓[2]						
CKM_RSA_X_509		✓			✓		
CKM_RSA_PKCS 참고 사항 참조 [8]		✓[1]	✖		✓[1]		✓[1]

메커니즘	함수						
CKM_RSA_PKCS_OAEP					✓[1]		✓[6]
CKM_SHA1_RSA_PKCS		✓[3.2]					
CKM_SHA224_RSA_PKCS		✓[3.2]					
CKM_SHA256_RSA_PKCS		✓[3.2]					
CKM_SHA384_RSA_PKCS		✓[2],[3.2]					
CKM_SHA512_RSA_PKCS		✓[3.2]					
CKM_RSA_PKCS_PSS		✓[1]					
CKM_SHA1_RSA_PKCS_PSS		✓[3.2]					
CKM_SHA224_RSA_PKCS_PSS		✓[3.2]					
CKM_SHA256_RSA_PKCS_PSS		✓[3.2]					

메커니즘	함수						
CKM_SHA384_RSA_PKCS_PSS		✓[2,3.2]					
CKM_SHA512_RSA_PKCS_PSS		✓[3.2]					
CKM_EC_KEY_PAIR_GEN	✓						
CKM_ECDSA		✓[1]					
CKM_ECDSA_SHA1		✓[3.2]					
CKM_ECDSA_SHA224		✓[3.2]					
CKM_ECDSA_SHA256		✓[3.2]					
CKM_ECDSA_SHA384		✓[3.2]					
CKM_ECDSA_SHA512		✓[3.2]					
CKM_ECDH1_DERIVE						✓[5]	
CKM_SP800_108_COUNTER_KDF						✓	

메커니즘	함수						
CKM_GENERIC_SECRET_KEY_GEN	✓						
CKM_AES_KEY_GEN	✓						
CKM_AES_ECB					✓		✖
CKM_AES_CTR					✓		✖
CKM_AES_CBC					✓[3.3]		✖
CKM_AES_CBC_PAD					✓		✖
CKM_DES3_KEY_GEN 참고 사항 참조 [8]	✓						
CKM_DES3_CBC 참고 사항 참조 [8]					✓[3.3]		✖
CKM_DES3_CBC_PAD 참고 사항 참조 [8]					✓		✖

메커니즘	함수						
CKM_DES3_ECB 참고 사항 참조 8					✓		✖
CKM_AES_GCM					✓[3.3, 4]		✓[7.1]
CKM_CLOUDHSM_AES_GCM					✓[7.1]		✓[7.1]
CKM_SHA_1				✓[3.1]			
CKM_SHA_1_HMAC		✓[3.3]					
CKM_SHA224				✓[3.1]			
CKM_SHA224_HMAC		✓[3.3]					
CKM_SHA256				✓[3.1]			
CKM_SHA256_HMAC		✓[3.3]					
CKM_SHA384				✓[3.1]			
CKM_SHA384_HMAC		✓[3.3]					

메커니즘	함수						
CKM_SHA512				✓[3.1]			
CKM_SHA512_HMAC		✓[3.3]					
CKM_RSA_AES_KEY_WRAP							✓
CKM_AES_KEY_WRAP							✓
CKM_AES_KEY_WRAP_PAD							✓
CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD							✓[7.1]
CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD							✓[7.1]
CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD							✓[7.1]

메커니즘 주석

- [1] 단일 부분 작업만.

- [2] 메커니즘은 기능상 CKM_RSA_PKCS_KEY_PAIR_GEN 메커니즘과 동일하지만, p 및 q 생성을 보다 확실히 보장합니다.
- [3.1] 은 클라이언트 SDK에 따라 해싱 AWS CloudHSM 접근 방식이 다릅니다. 클라이언트 SDK 3의 경우 해싱을 수행하는 위치는 데이터 크기와 단일 부분 또는 다중 부분 작업을 사용하는지 여부에 따라 달라집니다.

  클라이언트 SDK 3의 단일 부분 작업

  표 3.1에는 클라이언트 SDK 3의 각 메커니즘에 대한 최대 데이터 세트 크기가 나열되어 있습니다. 전체 해시는 HSM 내에서 계산됩니다. 16KB를 초과하는 데이터 크기는 지원되지 않습니다.

  표 3.1, 단일 부분 작업의 최대 데이터 세트 크기

메커니즘	최대 데이터 크기
CKM_SHA_1	16296
CKM_SHA224	16264
CKM_SHA256	16296
CKM_SHA384	16232
CKM_SHA512	16232

  멀티파트 작업 클라이언트 SDK 3

  16KB보다 큰 데이터 크기를 지원하지만 데이터 크기에 따라 해싱이 발생하는 위치가 결정됩니다. 16KB 미만의 데이터 버퍼는 HSM 내에서 해시됩니다. 16KB와 시스템의 최대 데이터 크기 사이의 버퍼는 소프트웨어에서 로컬로 해시됩니다. 기억하십시오: 해시 함수에는 암호화 비밀이 필요하지 않으므로 HSM 외부에서 안전하게 계산할 수 있습니다.
- [3.2] 해싱은 클라이언트 SDK에 따라 다르게 AWS CloudHSM 접근합니다. 클라이언트 SDK 3의 경우 해싱을 수행하는 위치는 데이터 크기와 단일 부분 또는 다중 부분 작업을 사용하는지 여부에 따라 달라집니다.

  단일 부분 작업 클라이언트 SDK 3

  표 3.2에는 클라이언트 SDK 3의 각 메커니즘에 대한 최대 데이터 세트 크기가 나열되어 있습니다. 16KB를 초과하는 데이터 크기는 지원되지 않습니다.

표 3.2, 단일 부분 작업의 최대 데이터 세트 크기

메커니즘	최대 데이터 크기
CKM_SHA1_RSA_PKCS	16296
CKM_SHA224_RSA_PKCS	16264
CKM_SHA256_RSA_PKCS	16296
CKM_SHA384_RSA_PKCS	16232
CKM_SHA512_RSA_PKCS	16232
CKM_SHA1_RSA_PKCS_PSS	16296
CKM_SHA224_RSA_PKCS_PSS	16264
CKM_SHA256_RSA_PKCS_PSS	16296
CKM_SHA384_RSA_PKCS_PSS	16232
CKM_SHA512_RSA_PKCS_PSS	16232
CKM_ECDSA_SHA1	16296
CKM_ECDSA_SHA224	16264
CKM_ECDSA_SHA256	16296
CKM_ECDSA_SHA384	16232
CKM_ECDSA_SHA512	16232

멀티파트 작업 클라이언트 SDK 3

16KB보다 큰 데이터 크기를 지원하지만 데이터 크기에 따라 해싱이 발생하는 위치가 결정됩니다. 16KB 미만의 데이터 버퍼는 HSM 내에서 해시됩니다. 16KB와 시스템의 최대 데이터 크기 사이의 버퍼는 소프트웨어에서 로컬로 해시됩니다. 기억하십시오: 해시 함수에는 암호화 비밀이 필요하지 않으므로 HSM 외부에서 안전하게 계산할 수 있습니다.

- [3.3] 다음 메커니즘 중 하나를 사용하여 데이터로 작업할 때 데이터 버퍼가 최대 데이터 크기를 초과하면 작업 결과에 오류가 생깁니다. 이러한 메커니즘의 경우 모든 데이터 처리는 HSM 내에서 이루어져야 합니다. 다음 표에는 각 메커니즘에 설정된 최대 데이터 크기가 나와 있습니다.

  표 3.3, 최대 데이터 세트 크기

메커니즘	최대 데이터 크기
CKM_SHA_1_HMAC	16288
CKM_SHA224_HMAC	16256
CKM_SHA256_HMAC	16288
CKM_SHA384_HMAC	16224
CKM_SHA512_HMAC	16224
CKM_AES_CBC	16272
CKM_AES_GCM	16224
CKM_CLOUDHSM_AES_GCM	16224
CKM_DES3_CBC	16280

- [4] AES-GCM 암호화를 수행할 때 HSM은 애플리케이션에서 IV(초기화 벡터) 데이터를 수락하지 않습니다. 생성되는 IV를 사용해야 합니다. HSM이 제공하는 12바이트 IV는 사용자가 제공하는 CK_GCM_PARAMS 파라미터 구조의 pIV 요소가 가리키는 메모리 참조에 기록됩니다. 사용자의 혼동을 피하기 위해 버전 1.1.1 이상인 PKCS#11 SDK는 AES-GCM 암호화가 초기화될 때 pIV가 초기화된 버퍼를 가리키는지 확인합니다.
- [5] 클라이언트 SDK 3만 해당됩니다. 메커니즘은 SSL/TLS 오프로드 사례를 지원하기 위해 구현됐으며 HSM 내에서만 부분적으로 실행됩니다. 이 메커니즘을 사용하기 전에 PKCS#11 SDK의 알려진 문제의 “문제: ECDH 키 파생은 부분적으로 HSM 내에서만 실행됩니다”를 참조하십시오. CKM_ECDH1_DERIVE는 secp521r1(P-521) 곡선을 지원하지 않습니다.
- [6] 다음 CK_MECHANISM_TYPE 및 CK_RSA_PKCS_MGF_TYPE은 CKM_RSA_PKCS_OAEP의 경우 CK_RSA_PKCS_OAEP_PARAMS로 지원됩니다.
  - CKM_SHA_1CKG_MGF1_SHA1을 사용하는
  - CKG_MGF1_SHA224를 사용하는 CKM_SHA224

  - CKG_MGF1_SHA256를 사용하는 CKM_SHA256
  - CKM_MGF1_SHA384를 사용하는 CKM_SHA384
  - CKM_MGF1_SHA512를 사용하는 CKM_SHA512
- [7.1] 공급자 정의 메커니즘. CloudHSM 공급자 정의 메커니즘을 사용하려면 컴파일하는 동안 PKCS #11 애플리케이션에 /opt/cloudhsm/include/pkcs11t.h가 포함되어 있어야 합니다.

  CKM_CLOUDHSM_AES_GCM: 이 독점 메커니즘은 CKM_AES_GCM 표준에 대한 프로그래밍 방식으로 안전한 대안입니다. 이 메커니즘은 암호 초기화 중에 제공되는 CK_GCM_PARAMS 구조로 다시 암호 텍스트를 쓰는 대신 암호 텍스트 앞에 HSM에 의해 생성된 IV를 추가합니다. 이 메커니즘은 C_Encrypt, C_WrapKey, C_Decrypt 및 C_UnwrapKey 함수와 함께 사용할 수 있습니다. 이 메커니즘을 사용할 때는 CK_GCM_PARAMS 구문의 piV 변수를 NULL로 설정해야 합니다. C_Decrypt 및 C_UnwrapKey와 함께 이 메커니즘을 사용할 경우, IV는 언래핑되는 암호화 텍스트 앞에 추가될 것으로 예상됩니다.

  CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD: PKCS #5 패딩을 사용하는 AES 키 래핑

  CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD: 제로 패딩을 사용하는 AES 키 래핑

  AES 키 래핑에 대한 추가 정보는 AES 키 래핑을 참조하십시오.
- [8] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에는 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단 섹션을 참조하십시오.

지원되는 API 작업(클라이언트 SDK 3)

PKCS #11 라이브러리는 다음과 같은 PKCS #11 API 작업을 지원합니다.

- C_CloseAllSessions
- C_CloseSession
- C_CreateObject
- C_Decrypt
- C_DecryptFinal
- C_DecryptInit
- C_DecryptUpdate
- C_DeriveKey

- C_DestroyObject
- C_Digest
- C_DigestFinal
- C_DigestInit
- C_DigestUpdate
- C_Encrypt
- C_EncryptFinal
- C_EncryptInit
- C_EncryptUpdate
- C_Finalize
- C_FindObjects
- C_FindObjectsFinal
- C_FindObjectsInit
- C_GenerateKey
- C_GenerateKeyPair
- C_GenerateRandom
- C_GetAttributeValue
- C_GetFunctionList
- C_GetInfo
- C_GetMechanismInfo
- C_GetMechanismList
- C_GetSessionInfo
- C_GetSlotInfo
- C_GetSlotList
- C_GetTokenInfo
- C_Initialize
- C_Login
- C_Logout
- C_OpenSession
- C_Sign

- C_SignFinal
- C_SignInit
- C_SignRecover (클라이언트 SDK 3만 지원)
- C_SignRecoverInit (클라이언트 SDK 3만 지원)
- C_SignUpdate
- C_UnWrapKey
- C_Verify
- C_VerifyFinal
- C_VerifyInit
- C_VerifyRecover (클라이언트 SDK 3만 지원)
- C_VerifyRecoverInit (클라이언트 SDK 3만 지원)
- C_VerifyUpdate
- C_WrapKey

지원되는 키 속성(클라이언트 SDK 3)

키 객체는 퍼블릭 키, 프라이빗 키 또는 비밀 키일 수 있습니다. 키 객체에 대해 허용되는 작업은 속성을 통해 지정됩니다. 속성은 키 객체가 생성될 때 정의됩니다. PKCS #11 라이브러리를 사용하는 경우 PKCS #11 표준에 지정된 대로 기본값을 할당합니다.

AWS CloudHSM PKCS #11 사양에 나열된 모든 속성을 지원하지는 않습니다. 우리는 우리가 지원하는 모든 속성에 대한 사양을 준수합니다. 이러한 속성은 각 표에 나열되어 있습니다.

객체를 생성, 수정 또는 복사하는 C_CreateObject, C_GenerateKey, C_GenerateKeyPair, C_UnwrapKey, C_DeriveKey 등의 암호화 함수는 속성 템플릿을 파라미터 중 하나로 사용합니다. 객체 생성 중 속성 템플릿 전달에 대한 자세한 내용은 Generate keys through PKCS #11 library 샘플을 참조하십시오.

PKCS #11 라이브러리 속성 테이블 해석

PKCS #11 라이브러리 테이블에는 키 유형별로 다른 속성 목록이 포함되어 있습니다. 에서 특정 암호화 함수를 사용할 때 특정 키 유형에 대해 지정된 속성이 지원되는지 여부를 나타냅니다. AWS CloudHSM

범례:

- ✓는 CloudHSM이 특정 키 유형에 대해 해당 속성을 지원함을 나타냅니다.
- ✖는 CloudHSM이 특정 키 유형에 대해 해당 속성을 지원하지 않음을 나타냅니다.
- R은 속성 값이 특정 키 유형에 대해 읽기 전용으로 설정됨을 나타냅니다.
- S는 중요하므로 `GetAttributeValue`로 속성을 읽을 수 없음을 나타냅니다.
- 기본값 열의 빈 셀은 속성에 할당된 특정 기본값이 없음을 나타냅니다.

GenerateKeyPair

속성	키 유형				기본 값
	EC 프라이빗	EC 퍼블릭	RSA 프라이빗	RSA 퍼블릭	
CKA_CLASS	✓	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	✓	
CKA_LABEL	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	True
CKA_TOKEN	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✖	✓	✖	✓	False
CKA_DECRYPT	✓	✖	✓	✖	False

속성	키 유형				기본 값
CKA_DERIVE	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	✓	True
CKA_SIGN	✓	✖	✓	✖	False
CKA_SIGN_RECOVER	✖	✖	✓[3]	✖	
CKA_VERIFY	✖	✓	✖	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✓[4]	
CKA_WRAP	✖	✓	✖	✓	False
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	
CKA_TRUSTED	✖	✓	✖	✓	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	False
CKA_UNWRAP	✓	✖	✓	✖	False

속성	키 유형				기본 값
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	True
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	
CKA_EXTRACTABLE	✓	✖	✓	✖	True
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	
CKA_MODULUS	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✓[2]	
CKA_PRIME_1	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	

속성	키 유형				기본 값
CKA_EXPONENT_2	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	✓[2]	
CKA_EC_PARAMS	✖	✓[2]	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	

GenerateKey

속성	키 유형			기본 값
	AES	DES3	일반 보안	
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	

속성	키 유형			기본 값
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✓	✓	✖	False
CKA_DECRYPT	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	True
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	True
CKA_VERIFY_RECOVER	✖	✖	✖	

속성	키 유형			기본 값
CKA_WRAP	✓	✓	✖	False
CKA_WRAP_TEMPLATE	✓	✓	✖	
CKA_TRUSTED	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✓	✓	False
CKA_UNWRAP	✓	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✓	✖	
CKA_SENSITIVE	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	✖	✖	✖	
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_MODULUS	✖	✖	✖	

속성	키 유형			기본 값
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	

속성	키 유형			기본 값
CKA_CHECK_VALUE	R	R	R	

CreateObject

속성	키 유형							기본 값
	EC 프라이빗	EC 퍼블릭	RSA 프라이빗	RSA 퍼블릭	AES	DES3	일반 보안	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	R	R	False
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖	False

속성	키 유형							기본 값
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✖	✓[3]	✖	✖	✖	✖	False
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✓[4]	✖	✖	✖	
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖	False
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖	
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓	False

속성	키 유형							기본 값
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	R	R	R	
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	R	R	R	
CKA_MODULUS	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✓	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✓	✖	✖	✖	✖	

속성	키 유형							기본 값
CKA_EXPONENT_1	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✓[2]	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_EC_PARAMS	✓[2]	✓[2]	✖	✖	✖	✖	✖	
CKA_EC_POINT	✖	✓[2]	✖	✖	✖	✖	✖	
CKA_VALUE	✓[2]	✖	✖	✖	✓[2]	✓[2]	✓[2]	
CKA_VALUE_LEN	✖	✖	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	R	R	

UnwrapKey

속성	키 유형						기본 값
		EC 프라이빗	RSA 프라이빗	AES	DES3	일반 보안	

속성	키 유형					기본 값
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	False
CKA_TOKEN	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✖	✖	✓	✓	✖	False
CKA_DECRYPT	✖	✓	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	✓	✓	False

속성	키 유형					기본 값
CKA_SIGN_RECOVER	✖	✓[3]	✖	✖	✖	False
CKA_VERIFY	✖	✖	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✓	✓	✖	False
CKA_UNWRAP	✖	✓	✓	✓	✖	False
CKA_SENSITIVE	✓	✓	✓	✓	✓	True
CKA_EXTRACTABLE	✓	✓	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	
CKA_MODULUS	✖	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	✖	✖	

속성	키 유형					기본 값
CKA_PRIME_2	✖	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	

DeriveKey

속성	키 유형			기본 값
	AES	DES3	일반 보안	
CKA_CLASS	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✓	✓	✖	False
CKA_DECRYPT	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓[1]	✓[1]	✓[1]	True
CKA_SIGN	✓	✓	✓	False

속성	키 유형			기본 값
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	False
CKA_UNWRAP	✓	✓	✖	False
CKA_SENSITIVE	✓	✓	✓	True
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	

속성	키 유형			기본 값
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

속성	키 유형						
	EC 프라이빗	EC 퍼블릭	RSA 프라이빗	RSA 퍼블릭	AES	DES3	일반 보안
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓

속성	키 유형						
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓
CKA_SIGN_RECOVER	✖	✖	✓	✖	✖	✖	✖
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓
CKA_VERIFY_RECOVER	✖	✖	✖	✓	✖	✖	✖
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✓
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓

속성	키 유형						
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_NEVER_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_ALWAYS_SENSITIVE	R	R;	R	R	R	R	R
CKA_MODULUS	✖	✖	✓	✓	✖	✖	✖
CKA_MODULUS_BITS	✖	✖	✖	✓	✖	✖	✖
CKA_PRIME_1	✖	✖	S	✖	✖	✖	✖
CKA_PRIME_2	✖	✖	S	✖	✖	✖	✖
CKA_COEFFICIENT	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_1	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_2	✖	✖	S	✖	✖	✖	✖
CKA_PRIVATE_EXPONENT	✖	✖	S	✖	✖	✖	✖

속성	키 유형						
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
CKA_VALUE	S	✖	✖	✖	✓[2]	✓[2]	✓[2]
CKA_VALUE_LEN	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	✖

속성 주석

- [1] 이 속성은 펌웨어에서 부분적으로 지원되며 명시적으로 기본값으로만 설정되어야 합니다.
- [2] 필수 속성.
- [3] 클라이언트 SDK 3만 해당됩니다. CKA_SIGN_RECOVER 속성은 CKA_SIGN 속성에서 파생되었습니다. 설정된 경우 CKA_SIGN에 대해 설정된 것과 동일한 값으로만 설정할 수 있습니다. 설정되지 않은 경우 기본값 CKA_SIGN을 파생합니다. CloudHSM은 RSA 기반의 복구 가능한 서명 메커니즘만 지원하므로, 이 속성은 현재 RSA 퍼블릭 키에만 적용됩니다.
- [4] 클라이언트 SDK 3만 해당됩니다. CKA_VERIFY_RECOVER 속성은 CKA_VERIFY 속성에서 파생되었습니다. 설정된 경우 CKA_VERIFY에 대해 설정된 것과 동일한 값으로만 설정할 수 있습니다. 설정되지 않은 경우 기본값 CKA_VERIFY을 파생합니다. CloudHSM은 RSA 기반의 복구 가능한 서명 메커니즘만 지원하므로, 이 속성은 현재 RSA 퍼블릭 키에만 적용됩니다.

속성 수정

객체의 일부 속성은 객체가 생성된 후 수정할 수 있지만, 일부 속성은 수정할 수 없습니다. 속성을 수정하려면 cloudhsm_mgmt_util에서 setAttribute 명령을 사용하십시오. cloudhsm_mgmt_util에서 listAttribute 명령을 사용하여 속성 및 이를 나타내는 상수 목록을 파생시킬 수도 있습니다.

다음 목록은 객체 생성 후 수정할 수 있는 속성을 보여 줍니다.

- CKA_LABEL
- CKA_TOKEN

> **Note**
>
> 세션 키를 토큰 키로 변경하는 경우에만 수정이 허용됩니다. 속성 값을 변경하려면 key_mgmt_util에서 setAttribute 명령을 사용하십시오.

- CKA_ENCRYPT
- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

> **Note**
>
> 이 속성은 키 파생을 지원합니다. 모든 퍼블릭 키에 대해 False여야 하고 True로 설정할 수 없습니다. 보안 및 EC 프라이빗 키의 경우 True 또는 False로 설정할 수 있습니다.

- CKA_TRUSTED

> **Note**
>
> CO(Crypto Officer)만 이 속성을 True 또는 False로 설정할 수 있습니다.

- CKA_WRAP_WITH_TRUSTED

> **Note**
>
> 이 속성을 내보낼 수 있는 데이터 키에 적용하여 이 키를 CKA_TRUSTED로 표시된 키로만 래핑할 수 있도록 지정합니다. CKA_WRAP_WITH_TRUSTED는 true로 설정하면 속성이 읽기 전용이 되며 속성을 변경하거나 제거할 수 없습니다.

오류 코드 해석

템플릿에서 특정 키로 지원되지 않는 속성을 지정하면 오류가 발생합니다. 다음 표에 사양을 위반하면 발생되는 오류 코드가 나와 있습니다.

오류 코드	설명
CKR_TEMPLATE_INCONSISTENT	어떤 속성이 PKCS #11 사양을 준수하지만 CloudHSM에서 지원되지 않는 경우, 속성 템플릿에서 해당 속성을 지정하면 이 오류가 발생합니다.
CKR_ATTRIBUTE_TYPE_INVALID	PKCS #11 사양을 준수하지만 CloudHSM에서 지원되지 않는 속성의 값을 검색하면 이 오류가 발생합니다.
CKR_ATTRIBUTE_INCOMPLETE	속성 템플릿에서 필수 속성을 지정하지 않으면 이 오류가 발생합니다.
CKR_ATTRIBUTE_READ_ONLY	속성 템플릿에서 읽기 전용 속성을 지정하면 이 오류가 발생합니다.

PKCS #11 라이브러리의 코드 샘플(클라이언트 SDK 3)

위의 코드 샘플은 PKCS #11 라이브러리를 사용하여 기본 작업을 수행하는 방법을 GitHub 보여줍니다.

샘플 코드 사전 조건

샘플을 실행하기 전에 다음 단계를 수행하여 환경을 설정합니다.

- 클라이언트 SDK 3용 PKCS #11 라이브러리를 설치하고 구성합니다.
- CU(Cryptographic User) 설정 애플리케이션은 이 HSM 계정을 사용하여 HSM에서 코드 샘플을 실행합니다.

코드 샘플

PKCS #11 AWS CloudHSM 소프트웨어 라이브러리의 코드 샘플은 에서 사용할 수 있습니다. GitHub 이 리포지토리에는 암호화, 암호 해독, 서명 및 확인 등 PKCS #11을 사용하여 일반적인 작업을 수행하는 방법에 대한 예제가 포함되어 있습니다.

- 키 생성(AES, RSA, EC)
- 키 속성 나열
- AES GCM을 사용하여 데이터 암호화 및 암호화 해제
- AES_CTR을 사용하여 데이터 암호화 및 복호화
- 3DES를 사용하여 데이터 암호화 및 복호화
- RSA를 사용하여 데이터 서명 및 확인
- HMAC KDF를 사용하여 키 추출
- PKCS #5 패딩을 사용하는 AES로 키 래핑 및 언래핑
- 패딩을 사용하지 않는 AES로 키 래핑 및 언래핑
- 제로 패딩을 사용하는 AES로 키 래핑 및 언래핑
- AES-GCM을 사용하여 키 래핑 및 언래핑
- RSA를 사용하여 키 래핑 및 언래핑

OpenSSL 다이내믹 엔진용 클라이언트 SDK 3 설치

클라이언트 SDK 3을 클러스터에 연결하려면 클라이언트 데몬이 필요합니다. 다음을 지원합니다.

- 2048비트, 3072비트 및 4096비트 키를 위한 RSA 키 생성.
- RSA 서명/확인.
- RSA 암호화/암호 해독.
- 암호를 통한 보안과 FIPS 확인이 지원되는 난수 생성.

주제

- 클라이언트 SDK 3을 사용하는 OpenSSL 동적 엔진의 사전 요구 사항
- Client SDK 3용 OpenSSL Dynamic Engine 설치하기
- 클라이언트 SDK 3용 OpenSSL Dynamic Engine 사용하기

클라이언트 SDK 3을 사용하는 OpenSSL 동적 엔진의 사전 요구 사항

지원되는 플랫폼에 대한 자세한 내용은 Client SDK 3 지원 플랫폼 섹션을 참조하세요.

OpenSSL용 AWS CloudHSM 동적 엔진을 사용하려면 먼저 클라이언트가 필요합니다. AWS CloudHSM

클라이언트는 클러스터의 HSM과 end-to-end 암호화된 통신을 설정하는 데몬이며, OpenSSL 엔진은 클라이언트와 로컬로 통신합니다. 클라이언트를 설치 및 구성하려면 을 참조하십시오. AWS CloudHSM 클라이언트 설치(Linux) 그리고 다음 명령을 사용하여 시작합니다.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl cloudhsm-client start
```

CentOS 6

```
$ sudo systemctl start cloudhsm-client
```

CentOS 7

```
$ sudo systemctl cloudhsm-client start
```

RHEL 6

```
$ sudo systemctl start cloudhsm-client
```

RHEL 7

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Client SDK 3용 OpenSSL Dynamic Engine 설치하기

다음 단계는 OpenSSL용 AWS CloudHSM 동적 엔진을 설치하고 구성하는 방법을 설명합니다. 업그레이드에 대한 자세한 내용은 클라이언트 SDK 3 업그레이드을 참조하십시오.

OpenSSL 엔진을 설치 및 구성하려면

1. 다음 명령을 사용하여 OpenSSL 엔진을 다운로드하고 설치합니다.

   Amazon Linux

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
   ```

   Amazon Linux 2

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
   ```

   CentOS 6

   ```
   $ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
   ```

   ```
   $ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
   ```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

RHEL 6

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-dyn-latest.el7.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-dyn_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-dyn_latest_amd64.deb
```

`/opt/cloudhsm/lib/libcloudhsm_openssl.so`에 OpenSSL 엔진이 설치되어 있습니다.

2. 다음 명령을 사용하여 CU(Crypto User)의 인증서가 포함된 `n3fips_password`라는 환경 변수를 설정합니다.

```
$ export n3fips_password=<HSM user name>:<password>
```

클라이언트 SDK 3용 OpenSSL Dynamic Engine 사용하기

OpenSSL 통합 애플리케이션에서 OpenSSL용 AWS CloudHSM 동적 엔진을 사용하려면 애플리케이션에서 이름이 지정된 OpenSSL 동적 엔진을 사용해야 합니다. `cloudhsm` Dynamic Engine을 위한 공유 라이브러리는 `/opt/cloudhsm/lib/libcloudhsm_openssl.so`에 위치합니다.

OpenSSL 명령줄에서 OpenSSL용 AWS CloudHSM 동적 엔진을 사용하려면 `-engine` 옵션을 사용하여 OpenSSL 동적 엔진을 명명하도록 지정합니다. `cloudhsm` 예:

```
$ openssl s_server -cert server.crt -key server.key -engine cloudhsm
```

JCE 공급자용 클라이언트 SDK 3

JCE 제공자는 AWS CloudHSM JCE (자바 암호화 확장) 공급자 프레임워크를 기반으로 구축된 공급자 구현입니다. JCE를 사용하면 자바 개발자 키트(Java Development Kit)를 사용하여 암호화 작업을 수행할 수 있습니다. 이 안내서에서는 AWS CloudHSM JCE 제공자를 JCE 제공자라고도 합니다. JCE 공급자와 JDK를 사용하여 암호화 작업을 HSM으로 오프로드하십시오.

주제

- 클라이언트 SDK 3용 AWS CloudHSM JCE 공급자 설치 및 사용
- 지원되는 메커니즘(클라이언트 SDK 3
- Client SDK 3에 지원되는 Java 키 속성
- Java for Client SDK 3용 AWS CloudHSM 소프트웨어 라이브러리의 코드 샘플
- 클라이언트 SDK 3에 AWS CloudHSM KeyStore 자바 클래스 사용

클라이언트 SDK 3용 AWS CloudHSM JCE 공급자 설치 및 사용

JCE 공급자를 사용하려면 먼저 클라이언트가 필요합니다. AWS CloudHSM

클라이언트는 클러스터의 HSM과 end-to-end 암호화된 통신을 설정하는 데몬입니다. JCE 공급자는 클라이언트와 로컬로 통신합니다. AWS CloudHSM 클라이언트 패키지를 설치 및 구성하지 않았다면, 지금 의 단계에 따라 설치하세요. 클라이언트 설치(Linux) 클라이언트를 설치 및 구성한 후 다음 명령을 사용하여 시작합니다.

JCE 공급자는 Linux 및 호환 운영 체제에서만 지원됩니다.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo systemctl cloudhsm-client start
```

CentOS 7

```
$ sudo systemctl cloudhsm-client start
```

CentOS 8

```
$ sudo systemctl cloudhsm-client start
```

RHEL 7

```
$ sudo systemctl cloudhsm-client start
```

RHEL 8

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo systemctl cloudhsm-client start
```

주제

- JCE 공급자 설치
- 설치 검증
- JCE 공급자에게 자격 증명 제공
- JCE 공급자의 키 관리 기본 사항

JCE 공급자 설치

다음 명령을 사용하여 JCE 공급자를 다운로드하고 설치합니다. 이 공급자는 Linux 및 호환 운영 체제에서만 지원됩니다.

> **Note**
>
> 업그레이드에 대한 자세한 내용은 클라이언트 SDK 3 업그레이드 단원을 참조하십시오.

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-jce-latest.el6.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el6.x86_64.rpm
```

Amazon Linux 2

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

CentOS 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el8.x86_64.rpm
```

RHEL 7

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL7/cloudhsm-client-jce-latest.el7.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el7.x86_64.rpm
```

RHEL 8

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL8/cloudhsm-client-jce-latest.el8.x86_64.rpm
```

```
$ sudo yum install ./cloudhsm-client-jce-latest.el8.x86_64.rpm
```

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-jce_latest_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-jce_latest_amd64.deb
```

Ubuntu 18.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Bionic/cloudhsm-client-jce_latest_u18.04_amd64.deb
```

```
$ sudo apt install ./cloudhsm-client-jce_latest_u18.04_amd64.deb
```

앞의 명령을 실행하면 다음과 같은 JCE 공급자 파일을 찾을 수 있습니다.

- /opt/cloudhsm/java/cloudhsm-*version*.jar
- /opt/cloudhsm/java/cloudhsm-test-*version*.jar
- /opt/cloudhsm/java/hamcrest-all-1.3.jar
- /opt/cloudhsm/java/junit.jar
- /opt/cloudhsm/java/log4j-api-2.17.1.jar
- /opt/cloudhsm/java/log4j-core-2.17.1.jar
- /opt/cloudhsm/lib/libcaviumjca.so

설치 검증

HSM에서 기본 작업을 수행하여 설치를 검증합니다.

JCE 공급자 설치를 확인하려면

1. (선택 사항) 환경에 Java가 아직 설치되어 있지 않은 경우 다음 명령을 사용하여 설치합니다.

   Linux (and compatible libraries)

   ```
   $ sudo yum install java-1.8.0-openjdk
   ```

   Ubuntu

   ```
   $ sudo apt-get install openjdk-8-jre
   ```

2. 다음 명령을 사용하여 필요한 환경 변수를 설정합니다. *<HSM user name>* 및 *<password>*를 CU(Crypto User)의 자격 증명으로 바꿉니다.

   ```
   $ export LD_LIBRARY_PATH=/opt/cloudhsm/lib
   ```

   ```
   $ export HSM_PARTITION=PARTITION_1
   ```

   ```
   $ export HSM_USER=<HSM user name>
   ```

   ```
   $ export HSM_PASSWORD=<password>
   ```

3. 다음 명령을 사용하여 기본 기능 테스트를 실행합니다. 성공하면 명령이 다음과 비슷하게 출력됩니다.

```
$ java8 -classpath "/opt/cloudhsm/java/*" org.junit.runner.JUnitCore
 TestBasicFunctionality

JUnit version 4.11
.2018-08-20 17:53:48,514 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:33) - Adding provider.
2018-08-20 17:53:48,612 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:42) - Logging in.
2018-08-20 17:53:48,612 INFO [main] cfm2.LoginManager (LoginManager.java:104) -
 Looking for credentials in HsmCredentials.properties
2018-08-20 17:53:48,612 INFO [main] cfm2.LoginManager (LoginManager.java:122) -
 Looking for credentials in System.properties
2018-08-20 17:53:48,613 INFO [main] cfm2.LoginManager (LoginManager.java:130) -
 Looking for credentials in System.env
 SDK Version: 2.03
2018-08-20 17:53:48,655 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:54) - Generating AES Key with key size 256.
2018-08-20 17:53:48,698 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:63) - Encrypting with AES Key.
2018-08-20 17:53:48,705 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:84) - Deleting AES Key.
2018-08-20 17:53:48,707 DEBUG [main] TestBasicFunctionality
 (TestBasicFunctionality.java:92) - Logging out.

Time: 0.205

OK (1 test)
```

JCE 공급자에게 자격 증명 제공

애플리케이션에서 인증서를 사용하려면 먼저 HSM이 Java 애플리케이션을 인증해야 합니다. 애플리케이션마다 한 세션을 사용할 수 있습니다. HSM이 명시적 로그인이나 암시적 로그인 방법을 사용하여 세션을 인증합니다.

명시적 로그인 – 이 메소드를 사용하면 애플리케이션에서 직접 CloudHSM 자격 증명을 제공할 수 있습니다. `LoginManager.login()` 메서드가 사용됩니다. 이때 CU 사용자 이름, 암호 및 HSM 파티션 ID를 전달합니다. 명시적 로그인 방법 사용에 대한 자세한 내용은 Login to an HSM 코드 예제를 참조하십시오.

암시적 로그인 - 이 메서드를 사용하면 새 속성 파일, 시스템 속성 또는 환경 변수로 CloudHSM 자격 증명을 설정할 수 있습니다.

- 새 속성 파일 - HsmCredentials.properties라는 이름의 새 파일을 생성하고 애플리케이션의 CLASSPATH에 추가합니다. 파일에 다음을 포함해야 합니다.

```
HSM_PARTITION = PARTITION_1
HSM_USER = <HSM user name>
HSM_PASSWORD = <password>
```

- 시스템 속성 – 애플리케이션을 실행할 때 시스템 속성을 통해 자격 증명을 설정합니다. 다음 예제와 같이 이 작업을 두 가지 방법으로 수행할 수 있습니다.

```
$ java -DHSM_PARTITION=PARTITION_1 -DHSM_USER=<HSM user name> -
DHSM_PASSWORD=<password>
```

```
System.setProperty("HSM_PARTITION","PARTITION_1");
System.setProperty("HSM_USER","<HSM user name>");
System.setProperty("HSM_PASSWORD","<password>");
```

- 환경 변수 – 자격 증명을 환경 변수로 설정합니다.

```
$ export HSM_PARTITION=PARTITION_1
$ export HSM_USER=<HSM user name>
$ export HSM_PASSWORD=<password>
```

애플리케이션이 자격 증명을 제공하지 않거나 HSM이 세션을 인증하기 전에 작업을 시도하면 자격 증명을 사용할 수 없기도 합니다. 이러한 경우 Java용 CloudHSM 소프트웨어 라이브러리가 다음 순서로 자격 증명을 검색합니다.

1. HsmCredentials.properties
2. 시스템 속성
3. 환경 변수

오류 처리

암시적 로그인 방법보다 명시적 로그인 방법에서 오류 처리가 더 쉽습니다. LoginManager 클래스를 사용하면 애플리케이션이 오류를 처리하는 방법을 더 세부적으로 제어할 수 있습니다. 암시적 로그인 방법을 사용하면 자격 증명이 잘못되거나 HSM의 세션 인증에 문제가 있을 때 오류 처리에서 이해하기 어려워집니다.

JCE 공급자의 키 관리 기본 사항

JCE 공급자의 키 관리에 대한 기본 사항은 키 가져오기, 키 내보내기, 핸들별 키 로드 또는 키 삭제를 포함합니다. 키 관리에 대한 자세한 내용은 Manage keys 코드 예제를 참조하십시오.

또한 코드 샘플에서 JCE 제공자 코드 예시를 더 확인할 수 있습니다.

지원되는 메커니즘(클라이언트 SDK 3

에서 지원하는 Java 암호화 아키텍처 (JCA) 인터페이스 및 엔진 클래스에 대한 자세한 내용은 다음 항목을 참조하십시오. AWS CloudHSM

주제

- 지원되는 키
- 지원되는 암호
- 지원되는 다이제스트
- 지원되는 해시 기반 메시지 인증 코드(HMAC) 알고리즘
- 지원되는 서명/확인 메커니즘
- 메커니즘 주석

지원되는 키

Java용 AWS CloudHSM 소프트웨어 라이브러리를 사용하면 다음과 같은 키 유형을 생성할 수 있습니다.

- AES - 128비트, 192비트 및 256비트 AES 키입니다.
- DeSede — 92비트 3DES 키. 예정된 변경 사항은 1 아래 참고를 참조하세요.
- NIST 곡선 secp256r1(P-256), secp384r1(P-384) 및 secp256k1(블록체인)용 ECC 키 페어입니다.
- RSA - 2048비트 ~ 4096비트의 RSA 키입니다(256비트씩 증분).

표준 파라미터 외에도 생성되는 각 키에 대해 다음의 파라미터도 지원됩니다.

- Label: 키를 검색하는 데 사용할 수 있는 키 라벨.
- isExtractable: HSM에서 키를 내보낼 수 있는지 여부를 나타냅니다.
- isPersistent: 현재 세션이 종료될 때 키가 HSM에 남아 있는지 여부를 나타냅니다.

> **Note**
>
> Java 라이브러리 버전 3.1은 파라미터를 보다 자세히 지정할 수 있는 기능을 제공합니다. 자세한 내용은 지원되는 Java 속성을 참조하세요.

지원되는 암호

Java용 AWS CloudHSM 소프트웨어 라이브러리는 다음과 같은 알고리즘, 모드 및 패딩 조합을 지원합니다.

알고리즘	Mode	패딩	참고

AES	CBC	`AES/CBC/NoPadding`
		
		`AES/CBC/PKCS5Padding`
			`Cipher.ENCRYPT_MODE` 와 `Cipher.DECRYPT_MODE` 를 구현합니다.

AES	ECB	`AES/ECB/NoPadding`
		
		`AES/ECB/PKCS5Padding`
			`Cipher.ENCRYPT_MODE` 와 `Cipher.DECRYPT_MODE` 를 구현합니다. 변환 AES를 사용합니다.

AES	CTR	`AES/CTR/NoPadding`	`Cipher.ENCRYPT_MODE` 와 `Cipher.DECRYPT_MODE` 를 구현합니다.
AES	GCM	`AES/GCM/NoPadding`	`Cipher.ENCRYPT_MODE` 와 `Cipher.DECRYPT_MODE` 및 `Cipher.WRAP_MODE` 와

알고리즘	Mode	패딩	참고
			`Cipher.UNWRAP_MODE` 를 구현합니다. AES-GCM 암호화를 수행할 때 HSM은 요청에서 초기화 벡터(IV)를 무시하고 HSM이 생성하는 IV를 사용합니다. 작업이 완료되면 `Cipher.getIV()` 를 호출하여 IV를 가져와야 합니다.

AESWrap	ECB	`AESWrap/ECB/ZeroPadding`
		
		`AESWrap/ECB/NoPadding`
		
		`AESWrap/ECB/PKCS5Padding`
			`Cipher.WRAP_MODE` 와 `Cipher.UNWRAP_MODE` 를 구현합니다. 변환 AES를 사용합니다.

알고리즘	Mode	패딩	참고

DESede (Triple DES)	CBC	`DESede/CBC/NoPadding`
		
		`DESede/CBC/PKCS5Padding`
			`Cipher.ENCRYPT_MODE` 와 `Cipher.DECRYPT_MODE` 를 구현합니다.
			
			키 생성 루틴은 168비트 또는 192비트의 크기를 허용합니다. 하지만 내부적으로 모든 DESede 키는 192비트입니다.
			
			예정된 변경 사항은 1 아래 참고를 참조하세요.

DESede (Triple DES)	ECB	`DESede/ECB/NoPadding`
		
		`DESede/ECB/PKCS5Padding`
			`Cipher.ENCRYPT_MODE` 와 `Cipher.DECRYPT_MODE` 를 구현합니다.
			
			키 생성 루틴은 168비트 또는 192비트의 크기를 허용합니다. 하지만 내부적으로 모든 DESede 키는 192비트입니다.
			
			예정된 변경 사항은 1 아래 참고를 참조하세요.

알고리즘	Mode	패딩	참고

RSA	ECB	`RSA/ECB/NoPadding`
		
		`RSA/ECB/PKCS1Padding`
			`Cipher.ENCRYPT_MODE` 와 `Cipher.DECRYPT_MODE` 를 구현합니다.
			
			예정된 변경 사항은 1 아래 참고를 참조하세요.

알고리즘	Mode	패딩	참고
RSA	ECB	RSA/ECB/OAEPPadding RSA/ECB/OAEPWithSHA-1ANDMGF1Padding RSA/ECB/OAEPWithSHA-224ANDMGF1Padding RSA/ECB/OAEPWithSHA-256ANDMGF1Padding RSA/ECB/OAEPWithSHA-384ANDMGF1Padding RSA/ECB/OAEPWithSHA-512ANDMGF1Padding	Cipher.ENCRYPT_MODE , Cipher.DECRYPT_MODE , Cipher.WRAP_MODE , Cipher.UNWRAP_MODE 를 구현합니다. OAEPPadding 은 SHA-1 패딩 유형을 포함하는 OAEP입니다.
RSAAESWrap	ECB	OAEPPADDING	Cipher.WRAP_Mode 와 Cipher.UNWRAP_MODE 를 구현합니다.

지원되는 다이제스트

Java용 AWS CloudHSM 소프트웨어 라이브러리는 다음 메시지 다이제스트를 지원합니다.

- `SHA-1`
- `SHA-224`
- `SHA-256`
- `SHA-384`
- `SHA-512`

> **Note**
>
> 크기가 16KB 미만인 데이터는 HSM에서 해싱되고, 더 큰 데이터는 소프트웨어에서 로컬로 해싱됩니다.

지원되는 해시 기반 메시지 인증 코드(HMAC) 알고리즘

Java용 AWS CloudHSM 소프트웨어 라이브러리는 다음 HMAC 알고리즘을 지원합니다.

- `HmacSHA1`
- `HmacSHA224`
- `HmacSHA256`
- `HmacSHA384`
- `HmacSHA512`

지원되는 서명/확인 메커니즘

Java용 AWS CloudHSM 소프트웨어 라이브러리는 다음 유형의 서명 및 검증을 지원합니다.

RSA 서명 유형

- `NONEwithRSA`
- `SHA1withRSA`
- `SHA224withRSA`
- `SHA256withRSA`

- SHA384withRSA
- SHA512withRSA
- SHA1withRSA/PSS
- SHA224withRSA/PSS
- SHA256withRSA/PSS
- SHA384withRSA/PSS
- SHA512withRSA/PSS

ECDSA 서명 유형

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

메커니즘 주석

[1] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에서는 이 기능이 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단를 참조하세요.

Client SDK 3에 지원되는 Java 키 속성

이 주제에서는 Java 라이브러리 버전 3.1의 독점적 확장을 사용하여 키 속성을 설정하는 방법에 대해 설명합니다. 이 확장을 사용하여 다음 작업 중에 지원되는 키 속성 및 해당 값을 설정할 수 있습니다:

- 키 생성
- 키 가져오기
- 키 언래핑

> **Note**
>
> 사용자 지정 키 속성을 설정하기 위한 확장은 선택적 기능입니다. Java 라이브러리 버전 3.0에서 작동하는 코드가 이미 있으면 해당 코드를 수정할 필요가 없습니다. 사용자가 만든 키는 이전과 동일한 속성을 계속 포함합니다.

주제

속성 이해

키 속성을 사용하여 공개 키, 개인 키 또는 보안 키를 포함하여 키 객체에 허용되는 작업을 지정합니다. 키 객체 생성 작업 중에 키 속성과 값을 정의합니다.

그러나 자바 암호화 확장 (JCE)는 키 속성에 대한 값을 설정하는 방법을 지정하지 않으므로 대부분의 작업이 기본적으로 허용되었습니다. 이와 반대로 PKCS# 11 표준은 더 제한적인 기본값을 가진 포괄적인 속성 집합을 정의합니다. Java 라이브러리 버전 3.1부터 CloudHSM은 일반적으로 사용되는 속성에 대해 보다 제한적인 값을 설정할 수 있는 독점적 확장을 제공합니다.

지원되는 속성

아래 표에 나열된 속성에 대한 값을 설정할 수 있습니다. 제한적으로 만들려는 속성의 값만 설정하는 것이 좋습니다. 값을 지정하지 않으면 CloudHSM은 아래 표에 지정된 기본값을 사용합니다. 기본값 열의 빈 셀은 속성에 할당된 특정 기본값이 없음을 나타냅니다.

속성	기본 값			참고
	대칭 키	키 페어의 공개 키	키 페어의 개인 키	
CKA_TOKEN	FALSE	FALSE	FALSE	클러스터의 모든 HSM에 복제되고 백업에 포함된

속성	기본 값			참고
				영구 키입니다. CKA_TOKEN = FALSE는 HSM 하나에만 로드되고 HSM에 대한 연결이 끊어지면 자동으로 지워지는 세션 키를 의미합니다.
CKA_LABEL				사용자 정의 문자열입니다. 이를 통해 HSM에서 키를 편리하게 식별할 수 있습니다.
CKA_EXTRACTABLE	TRUE		TRUE	True는 이 키를 HSM에서 내보낼 수 있음을 나타냅니다.
CKA_ENCRYPT	TRUE	TRUE		True는 키를 사용하여 버퍼를 암호화할 수 있음을 나타냅니다.
CKA_DECRYPT	TRUE		TRUE	True는 키를 사용하여 버퍼의 암호를 해독할 수 있음을 나타냅니다. 일반적으로 CKA_WRAP이 true로 설정된 키의 경우 FALSE로 설정됩니다.

속성	기본 값			참고
CKA_WRAP	TRUE	TRUE		True는 키를 사용하여 다른 키를 래핑할 수 있음을 나타냅니다. 일반적으로 개인 키의 경우 FALSE로 설정됩니다.
CKA_UNWRAP	TRUE		TRUE	True는 키를 사용하여 다른 키를 언래핑(가져오기)할 수 있음을 나타냅니다.
CKA_SIGN	TRUE		TRUE	True는 키를 사용하여 메시지 다이제스트에 서명할 수 있음을 나타냅니다. 일반적으로 아카이브 완료된 개인 키와 공개 키의 경우 FALSE로 설정됩니다.
CKA_VERIFY	TRUE	TRUE		True는 키를 사용하여 서명을 확인할 수 있음을 나타냅니다. 일반적으로 개인 키의 경우 FALSE로 설정됩니다.

속성	기본 값			참고
CKA_PRIVATE	TRUE	TRUE	TRUE	True는 사용자가 인증될 때까지 사용자가 키에 액세스할 수 없음을 나타냅니다. 이해를 돕기 위해 설명하자면 사용자는 인증될 때까지 CloudHSM의 키에 액세스할 수 없습니다. 이 속성이 FALSE로 설정되는 경우에도 마찬가지입니다.

> **Note**
>
> PKCS #11 라이브러리의 속성에 대한 광범위한 지원을 받을 수 있습니다. 자세한 내용은 지원되는 PKCS #11 속성을 참조하십시오.

키에 대한 속성 설정

CloudHsmKeyAttributesMap은 Java Map과 유사한 객체로, 키 객체에 대한 속성 값을 설정하는 데 사용할 수 있습니다. CloudHsmKeyAttributesMap 함수에 대한 메서드는 Java Map 조작에 사용되는 메서드와 비슷합니다.

속성에 대한 사용자 지정 값을 설정하기 위해 다음 두 가지 옵션이 제공됩니다.

- 다음 표에 나열된 메서드 사용
- 이 문서 뒷부분에 설명된 빌더 패턴 사용

속성 맵 객체는 속성을 설정하기 위한 다음과 같은 메서드를 지원합니다:

Operation	반환 값	**CloudHSMKeyAttributesMap** 메서드
기존 키에 대한 키 속성 값 가져오기	객체(값 포함) 또는 null	get(keyAttribute)
키 속성 하나의 값 채우기	키 속성과 연결된 이전 값 또는 키 속성에 대한 매핑이 없는 경우에는 null	put(keyAttribute, 값)
여러 키 속성에 대한 값 채우기	N/A	PuTall () keyAttributesMap
속성 맵에서 키-값 페어 제거	키 속성과 연결된 이전 값 또는 키 속성에 대한 매핑이 없는 경우에는 null	remove(keyAttribute)

> **Note**
>
> 명시적으로 지정하지 않은 속성은 the section called “지원되는 속성”의 이전 표에 나열된 기본 값으로 설정됩니다.

빌더 패턴 예시

개발자는 일반적으로 빌더 패턴을 통해 더 편리하게 클래스를 활용합니다. 예를 들면 다음과 같습니다.

```
import com.amazonaws.cloudhsm.CloudHsmKeyAttributes;
import com.amazonaws.cloudhsm.CloudHsmKeyAttributesMap;
import com.amazonaws.cloudhsm.CloudHsmKeyPairAttributesMap;

CloudHsmKeyAttributesMap keyAttributesSessionDecryptionKey =
   new CloudHsmKeyAttributesMap.Builder()
      .put(CloudHsmKeyAttributes.CKA_LABEL, "ExtractableSessionKeyEncryptDecrypt")
      .put(CloudHsmKeyAttributes.CKA_WRAP, false)
      .put(CloudHsmKeyAttributes.CKA_UNWRAP, false)
      .put(CloudHsmKeyAttributes.CKA_SIGN, false)
      .put(CloudHsmKeyAttributes.CKA_VERIFY, false)
      .build();
```

```
CloudHsmKeyAttributesMap keyAttributesTokenWrappingKey =
   new CloudHsmKeyAttributesMap.Builder()
      .put(CloudHsmKeyAttributes.CKA_LABEL, "TokenWrappingKey")
      .put(CloudHsmKeyAttributes.CKA_TOKEN, true)
      .put(CloudHsmKeyAttributes.CKA_ENCRYPT, false)
      .put(CloudHsmKeyAttributes.CKA_DECRYPT, false)
      .put(CloudHsmKeyAttributes.CKA_SIGN, false)
      .put(CloudHsmKeyAttributes.CKA_VERIFY, false)
      .build();
```

개발자는 사전 정의된 속성 집합을 키 템플릿에서 모범 사례를 적용하는 편리한 방법으로 활용할 수도 있습니다. 예를 들면 다음과 같습니다.

```
//best practice template for wrapping keys

CloudHsmKeyAttributesMap commonKeyAttrs = new CloudHsmKeyAttributesMap.Builder()
    .put(CloudHsmKeyAttributes.CKA_EXTRACTABLE, false)
    .put(CloudHsmKeyAttributes.CKA_DECRYPT, false)
    .build();

// initialize a new instance of CloudHsmKeyAttributesMap by copying commonKeyAttrs
// but with an appropriate label

CloudHsmKeyAttributesMap firstKeyAttrs = new CloudHsmKeyAttributesMap(commonKeyAttrs);
firstKeyAttrs.put(CloudHsmKeyAttributes.CKA_LABEL, "key label");

// alternatively, putAll() will overwrite existing values to enforce conformance

CloudHsmKeyAttributesMap secondKeyAttrs = new CloudHsmKeyAttributesMap();
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_DECRYPT, true);
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_ENCRYPT, true);
secondKeyAttrs.put(CloudHsmKeyAttributes.CKA_LABEL, “safe wrapping key”);
secondKeyAttrs.putAll(commonKeyAttrs); // will overwrite CKA_DECRYPT to be FALSE
```

키 페어에 대한 속성 설정

Java 클래스 `CloudHsmKeyPairAttributesMap`을 사용하여 키 페어에 대한 키 속성을 처리합니다. `CloudHsmKeyPairAttributesMap`은 두 개의 `CloudHsmKeyAttributesMap` 객체를 캡슐화합니다. 하나는 퍼블릭 키용이고 다른 하나는 개인 키용입니다.

퍼블릭 키와 개인 키에 대해 별도로 개별 속성을 설정하려면 키의 해당하는 `CloudHsmKeyAttributes` 맵 객체에 `put()` 메서드를 사용하면 됩니다. `getPublic()` 메서드를 사

용하여 퍼블릭 키에 대한 속성 맵을 검색하고 `getPrivate()`을 사용하여 개인 키에 대한 속성 맵을 검색합니다. 키 페어 속성 맵에 `putAll()`을 인수로 사용하여 퍼블릭 및 개인 키 페어에 대한 여러 키 속성 값을 함께 채웁니다.

빌더 패턴 예시

개발자는 일반적으로 빌더 패턴을 통해 더 편리하게 키 속성을 설정합니다. 예:

```
import com.amazonaws.cloudhsm.CloudHsmKeyAttributes;
import com.amazonaws.cloudhsm.CloudHsmKeyAttributesMap;
import com.amazonaws.cloudhsm.CloudHsmKeyPairAttributesMap;

//specify attributes up-front
CloudHsmKeyAttributesMap keyAttributes =
    new CloudHsmKeyAttributesMap.Builder()
        .put(CloudHsmKeyAttributes.CKA_SIGN, false)
        .put(CloudHsmKeyAttributes.CKA_LABEL, "PublicCertSerial12345")
        .build();

CloudHsmKeyPairAttributesMap keyPairAttributes =
    new CloudHsmKeyPairAttributesMap.Builder()
        .withPublic(keyAttributes)
        .withPrivate(
            new CloudHsmKeyAttributesMap.Builder() //or specify them inline
                .put(CloudHsmKeyAttributes.CKA_LABEL, "PrivateCertSerial12345")
                .put (CloudHSMKeyAttributes.CKA_WRAP, FALSE)
                .build()
        )
        .build();
```

> **Note**
>
> 이 전용 확장에 대한 자세한 내용은 Javadoc 아카이브 및 샘플을 참조하십시오. GitHub Javadoc을 탐색하려면 아카이브를 다운로드하고 확장합니다.

모두 통합

키 작업을 사용하여 키 속성을 지정하려면 다음 단계를 수행합니다.

1. 대칭 키의 경우 `CloudHsmKeyAttributesMap`을, 키 페어의 경우 `CloudHsmKeyPairAttributesMap`을 인스턴스화합니다.

2. 필수 키 속성 및 값을 사용하여 1단계의 속성 객체를 정의합니다.
3. 특정 키 유형에 해당하는 Cavium*ParameterSpec 클래스를 인스턴스화하고 이 구성된 속성 객체를 생성자에게 전달합니다.
4. 이 Cavium*ParameterSpec 객체를 해당하는 암호화 클래스 또는 메서드에 전달합니다.

참고로 다음 표에는 사용자 지정 키 속성을 지원하는 Cavium*ParameterSpec 클래스 및 메서드가 포함되어 있습니다.

키 유형	파라미터 사양 클래스	예제 생성자
기본 클래스	CaviumKeyGenAlgorithmParameterSpec	CaviumKeyGenAlgorithmParameterSpec(CloudHsmKeyAttributesMap keyAttributesMap)
DES	CaviumDESKeyGenParameterSpec	CaviumDESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap keyAttributesMap)
RSA	CaviumRSAKeyGenParameterSpec	CaviumRSAKeyGenParameterSpec(int keysize, BigInteger publicExponent, CloudHsmKeyPairAttributesMap keyPairAttributesMap)
Secret	CaviumGenericSecretKeyGenParameterSpec	CaviumGenericSecretKeyGenParameterSpec(int size, CloudHsmKeyAttribu

키 유형	파라미터 사양 클래스	예제 생성자
		tesMap key AttributesMap)
AES	CaviumAESKeyGenParameterSpec	CaviumAESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap key AttributesMap)
EC	CaviumECGenParameterSpec	CaviumECGenParameterSpec(String stdName, CloudHsmKeyPairAttributesMap keyPairAttributesMap)

샘플 코드: 키 생성 및 래핑

다음 간단한 코드 샘플은 키 생성과 키 래핑의 두 가지 작업 단계를 보여줍니다.

```
// Set up the desired key attributes

KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
CaviumAESKeyGenParameterSpec keyAttributes = new CaviumAESKeyGenParameterSpec(
    256,
    new CloudHsmKeyAttributesMap.Builder()
        .put(CloudHsmKeyAttributes.CKA_LABEL, "MyPersistentAESKey")
        .put(CloudHsmKeyAttributes.CKA_EXTRACTABLE, true)
        .put(CloudHsmKeyAttributes.CKA_TOKEN, true)
        .build()
);

// Assume we already have a handle to the myWrappingKey
// Assume we already have the wrappedBytes to unwrap

// Unwrap a key using Custom Key Attributes
```

```
CaviumUnwrapParameterSpec unwrapSpec = new
 CaviumUnwrapParameterSpec(myInitializationVector, keyAttributes);

Cipher unwrapCipher = Cipher.getInstance("AESWrap", "Cavium");
unwrapCipher.init(Cipher.UNWRAP_MODE, myWrappingKey, unwrapSpec);
Key unwrappedKey = unwrapCipher.unwrap(wrappedBytes, "AES", Cipher.SECRET_KEY);
```

Java for Client SDK 3용 AWS CloudHSM 소프트웨어 라이브러리의 코드 샘플

사전 조건

샘플을 실행하기 전에 사용자 환경을 설정해야 합니다.

- Java 암호화 확장(JCE) 공급자와 AWS CloudHSM 클라이언트 패키지를 설치하고 구성합니다.
- 유효한 HSM 사용자 이름과 암호를 설정합니다. 이러한 작업을 수행하기 위해서는 CU(Cryptographic User) 권한이면 충분합니다. 각 예제에서 애플리케이션은 이러한 자격 증명을 사용하여 HSM에 로그인합니다.
- JCE 공급자에게 자격 증명을 제공하는 방법을 결정하십시오.

코드 샘플

다음 코드 샘플은 AWS CloudHSM JCE 공급자를 사용하여 기본 태스크를 수행하는 방법을 보여줍니다. 에서 더 많은 코드 샘플을 사용할 수 GitHub있습니다.

- HSM에 로그인
- 키 관리
- AES 키 생성
- AES GCM을 사용하여 암호화 및 해독
- AES-CTR을 사용하여 암호화 및 해독
- D3DES-ECB로 암호화 및 해독(1 참고 참조)
- AES-GCM을 사용하여 키 래핑 및 언래핑
- AES를 사용하여 키 래핑 및 언래핑
- RSA를 사용하여 키 래핑 및 언래핑
- 지원되는 키 속성 사용
- 키 스토어의 키 나열

- CloudHSM 키 스토어 사용
- 다중 스레드 샘플에서 메시지에 서명
- EC 키를 사용한 서명 및 확인

[1] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에는 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단를 참조하세요.

클라이언트 SDK 3에 AWS CloudHSM KeyStore 자바 클래스 사용

이 AWS CloudHSM **`KeyStore`** 클래스는 keytool 및 jarsigner와 같은 애플리케이션을 통해 키에 액세스할 수 있는 특수 목적의 PKCS12 AWS CloudHSM 키 저장소를 제공합니다. 이 키 스토어는 키 데이터와 함께 인증서를 저장하고 AWS CloudHSM에 저장된 키 데이터와 상호 연관시킬 수 있습니다.

> **Note**
>
> 인증서는 공개 정보이고 암호화 키의 저장 용량을 극대화하기 위해 HSM에 인증서를 저장하는 AWS CloudHSM 것을 지원하지 않습니다.

이 AWS CloudHSM `KeyStore` 클래스는 JCE (Java 암호화 확장) 의 SPI (`KeyStore`서비스 공급자 인터페이스) 를 구현합니다. 사용에 `KeyStore` 대한 자세한 내용은 클래스를 참조하십시오. KeyStore

적절한 키 스토어 선택

JCE (AWS CloudHSM Java 암호화 확장) 공급자는 모든 트랜잭션을 HSM으로 전달하는 기본 패스스루 읽기 전용 키 저장소를 제공합니다. 이 기본 키 스토어는 특수 용도와는 다릅니다. AWS CloudHSM KeyStore 대부분의 상황에서 기본값을 사용하여 더 나은 런타임 성능과 처리량을 얻을 수 있습니다. 키 작업을 HSM으로 AWS CloudHSM KeyStore 오프로드하는 것 외에도 인증서 및 인증서 기반 작업에 대한 지원이 필요한 애플리케이션에만 사용해야 합니다.

두 키 저장소 모두 작업을 위해 JCE 공급자를 사용하지만, 이 둘은 독립 개체이며 서로 정보를 교환하지 않습니다.

다음과 같이 Java 애플리케이션에 대한 기본 키 스토어를 로드합니다.

```
KeyStore ks = KeyStore.getInstance("Cavium");
```

다음과 같이 특수 목적의 CloudHSM을 로드합니다. KeyStore

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

초기화 중 AWS CloudHSM KeyStore

JCE AWS CloudHSM KeyStore 제공자에 로그인하는 것과 같은 방법으로 로그인합니다. 환경 변수 또는 시스템 속성 파일을 사용할 수 있으며, KeyStore CloudHSM을 사용하기 전에 로그인해야 합니다. JCE 공급자를 사용하여 HSM에 로그인하는 예는 HSM에 로그인을 참조하십시오.

원하는 경우, 암호를 지정하여 키 스토어 데이터를 보유하는 로컬 PKCS12 파일을 암호화할 수 있습니다. AWS CloudHSM 키 저장소를 생성할 때 암호를 설정하고 load, set 및 get 메서드를 사용할 때 암호를 입력합니다.

다음과 같이 새 CloudHSM 객체를 인스턴스화합니다. KeyStore

```
ks.load(null, null);
```

`store` 메서드를 사용하여 파일에 키 스토어 데이터를 씁니다. 이 시점부터 다음과 같이 소스 파일 및 암호와 함께 `load` 메서드를 사용하여 기존 키 스토어를 로드할 수 있습니다.

```
ks.load(inputStream, password);
```

사용 AWS CloudHSM KeyStore

KeyStore CloudHSM 객체는 일반적으로 jarsigner 또는 keytool과 같은 타사 애플리케이션을 통해 사용됩니다. 또한 코드를 사용하여 직접 객체에 액세스 할 수도 있습니다.

AWS CloudHSM KeyStore JCE 클래스 KeyStore 사양을 준수하며 다음 함수를 제공합니다.

- `load`

  지정된 입력 스트림에서 키 스토어를 로드합니다. 키 스토어를 저장할 때 암호가 설정된 경우, 로드가 성공하려면 이와 동일한 암호가 제공되어야 합니다. 새로운 빈 키 스토어를 초기화하려면 두 파라미터를 모두 null로 설정합니다.

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  ks.load(inputStream, password);
  ```

- `aliases`

지정된 키 스토어 인스턴스에 있는 모든 항목의 별칭 이름의 열거를 반환합니다. 결과는 PKCS12 파일에 로컬로 저장된 객체와 HSM에 있는 객체를 포함합니다.

샘플 코드:

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();)
 {
  String label = entry.nextElement();
  System.out.println(label);
 }
```

- `ContainsAlias`

  키 스토어가 지정된 별칭이 있는 객체에 최소 하나 이상 액세스할 수 있는 경우 true를 반환합니다. 키 스토어는 PKCS12 파일에 로컬로 저장된 객체와 HSM에 상주하는 객체를 확인합니다.

- `DeleteEntry`

  로컬 PKCS12 파일에서 인증서 항목을 삭제합니다. 를 사용하여 HSM에 저장된 주요 데이터를 삭제할 수 없습니다. AWS CloudHSM KeyStore CloudHSM의 key_mgmt_util 도구를 사용하여 키를 삭제할 수 있습니다.

- `GetCertificate`

  사용 가능한 경우 별칭과 연결된 인증서를 반환합니다. 별칭이 없거나 인증서가 아닌 객체를 참조하는 경우, 함수는 NULL을 반환합니다.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
Certificate cert = ks.getCertificate(alias)
```

- `GetCertificateAlias`

  데이터가 지정된 인증서와 일치하는 첫 번째 키 스토어 항목의 이름(별칭)을 반환합니다.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
String alias = ks.getCertificateAlias(cert)
```

- `GetCertificateChain`

  지정된 별칭과 연결된 인증서 체인을 반환합니다. 별칭이 없거나 인증서가 아닌 객체를 참조하는 경우, 함수는 NULL을 반환합니다.

- `GetCreationDate`

  지정된 별칭에 의해 식별된 항목의 생성 날짜를 반환합니다. 생성 날짜를 사용할 수 없는 경우 함수는 인증서가 유효해지는 날짜를 반환합니다.

- `GetKey`

  GetKey HSM으로 전달되고 지정된 레이블에 해당하는 키 객체를 반환합니다. HSM을 `getKey` 직접 쿼리하므로 HSM에서 키를 생성했는지 여부에 관계없이 HSM의 모든 키에 사용할 수 있습니다. KeyStore

  ```
  Key key = ks.getKey(keyLabel, null);
  ```

- `IsCertificateEntry`

  지정된 별칭이 있는 항목이 인증서 항목을 나타내는지 확인합니다.

- `IsKeyEntry`

  지정된 별칭이 있는 항목이 키 항목을 나타내는지 확인합니다. 이 작업은 PKCS12 파일과 HSM에서 모두 별칭을 검색합니다.

- `SetCertificateEntry`

  지정된 인증서를 지정된 별칭에 할당합니다. 지정된 별칭이 키 또는 인증서를 식별하는 데 이미 사용 중인 경우 `KeyStoreException`이 발생합니다. JCE 코드를 사용하여 키 개체를 가져온 다음 KeyStore `SetKeyEntry` 메서드를 사용하여 인증서를 키에 연결할 수 있습니다.

- `byte[]` 키가 있는 `SetKeyEntry`

  이 API는 현재 클라이언트 SDK 3에서 지원되지 않습니다.

- `Key` 객체가 있는 `SetKeyEntry`

  지정된 키를 지정된 별칭에 할당하고 HSM 내부에 저장합니다. `Key` 객체가 `CaviumKey` 유형이 아닌 경우, 키가 추출 가능한 세션 키로 HSM에 가져오기 됩니다.

  `Key` 객체가 `PrivateKey` 유형인 경우 해당 인증서 체인이 함께 제공되어야 합니다.

  별칭이 이미 존재하는 경우 `SetKeyEntry` 호출은 `KeyStoreException`을 발생시키고 키를 덮어쓰지 못하게 합니다. 키를 덮어써야 하는 경우 해당 목적으로 KMU 또는 JCE를 사용하십시오.

- `EngineSize`

  키 스토어의 항목 수를 반환합니다.

- `Store`

  키 스토어를 지정된 출력 스트림에 PKCS12 파일로 저장하고 지정된 암호로 보호합니다. 또한, 모든 로드된 키(`setKey` 호출을 사용하여 설정됨)를 유지합니다.

AWS CloudHSM과 타사 애플리케이션 통합

의 일부 사용 사례에는 타사 소프트웨어 애플리케이션을 클러스터의 HSM과 통합하는 것이 AWS CloudHSM AWS CloudHSM 포함됩니다. 타사 소프트웨어를 와 통합하면 다양한 AWS CloudHSM보안 관련 목표를 달성할 수 있습니다. 다음 주제에서는 이러한 목표 일부를 달성하는 방법에 대해 설명합니다.

주제

- SSL/TLS 오프로드를 통해 웹 서버 보안을 개선하세요. AWS CloudHSM
- AWS CloudHSM을 사용하여 Windows Server를 인증 기관(CA)으로 구성하기
- AWS CloudHSM을 사용하는 Oracle Database TDE(Transparent Data Encryption)
- SignTool Microsoft와 함께 AWS CloudHSM 사용하여 파일에 서명하세요.
- Java Keytool 및 Jarsigner
- 기타 타사 벤더 통합

SSL/TLS 오프로드를 통해 웹 서버 보안을 개선하세요. AWS CloudHSM

웹 서버와 해당 클라이언트(웹 브라우저)는 SSL(Secure Sockets Layer) 또는 TLS(전송 계층 보안) 프로토콜을 사용하여 웹 서버의 ID를 확인하고 인터넷을 통해 웹 페이지 또는 기타 데이터를 송수신하는 보안 연결을 설정할 수 있습니다. 이를 일반적으로 HTTPS라고 합니다. 웹 서버는 퍼블릭-프라이빗 키 페어 및 SSL/TLS 퍼블릭 키 인증서를 사용하여 각 클라이언트에 HTTPS 세션을 설정합니다. 이 프로세스에는 웹 서버의 계산 작업이 많이 필요하지만 이 중 일부를 AWS CloudHSM 클러스터로 오프로드할 수 있는데, 이를 SSL 가속이라고 합니다. 오프로드를 사용하면 웹 서버의 컴퓨팅 부담을 줄일 수 있으며 서버의 프라이빗 키를 HSM에 저장하여 보안을 강화할 수 있습니다.

다음 항목에서는 SSL/TLS 오프로드를 통한 AWS CloudHSM 작동 방식에 대한 개요와 다음 플랫폼에서 SSL/TLS 오프로드를 설정하기 위한 자습서를 제공합니다. AWS CloudHSM

Linux의 경우 NGINX 또는 Apache HTTP Server 웹 서버 소프트웨어에서 OpenSSL Dynamic Engine 사용

Windows의 경우 Windows Server용 IIS(인터넷 정보 서비스) 웹 서버 소프트웨어 사용

주제

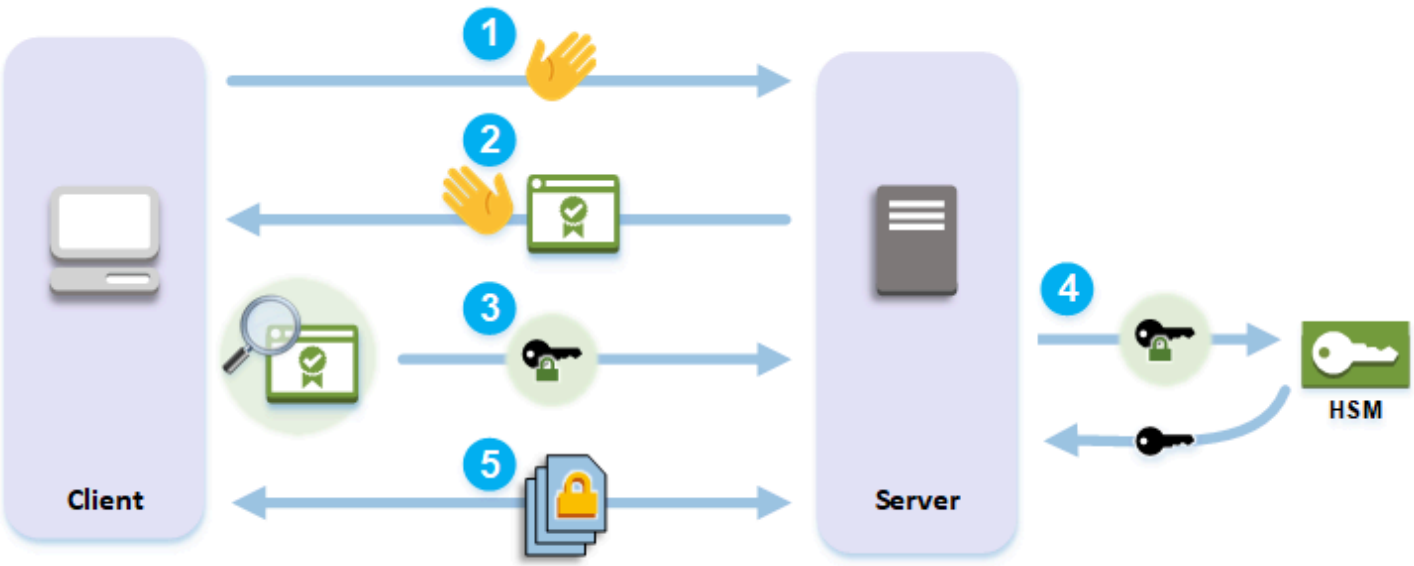
- SSL/TLS 오프로드 작동 방식 AWS CloudHSM

- Linux에서의 SSL/TLS 오프로드
- Windows에서 SSL/TLS 오프로드를 위해 CNG와 함께 IIS 사용
- Elastic Load Balancing을 사용하여 로드 밸런서 추가(선택 사항)

SSL/TLS 오프로드 작동 방식 AWS CloudHSM

HTTPS 연결을 설정하기 위해 웹 서버는 클라이언트와의 핸드셰이크 프로세스를 수행합니다. 이 프로세스의 일환으로 서버는 다음 그림에 나온 것처럼 암호화 처리 일부를 HSM에 오프로딩합니다. 프로세스의 각 단계는 그림 아래 설명되어 있습니다.

> **Note**
>
> 다음 이미지와 프로세스는 서버 확인 및 키 교환에 RSA가 사용된다고 가정합니다. RSA 대신 Diffie-Hellman을 사용하는 경우 프로세스가 약간 다릅니다.



1. 클라이언트가 서버에 hello 메시지를 전송합니다.
2. 서버는 hello 메시지로 응답하고 서버의 인증서를 전송합니다.
3. 클라이언트는 다음 작업을 수행합니다.
   a. 클라이언트가 신뢰하는 루트 인증서로 SSL/TLS 서버의 인증서가 서명되어 있는지 확인합니다.
   b. 서버 인증서에서 퍼블릭 키를 추출합니다.
   c. premaster secret을 생성하여 서버의 퍼블릭 키로 암호화합니다.

d. 암호화된 premaster secret을 서버로 전송합니다.

4. 클라이언트의 premaster secret 암호를 해독하기 위하여 서버가 이를 HSM에 전송합니다. HSM은 HSM에 있는 프라이빗 키를 사용하여 premaster secret을 암호화 해제한 다음 premaster secret을 서버로 전송합니다. 클라이언트와 서버는 각각 독립적으로 premaster secret과 hello 메시지의 일부 정보를 사용하여 master secret을 계산합니다.
5. 핸드셰이크 프로세스가 종료됩니다. 세션의 나머지에서는 클라이언트와 서버 사이에서 전송되는 모든 메시지가 master secret의 파생문으로 암호화됩니다.

를 사용하여 SSL/TLS 오프로드를 구성하는 방법을 알아보려면 다음 주제 중 하나를 참조하십시오. AWS CloudHSM

- Linux에서의 SSL/TLS 오프로드
- Windows에서 SSL/TLS 오프로드를 위해 CNG와 함께 IIS 사용

Linux에서의 SSL/TLS 오프로드

를 사용하면 AWS CloudHSM NGINX, Apache 및 Tomcat을 사용하여 Linux에서 SSL/TLS 오프로드를 수행할 수 있습니다. 자세한 내용은 아래의 관련 주제를 참조하십시오.

주제

- Linux에서 SSL/TLS 오프로드를 위해 OpenSSL과 함께 NGINX 또는 Apache를 사용
- 리눅스에서 SSL/TLS 오프로드를 위한 JSSE 포함 Tomcat 사용

Linux에서 SSL/TLS 오프로드를 위해 OpenSSL과 함께 NGINX 또는 Apache를 사용

이 항목에서는 Linux 웹 서버에서 SSL/TLS 오프로드를 설정하는 step-by-step 방법에 대한 지침을 제공합니다. AWS CloudHSM

주제

- 개요
- 1단계: 사전 조건 설정
- 단계 2: 프라이빗 키 및 SSL/TLS 인증서 생성 또는 가져오기
- 3단계: 웹 서버 구성하기
- 4단계: HTTPS 트래픽 활성화 및 인증서 확인하기

개요

Linux에서는 HTTPS를 지원하기 위해 NGINX 및 Apache HTTP Server 웹 서버 소프트웨어가 OpenSSL에 통합되어 있습니다. OpenSSL용AWS CloudHSM Dynamic Engine은 웹 서버 소프트웨어가 암호화 오프로딩 및 키 스토리지를 위해 클러스터에서 HSM을 사용할 수 있도록 인터페이스를 제공합니다. OpenSSL 엔진은 웹 서버를 AWS CloudHSM 클러스터에 연결하는 브리지입니다.

이 자습서를 완료하려면 Linux에서 NGINX 웹 서버 소프트웨어를 사용할지, Apache 웹 서버 소프트웨어를 사용할지 먼저 선택해야 합니다. 그런 다음 이 자습서에서는 다음을 수행하는 방법을 보여줍니다.

- Amazon EC2 인스턴스에 웹 서버 소프트웨어를 설치합니다.
- AWS CloudHSM 클러스터에 프라이빗 키가 저장되는 HTTPS를 지원하도록 웹 서버 소프트웨어를 구성합니다.
- (선택 사항) Amazon EC2를 사용하여 두 번째 웹 서버 인스턴스를 만들고 Elastic Load Balancing을 사용하여 로드 밸런서를 만듭니다. 로드 밸런서를 사용하면 여러 서버에 부하를 분산하여 성능을 향상할 수 있습니다. 또한 하나 이상의 서버에 장애가 발생할 경우 중복성과 더 높은 가용성을 제공할 수 있습니다.

시작할 준비가 되면 1단계: 사전 조건 설정로 이동합니다.

1단계: 사전 조건 설정

플랫폼마다 필요한 필수 조건이 다릅니다. 플랫폼에 맞는 아래의 사전 조건 섹션을 사용하세요.

주제

- 클라이언트 SDK 사전 조건 5
- Client SDK 3의 필요 조건

클라이언트 SDK 사전 조건 5

클라이언트 SDK 5으로 웹 서버 SSL/TLS 오프로드를 설정하려면 다음이 필요합니다.

- 두 개 이상의 하드웨어 보안 모듈 (AWS CloudHSM HSM) 이 있는 활성 클러스터

> **Note**
>
> 단일 HSM 클러스터를 사용할 수 있지만 먼저 클라이언트 키 내구성을 비활성화해야 합니다. 자세한 내용은 클라이언트 키 내구성 설정 관리 및 클라이언트 SDK 5 구성 도구를 참조하세요.

- 다음 소프트웨어가 설치된 Linux 운영 체제를 실행하는 Amazon EC2 인스턴스.
  - 웹 서버(NGINX 또는 아파치)
  - 클라이언트 SDK 5용 OpenSSL Dynamic Engine
- HSM에서 웹 서버의 프라이빗 키를 소유하고 관리할 CU(Crypto User)입니다.

HSM에서 Linux 웹 서버 인스턴스를 설정하고 CU를 생성하려면

1. 에 대한 OpenSSL 동적 엔진을 설치하고 구성합니다. AWS CloudHSM OpenSSL 동적 엔진 설치에 대한 자세한 내용은 클라이언트용 OpenSSL 동적 엔진 SDK 5를 참조하세요.
2. 클러스터에 액세스할 수 있는 EC2 Linux 인스턴스에서 NGINX 또는 Apache 웹 서버를 설치하세요.

   Amazon Linux

   - NGINX

     ```
     $ sudo yum install nginx
     ```

   - Apache

     ```
     $ sudo yum install httpd24 mod24_ssl
     ```

   Amazon Linux 2

   - Amazon Linux 2에서 최신 버전의 NGINX를 다운로드하는 방법에 대한 자세한 내용은 NGINX 웹 사이트를 참조하세요.

     Amazon Linux 2에서 사용할 수 있는 NGINX의 최신 버전은 OpenSSL의 시스템 버전보다 최신 버전의 OpenSSL을 사용합니다. NGINX를 설치한 후에는 OpenSSL 동적 엔진 라이브러

리에서 이 버전의 AWS CloudHSM OpenSSL이 예상하는 위치로 연결되는 심볼릭 링크를 생성해야 합니다.

```
$ sudo ln -sf /opt/cloudhsm/lib/libcloudhsm_openssl_engine.so /usr/lib64/
engines-1.1/cloudhsm.so
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

CentOS 7

- CentOS 7에서 최신 버전의 NGINX를 다운로드하는 방법에 대한 자세한 내용은 NGINX 웹 사이트를 참조하세요.

  CentOS 7에서 사용할 수 있는 NGINX의 최신 버전은 OpenSSL의 시스템 버전보다 최신 버전의 OpenSSL을 사용합니다. NGINX를 설치한 후에는 OpenSSL 동적 엔진 라이브러리에서 이 버전의 AWS CloudHSM OpenSSL이 예상하는 위치로 연결되는 심볼릭 링크를 생성해야 합니다.

```
$ sudo ln -sf /opt/cloudhsm/lib/libcloudhsm_openssl_engine.so /usr/lib64/
engines-1.1/cloudhsm.so
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Red Hat 7

- Red Hat 7에서 최신 버전의 NGINX를 다운로드하는 방법에 대한 자세한 내용은 NGINX 웹 사이트를 참조하세요.

  Red Hat 7에서 사용할 수 있는 NGINX의 최신 버전은 OpenSSL의 시스템 버전보다 최신 버전의 OpenSSL을 사용합니다. NGINX를 설치한 후에는 OpenSSL 동적 엔진 라이브러리에서 이 버전의 AWS CloudHSM OpenSSL이 예상하는 위치로 연결되는 심볼릭 링크를 생성해야 합니다.

```
$ sudo ln -sf /opt/cloudhsm/lib/libcloudhsm_openssl_engine.so /usr/lib64/
engines-1.1/cloudhsm.so
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

CentOS 8

- NGINX

```
$ sudo yum install nginx
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Red Hat 8

- NGINX

```
$ sudo yum install nginx
```

- Apache

```
$ sudo yum install httpd mod_ssl
```

Ubuntu 18.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 20.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 22.04

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

3. CloudHSM CLI를 사용하여 CU를 생성합니다. HSM 사용자 관리에 대한 자세한 내용은 CloudHSM CLI를 사용한 HSM 사용자 관리를 참조하십시오.

> **Tip**
>
> CU의 사용자 이름과 암호를 기록합니다. 나중에 웹 서버용 HTTPS 프라이빗 키와 인증서를 생성하거나 가져올 때 이 정보가 필요합니다.

이 단계들을 완료한 후 단계 2: 프라이빗 키 및 SSL/TLS 인증서 생성 또는 가져오기로 이동합니다.

참고

- 보안이 강화된 리눅스(SELinux) 및 웹 서버를 사용하려면 클라이언트 SDK 5가 HSM과 통신하는 데 사용하는 포트인 포트 2223에서 아웃바운드 TCP 연결을 허용해야 합니다.
- 클러스터를 생성 및 활성화하고 EC2 인스턴스에 클러스터 액세스 권한을 부여하려면 AWS CloudHSM시작하기의 단계를 완료하십시오. 시작하기에서는 HSM 1개와 Amazon EC2 클라이언트 인스턴스 1개로 활성 클러스터를 생성하는 step-by-step 방법에 대한 지침을 제공합니다. 이 클라이언트 인스턴스를 웹 서버로 사용할 수 있습니다.
- 클라이언트 키 내구성을 비활성화하지 않으려면 클러스터에 HSM을 두 개 이상 추가하십시오. 자세한 내용은 HSM 추가 섹션을 참조하십시오.

- SSH 또는 PuTTY를 사용하여 클라이언트 인스턴스에 연결할 수 있습니다. 자세한 정보는 Amazon EC2 설명서의 SSH를 사용하여 Linux 인스턴스에 연결과 PuTTY를 사용하여 Windows에서 Linux 인스턴스에 연결 단원을 참조하세요.

Client SDK 3의 필요 조건

클라이언트 SDK 3으로 웹 서버 SSL/TLS 오프로드를 설정하려면 다음이 필요합니다.

- HSM이 하나 이상 있는 액티브 AWS CloudHSM 클러스터.
- 다음 소프트웨어가 설치된 Linux 운영 체제를 실행하는 Amazon EC2 인스턴스.
  - AWS CloudHSM 클라이언트 및 명령줄 도구.
  - NGINX 또는 Apache 웹 서버 애플리케이션.
  - OpenSSL을 위한 AWS CloudHSM 다이나믹 엔진.
- HSM에서 웹 서버의 프라이빗 키를 소유하고 관리할 CU(Crypto User)입니다.

HSM에서 Linux 웹 서버 인스턴스를 설정하고 CU를 생성하려면

1. 시작하기의 단계를 수행하세요. 그러면 하나의 HSM과 Amazon EC2 클라이언트 인스턴스가 있는 활성 클러스터가 생깁니다. EC2 인스턴스는 명령줄 도구를 사용하여 구성됩니다. 이 클라이언트 인스턴스를 웹 서버로 사용합니다.
2. 클라이언트 인스턴스에 연결합니다. 자세한 정보는 Amazon EC2 설명서의 SSH를 사용하여 Linux 인스턴스에 연결과 PuTTY를 사용하여 Windows에서 Linux 인스턴스에 연결 단원을 참조하세요.
3. 클러스터에 액세스할 수 있는 EC2 Linux 인스턴스에서 NGINX 또는 Apache 웹 서버를 설치하세요.

   Amazon Linux

   - NGINX

     ```
     $ sudo yum install nginx
     ```

   - Apache

     ```
     $ sudo yum install httpd24 mod24_ssl
     ```

Amazon Linux 2

- NGINX 버전 1.19는 Amazon Linux 2의 클라이언트 SDK 3 엔진과 호환되는 NGINX의 최신 버전입니다.

  자세한 내용을 확인하고 NGINX 버전 1.19를 다운로드하려면 NGINX 웹 사이트를 참조하세요.

- Apache

```
$ sudo yum install httpd mod_ssl
```

CentOS 7

- NGINX 버전 1.19는 CentOS 7의 클라이언트 SDK 3 엔진과 호환되는 NGINX의 최신 버전입니다.

  자세한 내용을 확인하고 NGINX 버전 1.19를 다운로드하려면 NGINX 웹 사이트를 참조하세요.

- Apache

```
$ sudo yum install httpd mod_ssl
```

Red Hat 7

- NGINX 버전 1.19는 Red Hat 7의 클라이언트 SDK 3 엔진과 호환되는 NGINX의 최신 버전입니다.

  자세한 내용을 확인하고 NGINX 버전 1.19를 다운로드하려면 NGINX 웹 사이트를 참조하세요.

- Apache

```
$ sudo yum install httpd mod_ssl
```

Ubuntu 16.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

Ubuntu 18.04

- NGINX

```
$ sudo apt install nginx
```

- Apache

```
$ sudo apt install apache2
```

4. (선택 사항) 클러스터에 HSM을 더 추가합니다. 자세한 정보는 HSM 추가을 참조하세요.
5. cloudhsm_mgmt_util을 사용하여 CU를 생성합니다. 자세한 정보는 HSM 사용자 관리을 참조하세요. CU의 사용자 이름과 암호를 기록합니다. 나중에 웹 서버용 HTTPS 프라이빗 키와 인증서를 생성하거나 가져올 때 이 정보가 필요합니다.

이 단계들을 완료한 후 단계 2: 프라이빗 키 및 SSL/TLS 인증서 생성 또는 가져오기로 이동합니다.

단계 2: 프라이빗 키 및 SSL/TLS 인증서 생성 또는 가져오기

HTTPS를 활성화하려면 웹 서버 애플리케이션(NGINX 또는 Apache)에 프라이빗 키와 해당 SSL/TLS 인증서가 필요합니다. 웹 서버 SSL/TLS 오프로드를 함께 사용하려면 클러스터의 HSM에 AWS CloudHSM개인 키를 저장해야 합니다. AWS CloudHSM 이 작업은 다음 중 한 가지 방법으로 수행할 수 있습니다.

- 아직 프라이빗 키와 해당 인증서가 없다면 HSM에서 프라이빗 키를 생성하세요. 프라이빗 키를 사용하여 SSL/TLS 인증서를 생성하는 데 사용하는 인증서 사인 요청(CSR)을 생성합니다.
- 프라이빗 키와 해당 인증서가 이미 있는 경우 프라이빗 키를 HSM으로 가져옵니다.

이전 방법 중 어떤 방법을 선택하든 관계없이 HSM에서 가짜 PEM 프라이빗 키를 내보냅니다. 즉 HSM에 저장된 프라이빗 키에 대한 참조를 포함하는 PEM 형식의 프라이빗 키 파일입니다(실제 프라이빗 키가 아님). 웹 서버는 SSL/TLS 오프로드 중에 가짜 PEM 프라이빗 키 파일을 사용하여 HSM의 프라이빗 키를 식별합니다.

다음 중 하나를 수행하십시오.

- 프라이빗 키 및 인증서 생성
- 기존 프라이빗 키 및 인증서 가져오기

프라이빗 키 및 인증서 생성

프라이빗 키 생성

이 섹션에서는 Client SDK 3의 키 관리 유틸리티(KMU)를 사용하여 키 페어를 생성하는 방법을 보여줍니다. HSM 내부에 키 페어가 생성되면 이를 가짜 PEM 파일로 내보내고 해당 인증서를 생성할 수 있습니다.

키 관리 유틸리티(KMU)로 생성된 프라이빗 키는 Client SDK 3 및 Client SDK 5 모두에서 사용할 수 있습니다.

키 관리 유틸리티(KMU) 설치 및 구성

1. 클라이언트 인스턴스에 연결합니다.
2. Client SDK 3 설치 및 구성.
3. 다음 명령을 실행하여 클라이언트를 시작합니다. AWS CloudHSM

   Amazon Linux

   ```
   $ sudo start cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client start
   ```

   CentOS 7

   ```
   $ sudo service cloudhsm-client start
   ```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 22.04 LTS

```
$ sudo service cloudhsm-client start
```

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

4. 다음 명령을 실행하여 key_mgmt_util 명령줄 도구를 시작합니다.

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. 다음 명령을 실행하여 HSM에 로그인합니다. *<user name>* 및 *<password>*를 CU(Cryptographic User)의 사용자 이름과 암호로 바꿉니다.

```
Command: loginHSM -u CU -s <user name> -p <password>>
```

프라이빗 키 생성

사용 사례에 따라 RSA 또는 EC 키 페어를 생성할 수 있습니다. 다음 중 하나를 수행하십시오.

- HSM에서 RSA 프라이빗 키를 생성하려면

  genRSAKeyPair 명령을 사용하여 RSA 키 페어를 생성합니다. 이 예제에서는 모듈러스가 2048이고 퍼블릭 지수가 65537이고 레이블이 *tls_rsa_keypair*인 RSA 키 페어를 생성합니다.

  ```
  Command: genRSAKeyPair -m 2048 -e 65537 -l tls_rsa_keypair
  ```

  명령이 성공하면 RSA 키 페어가 성공적으로 생성되었음을 나타내는 다음 출력이 표시됩니다.

  ```
  Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

                  Cfm3GenerateKeyPair:    public key handle: 7    private key handle: 8

                  Cluster Status:
                  Node id 1 status: 0x00000000 : HSM Return: SUCCESS
  ```

- HSM에서 EC 프라이빗 키를 생성하려면

  genECCKeyPair 명령을 사용하여 EC 키 쌍을 생성합니다. 이 예제에서는 곡선 ID가 2(NID_X9_62_prime256v1 곡선에 해당)이고 레이블이 *tls_ec_keypair*인 EC 키 페어를 생성합니다.

  ```
  Command: genECCKeyPair -i 2 -l tls_ec_keypair
  ```

  명령이 성공하면 EC 키 페어가 성공적으로 생성되었음을 나타내는 다음 출력이 표시됩니다.

  ```
  Cfm3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

                  Cfm3GenerateKeyPair:    public key handle: 7    private key handle: 8

                  Cluster Status:
                  Node id 1 status: 0x00000000 : HSM Return: SUCCESS
  ```

가짜 PEM 프라이빗 키 파일 내보내기

HSM에 프라이빗 키가 있으면 가짜 PEM 프라이빗 키 파일을 내보내야 합니다. 이 파일에은 실제 키 데이터가 포함하지 않지만 OpenSSL 동적 엔진이 HSM의 프라이빗 키를 식별할 수 있도록 해줍니다. 그

런 다음 프라이빗 키를 사용하여 CSR(인증서 서명 요청)을 생성하고 CSR에 서명하여 인증서를 생성할 수 있습니다.

> **Note**
>
> 키 관리 유틸리티(KMU)로 생성된 가짜 PEM 파일은 Client SDK 3 및 Client SDK 5 모두에서 사용할 수 있습니다.

가짜 PEM으로 내보내려는 키에 해당하는 키 핸들을 식별한 후 다음 명령을 실행하여 프라이빗 키를 가짜 PEM 형식으로 내보내고 파일에 저장합니다. 다음 값을 사용자의 값으로 대체합니다.

- *<private_key_handle>* – 생성된 프라이빗 키 처리 이 핸들은 이전 단계의 키 생성 명령 중 하나에 의해 생성되었습니다. 이전 예제에서 프라이빗 키의 핸들은 8입니다.
- *<web_server_fake_PEM.key>* – 가짜 PEM 키가 기록될 파일의 이름입니다.

```
Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
```

Exit

다음 명령을 실행하여 key_mgmt_util을 중지합니다.

```
Command: exit
```

이제 이전 명령에서 *<web_server_fake_PEM.key>*로 지정된 경로에 시스템에 새 파일이 있어야 합니다. 이 파일은 가짜 PEM 프라이빗 키 파일입니다.

자체 사인된 인증서를 생성합니다.

가짜 PEM 프라이빗 키를 생성한 후에는 이 파일을 사용하여 인증서 사인 요청(CSR) 및 인증서를 생성할 수 있습니다.

프로덕션 환경에서는 일반적으로 CA(인증 기관)을 사용하여 CSR에서 인증서를 생성합니다. 테스트 환경에는 CA가 필요하지 않습니다. CA를 사용하는 경우 CA에 CSR 파일을 보내고 CA가 웹 서버에서 HTTPS용으로 제공하는 서명된 SSL/TLS 인증서를 사용하세요.

CA를 사용하는 대신 AWS CloudHSM OpenSSL 동적 엔진을 사용하여 자체 서명된 인증서를 생성할 수 있습니다. 자체 사인된 인증서는 브라우저에서 신뢰하지 않으며 프로덕션 환경에서 사용해서는 안 됩니다. 테스트 환경에서는 이러한 인증서를 사용할 수 있습니다.

> **Warning**
>
> 자체 사인된 인증서는 테스트 환경에서만 사용해야 합니다. 프로덕션 환경의 경우 인증 기관과 같은 추가 보안 방법을 사용하여 인증서를 생성하십시오.

OpenSSL Dynamic Engine 설치 및 구성

1. 클라이언트 인스턴스에 연결합니다.
2. 설치 및 구성하려면 다음 중 하나를 수행하십시오.
   - the section called “OpenSSL Dynamic Engine 설치”
   - the section called “OpenSSL Dynamic Engine”

인증서 생성

1. 이전 단계에서 생성한 가짜 PEM 파일의 사본을 확보하십시오.
2. CSR 생성

   다음 명령을 실행하여 AWS CloudHSM OpenSSL 동적 엔진을 사용하여 인증서 서명 요청 (CSR)을 생성합니다. *<web_server_fake_PEM.key>*를 가짜 PEM 프라이빗 키가 포함된 파일 이름으로 바꿉니다. *<web_server.csr>*을 CSR이 포함된 파일의 이름으로 바꿉니다.

   `req` 명령은 대화식입니다. 각 필드에 응답합니다. 필드 정보가 SSL/TLS 인증서에 복사됩니다.

   ```
   $ openssl req -engine cloudhsm -new -key <web_server_fake_PEM.key> -
   out <web_server.csr>
   ```

3. 자체 서명된 인증서 생성

   다음 명령을 실행하여 AWS CloudHSM OpenSSL 동적 엔진을 사용하여 HSM의 개인 키로 CSR에 서명합니다. 이렇게 하면 자체 사인된 인증서가 생성됩니다. 명령의 다음 값을 사용자의 값으로 바꿉니다.

   - *<web_server.csr>* – CSR이 포함된 파일 이름
   - *<web_server_fake_PEM.key>* – 가짜 PEM 프라이빗 키가 포함된 파일 이름
   - *<web_server.crt>* – 웹 서버 인증서가 포함될 파일 이름

```
$ openssl x509 -engine cloudhsm -req -days 365 -in <web_server.csr> -
signkey <web_server_fake_PEM.key> -out <web_server.crt>
```

이 단계들을 완료한 후 3단계: 웹 서버 구성하기로 이동합니다.

기존 프라이빗 키 및 인증서 가져오기

웹 서버에서 HTTPS에 사용할 프라이빗 키와 해당 SSL/TLS 인증서가 이미 있을 수도 있습니다. 그런 경우, 이 섹션의 단계에 따라 해당 키를 HSM으로 가져올 수 있습니다.

> **Note**
>
> 프라이빗 키 가져오기 및 Client SDK 호환성에 대한 몇 가지 참고 사항:
>
> - 기존 프라이빗 키를 가져오려면 Client SDK 3이 필요합니다.
> - Client SDK 5와 함께 Client SDK 3의 프라이빗 키를 사용할 수 있습니다.
> - Client SDK 3용 OpenSSL 동적 엔진은 최신 Linux 플랫폼을 지원하지 않지만 Client SDK 5용 OpenSSL 동적 엔진 구현은 지원합니다. Client SDK 3과 함께 제공되는 KMU(키 관리 유틸리티)를 사용하여 기존 프라이빗 키를 가져온 다음 해당 프라이빗 키를 사용하고 Client SDK 5로 OpenSSL Dynamic Engine을 구현하여 최신 Linux 플랫폼에서 SSL/TLS 오프로드를 지원할 수 있습니다. .

Client SDK 3을 사용하여 기존 프라이빗 키를 HSM으로 가져오려면

1. Amazon EC2 클라이언트 인스턴스에 연결합니다. 필요한 경우, 기존 프라이빗 키와 인증서를 인스턴스로 복사합니다.
2. Client SDK 3 설치 및 구성
3. 다음 명령을 실행하여 클라이언트를 시작합니다. AWS CloudHSM

   Amazon Linux

   ```
   $ sudo start cloudhsm-client
   ```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 20.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

4. 다음 명령을 실행하여 key_mgmt_util 명령줄 도구를 시작합니다.

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. 다음 명령을 실행하여 HSM에 로그인합니다. *<user name>* 및 *<password>*를 CU(Cryptographic User)의 사용자 이름과 암호로 바꿉니다.

```
Command: loginHSM -u CU -s <user name> -p <password>
```

6. 다음 명령을 실행하여 프라이빗 키를 HSM으로 가져옵니다.

   a. 다음 명령을 실행하여 현재 세션에만 유효한 대칭 래핑 키를 생성합니다. 명령과 출력은 다음과 같습니다.

```
Command: genSymKey -t 31 -s 16 -sess -l wrapping_key_for_import

Cfm3GenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS
Symmetric Key Created.  Key Handle: 6
Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

   b. 다음 명령을 실행하여 기존 프라이빗 키를 HSM으로 가져옵니다. 명령과 출력은 다음과 같습니다. 다음 값을 사용자의 값으로 대체합니다.

   - *<web_server_existing.key>* – 프라이빗 키가 포함된 파일 이름
   - *<web_server_imported_key>* – 가져온 프라이빗 키의 레이블
   - *<wrapping_key_handle>* – 이전 명령에서 생성된 래핑 키 처리 이전 예제에서 래핑 키 핸들은 6입니다.

```
Command: importPrivateKey -f <web_server_existing.key> -
l <web_server_imported_key> -w <wrapping_key_handle>

BER encoded key length is 1219
Cfm3WrapHostKey returned: 0x00 : HSM Return: SUCCESS
Cfm3CreateUnwrapTemplate returned: 0x00 : HSM Return: SUCCESS
Cfm3UnWrapKey returned: 0x00 : HSM Return: SUCCESS
Private Key Unwrapped.  Key Handle: 8
Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
```

7. 다음 명령을 실행하여 가짜 PEM 형식의 프라이빗 키를 내보내고 파일에 저장합니다. 다음 값을 사용자의 값으로 대체합니다.

   - *<private_key_handle>* – 가져온 프라이빗 키의 처리. 이 핸들은 이전 단계에서 두 번째 명령으로 생성되었습니다. 이전 예제에서 프라이빗 키의 핸들은 8입니다.
   - *<web_server_fake_PEM.key>* – 내보낸 가짜 PEM 프라이빗 키가 포함된 파일의 이름.

   ```
   Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
   ```

8. 다음 명령을 실행하여 key_mgmt_util을 중지합니다.

   ```
   Command: exit
   ```

이 단계들을 완료한 후 3단계: 웹 서버 구성하기로 이동합니다.

3단계: 웹 서버 구성하기

이전 단계에서 생성한 HTTPS 인증서와 이에 해당되는 가짜 PEM 프라이빗 키를 사용하려면 웹 서버 소프트웨어의 구성을 업데이트하십시오. 시작하기 전에 기존 인증서와 키를 백업해야 한다는 점을 잊지 마십시오. 그러면 AWS CloudHSM이 지원되는 SSL/TLS 오프로드의 Linux 웹 서버 소프트웨어 설정이 완료됩니다.

다음 섹션 중 하나에 있는 단계를 완료합니다.

주제

- NGINX 웹 서버를 구성합니다
- Apache 웹 서버 구성하기

NGINX 웹 서버를 구성합니다

이 섹션을 사용하여 지원되는 플랫폼에서 NGINX를 구성합니다.

NGINX용 웹 서버 구성을 업데이트하려면

1. 클라이언트 인스턴스에 연결합니다.
2. 다음 명령을 실행하여 웹 서버 인증서 및 가짜 PEM 프라이빗 키에 필요한 디렉터리를 생성합니다.

```
$ sudo mkdir -p /etc/pki/nginx/private
```

3. 다음 명령을 실행하여 웹 서버 인증서를 필요한 위치에 복사합니다. *<web_server.crt>*를 웹 서버 인증서 이름으로 바꿉니다.

```
$ sudo cp <web_server.crt> /etc/pki/nginx/server.crt
```

4. 다음 명령을 실행하여 가짜 PEM 프라이빗 키를 필요한 위치에 복사합니다. *<web_server_fake_PEM.key>*를 가짜 PEM 프라이빗 키가 포함된 파일 이름으로 바꿉니다.

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/nginx/private/server.key
```

5. 다음 명령을 실행하여 nginx라는 이름의 사용자가 파일을 읽을 수 있도록 파일 소유권을 변경합니다.

```
$ sudo chown nginx /etc/pki/nginx/server.crt /etc/pki/nginx/private/server.key
```

6. 다음 명령을 실행하여 `/etc/nginx/nginx.conf` 파일을 백업합니다.

```
$ sudo cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

7. NGINX 구성을 업데이트합니다.

> **Note**
>
> 각 클러스터는 모든 NGINX 웹 서버에서 최대 1000개의 NGINX 작업자 프로세스를 지원할 수 있습니다.

Amazon Linux

텍스트 편집기를 사용하여 `/etc/nginx/nginx.conf` 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다. 파일의 맨 위에 다음 행을 추가합니다.

- 클라이언트 SDK 3을 사용하는 경우

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 클라이언트 SDK 5를 사용하는 경우

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

그런 다음 파일의 TLS 섹션에 다음을 추가합니다.

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
```

```
}
```

Amazon Linux 2

텍스트 편집기를 사용하여 /etc/nginx/nginx.conf 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다. 파일의 맨 위에 다음 행을 추가합니다.

- 클라이언트 SDK 3을 사용하는 경우

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 클라이언트 SDK 5를 사용하는 경우

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

그런 다음 파일의 TLS 섹션에 다음을 추가합니다.

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
```

```
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

CentOS 7

텍스트 편집기를 사용하여 /etc/nginx/nginx.conf 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다. 파일의 맨 위에 다음 행을 추가합니다.

- 클라이언트 SDK 3을 사용하는 경우

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 클라이언트 SDK 5를 사용하는 경우

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

그런 다음 파일의 TLS 섹션에 다음을 추가합니다.

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;
```

```
    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

CentOS 8

텍스트 편집기를 사용하여 `/etc/nginx/nginx.conf` 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다. 파일의 맨 위에 다음 행을 추가합니다.

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

그런 다음 파일의 TLS 섹션에 다음을 추가합니다.

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Red Hat 7

텍스트 편집기를 사용하여 /etc/nginx/nginx.conf 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다. 파일의 맨 위에 다음 행을 추가합니다.

- 클라이언트 SDK 3을 사용하는 경우

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 클라이언트 SDK 5를 사용하는 경우

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

그런 다음 파일의 TLS 섹션에 다음을 추가합니다.

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
```

```
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Red Hat 8

텍스트 편집기를 사용하여 /etc/nginx/nginx.conf 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다. 파일의 맨 위에 다음 행을 추가합니다.

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

그런 다음 파일의 TLS 섹션에 다음을 추가합니다.

```
# Settings for a TLS enabled server.
server {
    listen       443 ssl http2 default_server;
    listen       [::]:443 ssl http2 default_server;
    server_name  _;
    root         /usr/share/nginx/html;

    ssl_certificate "/etc/pki/nginx/server.crt";
    ssl_certificate_key "/etc/pki/nginx/private/server.key";
    # It is *strongly* recommended to generate unique DH parameters
    # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem 2048
    #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
    ssl_session_cache shared:SSL:1m;
    ssl_session_timeout  10m;
    ssl_protocols TLSv1.2 TLSv1.3;
    ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
```

```
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
    ssl_prefer_server_ciphers on;

    # Load configuration files for the default server block.
    include /etc/nginx/default.d/*.conf;

    location / {
    }

    error_page 404 /404.html;
    location = /40x.html {
    }

    error_page 500 502 503 504 /50x.html;
    location = /50x.html {
    }
}
```

Ubuntu 16.04 LTS

텍스트 편집기를 사용하여 /etc/nginx/nginx.conf 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다. 파일의 맨 위에 다음 행을 추가합니다.

```
ssl_engine cloudhsm;
    env n3fips_password;
```

그런 다음 파일의 TLS 섹션에 다음을 추가합니다.

```
# Settings for a TLS enabled server.
    server {
        listen       443 ssl http2 default_server;
        listen       [::]:443 ssl http2 default_server;
        server_name  _;
        root         /usr/share/nginx/html;

        ssl_certificate "/etc/pki/nginx/server.crt";
        ssl_certificate_key "/etc/pki/nginx/private/server.key";
        # It is *strongly* recommended to generate unique DH parameters
```

```
        # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem
 2048
        #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
        ssl_session_cache shared:SSL:1m;
        ssl_session_timeout  10m;
        ssl_protocols TLSv1.2;
        ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-
RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-
GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
        ssl_prefer_server_ciphers on;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
        }

        error_page 404 /404.html;
        location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
        }
    }
```

Ubuntu 18.04 LTS

텍스트 편집기를 사용하여 /etc/nginx/nginx.conf 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다. 파일의 맨 위에 다음 행을 추가합니다.

```
ssl_engine cloudhsm;
    env CLOUDHSM_PIN;
```

그런 다음 파일의 TLS 섹션에 다음을 추가합니다.

```
# Settings for a TLS enabled server.
    server {
```

```
        listen       443 ssl http2 default_server;
        listen       [::]:443 ssl http2 default_server;
        server_name  _;
        root         /usr/share/nginx/html;

        ssl_certificate "/etc/pki/nginx/server.crt";
        ssl_certificate_key "/etc/pki/nginx/private/server.key";
        # It is *strongly* recommended to generate unique DH parameters
        # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem
 2048
        #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
        ssl_session_cache shared:SSL:1m;
        ssl_session_timeout  10m;
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-
RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-
GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
        ssl_prefer_server_ciphers on;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
        }

        error_page 404 /404.html;
        location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
        }
    }
```

Ubuntu 20.04 LTS

텍스트 편집기를 사용하여 `/etc/nginx/nginx.conf` 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다. 파일의 맨 위에 다음 행을 추가합니다.

```
ssl_engine cloudhsm;
    env CLOUDHSM_PIN;
```

그런 다음 파일의 TLS 섹션에 다음을 추가합니다.

```
# Settings for a TLS enabled server.
    server {
        listen       443 ssl http2 default_server;
        listen       [::]:443 ssl http2 default_server;
        server_name  _;
        root         /usr/share/nginx/html;

        ssl_certificate "/etc/pki/nginx/server.crt";
        ssl_certificate_key "/etc/pki/nginx/private/server.key";
        # It is *strongly* recommended to generate unique DH parameters
        # Generate them with: openssl dhparam -out /etc/pki/nginx/dhparams.pem
 2048
        #ssl_dhparam "/etc/pki/nginx/dhparams.pem";
        ssl_session_cache shared:SSL:1m;
        ssl_session_timeout  10m;
        ssl_protocols TLSv1.2 TLSv1.3;
        ssl_ciphers "ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-
SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-
RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-
GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA";
        ssl_prefer_server_ciphers on;

        # Load configuration files for the default server block.
        include /etc/nginx/default.d/*.conf;

        location / {
        }

        error_page 404 /404.html;
        location = /40x.html {
        }

        error_page 500 502 503 504 /50x.html;
        location = /50x.html {
        }
```

```
    }
```

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

파일을 저장합니다.

8. `systemd` 구성 파일을 백업한 후, `EnvironmentFile` 경로를 설정합니다.

Amazon Linux

작업이 필요하지 않습니다.

Amazon Linux 2

1. `nginx.service` 파일을 백업합니다.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 텍스트 편집기에서 `/lib/systemd/system/nginx.service` 파일을 연 후, [서비스] 섹션 아래에 다음 경로를 추가합니다.

```
EnvironmentFile=/etc/sysconfig/nginx
```

CentOS 7

작업이 필요하지 않습니다.

CentOS 8

1. `nginx.service` 파일을 백업합니다.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 텍스트 편집기에서 `/lib/systemd/system/nginx.service` 파일을 연 후, [서비스] 섹션 아래에 다음 경로를 추가합니다.

```
EnvironmentFile=/etc/sysconfig/nginx
```

Red Hat 7

작업이 필요하지 않습니다.

Red Hat 8

1. nginx.service 파일을 백업합니다.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 텍스트 편집기에서 /lib/systemd/system/nginx.service 파일을 연 후, [서비스] 섹션 아래에 다음 경로를 추가합니다.

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 16.04

1. nginx.service 파일을 백업합니다.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 텍스트 편집기에서 /lib/systemd/system/nginx.service 파일을 연 후, [서비스] 섹션 아래에 다음 경로를 추가합니다.

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 18.04

1. nginx.service 파일을 백업합니다.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 텍스트 편집기에서 /lib/systemd/system/nginx.service 파일을 연 후, [서비스] 섹션 아래에 다음 경로를 추가합니다.

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 20.04 LTS

1. nginx.service 파일을 백업합니다.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 텍스트 편집기에서 /lib/systemd/system/nginx.service 파일을 연 후, [서비스] 섹션 아래에 다음 경로를 추가합니다.

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

9. /etc/sysconfig/nginx 파일이 있는지 여부를 확인한 후, 다음 중 하나를 수행합니다.

- 파일이 있는 경우 다음 명령을 실행하여 파일을 백업합니다.

```
$ sudo cp /etc/sysconfig/nginx /etc/sysconfig/nginx.backup
```

- 파일이 없는 경우 텍스트 편집기를 연 후 /etc/sysconfig/ 폴더에 nginx라는 파일을 생성합니다.

10. NGINX 환경을 구성합니다.

> **Note**
>
> 클라이언트 SDK 5는 CU의 자격 증명을 저장하기 위한 CLOUDHSM_PIN 환경 변수를 도입합니다.

Amazon Linux

텍스트 편집기에서 `/etc/sysconfig/nginx` 파일을 엽니다. 이때 Linux 루트 권한이 필요합니다. 암호화 사용자(Cryptography User) 자격 증명 추가하기:

- 클라이언트 SDK 3을 사용하는 경우

```
n3fips_password=<CU user name>:<password>
```

- 클라이언트 SDK 5를 사용하는 경우

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

파일을 저장합니다.

Amazon Linux 2

텍스트 편집기에서 `/etc/sysconfig/nginx` 파일을 엽니다. 이때 Linux 루트 권한이 필요합니다. 암호화 사용자(Cryptography User) 자격 증명 추가하기:

- 클라이언트 SDK 3을 사용하는 경우

```
n3fips_password=<CU user name>:<password>
```

- 클라이언트 SDK 5를 사용하는 경우

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

파일을 저장합니다.

CentOS 7

텍스트 편집기에서 `/etc/sysconfig/nginx` 파일을 엽니다. 이때 Linux 루트 권한이 필요합니다. 암호화 사용자(Cryptography User) 자격 증명 추가하기:

- 클라이언트 SDK 3을 사용하는 경우

```
n3fips_password=<CU user name>:<password>
```

- 클라이언트 SDK 5를 사용하는 경우

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

파일을 저장합니다.

CentOS 8

텍스트 편집기에서 `/etc/sysconfig/nginx` 파일을 엽니다. 이때 Linux 루트 권한이 필요합니다. 암호화 사용자(Cryptography User) 자격 증명 추가하기:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

파일을 저장합니다.

Red Hat 7

텍스트 편집기에서 `/etc/sysconfig/nginx` 파일을 엽니다. 이때 Linux 루트 권한이 필요합니다. 암호화 사용자(Cryptography User) 자격 증명 추가하기:

- 클라이언트 SDK 3을 사용하는 경우

```
n3fips_password=<CU user name>:<password>
```

- 클라이언트 SDK 5를 사용하는 경우

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

파일을 저장합니다.

Red Hat 8

텍스트 편집기에서 /etc/sysconfig/nginx 파일을 엽니다. 이때 Linux 루트 권한이 필요합니다. 암호화 사용자(Cryptography User) 자격 증명 추가하기:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

파일을 저장합니다.

Ubuntu 16.04 LTS

텍스트 편집기에서 /etc/sysconfig/nginx 파일을 엽니다. 이때 Linux 루트 권한이 필요합니다. 암호화 사용자(Cryptography User) 자격 증명 추가하기:

```
n3fips_password=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

파일을 저장합니다.

Ubuntu 18.04 LTS

텍스트 편집기에서 /etc/sysconfig/nginx 파일을 엽니다. 이때 Linux 루트 권한이 필요합니다. 암호화 사용자(Cryptography User) 자격 증명 추가하기:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

파일을 저장합니다.

Ubuntu 20.04 LTS

텍스트 편집기에서 /etc/sysconfig/nginx 파일을 엽니다. 이때 Linux 루트 권한이 필요합니다. 암호화 사용자(Cryptography User) 자격 증명 추가하기:

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

파일을 저장합니다.

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

11. NGINX 웹 서버를 시작합니다.

Amazon Linux

텍스트 편집기에서 `/etc/sysconfig/nginx` 파일을 엽니다. 이때 Linux 루트 권한이 필요합니다. 암호화 사용자(Cryptography User) 자격 증명 추가하기:

```
$ sudo service nginx start
```

Amazon Linux 2

실행 중인 NGINX 프로세스를 모두 중지하기

```
$ sudo systemctl stop nginx
```

`systemd` 구성을 다시 로드하여 최신 변경 사항 찾아내기

```
$ sudo systemctl daemon-reload
```

NGINX 프로세스 시작하기

```
$ sudo systemctl start nginx
```

CentOS 7

실행 중인 NGINX 프로세스를 모두 중지하기

```
$ sudo systemctl stop nginx
```

`systemd` 구성을 다시 로드하여 최신 변경 사항 찾아내기

```
$ sudo systemctl daemon-reload
```

NGINX 프로세스 시작하기

```
$ sudo systemctl start nginx
```

CentOS 8

실행 중인 NGINX 프로세스를 모두 중지하기

```
$ sudo systemctl stop nginx
```

systemd 구성을 다시 로드하여 최신 변경 사항 찾아내기

```
$ sudo systemctl daemon-reload
```

NGINX 프로세스 시작하기

```
$ sudo systemctl start nginx
```

Red Hat 7

실행 중인 NGINX 프로세스를 모두 중지하기

```
$ sudo systemctl stop nginx
```

systemd 구성을 다시 로드하여 최신 변경 사항 찾아내기

```
$ sudo systemctl daemon-reload
```

NGINX 프로세스 시작하기

```
$ sudo systemctl start nginx
```

Red Hat 8

실행 중인 NGINX 프로세스를 모두 중지하기

```
$ sudo systemctl stop nginx
```

systemd 구성을 다시 로드하여 최신 변경 사항 찾아내기

```
$ sudo systemctl daemon-reload
```

NGINX 프로세스 시작하기

```
$ sudo systemctl start nginx
```

Ubuntu 16.04 LTS

실행 중인 NGINX 프로세스를 모두 중지하기

```
$ sudo systemctl stop nginx
```

systemd 구성을 다시 로드하여 최신 변경 사항 찾아내기

```
$ sudo systemctl daemon-reload
```

NGINX 프로세스 시작하기

```
$ sudo systemctl start nginx
```

Ubuntu 18.04 LTS

실행 중인 NGINX 프로세스를 모두 중지하기

```
$ sudo systemctl stop nginx
```

systemd 구성을 다시 로드하여 최신 변경 사항 찾아내기

```
$ sudo systemctl daemon-reload
```

NGINX 프로세스 시작하기

```
$ sudo systemctl start nginx
```

Ubuntu 20.04 LTS

실행 중인 NGINX 프로세스를 모두 중지하기

```
$ sudo systemctl stop nginx
```

systemd 구성을 다시 로드하여 최신 변경 사항 찾아내기

```
$ sudo systemctl daemon-reload
```

NGINX 프로세스 시작하기

```
$ sudo systemctl start nginx
```

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

12. (선택 사항) 시작 시 NGINX를 시작하도록 플랫폼을 구성합니다.

Amazon Linux

```
$ sudo chkconfig nginx on
```

Amazon Linux 2

```
$ sudo systemctl enable nginx
```

CentOS 7

작업이 필요하지 않습니다.

CentOS 8

```
$ sudo systemctl enable nginx
```

Red Hat 7

작업이 필요하지 않습니다.

Red Hat 8

```
$ sudo systemctl enable nginx
```

Ubuntu 16.04 LTS

```
$ sudo systemctl enable nginx
```

Ubuntu 18.04 LTS

```
$ sudo systemctl enable nginx
```

Ubuntu 20.04 LTS

```
$ sudo systemctl enable nginx
```

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

웹 서버 구성을 업데이트한 후에 4단계: HTTPS 트래픽 활성화 및 인증서 확인하기 단원으로 이동합니다.

Apache 웹 서버 구성하기

이 섹션을 사용하여 지원되는 플랫폼에서 Apache를 구성합니다.

Apache용 웹 서버 구성을 업데이트하려면

1. Amazon EC2 클라이언트 인스턴스에 연결합니다.
2. 플랫폼의 인증서 및 프라이빗 키의 기본 위치를 정의하십시오.

Amazon Linux

`/etc/httpd/conf.d/ssl.conf` 파일에 다음 값이 존재하는지 확인하십시오.

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Amazon Linux 2

/etc/httpd/conf.d/ssl.conf 파일에 다음 값이 존재하는지 확인하십시오.

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

CentOS 7

/etc/httpd/conf.d/ssl.conf 파일에 다음 값이 존재하는지 확인하십시오.

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

CentOS 8

/etc/httpd/conf.d/ssl.conf 파일에 다음 값이 존재하는지 확인하십시오.

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 7

/etc/httpd/conf.d/ssl.conf 파일에 다음 값이 존재하는지 확인하십시오.

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 8

/etc/httpd/conf.d/ssl.conf 파일에 다음 값이 존재하는지 확인하십시오.

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Ubuntu 16.04 LTS

/etc/apache2/sites-available/default-ssl.conf 파일에 다음 값이 존재하는지 확인하십시오.

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 18.04 LTS

`/etc/apache2/sites-available/default-ssl.conf` 파일에 다음 값이 존재하는지 확인하십시오.

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 20.04 LTS

`/etc/apache2/sites-available/default-ssl.conf` 파일에 다음 값이 존재하는지 확인하십시오.

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

3. 웹 서버 인증서를 플랫폼의 필수 위치에 복사하십시오.

Amazon Linux

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

*<web_server.crt>*를 웹 서버 인증서 이름으로 바꿉니다.

Amazon Linux 2

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

*<web_server.crt>*를 웹 서버 인증서 이름으로 바꿉니다.

CentOS 7

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

*<web_server.crt>*를 웹 서버 인증서 이름으로 바꿉니다.

CentOS 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

*<web_server.crt>*를 웹 서버 인증서 이름으로 바꿉니다.

Red Hat 7

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

*<web_server.crt>*를 웹 서버 인증서 이름으로 바꿉니다.

Red Hat 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

*<web_server.crt>*를 웹 서버 인증서 이름으로 바꿉니다.

Ubuntu 16.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

*<web_server.crt>*를 웹 서버 인증서 이름으로 바꿉니다.

Ubuntu 18.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

*<web_server.crt>*를 웹 서버 인증서 이름으로 바꿉니다.

Ubuntu 20.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

*<web_server.crt>*를 웹 서버 인증서 이름으로 바꿉니다.

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

4. 개인 PEM 프라이빗 키를 플랫폼에 필요한 위치로 복사합니다.

Amazon Linux

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

*<web_server_fake_PEM.key>*를 가짜 PEM 프라이빗 키가 포함된 파일 이름으로 바꿉니다.

Amazon Linux 2

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

*<web_server_fake_PEM.key>*를 가짜 PEM 프라이빗 키가 포함된 파일 이름으로 바꿉니다.

CentOS 7

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

*<web_server_fake_PEM.key>*를 가짜 PEM 프라이빗 키가 포함된 파일 이름으로 바꿉니다.

CentOS 8

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

*<web_server_fake_PEM.key>*를 가짜 PEM 프라이빗 키가 포함된 파일 이름으로 바꿉니다.

Red Hat 7

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

*<web_server_fake_PEM.key>*를 가짜 PEM 프라이빗 키가 포함된 파일 이름으로 바꿉니다.

Red Hat 8

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

*<web_server_fake_PEM.key>*를 가짜 PEM 프라이빗 키가 포함된 파일 이름으로 바꿉니다.

Ubuntu 16.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

*<web_server_fake_PEM.key>*를 가짜 PEM 프라이빗 키가 포함된 파일 이름으로 바꿉니다.

Ubuntu 18.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

*<web_server_fake_PEM.key>*를 가짜 PEM 프라이빗 키가 포함된 파일 이름으로 바꿉니다.

Ubuntu 20.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

*<web_server_fake_PEM.key>*를 가짜 PEM 프라이빗 키가 포함된 파일 이름으로 바꿉니다.

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

5. 플랫폼에서 필요한 경우 이러한 파일의 소유권을 변경하십시오.

Amazon Linux

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

apache라는 이름의 사용자에게 읽기 권한을 제공합니다.

Amazon Linux 2

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

apache라는 이름의 사용자에게 읽기 권한을 제공합니다.

CentOS 7

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

apache라는 이름의 사용자에게 읽기 권한을 제공합니다.

CentOS 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

apache라는 이름의 사용자에게 읽기 권한을 제공합니다.

Red Hat 7

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

apache라는 이름의 사용자에게 읽기 권한을 제공합니다.

Red Hat 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

apache라는 이름의 사용자에게 읽기 권한을 제공합니다.

Ubuntu 16.04 LTS

작업이 필요하지 않습니다.

Ubuntu 18.04 LTS

작업이 필요하지 않습니다.

Ubuntu 20.04 LTS

작업이 필요하지 않습니다.

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

6. 플랫폼의 Apache 지시어를 구성합니다.

Amazon Linux

이 플랫폼의 SSL 파일을 찾으십시오.

```
/etc/httpd/conf.d/ssl.conf
```

이 파일에는 서버 실행 방법을 정의하는 Apache 지시어가 들어 있습니다. 왼쪽에 지시어가 표시되고 그 뒤에 값이 표시됩니다. 텍스트 편집기를 사용하여 이 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다.

다음 지시어를 다음 값으로 업데이트하거나 입력하십시오.

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

파일을 저장합니다.

Amazon Linux 2

이 플랫폼의 SSL 파일을 찾으십시오.

```
/etc/httpd/conf.d/ssl.conf
```

이 파일에는 서버 실행 방법을 정의하는 Apache 지시어가 들어 있습니다. 왼쪽에 지시어가 표시되고 그 뒤에 값이 표시됩니다. 텍스트 편집기를 사용하여 이 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다.

다음 지시어를 다음 값으로 업데이트하거나 입력하십시오.

```
SSLCryptoDevice cloudhsm
```

```
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

파일을 저장합니다.

CentOS 7

이 플랫폼의 SSL 파일을 찾으십시오.

```
/etc/httpd/conf.d/ssl.conf
```

이 파일에는 서버 실행 방법을 정의하는 Apache 지시어가 들어 있습니다. 왼쪽에 지시어가 표시되고 그 뒤에 값이 표시됩니다. 텍스트 편집기를 사용하여 이 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다.

다음 지시어를 다음 값으로 업데이트하거나 입력하십시오.

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

파일을 저장합니다.

CentOS 8

이 플랫폼의 SSL 파일을 찾으십시오.

```
/etc/httpd/conf.d/ssl.conf
```

이 파일에는 서버 실행 방법을 정의하는 Apache 지시어가 들어 있습니다. 왼쪽에 지시어가 표시되고 그 뒤에 값이 표시됩니다. 텍스트 편집기를 사용하여 이 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다.

다음 지시어를 다음 값으로 업데이트하거나 입력하십시오.

```
SSLCryptoDevice cloudhsm
SSLProtocol TLSv1.2 TLSv1.3
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProxyCipherSuite HIGH:!aNULL
```

파일을 저장합니다.

Red Hat 7

이 플랫폼의 SSL 파일을 찾으십시오.

```
/etc/httpd/conf.d/ssl.conf
```

이 파일에는 서버 실행 방법을 정의하는 Apache 지시어가 들어 있습니다. 왼쪽에 지시어가 표시되고 그 뒤에 값이 표시됩니다. 텍스트 편집기를 사용하여 이 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다.

다음 지시어를 다음 값으로 업데이트하거나 입력하십시오.

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

파일을 저장합니다.

Red Hat 8

이 플랫폼의 SSL 파일을 찾으십시오.

```
/etc/httpd/conf.d/ssl.conf
```

이 파일에는 서버 실행 방법을 정의하는 Apache 지시어가 들어 있습니다. 왼쪽에 지시어가 표시되고 그 뒤에 값이 표시됩니다. 텍스트 편집기를 사용하여 이 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다.

다음 지시어를 다음 값으로 업데이트하거나 입력하십시오.

```
SSLCryptoDevice cloudhsm
SSLProtocol TLSv1.2 TLSv1.3
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProxyCipherSuite HIGH:!aNULL
```

파일을 저장합니다.

Ubuntu 16.04 LTS

이 플랫폼의 SSL 파일을 찾으십시오.

```
/etc/apache2/mods-available/ssl.conf
```

이 파일에는 서버 실행 방법을 정의하는 Apache 지시어가 들어 있습니다. 왼쪽에 지시어가 표시되고 그 뒤에 값이 표시됩니다. 텍스트 편집기를 사용하여 이 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다.

다음 지시어를 다음 값으로 업데이트하거나 입력하십시오.

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
```

파일을 저장합니다.

SSL 모듈 및 기본 SSL 사이트 구성을 활성화합니다.

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 18.04 LTS

이 플랫폼의 SSL 파일을 찾으십시오.

```
/etc/apache2/mods-available/ssl.conf
```

이 파일에는 서버 실행 방법을 정의하는 Apache 지시어가 들어 있습니다. 왼쪽에 지시어가 표시되고 그 뒤에 값이 표시됩니다. 텍스트 편집기를 사용하여 이 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다.

다음 지시어를 다음 값으로 업데이트하거나 입력하십시오.

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-
RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-
SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-
AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProtocol TLSv1.2 TLSv1.3
```

파일을 저장합니다.

SSL 모듈 및 기본 SSL 사이트 구성을 활성화합니다.

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 20.04 LTS

이 플랫폼의 SSL 파일을 찾으십시오.

```
/etc/apache2/mods-available/ssl.conf
```

이 파일에는 서버 실행 방법을 정의하는 Apache 지시어가 들어 있습니다. 왼쪽에 지시어가 표시되고 그 뒤에 값이 표시됩니다. 텍스트 편집기를 사용하여 이 파일을 편집합니다. 이때 Linux 루트 권한이 필요합니다.

다음 지시어를 다음 값으로 업데이트하거나 입력하십시오.

```
SSLCryptoDevice cloudhsm
SSLCipherSuite ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-AES128-SHA256:ECDHE-RSA-AES256-SHA384:DHE-RSA-AES128-SHA:DHE-RSA-AES256-SHA:DHE-RSA-AES128-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-SHA384:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-ECDSA-AES256-SHA:ECDHE-ECDSA-AES128-SHA
SSLProtocol TLSv1.2 TLSv1.3
```

파일을 저장합니다.

SSL 모듈 및 기본 SSL 사이트 구성을 활성화합니다.

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

7. 플랫폼의 환경 값 파일을 구성합니다.

Amazon Linux

작업이 필요하지 않습니다. `/etc/sysconfig/httpd`에 환경 값이 입력됩니다.

Amazon Linux 2

httpd 서비스 파일을 엽니다.

```
/lib/systemd/system/httpd.service
```

`[Service]` 섹션 아래에 다음을 추가합니다.

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 7

httpd 서비스 파일을 엽니다.

```
/lib/systemd/system/httpd.service
```

[Service] 섹션 아래에 다음을 추가합니다.

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 8

httpd 서비스 파일을 엽니다.

```
/lib/systemd/system/httpd.service
```

[Service] 섹션 아래에 다음을 추가합니다.

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 7

httpd 서비스 파일을 엽니다.

```
/lib/systemd/system/httpd.service
```

[Service] 섹션 아래에 다음을 추가합니다.

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 8

httpd 서비스 파일을 엽니다.

```
/lib/systemd/system/httpd.service
```

[Service] 섹션 아래에 다음을 추가합니다.

```
EnvironmentFile=/etc/sysconfig/httpd
```

Ubuntu 16.04 LTS

작업이 필요하지 않습니다. /etc/sysconfig/httpd에 환경 값이 입력됩니다.

Ubuntu 18.04 LTS

작업이 필요하지 않습니다. /etc/sysconfig/httpd에 환경 값이 입력됩니다.

Ubuntu 20.04 LTS

작업이 필요하지 않습니다. /etc/sysconfig/httpd에 환경 값이 입력됩니다.

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

8. 플랫폼의 환경 변수를 저장하는 파일에서 암호화 사용자(cryptographic user)의 자격 증명이 포함된 환경 변수를 설정합니다.

Amazon Linux

텍스트 편집기를 사용하여 /etc/sysconfig/httpd을 편집합니다.

- 클라이언트 SDK 3을 사용하는 경우

```
n3fips_password=<CU user name>:<password>
```

- 클라이언트 SDK 5를 사용하는 경우

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

Amazon Linux 2

텍스트 편집기를 사용하여 /etc/sysconfig/httpd을 편집합니다.

- 클라이언트 SDK 3을 사용하는 경우

```
n3fips_password=<CU user name>:<password>
```

- 클라이언트 SDK 5를 사용하는 경우

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

CentOS 7

텍스트 편집기를 사용하여 /etc/sysconfig/httpd을 편집합니다.

- 클라이언트 SDK 3을 사용하는 경우

```
n3fips_password=<CU user name>:<password>
```

- 클라이언트 SDK 5를 사용하는 경우

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

CentOS 8

텍스트 편집기를 사용하여 /etc/sysconfig/httpd을 편집합니다.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

Red Hat 7

텍스트 편집기를 사용하여 /etc/sysconfig/httpd을 편집합니다.

- 클라이언트 SDK 3을 사용하는 경우

```
n3fips_password=<CU user name>:<password>
```

- 클라이언트 SDK 5를 사용하는 경우

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

Red Hat 8

텍스트 편집기를 사용하여 /etc/sysconfig/httpd을 편집합니다.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

> **Note**
>
> 클라이언트 SDK 5는 CU의 자격 증명을 저장하기 위한 CLOUDHSM_PIN 환경 변수를 도입합니다.

Ubuntu 16.04 LTS

텍스트 편집기를 사용하여 /etc/apache2/envvars을 편집합니다.

```
export n3fips_password=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

Ubuntu 18.04 LTS

텍스트 편집기를 사용하여 /etc/apache2/envvars을 편집합니다.

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<##>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

> **Note**
>
> 클라이언트 SDK 5는 CU의 자격 증명을 저장하기 위한 CLOUDHSM_PIN 환경 변수를 도입합니다. 클라이언트 SDK 3에서 CU 자격 증명을 n3fips_password 환경 변수에 저장했습니다. 클라이언트 SDK 5는 두 환경 변수를 모두 지원하지만 CLOUDHSM_PIN을 사용하는 것이 좋습니다.

Ubuntu 20.04 LTS

텍스트 편집기를 사용하여 /etc/apache2/envvars을 편집합니다.

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

<CU ### ##> 및 *<###>*를 암호화 사용자(Cryptography User) 자격 증명으로 바꿉니다.

> **Note**
>
> 클라이언트 SDK 5는 CU의 자격 증명을 저장하기 위한 CLOUDHSM_PIN 환경 변수를 도입합니다. 클라이언트 SDK 3에서 CU 자격 증명을 n3fips_password 환경 변수에 저장했습니다. 클라이언트 SDK 5는 두 환경 변수를 모두 지원하지만 CLOUDHSM_PIN을 사용하는 것이 좋습니다.

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

9. Apache 웹 서버를 시작합니다.

Amazon Linux

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Amazon Linux 2

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

CentOS 7

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

CentOS 8

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Red Hat 7

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Red Hat 8

```
$ sudo systemctl daemon-reload
$ sudo service httpd start
```

Ubuntu 16.04 LTS

```
$ sudo service apache2 start
```

Ubuntu 18.04 LTS

```
$ sudo service apache2 start
```

Ubuntu 20.04 LTS

```
$ sudo service apache2 start
```

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

10. (선택 사항) 시작 시 Apache를 시작하도록 플랫폼을 구성합니다.

Amazon Linux

```
$ sudo chkconfig httpd on
```

Amazon Linux 2

```
$ sudo chkconfig httpd on
```

CentOS 7

```
$ sudo chkconfig httpd on
```

CentOS 8

```
$ systemctl enable httpd
```

Red Hat 7

```
$ sudo chkconfig httpd on
```

Red Hat 8

```
$ systemctl enable httpd
```

Ubuntu 16.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 18.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 20.04 LTS

```
$ sudo systemctl enable apache2
```

Ubuntu 22.04 LTS

OpenSSL Dynamic Engine에 대한 지원은 아직 제공되지 않습니다.

웹 서버 구성을 업데이트한 후에 4단계: HTTPS 트래픽 활성화 및 인증서 확인하기 단원으로 이동합니다.

4단계: HTTPS 트래픽 활성화 및 인증서 확인하기

SSL/TLS 오프로드를 사용하도록 웹 서버를 구성한 후 인바운드 HTTPS 트래픽을 허용하는 AWS CloudHSM보안 그룹에 웹 서버 인스턴스를 추가합니다. 이렇게 하면 웹 브라우저와 같은 클라이언트가 웹 서버와 HTTPS 연결을 설정할 수 있습니다. 그런 다음 웹 서버에 HTTPS 연결을 설정하고 SSL/TLS 오프로드를 위해 구성한 인증서를 사용하고 있는지 확인합니다. AWS CloudHSM

주제

- 인바운드 HTTPS 연결 활성화
- HTTPS가 사용자가 구성한 인증서를 사용하는지 확인

인바운드 HTTPS 연결 활성화

클라이언트(예: 웹 서버)에서 웹 서버에 연결하려면 인바운드 HTTPS 연결을 허용하는 보안 그룹을 생성합니다. 구체적으로 포트 443에서 인바운드 TCP 연결을 허용해야 합니다. 이 보안 그룹을 웹 서버에 할당합니다.

HTTPS용 보안 그룹을 생성하여 웹 서버에 할당하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 보안 그룹을 선택합니다.
3. 보안 그룹 생성을 선택합니다.
4. 보안 그룹 생성에서 다음을 수행합니다.
    a. 보안 그룹 이름에 생성하려는 보안 그룹의 이름을 입력합니다.
    b. (선택 사항) 생성하려는 보안 그룹에 대한 설명을 입력합니다.
    c. 웹 서버 Amazon EC2 인스턴스가 포함된 VPC를 VPC로 선택합니다.
    d. 규칙 추가를 선택합니다.
    e. 드롭다운 창에서 HTTPS를 유형으로 선택합니다.
    f. 소스에 소스 위치를 입력합니다.
    g. 보안 그룹 생성을 선택합니다.
5. 탐색 창에서 인스턴스(Instances)를 선택합니다.
6. 웹 서버 인스턴스 옆에 있는 확인란을 선택합니다.

7. 페이지 상단의 작업 드롭다운 메뉴를 선택합니다. 보안을 선택한 다음 보안 그룹 변경을 선택합니다.
8. 연결된 보안 그룹에서 검색 상자를 선택하고 HTTPS용으로 생성한 보안 그룹을 선택합니다. 그런 다음 보안 그룹 추가를 선택합니다.
9. 저장을 선택합니다.

HTTPS가 사용자가 구성한 인증서를 사용하는지 확인

웹 서버를 보안 그룹에 추가한 후 SSL/TLS 오프로드가 자체 서명된 인증서를 사용하고 있는지 확인할 수 있습니다. 웹 브라우저 또는 OpenSSL s_client와 같은 도구를 사용하여 이 작업을 수행할 수 있습니다.

웹 브라우저를 사용하여 SSL/TLS 오프로드를 확인하려면

1. 웹 브라우저를 사용하여 서버의 퍼블릭 DNS 이름 또는 IP 주소를 사용해 웹 서버에 연결합니다. 주소 표시줄의 URL이 https://로 시작하는지 확인합니다. 예를 들어 **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**입니다.

   > **Tip**
   >
   > Amazon Route 53과 같은 DNS 서비스를 사용하여 웹사이트의 도메인 이름(예: https://www.example.com/)을 웹 서버로 라우팅할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서 또는 DNS 서비스 설명서의 Amazon EC2 인스턴스로 트래픽 라우팅을 참조하십시오.

2. 웹 브라우저를 사용하여 웹 서버 인증서를 봅니다. 자세한 내용은 다음을 참조하십시오.

   - Mozilla Firefox의 경우 Mozilla Support 웹 사이트의 View a Certificate(인증서 보기)를 참조하십시오.
   - Google Chrome의 경우 Google Tools for Web Developers 웹 사이트의 보안 문제 이해를 참조하십시오.

   다른 웹 브라우저에도 웹 서버 인증서를 보는 데 사용할 수 있는 유사한 기능이 있을 수 있습니다.
3. SSL/TLS 인증서가 웹 서버에서 사용하도록 구성한 것인지 확인합니다.

OpenSSL s_client를 사용하여 SSL/TLS 오프로드를 확인하려면

1. 다음 OpenSSL 명령을 실행하여 HTTPS를 사용해 웹 서버에 연결합니다. *<server name>*을 웹 서버의 퍼블릭 DNS 이름 또는 IP 주소로 바꿉니다.

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> Amazon Route 53과 같은 DNS 서비스를 사용하여 웹사이트의 도메인 이름(예: https://www.example.com/)을 웹 서버로 라우팅할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서 또는 DNS 서비스 설명서의 Amazon EC2 인스턴스로 트래픽 라우팅을 참조하십시오.

2. SSL/TLS 인증서가 웹 서버에서 사용하도록 구성한 것인지 확인합니다.

이제 웹 사이트가 HTTPS로 보안됩니다. 웹 서버의 개인 키는 클러스터의 HSM에 저장됩니다. AWS CloudHSM

로드 밸런서를 추가하려면 Elastic Load Balancing을 사용하여 로드 밸런서 추가(선택 사항) 섹션을 참조하십시오.

리눅스에서 SSL/TLS 오프로드를 위한 JSSE 포함 Tomcat 사용

이 항목에서는 JCE SDK와 함께 Java 보안 소켓 확장 (JSSE) 을 사용하여 SSL/TLS 오프로드를 설정하는 step-by-step 방법에 대한 지침을 제공합니다. AWS CloudHSM

주제

- 개요
- 1단계: 사전 조건 설정
- 단계 2: 프라이빗 키 및 SSL/TLS 인증서 생성 또는 가져오기
- 3단계: Tomcat 웹 서버 구성
- 4단계: HTTPS 트래픽 활성화 및 인증서 확인하기

개요

에서 톰캣 웹 서버는 리눅스에서 AWS CloudHSM작동하여 HTTPS를 지원합니다. AWS CloudHSM JCE SDK는 JSSE (자바 보안 소켓 확장) 와 함께 사용할 수 있는 인터페이스를 제공하여 이러한 웹 서버에서 HSM을 사용할 수 있도록 합니다. AWS CloudHSM JCE는 JSSE를 AWS CloudHSM 클러스터에 연결하는 브리지입니다. JSSE는 보안 소켓 계층(SSL) 및 전송 계층 보안(TLS) 프로토콜을 위한 Java API입니다.

1단계: 사전 조건 설정

Linux에서 SSL/TLS 오프로드를 위해 Tomcat 웹 서버를 사용하려면 다음 사전 요구 사항을 따르십시오. AWS CloudHSM 클라이언트 SDK 5 및 Tomcat 웹 서버를 사용하여 웹 서버 SSL/TLS 오프로드를 설정하려면 이러한 필수 조건을 충족해야 합니다.

> **Note**
>
> 플랫폼마다 필요한 필수 조건이 다릅니다. 항상 플랫폼에 맞는 올바른 설치 단계를 따릅니다.

사전 조건

- Tomcat 웹 서버가 설치된 Linux 운영 체제를 실행하는 Amazon EC2 인스턴스입니다.
- HSM에서 웹 서버의 프라이빗 키를 소유하고 관리할 CU(Crypto User)입니다.
- JCE for Client SDK 5가 설치 및 구성된 하드웨어 보안 모듈 (HSM) 이 두 개 이상 있는 활성 AWS CloudHSM 클러스터

  > **Note**
  >
  > 단일 HSM 클러스터를 사용할 수 있지만 먼저 클라이언트 키 내구성을 비활성화해야 합니다. 자세한 내용은 클라이언트 키 내구성 설정 관리 및 클라이언트 SDK 5 구성 도구를 참조하십시오.

필수 조건 충족 방법

1. 두 개 이상의 하드웨어 보안 모듈 (HSM) 이 있는 활성 AWS CloudHSM AWS CloudHSM 클러스터에 JCE를 설치하고 구성합니다. 설치에 대한 자세한 내용은 클라이언트 SDK 5용 JCE를 참조하십시오.

2. AWS CloudHSM 클러스터에 액세스할 수 있는 EC2 Linux 인스턴스에서 Apache Tomcat 지침에 따라 Tomcat 웹 서버를 다운로드하고 설치합니다.
3. CloudHSM CLI를 사용하여 CU(Crypto User)를 생성합니다. HSM 사용자 관리에 대한 자세한 내용은 CloudHSM CLI를 사용한 HSM 사용자 관리를 참조하십시오.

> **Tip**
>
> CU의 사용자 이름과 암호를 기록합니다. 나중에 웹 서버용 HTTPS 프라이빗 키와 인증서를 생성하거나 가져올 때 이 정보가 필요합니다.

4. Java Keytool을 사용하여 JCE를 설정하려면 클라이언트 SDK 5를 사용하여 Java Keytool 및 Jarsigner와 통합의 지침을 따릅니다.

이 단계들을 완료한 후 단계 2: 프라이빗 키 및 SSL/TLS 인증서 생성 또는 가져오기로 이동합니다.

참고

- 보안이 강화된 리눅스(SELinux) 및 웹 서버를 사용하려면 클라이언트 SDK 5가 HSM과 통신하는 데 사용하는 포트인 포트 2223에서 아웃바운드 TCP 연결을 허용해야 합니다.
- 클러스터를 생성 및 활성화하고 EC2 인스턴스에 클러스터 액세스 권한을 부여하려면 AWS CloudHSM시작하기의 단계를 완료하십시오. 이 섹션에서는 HSM 1개와 Amazon EC2 클라이언트 인스턴스 1개로 액티브 클러스터를 생성하는 step-by-step 방법에 대한 지침을 제공합니다. 이 클라이언트 인스턴스를 웹 서버로 사용할 수 있습니다.
- 클라이언트 키 내구성을 비활성화하지 않으려면 클러스터에 HSM을 두 개 이상 추가하십시오. 자세한 내용은 HSM 추가 섹션을 참조하십시오.
- SSH 또는 PuTTY를 사용하여 클라이언트 인스턴스에 연결할 수 있습니다. 자세한 정보는 Amazon EC2 설명서의 SSH를 사용하여 Linux 인스턴스에 연결과 PuTTY를 사용하여 Windows에서 Linux 인스턴스에 연결 단원을 참조하세요.

단계 2: 프라이빗 키 및 SSL/TLS 인증서 생성 또는 가져오기

HTTPS를 활성화하려면 Tomcat 웹 서버 애플리케이션에서 프라이빗 키와 해당되는 SSL/TLS 인증서가 필요합니다. 웹 서버 SSL/TLS 오프로드를 함께 사용하려면 클러스터의 HSM에 AWS CloudHSM개인 키를 저장해야 합니다. AWS CloudHSM

> **Note**
>
> 아직 프라이빗 키와 해당 인증서가 없다면 HSM에서 프라이빗 키를 생성하세요. 프라이빗 키를 사용하여 SSL/TLS 인증서를 생성하는 데 사용하는 인증서 사인 요청(CSR)을 생성합니다.

HSM의 개인 키에 대한 참조와 관련 인증서를 포함하는 로컬 AWS CloudHSM KeyStore 파일을 생성합니다. 웹 서버는 SSL/TLS 오프로드 중에 이 AWS CloudHSM KeyStore 파일을 사용하여 HSM의 개인 키를 식별합니다.

주제

- 프라이빗 키 생성
- 자체 사인된 인증서를 생성합니다.

프라이빗 키 생성

이 섹션에서는 From JDK를 사용하여 키페어를 생성하는 방법을 보여줍니다. KeyTool HSM 내에서 키 쌍을 생성한 후에는 이를 KeyStore 파일로 내보내고 해당 인증서를 생성할 수 있습니다.

사용 사례에 따라 RSA 또는 EC 키 페어를 생성할 수 있습니다. 다음 단계에서는 RSA 키 쌍을 생성하는 방법을 보여 줍니다.

의 **genkeypair** KeyTool 명령을 사용하여 RSA 키 쌍을 생성합니다.

1. 아래의 *<VARIABLES>*을 특정 데이터로 바꾼 후 다음 명령을 사용하여 HSM의 프라이빗 키를 참조하는 `jsse_keystore.keystore`라는 이름의 키스토어 파일을 생성합니다.

```
$ keytool -genkeypair -alias <UNIQUE ALIAS FOR KEYS> -keyalg <KEY ALGORITHM> -
keysize <KEY SIZE> -sigalg <SIGN ALGORITHM> \
        -keystore <PATH>/<JSSE KEYSTORE NAME>.keystore -storetype CLOUDHSM \
        -dname CERT_DOMAIN_NAME \
        -J-classpath '-J'$JAVA_LIB'/*:/opt/cloudhsm/java/*:./*' \
        -provider "com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider" \
        -providerpath "$CLOUDHSM_JCE_LOCATION" \
        -keypass <KEY PASSWORD> -storepass <KEYSTORE PASSWORD>
```

   - ***<PATH>***: 키스토어 파일을 생성하려는 경로입니다.
   - ***<UNIQUE ALIAS FOR KEYS>***: HSM에서 키를 고유하게 식별하는 데 사용됩니다. 이 별칭은 키의 LABEL 속성으로 설정됩니다.

- ***<KEY PASSWORD>***: 키에 대한 참조를 로컬 Keystore 파일에 저장하며, 이 암호는 해당 로컬 참조를 보호합니다.
- ***<KEYSTORE PASSWORD>***: 로컬 키스토어 파일의 암호입니다.
- ***<JSSE KEYSTORE NAME>***: 키스토어 파일의 이름입니다.
- ***<CERT DOMAIN NAME>***: X.500 고유 이름.
- ***<KEY ALGORITHM>***: 키 페어를 생성하는 키 알고리즘(예: RSA 및 EC).
- ***<KEY SIZE>***: 키 페어를 생성하기 위한 키 크기(예: 2048, 3072, 4096).
- ***<SIGN ALGORITHM>***: 키 쌍을 생성하기 위한 키 크기(예: RSA를 사용하는 SHA1, RSA를 사용하는 SHA224, RSA를 사용하는 SHA256, RSA를 사용하는 SHA384, RSA를 사용하는 SHA512).

2. 명령이 성공했는지 확인하려면 다음 명령을 입력하고 RSA 키 쌍이 성공적으로 생성되었는지 확인합니다.

```
$ ls <PATH>/<JSSE KEYSTORE NAME>.keystore
```

자체 사인된 인증서를 생성합니다.

키스토어 파일과 함께 개인 키를 생성한 후에는 이 파일을 사용하여 인증서 서명 요청(CSR) 및 인증서를 생성할 수 있습니다.

프로덕션 환경에서는 일반적으로 CA(인증 기관)을 사용하여 CSR에서 인증서를 생성합니다. 테스트 환경에는 CA가 필요하지 않습니다. CA를 사용하는 경우 CA에 CSR 파일을 보내고 CA가 웹 서버에서 HTTPS용으로 제공하는 서명된 SSL/TLS 인증서를 사용하세요.

CA를 사용하는 대신 를 사용하여 자체 서명된 인증서를 KeyTool 만들 수 있습니다. 자체 사인된 인증서는 브라우저에서 신뢰하지 않으며 프로덕션 환경에서 사용해서는 안 됩니다. 테스트 환경에서는 이러한 인증서를 사용할 수 있습니다.

> **Warning**
>
> 자체 사인된 인증서는 테스트 환경에서만 사용해야 합니다. 프로덕션 환경의 경우 인증 기관과 같은 추가 보안 방법을 사용하여 인증서를 생성하세요.

인증서 생성

1. 이전 단계에서 생성한 키스토어 파일의 사본을 확보하세요.

2. 다음 명령을 실행하여 를 사용하여 인증서 서명 요청 (CSR) 을 생성합니다. KeyTool

```
$ keytool -certreq -keyalg RSA -alias unique_alias_for_key -file certreq.csr \
        -keystore <JSSE KEYSTORE NAME>.keystore -storetype CLOUDHSM \
        -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \
        -keypass <KEY PASSWORD> -storepass <KEYSTORE PASSWORD>
```

> **Note**
>
> 인증서 서명 요청의 출력 파일은 `certreq.csr`입니다.

인증서에 서명

- 아래 *<VARIABLES>*를 특정 데이터로 대체한 후 다음 명령을 실행하여 HSM의 프라이빗 키로 CSR에 서명합니다. 이렇게 하면 자체 사인된 인증서가 생성됩니다.

```
$ keytool -gencert -infile certreq.csr -outfile certificate.crt \
    -alias <UNIQUE ALIAS FOR KEYS> -keypass <KEY_PASSWORD> -
storepass <KEYSTORE_PASSWORD> -sigalg SIG_ALG \
    -storetype CLOUDHSM -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \
    -keystore jsse_keystore.keystore
```

> **Note**
>
> `certificate.crt`는 별칭의 프라이빗 키를 사용하는 서명된 인증서입니다.

키스토어에서 인증서 가져오기

- 아래 *<VARIABLES>*를 특정 데이터로 바꾼 후 다음 명령을 실행하여 서명된 인증서를 신뢰할 수 있는 인증서로 가져옵니다. 이 단계에서는 별칭으로 식별되는 키스토어 항목에 인증서를 저장합니다.

```
$ keytool -import -alias <UNIQUE ALIAS FOR KEYS> -keystore jsse_keystore.keystore \
    -file certificate.crt -storetype CLOUDHSM \
    -v -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \
    -keypass <KEY PASSWORD> -storepass <KEYSTORE_PASSWORD>
```

인증서를 PEM으로 변환합니다.

- 다음 명령을 실행하여 서명된 인증서 파일(.crt)을 PEM으로 변환합니다. PEM 파일은 http 클라이언트에서 요청을 보내는 데 사용됩니다.

```
$ openssl x509 -inform der -in certificate.crt -out certificate.pem
```

이 단계를 완료한 후 3단계: 웹 서버 구성으로 이동합니다.

3단계: Tomcat 웹 서버 구성

이전 단계에서 생성한 HTTPS 인증서와 해당 PEM 파일을 사용하려면 웹 서버 소프트웨어의 구성을 업데이트합니다. 시작하기 전에 기존 인증서와 키를 백업해야 한다는 점을 잊지 마십시오. 그러면 AWS CloudHSM이 지원되는 SSL/TLS 오프로드의 Linux 웹 서버 소프트웨어 설정이 완료됩니다. 자세한 내용은 Apache Tomcat 9 구성 참조를 참조하십시오.

서버 중지

- 아래의 *<VARIABLES>*를 특정 데이터로 바꾼 후 구성을 업데이트하기 전에 다음 명령을 실행하여 Tomcat 서버를 중지하십시오.

```
$ /<TOMCAT DIRECTORY>/bin/shutdown.sh
```

  - ***<TOMCAT DIRECTORY>***: Tomcat 설치 디렉터리입니다.

Tomcat의 Classpath 업데이트

1. 클라이언트 인스턴스에 연결합니다.
2. Tomcat 설치 폴더를 찾습니다.
3. 아래 *<VARIABLES>*를 특정 데이터로 바꾼 후 다음 명령을 사용하여 Tomcat/bin/catalina.sh 파일에 있는 Tomcat classpath에 Java 라이브러리 및 Cloudhsm Java 경로를 추가합니다.

```
$ sed -i 's@CLASSPATH="$CLASSPATH""$CATALINA_HOME"\/bin\/
bootstrap.jar@CLASSPATH="$CLASSPATH""$CATALINA_HOME"\/bin\/bootstrap.jar:'"
        <JAVA LIBRARY>"'\/*:\/opt\/cloudhsm\/java\/*:.\/*@' <TOMCAT PATH> /bin/
catalina.sh
```

  - ***<JAVA LIBRARY>***: Java JRE 라이브러리 위치입니다.

- ***<TOMCAT PATH>***: Tomcat 설치 폴더입니다.

서버 구성에 HTTPS 커넥터를 추가합니다.

1. Tomcat 설치 폴더로 이동합니다.
2. 아래의 *<VARIABLES>*를 특정 데이터로 바꾼 후 다음 명령을 사용하여 필수 구성 요소에서 생성된 인증서를 사용하도록 HTTPS 커넥터를 추가합니다.

```
$ sed -i '/<Connector port="8080"/i <Connector port=\"443\" maxThreads=\"200\"
 scheme=\"https\" secure=\"true\" SSLEnabled=\"true\" keystoreType=\"CLOUDHSM\"
 keystoreFile=\"
       <CUSTOM DIRECTORY>/<JSSE KEYSTORE NAME>.keystore\" keystorePass=\"<KEYSTORE
 PASSWORD>\" keyPass=\"<KEY PASSWORD>
       \" keyAlias=\"<UNIQUE ALIAS FOR KEYS>" clientAuth=\"false\" sslProtocol=
\"TLS\"/>' <TOMCAT PATH>/conf/server.xml
```

- ***<CUSTOM DIRECTORY>***: 키스토어 파일이 위치한 디렉터리입니다.
- ***<JSSE KEYSTORE NAME>***: 키스토어 파일의 이름입니다.
- ***<KEYSTORE PASSWORD>***: 로컬 키스토어 파일의 암호입니다.
- ***<KEY PASSWORD>***: 키에 대한 참조를 로컬 Keystore 파일에 저장하며, 이 암호는 해당 로컬 참조를 보호합니다.
- ***<UNIQUE ALIAS FOR KEYS>***: HSM에서 키를 고유하게 식별하는 데 사용됩니다. 이 별칭은 키의 LABEL 속성으로 설정됩니다.
- ***<TOMCAT PATH>***: Tomcat 폴더의 경로입니다.

서버 시작

- 아래 *<VARIABLES>*를 특정 데이터로 바꾼 후 다음 명령을 사용하여 Tomcat 서버를 시작합니다.

```
$ /<TOMCAT DIRECTORY>/bin/startup.sh
```

> **Note**
>
> ***<TOMCAT DIRECTORY>***는 Tomcat 설치 디렉터리의 이름입니다.

웹 서버 구성을 업데이트한 후에 4단계: HTTPS 트래픽 활성화 및 인증서 확인하기 단원으로 이동합니다.

4단계: HTTPS 트래픽 활성화 및 인증서 확인하기

SSL/TLS 오프로드를 사용하도록 웹 서버를 구성한 후 인바운드 HTTPS 트래픽을 허용하는 AWS CloudHSM보안 그룹에 웹 서버 인스턴스를 추가합니다. 이렇게 하면 웹 브라우저와 같은 클라이언트가 웹 서버와 HTTPS 연결을 설정할 수 있습니다. 그런 다음 웹 서버에 HTTPS 연결을 설정하고 SSL/TLS 오프로드를 위해 구성한 인증서를 사용하고 있는지 확인합니다. AWS CloudHSM

주제

- 인바운드 HTTPS 연결 활성화
- HTTPS가 사용자가 구성한 인증서를 사용하는지 확인

인바운드 HTTPS 연결 활성화

클라이언트(예: 웹 서버)에서 웹 서버에 연결하려면 인바운드 HTTPS 연결을 허용하는 보안 그룹을 생성합니다. 구체적으로 포트 443에서 인바운드 TCP 연결을 허용해야 합니다. 이 보안 그룹을 웹 서버에 할당합니다.

HTTPS용 보안 그룹을 생성하여 웹 서버에 할당하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 보안 그룹을 선택합니다.
3. 보안 그룹 생성을 선택합니다.
4. 보안 그룹 생성에서 다음을 수행합니다.
    a. 보안 그룹 이름에 생성하려는 보안 그룹의 이름을 입력합니다.
    b. (선택 사항) 생성하려는 보안 그룹에 대한 설명을 입력합니다.
    c. 웹 서버 Amazon EC2 인스턴스가 포함된 VPC를 VPC로 선택합니다.
    d. 규칙 추가를 선택합니다.
    e. 드롭다운 창에서 HTTPS를 유형으로 선택합니다.
    f. 소스에 소스 위치를 입력합니다.
    g. 보안 그룹 생성을 선택합니다.
5. 탐색 창에서 인스턴스(Instances)를 선택합니다.
6. 웹 서버 인스턴스 옆에 있는 확인란을 선택합니다.

7. 페이지 상단의 작업 드롭다운 메뉴를 선택합니다. 보안을 선택한 다음 보안 그룹 변경을 선택합니다.
8. 연결된 보안 그룹에서 검색 상자를 선택하고 HTTPS용으로 생성한 보안 그룹을 선택합니다. 그런 다음 보안 그룹 추가를 선택합니다.
9. 저장을 선택합니다.

HTTPS가 사용자가 구성한 인증서를 사용하는지 확인

웹 서버를 보안 그룹에 추가한 후 SSL/TLS 오프로드가 자체 서명된 인증서를 사용하고 있는지 확인할 수 있습니다. 웹 브라우저 또는 OpenSSL s_client와 같은 도구를 사용하여 이 작업을 수행할 수 있습니다.

웹 브라우저를 사용하여 SSL/TLS 오프로드를 확인하려면

1. 웹 브라우저를 사용하여 서버의 퍼블릭 DNS 이름 또는 IP 주소를 사용해 웹 서버에 연결합니다. 주소 표시줄의 URL이 https://로 시작하는지 확인합니다. 예를 들어 **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**입니다.

   > **Tip**
   >
   > Amazon Route 53과 같은 DNS 서비스를 사용하여 웹사이트의 도메인 이름(예: https://www.example.com/)을 웹 서버로 라우팅할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서 또는 DNS 서비스 설명서의 Amazon EC2 인스턴스로 트래픽 라우팅을 참조하십시오.

2. 웹 브라우저를 사용하여 웹 서버 인증서를 봅니다. 자세한 내용은 다음을 참조하십시오.

   - Mozilla Firefox의 경우 Mozilla Support 웹 사이트의 View a Certificate(인증서 보기)를 참조하십시오.
   - Google Chrome의 경우 Google Tools for Web Developers 웹 사이트의 보안 문제 이해를 참조하십시오.

   다른 웹 브라우저에도 웹 서버 인증서를 보는 데 사용할 수 있는 유사한 기능이 있을 수 있습니다.
3. SSL/TLS 인증서가 웹 서버에서 사용하도록 구성한 것인지 확인합니다.

OpenSSL s_client를 사용하여 SSL/TLS 오프로드를 확인하려면

1. 다음 OpenSSL 명령을 실행하여 HTTPS를 사용해 웹 서버에 연결합니다. *<server name>*을 웹 서버의 퍼블릭 DNS 이름 또는 IP 주소로 바꿉니다.

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> Amazon Route 53과 같은 DNS 서비스를 사용하여 웹사이트의 도메인 이름(예: https://www.example.com/)을 웹 서버로 라우팅할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서 또는 DNS 서비스 설명서의 Amazon EC2 인스턴스로 트래픽 라우팅을 참조하십시오.

2. SSL/TLS 인증서가 웹 서버에서 사용하도록 구성한 것인지 확인합니다.

이제 웹 사이트가 HTTPS로 보안됩니다. 웹 서버의 개인 키는 클러스터의 HSM에 저장됩니다. AWS CloudHSM

로드 밸런서를 추가하려면 Elastic Load Balancing을 사용하여 로드 밸런서 추가(선택 사항) 섹션을 참조하십시오.

Windows에서 SSL/TLS 오프로드를 위해 CNG와 함께 IIS 사용

이 자습서에서는 Windows 웹 step-by-step 서버에서 SSL/TLS 오프로드를 설정하는 방법에 대한 지침을 제공합니다. AWS CloudHSM

주제

- 개요
- 1단계: 사전 조건 설정
- 단계 2: 인증서 서명 요청(CSR) 및 인증서 생성
- 3단계: 웹 서버 구성하기
- 4단계: HTTPS 트래픽 활성화 및 인증서 확인하기

개요

Windows에서 Windows Server용 IIS(인터넷 정보 서비스) 웹 서버 애플리케이션은 기본적으로 HTTPS를 지원합니다. Microsoft의 CNG(Cryptography API: Next Generation)용AWS CloudHSM KSP(Key Storage Provider)는 IIS가 암호화 오프로딩 및 키 스토리지를 위해 클러스터에서 HSM을 사용할 수 있도록 인터페이스를 제공합니다. AWS CloudHSM KSP는 IIS를 클러스터에 연결하는 브리지입니다. AWS CloudHSM

이 자습서에서는 다음을 수행하는 방법을 보여줍니다.

- Amazon EC2 인스턴스에 웹 서버 소프트웨어를 설치합니다.
- AWS CloudHSM 클러스터에 프라이빗 키가 저장되는 HTTPS를 지원하도록 웹 서버 소프트웨어를 구성합니다.
- (선택 사항) Amazon EC2를 사용하여 두 번째 웹 서버 인스턴스를 만들고 Elastic Load Balancing을 사용하여 로드 밸런서를 만듭니다. 로드 밸런서를 사용하면 여러 서버에 부하를 분산하여 성능을 향상할 수 있습니다. 또한 하나 이상의 서버에 장애가 발생할 경우 중복성과 더 높은 가용성을 제공할 수 있습니다.

시작할 준비가 되면 1단계: 사전 조건 설정로 이동합니다.

1단계: 사전 조건 설정

를 사용하여 웹 서버 SSL/TLS 오프로드를 설정하려면 다음이 필요합니다. AWS CloudHSM

- HSM이 하나 이상 있는 활성 AWS CloudHSM 클러스터
- 다음 소프트웨어가 설치된 Window 운영 체제를 실행하는 Amazon EC2 인스턴스
  - Windows용 AWS CloudHSM 클라이언트 소프트웨어.
  - Windows Server용 IIS(인터넷 정보 서비스).
- HSM에서 웹 서버의 프라이빗 키를 소유하고 관리할 CU(Crypto User)입니다.

> **Note**
>
> 이 자습서에서는 Microsoft Windows Server 2016을 사용합니다. Microsoft Windows Server 2012 역시 지원되지만, Microsoft Windows Server 2012 R2는 지원되지 않습니다.

HSM에서 Windows Server 인스턴스를 설정하고 CU를 생성하려면

1. 시작하기의 단계를 수행하세요. Amazon EC2 클라이언트를 시작할 때 Windows Server 2016 또는 Windows Server 2012 AMI를 선택합니다. 해당 단계를 수행하면 한 개의 HSM이 있는 활성 클러스터가 생깁니다. 또한 Windows용 클라이언트 소프트웨어가 설치된 Windows Server를 실행하는 Amazon EC2 AWS CloudHSM 클라이언트 인스턴스도 있습니다.
2. (선택 사항) 클러스터에 HSM을 더 추가합니다. 자세한 내용은 HSM 추가 섹션을 참조하십시오.
3. Windows Server에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하십시오.
4. CloudHSM CLI를 사용하여 CU(Crypto User)를 생성합니다. CU의 사용자 이름과 암호를 기록합니다. 다음 단계에서 해당 정보가 필요합니다.

   > **Note**
   >
   > 사용자 생성에 대한 자세한 내용은 CloudHSM CLI를 사용하여 HSM 사용자 관리를 참조하십시오.

5. 이전 단계에서 생성한 CU 사용자 이름과 암호를 사용하여 HSM의 로그인 자격 증명을 설정합니다.
6. 5단계에서 Windows 자격 증명 관리자를 사용하여 HSM 자격 증명을 설정한 경우 psexec.exe에서 SysInternals 다운로드하여 다음 명령을 NT Authority\ SYSTEM으로 실행하십시오.

   ```
   psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
    --username <USERNAME> --password <PASSWORD>
   ```

   *<USERNAME>*과 *<PASSWORD>*를 HSM 자격 증명으로 바꿉니다.

Windows Server에 IIS를 설치하려면

1. 아직 역할 추가 및 키를 생성하지 않은 경우 Windows Server에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하십시오.
2. Windows Server에서 서버 관리자를 시작합니다.
3. 서버 관리자 대시보드에서 역할 및 기능 추가를 선택합니다.
4. 시작하기 전에 정보를 읽은 후 다음을 선택합니다.

5. Installation Type(설치 유형)에서 Role-based or feature-based installation(역할 기반 또는 기능 기반 설치)을 선택합니다. 이후 다음을 선택합니다.
6. 서버 선택에서 서버 풀에서 서버 선택을 선택합니다. 다음을 선택합니다.
7. 서버 역할의 경우 다음을 수행합니다.
    a. Web Server(IIS)(웹 서버(IIS))를 선택합니다.
    b. Add features that are required for Web Server (IIS)(웹 서버(IIS)에 필요한 기능 추가)에서 Add Features(기능 추가)를 선택합니다.
    c. 다음을 선택하여 서버 역할 선택을 마칩니다.
8. Features(기능)에 기본값을 적용합니다. 다음을 선택합니다.
9. Web Server Role (IIS)(웹 서버 역할(IIS)) 정보를 읽습니다. 다음을 선택합니다.
10. Select role services(역할 서비스 선택)에서 기본 설정을 그대로 수락하거나 필요에 따라 설정을 변경합니다. 다음을 선택합니다.
11. Confirmation(확인)에서 확인 정보를 읽은 후 Install(설치)을 선택합니다.
12. 설치가 완료되면 Close(닫기)를 클릭합니다.

이 단계들을 완료한 후 단계 2: 인증서 서명 요청(CSR) 및 인증서 생성로 이동합니다.

단계 2: 인증서 서명 요청(CSR) 및 인증서 생성

HTTPS를 활성화하려면 웹 서버에서 SSL/TLS 인증서와 해당되는 프라이빗 키가 필요합니다. SSL/TLS 오프로드를 사용하려면 클러스터의 HSM에 개인 AWS CloudHSM키를 저장합니다. AWS CloudHSM 이렇게 하려면 Microsoft의 CNG(Cryptography API: Next Generation)용AWS CloudHSM KSP(Key Storage Provider)를 사용하여 인증서 서명 요청(CSR)을 생성합니다. 그런 다음, 인증서 생성을 위해 CSR에 서명하는 인증 기관(CA)에 CSR을 제공합니다.

주제

CSR 생성

Windows 서버의 AWS CloudHSM KSP를 사용하여 CSR을 생성하십시오.

인증서 서명 요청(CSR)을 생성하려면

1. 아직 역할 추가 및 키를 생성하지 않은 경우 Windows Server에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하십시오.
2. 다음 명령을 사용하여 AWS CloudHSM 클라이언트 데몬을 시작합니다.

   Amazon Linux

   ```
   $ sudo start cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client start
   ```

   CentOS 7

   ```
   $ sudo service cloudhsm-client start
   ```

   CentOS 8

   ```
   $ sudo service cloudhsm-client start
   ```

   RHEL 7

   ```
   $ sudo service cloudhsm-client start
   ```

   RHEL 8

   ```
   $ sudo service cloudhsm-client start
   ```

   Ubuntu 16.04 LTS

   ```
   $ sudo service cloudhsm-client start
   ```

   Ubuntu 18.04 LTS

   ```
   $ sudo service cloudhsm-client start
   ```

Windows

- Windows 클라이언트 1.1.2+의 경우:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows 클라이언트 1.1.1 이상의 경우:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

3. Windows Server에서 텍스트 편집기를 사용하여 `IISCertRequest.inf`라는 인증서 요청을 생성합니다. 다음 예제에서는 `IISCertRequest.inf` 파일의 내용을 보여 줍니다. 이 파일에서 지정할 수 있는 섹션, 키 및 값에 대한 자세한 내용은 Microsoft 설명서를 참조하십시오. `ProviderName` 값은 변경하지 마십시오.

```
[Version]
Signature = "$Windows NT$"
[NewRequest]
Subject = "CN=example.com,C=US,ST=Washington,L=Seattle,O=ExampleOrg,OU=WebServer"
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = "Cavium Key Storage Provider"
KeyUsage = 0xf0
MachineKeySet = True
[EnhancedKeyUsageExtension]
OID=1.3.6.1.5.5.7.3.1
```

4. Windows certreq 명령을 사용해 이전 단계에서 생성한 `IISCertRequest.inf` 파일에서 CSR을 생성합니다. 다음 예제에서는 CSR을 `IISCertRequest.csr`이라는 파일에 저장합니다. 인증서 요청 파일에 다른 파일 이름을 사용한 경우 *`IIS CertRequest .inf#`* 적절한 파일 이름으로 바꾸십시오. 선택적으로 *`IIS CertRequest .csr# CSR`* 파일의 다른 파일 이름으로 바꿀 수 있습니다.

```
C:\>certreq -new IISCertRequest.inf IISCertRequest.csr
        SDK Version: 2.03

CertReq: Request Created
```

`IISCertRequest.csr` 파일에는 CSR이 포함되어 있습니다. 서명된 인증서를 얻기 위해서는 이 CSR이 필요합니다.

서명된 인증서를 받아 가져옵니다.

프로덕션 환경에서는 일반적으로 CA(인증 기관)을 사용하여 CSR에서 인증서를 생성합니다. 테스트 환경에는 CA가 필요하지 않습니다. CA를 사용하는 경우에는 CSR 파일(`IISCertRequest.csr`)을 CA에 전송하고 CA를 사용하여 서명된 SSL/TLS 인증서를 생성합니다.

CA 사용의 대안으로 OpenSSL 같은 도구를 사용하여 자체 서명 인증서를 생성할 수 있습니다.

> **Warning**
>
> 자체 사인된 인증서는 브라우저에서 신뢰하지 않으며 프로덕션 환경에서 사용해서는 안 됩니다. 테스트 환경에서는 이러한 인증서를 사용할 수 있습니다.

다음 절차에서는 자체 서명 인증서를 생성하고 이를 사용해 웹 서버의 CSR을 서명하는 방법을 보여줍니다.

자체 서명된 인증서를 생성하려면

1. OpenSSL 명령을 사용하여 프라이빗 키를 생성합니다. 선택적으로 *SelfSignedCA.key* 를 개인 키가 포함된 파일 이름으로 바꿀 수 있습니다.

```
openssl genrsa -aes256 -out SelfSignedCA.key 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................+++
.....................................+++
e is 65537 (0x10001)
Enter pass phrase for SelfSignedCA.key:
Verifying - Enter pass phrase for SelfSignedCA.key:
```

2. OpenSSL 명령을 사용하여 이전 단계에서 생성한 프라이빗 키를 통해 자체 서명 인증서를 생성합니다. 이것은 대화식 명령입니다. 화면의 지침을 읽고 프롬프트의 메시지를 따릅니다. *SelfSignedCA.key* 파일을 개인 키가 포함된 파일 이름 (다른 경우) 으로 바꾸십시오. 선택적으로 *SelfSignedca.crt#* 자체 서명된 인증서를 포함하는 파일 이름으로 바꿀 수 있습니다.

```
openssl req -new -x509 -days 365 -key SelfSignedCA.key -out SelfSignedCA.crt
```

```
Enter pass phrase for SelfSignedCA.key:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:
Email Address []:
```

자체 서명 인증서를 사용하여 웹 서버의 CSR을 서명하려면

- 다음 OpenSSL 명령을 사용해 프라이빗 키와 자체 서명 인증서를 통해 CSR을 서명합니다. 다음을 해당 데이터(다를 경우)가 포함된 파일 이름으로 바꿉니다.
  - *IIS CertRequest .csr* - 웹 서버의 CSR이 들어 있는 파일의 이름입니다.
  - *SelfSignedca.crt – ## ###* 인증서가 들어 있는 파일 이름
  - *SelfSignedCA.key* — 개인 키가 들어 있는 파일 이름
  - *IISCert.crt* – 웹 서버의 서명된 인증서를 포함하는 파일의 이름

```
openssl x509 -req -days 365 -in IISCertRequest.csr \
                            -CA SelfSignedCA.crt \
                            -CAkey SelfSignedCA.key \
                            -CAcreateserial \
                            -out IISCert.crt
Signature ok
subject=/ST=IIS-HSM/L=IIS-HSM/OU=IIS-HSM/O=IIS-HSM/CN=IIS-HSM/C=IIS-HSM
Getting CA Private Key
Enter pass phrase for SelfSignedCA.key:
```

이전 단계를 완료하고 나면 웹 서버에 대한 서명 인증서(`IISCert.crt`)와 자체 서명 인증서(`SelfSignedCA.crt`)를 얻게 됩니다. 이러한 파일을 얻었으면 3단계: 웹 서버 구성하기로 이동합니다.

3단계: 웹 서버 구성하기

이전 단계가 끝날 때 생성한 HTTPS 인증서를 사용하려면 IIS 웹사이트의 구성을 업데이트하십시오. 그러면 AWS CloudHSM이 지원되는 SSL/TLS 오프로드의 Windows 웹 서버 소프트웨어(IIS) 설정이 완료됩니다.

CSR 서명을 위해 자체 서명 인증서를 사용한 경우에는 먼저 신뢰할 수 있는 Windows 루트 인증 기관에 자체 서명 인증서를 가져와야 합니다.

자체 서명 인증서를 신뢰할 수 있는 Windows 루트 인증 기관에 가져오려면

1. 아직 역할 추가 및 키를 생성하지 않은 경우 Windows Server에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하십시오.
2. Windows Server에 자체 서명 인증서를 복사합니다.
3. Windows Server에서 제어판을 엽니다.
4. 제어판 검색에 **`certificates`**를 입력합니다. 그런 다음 컴퓨터 인증서 관리를 선택합니다.
5. 인증서 - 로컬 컴퓨터 창에서 신뢰할 수 있는 루트 인증 기관을 두 번 클릭합니다.
6. 인증서를 마우스 오른쪽 버튼으로 클릭하고 모든 작업, 가져오기를 선택합니다.
7. 인증서 가져오기 마법사에서 다음을 선택합니다.
8. 찾아보기에서 자체 서명 인증서를 찾아서 선택합니다. 이 자습서의 이전 단계에 나와 있는 지침에 따라 생성한 자체 서명 인증서는 이름이 `SelfSignedCA.crt`으로 지정됩니다. 열기를 선택합니다.
9. 다음을 선택합니다.
10. Certificate Store에서 다음 스토어에 모든 인증서 배치를 선택합니다. 그런 다음 인증서 저장소에 신뢰할 수 있는 루트 인증 기관이 선택되었는지 확인하십시오.
11. 다음을 선택한 다음 마침을 선택합니다.

IIS 웹사이트의 구성 업데이트

1. 아직 역할 추가 및 키를 생성하지 않은 경우 Windows Server에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하십시오.
2. AWS CloudHSM 클라이언트 데몬을 시작합니다.

3. 웹 서버의 서명된 인증서 복사 - 이 자습서의 이전 단계가 끝날 때 생성한 인증서를 Windows 서버에 복사합니다.
4. Windows Server에서는 다음 예제에서와 같이 Windows certreq 명령을 사용하여 서명 인증서를 수락합니다. *IISCert.crt*를 웹 서버의 서명 인증서가 포함된 파일 이름으로 바꿉니다.

```
C:\>certreq -accept IISCert.crt
        SDK Version: 2.03
```

5. Windows Server에서 서버 관리자를 시작합니다.
6. 서버 관리자 대시보드의 오른쪽 상단 모서리에서 도구, 인터넷 정보 서비스(IIS) 관리자를 선택합니다.
7. 인터넷 정보 서비스(IIS) 관리자 창에서 서버 이름을 두 번 클릭합니다. 그런 다음 사이트를 두 번 클릭합니다. 웹사이트를 선택합니다.
8. SSL 설정을 선택합니다. 그런 다음 창의 오른쪽에서 바인딩을 선택합니다.
9. 사이트 바인딩 창에서 추가를 선택합니다.
10. 유형에서 https를 선택합니다. SSL 인증서에서 이 자습서의 이전 단계가 끝날 때 생성한 HTTPS 인증서를 선택합니다.

> **Note**
>
> 이 인증서 바인딩 중에 오류가 발생하면, 서버를 다시 시작하고 이 단계를 다시 시도하십시오.

11. 확인를 선택합니다.

웹사이트의 구성을 업데이트한 후에 4단계: HTTPS 트래픽 활성화 및 인증서 확인하기로 이동합니다.

4단계: HTTPS 트래픽 활성화 및 인증서 확인하기

SSL/TLS 오프로드를 사용하도록 웹 서버를 구성한 후 인바운드 HTTPS 트래픽을 허용하는 AWS CloudHSM보안 그룹에 웹 서버 인스턴스를 추가합니다. 이렇게 하면 웹 브라우저와 같은 클라이언트가 웹 서버와 HTTPS 연결을 설정할 수 있습니다. 그런 다음 웹 서버에 HTTPS 연결을 설정하고 SSL/TLS 오프로드를 위해 구성한 인증서를 사용하고 있는지 확인합니다. AWS CloudHSM

주제

- 인바운드 HTTPS 연결 활성화

- HTTPS가 사용자가 구성한 인증서를 사용하는지 확인

인바운드 HTTPS 연결 활성화

클라이언트(예: 웹 서버)에서 웹 서버에 연결하려면 인바운드 HTTPS 연결을 허용하는 보안 그룹을 생성합니다. 구체적으로 포트 443에서 인바운드 TCP 연결을 허용해야 합니다. 이 보안 그룹을 웹 서버에 할당합니다.

HTTPS용 보안 그룹을 생성하여 웹 서버에 할당하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 탐색 창에서 보안 그룹을 선택합니다.
3. 보안 그룹 생성을 선택합니다.
4. 보안 그룹 생성에서 다음을 수행합니다.
    a. 보안 그룹 이름에 생성하려는 보안 그룹의 이름을 입력합니다.
    b. (선택 사항) 생성하려는 보안 그룹에 대한 설명을 입력합니다.
    c. 웹 서버 Amazon EC2 인스턴스가 포함된 VPC를 VPC로 선택합니다.
    d. 규칙 추가를 선택합니다.
    e. 드롭다운 창에서 HTTPS를 유형으로 선택합니다.
    f. 소스에 소스 위치를 입력합니다.
    g. 보안 그룹 생성을 선택합니다.
5. 탐색 창에서 인스턴스(Instances)를 선택합니다.
6. 웹 서버 인스턴스 옆에 있는 확인란을 선택합니다.
7. 페이지 상단의 작업 드롭다운 메뉴를 선택합니다. 보안을 선택한 다음 보안 그룹 변경을 선택합니다.
8. 연결된 보안 그룹에서 검색 상자를 선택하고 HTTPS용으로 생성한 보안 그룹을 선택합니다. 그런 다음 보안 그룹 추가를 선택합니다.
9. 저장을 선택합니다.

HTTPS가 사용자가 구성한 인증서를 사용하는지 확인

웹 서버를 보안 그룹에 추가한 후 SSL/TLS 오프로드가 자체 서명된 인증서를 사용하고 있는지 확인할 수 있습니다. 웹 브라우저 또는 OpenSSL s_client와 같은 도구를 사용하여 이 작업을 수행할 수 있습니다.

웹 브라우저를 사용하여 SSL/TLS 오프로드를 확인하려면

1. 웹 브라우저를 사용하여 서버의 퍼블릭 DNS 이름 또는 IP 주소를 사용해 웹 서버에 연결합니다. 주소 표시줄의 URL이 https://로 시작하는지 확인합니다. 예를 들어 **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**입니다.

   > **Tip**
   >
   > Amazon Route 53과 같은 DNS 서비스를 사용하여 웹사이트의 도메인 이름(예: https://www.example.com/)을 웹 서버로 라우팅할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서 또는 DNS 서비스 설명서의 Amazon EC2 인스턴스로 트래픽 라우팅을 참조하십시오.

2. 웹 브라우저를 사용하여 웹 서버 인증서를 봅니다. 자세한 내용은 다음을 참조하십시오.

   - Mozilla Firefox의 경우 Mozilla Support 웹 사이트의 View a Certificate(인증서 보기)를 참조하십시오.
   - Google Chrome의 경우 Google Tools for Web Developers 웹 사이트의 보안 문제 이해를 참조하십시오.

   다른 웹 브라우저에도 웹 서버 인증서를 보는 데 사용할 수 있는 유사한 기능이 있을 수 있습니다.

3. SSL/TLS 인증서가 웹 서버에서 사용하도록 구성한 것인지 확인합니다.

OpenSSL s_client를 사용하여 SSL/TLS 오프로드를 확인하려면

1. 다음 OpenSSL 명령을 실행하여 HTTPS를 사용해 웹 서버에 연결합니다. *<server name>*을 웹 서버의 퍼블릭 DNS 이름 또는 IP 주소로 바꿉니다.

   ```
   openssl s_client -connect <server name>:443
   ```

   > **Tip**
   >
   > Amazon Route 53과 같은 DNS 서비스를 사용하여 웹사이트의 도메인 이름(예: https://www.example.com/)을 웹 서버로 라우팅할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서 또는 DNS 서비스 설명서의 Amazon EC2 인스턴스로 트래픽 라우팅을 참조하십시오.

2. SSL/TLS 인증서가 웹 서버에서 사용하도록 구성한 것인지 확인합니다.

이제 웹 사이트가 HTTPS로 보안됩니다. 웹 서버의 개인 키는 클러스터의 HSM에 저장됩니다. AWS CloudHSM

로드 밸런서를 추가하려면 Elastic Load Balancing을 사용하여 로드 밸런서 추가(선택 사항) 섹션을 참조하십시오.

Elastic Load Balancing을 사용하여 로드 밸런서 추가(선택 사항)

웹 서버 하나를 사용하여 SSL/TLS 오프로드를 설정한 후 추가 웹 서버와 HTTPS 트래픽을 웹 서버로 라우팅하는 Elastic Load Balancing 로드 밸런서를 생성할 수 있습니다. 로드 밸런서는 둘 이상의 서버에 걸쳐 트래픽의 균형을 조정하여 개별 웹 서버의 부하를 줄일 수 있습니다. 또한 로드 밸런서는 웹 서버의 상태를 모니터링하여 정상적 서버로만 트래픽을 라우팅하므로 웹사이트의 가용성을 높일 수 있습니다. 웹 서버에 장애가 발생하면 로드 밸런서는 해당 웹 서버로의 트래픽 라우팅을 자동으로 중지합니다.

주제

- 두 번째 웹 서버의 서브넷 생성
- 두 번째 웹 서버 생성
- 로드 밸런서 생성

두 번째 웹 서버의 서브넷 생성

다른 웹 서버를 만들려면 먼저 기존 웹 서버 AWS CloudHSM 및 클러스터를 포함하는 동일한 VPC에 새 서브넷을 만들어야 합니다.

새 서브넷을 생성하려면

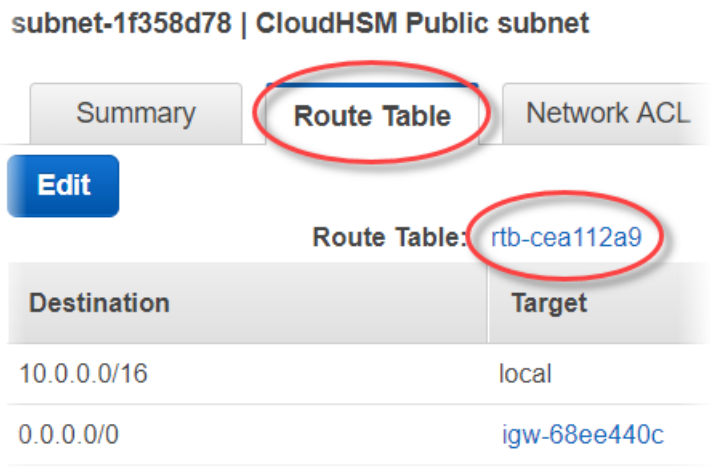
1. Amazon VPC 콘솔의 서브넷 섹션을 엽니다.
2. 서브넷 생성을 선택합니다.
3. [Create Subnet] 대화 상자에서 다음 작업을 수행합니다.
    a. Name 태그에 서브넷의 이름을 입력합니다.
    b. VPC의 경우 기존 웹 서버 및 클러스터가 포함된 AWS CloudHSM VPC를 선택합니다. AWS CloudHSM
    c. 가용 영역에서 기존 웹 서버가 포함된 가용 영역과는 다른 가용 영역을 선택합니다.

d. IPv4 CIDR 블록에 서브넷에 사용할 CIDR 블록을 입력합니다. 예를 들어 **10.0.10.0/24**를 입력합니다.

e. 예, 생성을 선택합니다.

4. 기존 웹 서버가 포함된 퍼블릭 서브넷 옆에 있는 확인란을 선택합니다. 이것은 이전 단계에서 생성한 퍼블릭 서브넷과 다릅니다.

5. 콘텐츠 창에서 라우팅 테이블 탭을 선택합니다. 그런 다음 라우팅 테이블의 링크를 선택합니다.



6. 라우팅 테이블 옆의 확인란을 선택합니다.

7. 서브넷 연결 탭을 선택합니다. 그런 다음 편집을 선택합니다.

8. 이 절차의 앞부분에서 생성한 퍼블릭 서브넷 옆에 있는 확인란을 선택합니다. 그런 다음 저장을 선택합니다.

두 번째 웹 서버 생성

다음 단계를 완료하여 기존 웹 서버와 동일한 구성으로 두 번째 웹 서버를 생성합니다.

두 번째 웹 서버를 생성하려면

1. Amazon EC2 콘솔의 인스턴스 섹션을 엽니다.

2. 기존 웹 서버 인스턴스 옆에 있는 확인란을 선택합니다.

3. 작업, 이미지, 이미지 생성을 차례로 선택합니다.

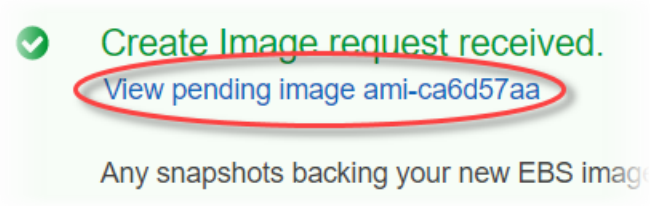
4. 이미지 생성 대화 상자에서 다음 작업을 수행합니다.

a. 이미지 이름에 이미지의 이름을 입력합니다.

b. 이미지 설명에 이미지에 대한 설명을 입력합니다.

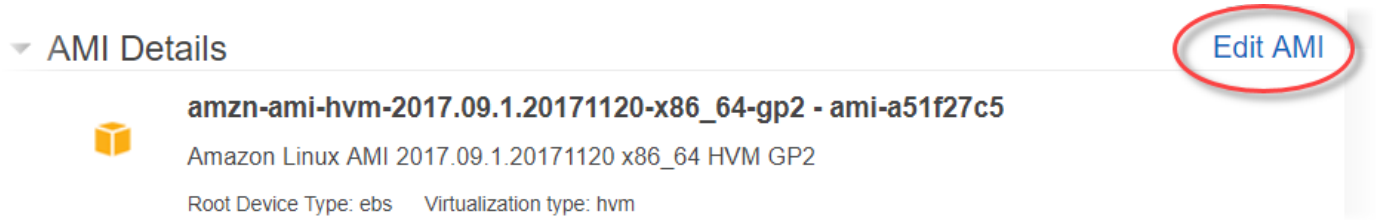
c. 이미지 생성을 선택합니다. 이 작업은 기존 웹 서버를 재부팅합니다.

d. 보류 중인 이미지 ami-*<AMI ID>* 보기 링크를 선택합니다.



상태 열에서 이미지 상태에 유의하십시오. 이미지 상태가 사용 가능이면(몇 분 정도 걸릴 수 있음) 다음 단계로 이동합니다.

5. 탐색 창에서 인스턴스를 선택합니다.
6. 기존 웹 서버 옆에 있는 확인란을 선택합니다.
7. 작업을 선택하고 기존 인스턴스를 기반으로 시작을 선택합니다.
8. AMI 편집을 선택합니다.



9. 왼쪽 탐색 창에서 나의 AMI를 선택합니다. 그 다음 검색 상자에서 텍스트를 삭제합니다.
10. 웹 서버 이미지 옆에 있는 선택을 선택합니다.
11. Yes, I want to continue with this AMI(예, 이 AMI를 계속 사용)(*<image name>* - ami-*<AMI ID>*)를 선택합니다.
12. 다음을 선택합니다.
13. 인스턴스 유형을 선택하고 Next: Configure Instance Details(다음: 인스턴스 정보 구성)를 선택합니다.
14. 3단계: 인스턴스 세부 정보 구성에서 다음을 수행합니다.

    a. 네트워크에서 기존의 웹 서버가 포함된 VPC를 선택합니다.

    b. 서브넷에서 두 번째 웹 서버를 위해 생성한 퍼블릭 서브넷을 선택합니다.

    c. 퍼블릭 IP 자동 할당에서 활성화를 선택합니다.

    d. 나머지 인스턴스 세부 정보를 원하는 대로 변경합니다. 그런 다음 Next: Add Storage(다음: 스토리지 추가)를 선택합니다.

15. 스토리지 설정을 원하는 대로 변경합니다. 다음: 태그 추가를 선택합니다.

16. 필요에 따라 태그를 추가하거나 편집합니다. 그러고 나서 다음: 보안 그룹 구성을 선택합니다.
17. 6단계: 보안 그룹 구성에서 다음을 수행합니다.

    a. 보안 그룹 할당에서 Select an existing security group(기존 보안 그룹 선택)을 선택합니다.

    b. cloudhsm-***<cluster ID>***-sg라는 보안 그룹 옆에 있는 확인란을 선택합니다. AWS CloudHSM 은 클러스터를 생성할 때 사용자를 대신하여 이 보안 그룹을 생성했습니다. 웹 서버 인스턴스가 클러스터의 HSM에 연결되도록 허용하려면 이 보안 그룹을 선택해야 합니다.

    c. 인바운드 HTTPS 트래픽을 허용하는 보안 그룹 옆에 있는 확인란을 선택합니다. 이 보안 그룹은 이전에 생성했습니다.

    d. (선택 사항) 네트워크에서 인바운드 SSH(Linux용) 또는 RDP(Windows용) 트래픽을 허용하는 보안 그룹 옆의 확인란을 선택합니다. 즉, 보안 그룹은 포트 22(Linux의 SSH용) 또는 포트 3389(Windows의 RDP용)에서 인바운드 TCP 트래픽을 허용해야 합니다. 그렇지 않으면 클라이언트 인스턴스에 연결할 수 있습니다. 이와 같은 보안 그룹이 없는 경우, 보안 그룹을 생성한 다음 이후 클라이언트 인스턴스에 할당해야 합니다.

    검토 및 시작을 선택합니다.
18. 인스턴스 세부 정보를 검토한 후 시작을 선택합니다.
19. 기존 키 페어를 사용하여 인스턴스를 시작할지, 새 키 페어를 생성할지 또는 키 페어 없이 인스턴스를 시작할지 선택합니다.

    - 기존 키 페어를 사용하려면 다음 작업을 수행하십시오.
      1. [Choose an existing key pair]를 선택합니다.
      2. [Select a key pair]의 경우 사용할 키 페어를 선택합니다.
      3. 선택한 프라이빗 키 파일(***<private key file name>***.pem)에 액세스할 수 있음을 확인합니다. 이 파일이 없으면 내 인스턴스에 로그인할 수 없습니다. 옆에 있는 확인란을 선택합니다.
    - 새 키 페어를 생성하려면 다음 작업을 수행하십시오.
      1. 새 키 페어 생성을 선택합니다.
      2. 키 페어 이름에 키 페어 이름을 입력합니다.
      3. [Download Key Pair]를 선택하고 안전하고 액세스 가능한 위치에 프라이빗 키 파일을 저장합니다.

> **Warning**
>
> 이 시점 이후에는 프라이빗 키 파일을 다시 다운로드할 수 없습니다. 지금 프라이빗 키 파일을 다운로드하지 않으면 클라이언트 인스턴스에 액세스할 수 없습니다.

- 키 페어 없이 인스턴스를 시작하려면 다음을 수행합니다.
  1. 키 페어 없이 계속을 선택합니다.
  2. 이 AMI에 기본 제공된 암호를 이미 알고 있는 경우가 아니면 이 인스턴스에 연결할 수 없는 것으로 알고 있습니다 옆에 있는 확인란을 선택합니다.

인스턴스 시작(Launch Instances)을 선택합니다.

로드 밸런서 생성

다음 단계를 완료하여 HTTPS 트래픽을 웹 서버로 라우팅하는 Elastic Load Balancing 로드 밸런서를 생성합니다.

로드 밸런서를 생성하려면

1. Amazon EC2 콘솔의 Load balancers(로드 밸런서) 섹션을 엽니다.
2. 로드 밸런서 생성을 선택합니다.
3. Network Load Balancer 섹션에서 생성을 선택합니다.
4. 1단계: 로드 밸런서 구성에서 다음을 수행합니다.
   a. 이름에 생성할 로드 밸런서 이름을 입력합니다.
   b. 리스너 섹션의 로드 밸런서 포트에서 값을 **443**으로 변경합니다.
   c. 가용 영역 섹션의 VPC에서 웹 서버가 포함된 VPC를 선택합니다.
   d. 가용 영역 섹션에서 웹 서버가 포함된 서브넷을 선택합니다.
   e. 다음: 라우팅 구성(Next: Configure Routing)을 선택합니다.
5. 2단계: 라우팅 구성에서 다음을 수행합니다.
   a. 이름에 생성하려는 대상 그룹의 이름을 입력합니다.
   b. 포트에서 값을 **443**으로 변경합니다.
   c. 다음: 대상 등록(Next: Register Targets)을 선택합니다.

6. 3단계: 대상 등록에서 다음을 수행합니다.

   a. 인스턴스 섹션에서 웹 서버 인스턴스 옆에 있는 확인란을 선택합니다. 그런 다음 등록된 항목에 추가를 선택합니다.

   b. 다음: 검토를 선택합니다.

7. 로드 밸런서 세부 정보를 검토한 후 생성을 선택합니다.

8. 로드 밸런서가 성공적으로 생성되면 닫기를 선택합니다.

위의 단계를 완료하면 Amazon EC2 콘솔에 Elastic Load Balancing 로드 밸런서가 표시됩니다.

로드 밸런서의 상태가 활성이면 로드 밸런서가 작동 중임을 확인할 수 있습니다. 즉, 로드 밸런서가 AWS CloudHSM을 사용한 SSL/TLS 오프로드로 HTTPS 트래픽을 웹 서버에 전송하고 있음을 확인할 수 있습니다. 웹 브라우저 또는 OpenSSL s_client와 같은 도구를 사용하여 이 작업을 수행할 수 있습니다.

로드 밸런서가 웹 브라우저에서 작동하고 있음을 확인하려면

1. Amazon EC2 콘솔에서 방금 생성한 로드 밸런서의 DNS 이름을 찾습니다. 그런 다음 DNS 이름을 선택하고 복사합니다.

2. Mozilla Firefox 또는 Google Chrome 같은 웹 브라우저로 로드 밸런서의 DNS 이름을 사용하여 로드 밸런서에 연결합니다. 주소 표시줄의 URL이 https://로 시작하는지 확인합니다.

   > **Tip**
   >
   > Amazon Route 53과 같은 DNS 서비스를 사용하여 웹사이트의 도메인 이름(예: https://www.example.com/)을 웹 서버로 라우팅할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서 또는 DNS 서비스 설명서의 Amazon EC2 인스턴스로 트래픽 라우팅을 참조하십시오.

3. 웹 브라우저를 사용하여 웹 서버 인증서를 봅니다. 자세한 내용은 다음을 참조하십시오.

   - Mozilla Firefox의 경우 Mozilla Support 웹 사이트의 View a Certificate(인증서 보기)를 참조하십시오.
   - Google Chrome의 경우 Google Tools for Web Developers 웹 사이트의 보안 문제 이해를 참조하십시오.

   다른 웹 브라우저에도 웹 서버 인증서를 보는 데 사용할 수 있는 유사한 기능이 있을 수 있습니다.

4. 인증서가 웹 서버에서 사용하도록 구성한 것인지 확인합니다.

로드 밸런서가 OpenSSL s_client에서 작동하는지 확인하려면

1. 다음 OpenSSL 명령으로 HTTPS를 사용하여 로드 밸런서에 연결합니다. *<DNS name>*을 로드 밸런서의 DNS 이름으로 바꿉니다.

```
openssl s_client -connect <DNS name>:443
```

> **Tip**
>
> Amazon Route 53과 같은 DNS 서비스를 사용하여 웹사이트의 도메인 이름(예: https://www.example.com/)을 웹 서버로 라우팅할 수 있습니다. 자세한 내용은 Amazon Route 53 개발자 안내서 또는 DNS 서비스 설명서의 Amazon EC2 인스턴스로 트래픽 라우팅을 참조하십시오.

2. 인증서가 웹 서버에서 사용하도록 구성한 것인지 확인합니다.

이제 웹 서버의 개인 키가 클러스터의 HSM에 저장되어 있는 HTTPS로 보호되는 웹 사이트가 생겼습니다. AWS CloudHSM 웹사이트에 2개의 웹 서버와 효율 및 가용성 개선을 돕는 로드 밸런서가 하나 있습니다.

AWS CloudHSM을 사용하여 Windows Server를 인증 기관(CA)으로 구성하기

퍼블릭 키 인프라(PKI)에서 인증 기관(CA)은 디지털 인증서를 발행하는 믿을 수 있는 단체입니다. 이러한 디지털 인증서는 퍼블릭 키 암호화 및 디지털 서명을 사용하여 퍼블릭 키를 ID(개인 또는 조직)에 바인딩합니다. CA를 운영하려면 CA에서 발급한 인증서에 서명하는 프라이빗 키를 보호하여 신뢰성을 유지해야 합니다. AWS CloudHSM 클러스터의 HSM에 프라이빗 키를 저장하고 HSM을 사용하여 암호화 서명 작업을 수행할 수 있습니다.

이 자습서에서는 Windows AWS CloudHSM Server를 사용하여 CA를 구성합니다. Windows 서버에 Windows용 AWS CloudHSM 클라이언트 소프트웨어를 설치한 후 Windows Server에 AD CS(Active Directory Certificate Services) 역할을 추가합니다. 이 역할을 구성할 때는 KSP (AWS CloudHSM 키 저장소 공급자) 를 사용하여 AWS CloudHSM 클러스터에 CA의 개인 키를 만들고 저장합니다. KSP는

Windows 서버를 클러스터에 연결하는 브리지입니다. AWS CloudHSM 마지막 단계에서는 Windows Server CA와 함께 인증서 서명 요청(CSR)을 서명합니다.

자세한 정보는 다음 주제를 참조하세요.

주제

- Windows Server CA 1단계: 필수 조건 설정하기
- Windows Server CA 2단계: AWS CloudHSM으로 Windows Server CA 생성하기
- 윈도우 서버 CA 3단계: 윈도우 서버 CA와 함께 인증서 서명 요청 (CSR) 에 서명합니다. AWS CloudHSM

Windows Server CA 1단계: 필수 조건 설정하기

Windows Server를 인증 기관 (CA) 으로 AWS CloudHSM설정하려면 다음이 필요합니다.

- HSM이 하나 이상 있는 활성 AWS CloudHSM 클러스터
- Windows용 AWS CloudHSM 클라이언트 소프트웨어가 설치된 Windows Server 운영 체제를 실행하는 Amazon EC2 인스턴스입니다. 이 자습서에서는 Microsoft Windows Server 2016을 사용합니다.
- HSM에서 CA 프라이빗 키를 소유하고 관리할 CU(Cryptographic User)입니다.

다음을 사용하여 Windows Server CA에 대한 사전 요구 사항을 설정하려면 AWS CloudHSM

1. 시작하기의 단계를 수행하세요. Amazon EC2 클라이언트를 시작할 때 Windows Server AMI를 선택합니다. 이 자습서에서는 Microsoft Windows Server 2016을 사용합니다. 해당 단계를 수행하면 한 개의 HSM이 있는 활성 클러스터가 생깁니다. 또한 Windows용 클라이언트 소프트웨어가 설치된 Windows Server를 실행하는 Amazon EC2 AWS CloudHSM 클라이언트 인스턴스도 있습니다.
2. (선택 사항) 클러스터에 HSM을 더 추가합니다. 자세한 정보는 HSM 추가을 참조하세요.
3. 클라이언트 인스턴스에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하십시오.
4. CloudHSM CLI로 HSM 사용자 관리 또는 CloudHSM 관리 유틸리티(CMU)로 HSM 사용자 관리를 사용하여 암호화 사용자(CU)를 생성합니다. CU의 사용자 이름과 암호를 기록합니다. 다음 단계에서 해당 정보가 필요합니다.
5. 이전 단계에서 생성한 CU 사용자 이름과 암호를 사용하여 HSM의 로그인 자격 증명을 설정합니다.

6. 5단계에서 Windows 자격 증명 관리자를 사용하여 HSM 자격 증명을 설정한 경우 psexec.exe에서 SysInternals 다운로드하여 다음 명령을 NT Authority\ SYSTEM으로 실행하십시오.

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
 --username <USERNAME> --password <PASSWORD>
```

<*USERNAME*>과 <*PASSWORD*>를 HSM 자격 증명으로 바꿉니다.

를 사용하여 Windows 서버 CA를 만들려면 AWS CloudHSM로 이동하십시오. Windows Server CA 생성

Windows Server CA 2단계: AWS CloudHSM으로 Windows Server CA 생성하기

Windows Server CA를 만들려면 Windows Server에 AD CS(Active Directory 인증서 서비스) 역할을 추가합니다. 이 역할을 추가하면 KSP (AWS CloudHSM 키 저장소 공급자) 를 사용하여 AWS CloudHSM 클러스터에 CA의 개인 키를 만들고 저장합니다.

> **Note**
>
> Windows Server CA를 만들 때 루트 CA 또는 종속 CA를 만들도록 선택할 수 있습니다. 일반적으로 퍼블릭 키 인프라 설계 및 조직의 보안 정책을 기준으로 결정합니다. 이 자습서에서는 단순 루트 CA를 만드는 방법을 설명합니다.

Windows Server에 AD CS 역할을 추가하고 CA의 프라이빗 키를 생성하려면

1. 아직 역할 추가 및 키를 생성하지 않은 경우 Windows Server에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하십시오.
2. Windows Server에서 서버 관리자를 시작합니다.
3. 서버 관리자 대시보드에서 역할 및 기능 추가를 선택합니다.
4. 시작하기 전에 정보를 읽은 후 다음을 선택합니다.
5. Installation Type(설치 유형)에서 Role-based or feature-based installation(역할 기반 또는 기능 기반 설치)을 선택합니다. 이후 다음을 선택합니다.
6. 서버 선택에서 서버 풀에서 서버 선택을 선택합니다. 다음을 선택합니다.

7. 서버 역할의 경우 다음을 수행합니다.

   a. Active Directory 인증서 서비스를 선택합니다.
   b. Active Directory 인증서 서비스에 필요한 기능을 추가하시겠습니까?라는 메시지가 표시되면 기능 추가를 선택합니다.
   c. 다음을 선택하여 서버 역할 선택을 마칩니다.
8. 기능에서 기본 설정을 그대로 수락하고 다음을 선택합니다.
9. AD CS의 경우 다음을 수행합니다.

   a. 다음을 선택합니다.
   b. Certification Authority를 선택한 다음 다음을 선택합니다.
10. 확인에서 확인 정보를 읽은 다음 설치를 선택합니다. 창을 닫습니다.
11. 강조 표시된 대상 서버에서 Active Directory 인증서 서비스 구성 링크를 선택합니다.
12. 자격 증명에서 표시된 자격 증명을 확인하거나 변경합니다. 다음을 선택합니다.
13. 역할 서비스에서 인증 기관을 선택합니다. 다음을 선택합니다.
14. 설치 유형에서 독립 CA를 선택합니다. 다음을 선택합니다.
15. CA 유형에서 루트 CA를 선택합니다. 다음을 선택합니다.

> **Note**
>
> 퍼블릭 키 인프라 설계 및 조직의 보안 정책을 기반으로 루트 CA 또는 하위 CA를 만들도록 선택할 수 있습니다. 이 자습서에서는 단순 루트 CA를 만드는 방법을 설명합니다.

16. 프라이빗 키에서 프라이빗 키 새로 만들기를 선택합니다. 다음을 선택합니다.
17. 암호화에서 다음을 수행합니다.

    a. 암호화 공급자 선택에서 메뉴의 Cavium 키 저장소 공급자 옵션 중 하나를 선택합니다. 이 옵션은 AWS CloudHSM KSP입니다. 예를 들어 RSA#Cavium Key Storage Provider를 선택할 수 있습니다.
    b. 키 길이에서 키 길이 옵션 중 하나를 선택합니다.
    c. 인증 기관에서 발급한 인증서에 서명하기 위한 해시 알고리즘을 선택합니다.에서 해시 알고리즘 옵션 중 하나를 선택합니다.

    다음을 선택합니다.
18. CA 이름에서 다음을 수행합니다.

a. (옵션) 일반 이름을 수정합니다.

b. (옵션) 고유 이름 접미사를 입력합니다.

다음을 선택합니다.

19. 유효 기간에 년, 월, 주 또는 일 단위로 기간을 지정합니다. 다음을 선택합니다.
20. 인증서 데이터베이스에서 기본값을 사용하거나 선택적으로 데이터베이스와 데이터베이스 로그의 위치를 변경할 수 있습니다. 다음을 선택합니다.
21. 확인에서 CA에 대한 정보를 검토하고 구성을 선택합니다.
22. 닫기를 선택한 다음 닫기를 차례로 선택합니다.

이제 Windows Server CA를 AWS CloudHSM사용할 수 있습니다. CA로 인증서 서명 요청(CSR)에 서명하는 방법을 배우려면 CSR 서명하기 섹션으로 이동합니다.

윈도우 서버 CA 3단계: 윈도우 서버 CA와 함께 인증서 서명 요청 (CSR) 에 서명합니다. AWS CloudHSM

Windows Server CA와 함께 AWS CloudHSM 사용하여 인증서 서명 요청 (CSR) 에 서명할 수 있습니다. 이러한 단계들을 완료하려면 유효한 CSR이 필요합니다. CSR은 다음을 포함한 다양한 방법으로 만들 수 있습니다.

- OpenSSL 사용
- Windows Server IIS(인터넷 정보 서비스) 관리자 사용
- Microsoft Management Console(MMC)에서 스냅인 인증서 사용
- Windows의 명령줄 유틸리티에서 certreq 사용

CSR 생성 단계는 본 자습서에서 다루지 않습니다. CSR이 있으면 Windows Server CA로 서명할 수 있습니다.

CSR로 Windows Server CA 서명

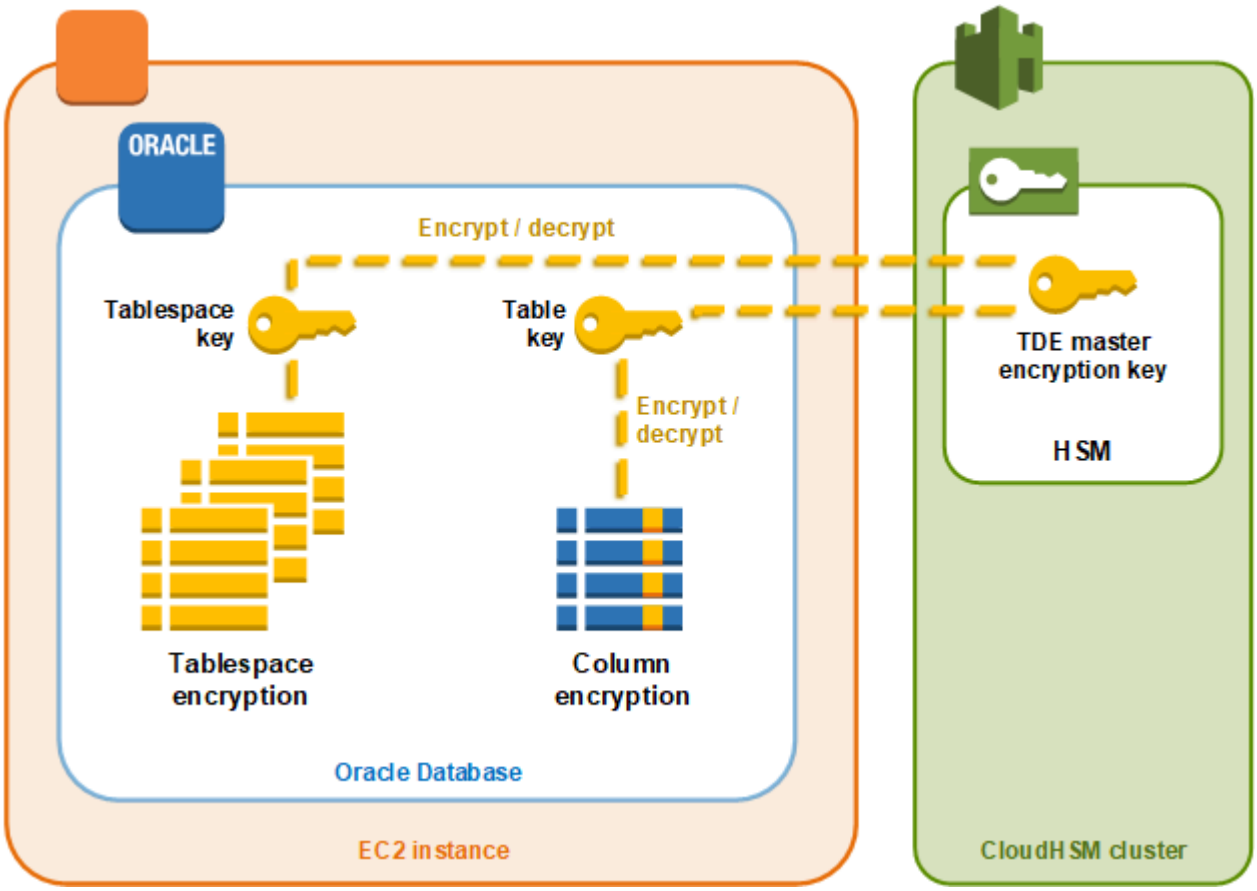
1. 아직 역할 추가 및 키를 생성하지 않은 경우 Windows Server에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하십시오.
2. Windows Server에서 서버 관리자를 시작합니다.
3. 오르쪽 상단 모서리에 서버 관리자 대시보드에서 도구, 인증 기관을 선택합니다.

4. 인증 기관 창에서 컴퓨터 이름을 선택합니다.
5. 작업 메뉴에서 모든 작업을 선택한 후 새 요청 제출을 선택합니다.
6. CSR 파일을 선택한 다음 열기를 선택합니다.
7. 인증 기관 창에서 대기 중인 요청을 두번 클릭합니다.
8. 대기 중인 요청을 선택합니다. 그런 다음 작업 메뉴에서 모든 작업을 선택하고 문제를 선택합니다.
9. 서명 인증서를 보려면 인증 기관 창에서 발급된 요청을 두번 클릭합니다.
10. (옵션) 서명 인증서를 파일에 내보내려면 다음 단계를 완료해야 합니다.
    a. 인증 기관 창에 보이는 인증서를 두번 클릭합니다.
    b. 세부 정보 탭을 선택한 다음 파일로 복사를 선택합니다.
    c. 인증서 내보내기 마법사에서 지침을 따릅니다.

이제 Windows Server CA가 생성되었으며 AWS CloudHSM, Windows Server CA에서 서명한 유효한 인증서가 생겼습니다.

AWS CloudHSM을 사용하는 Oracle Database TDE(Transparent Data Encryption)

TDE(Transparent Data Encryption)는 데이터베이스 파일을 암호화하는 데 사용됩니다. TDE를 사용하면 데이터베이스 소프트웨어는 데이터를 디스크에 저장하기 전에 암호화합니다. 데이터베이스의 테이블 열 또는 테이블스페이스의 데이터는 테이블 키 또는 테이블스페이스 키로 암호화됩니다. Oracle Database 소프트웨어의 일부 버전은 TDE를 제공합니다. Oracle TDE에서, 이 키는 TDE 마스터 암호화 키로 암호화됩니다. 클러스터의 HSM에 TDE 마스터 암호화 키를 저장하여 보안을 강화할 수 있습니다. AWS CloudHSM



이 솔루션에서는 Amazon EC2 인스턴스에 설치된 Oracle Database를 사용합니다. Oracle Database는 PKCS #11용AWS CloudHSM 소프트웨어 라이브러리와 통합되어 클러스터의 HSM에 TDE 마스터 키를 저장합니다.

> **Important**
>
> - Amazon EC2 인스턴스에 Oracle Database를 설치하는 것이 좋습니다.

Oracle TDE를 AWS CloudHSM과 통합하려면 다음 단계를 완료하십시오.

다음과 같은 Oracle TDE 통합을 구성하려면 AWS CloudHSM

1. 사전 조건 설정의 단계에 따라 환경을 준비합니다.
2. 의 단계에 따라 Oracle 데이터베이스를 데이터베이스 구성 구성하여 AWS CloudHSM 클러스터와 통합하십시오.

Oracle TDE 포함 AWS CloudHSM: 사전 요구 사항 설정

Oracle TDE 통합을 AWS CloudHSM수행하려면 다음이 필요합니다.

- HSM이 하나 이상 있는 활성 AWS CloudHSM 클러스터
- 다음 소프트웨어가 설치된 Amazon Linux 운영 체제를 실행하는 Amazon EC2 인스턴스:
  - AWS CloudHSM 클라이언트 및 명령줄 도구.
  - PKCS용 AWS CloudHSM 소프트웨어 라이브러리 #11.
  - 오라클 데이터베이스. AWS CloudHSM 오라클 TDE 통합을 지원합니다. Client SDK 5.6 이상은 Oracle Database 19c용 Oracle TDE를 지원합니다. Client SDK 3은 Oracle Database 버전 11g 및 12c용 Oracle TDE를 지원합니다.
- 클러스터의 HSM에서 TDE 마스터 암호화 키를 소유하고 관리할 CU(Cryptographic User)입니다.

다음 단계를 완료하여 사전 조건을 모두 설정합니다.

Oracle TDE 통합을 위한 사전 요구 사항을 설정하려면 AWS CloudHSM

1. 시작하기의 단계를 수행하세요. 이렇게 하면 HSM이 하나 있는 활성 클러스터가 생깁니다. 또한 Amazon Linux 운영 체제를 실행하는 Amazon EC2 인스턴스도 있습니다. AWS CloudHSM 클라이언트 및 명령줄 도구도 설치 및 구성됩니다.
2. (선택 사항) 클러스터에 HSM을 더 추가합니다. 자세한 내용은 HSM 추가 단원을 참조하십시오.
3. Amazon EC2 클라이언트 인스턴스에 연결하고 다음을 수행합니다.
   a. PKCS #11 AWS CloudHSM 소프트웨어 라이브러리를 설치합니다.
   b. Oracle Database를 설치합니다. 자세한 내용은 Oracle Database 설명서를 참조하십시오. Client SDK 5.6 이상은 Oracle Database 19c용 Oracle TDE를 지원합니다. Client SDK 3은 Oracle Database 버전 11g 및 12c용 Oracle TDE를 지원합니다.
   c. cloudhsm_mgmt_util 명령줄 도구를 사용하여 클러스터에서 CU(Cryptographic User)를 생성합니다. CU 생성에 대한 자세한 내용은 CMU를 사용하여 HSM 사용자를 관리 방법 및 HSM 사용자 관리 항목을 참조하십시오.

이 단계들을 완료한 후 데이터베이스 구성할 수 있습니다.

Oracle TDE 사용 AWS CloudHSM: 데이터베이스 구성 및 마스터 암호화 키 생성

Oracle TDE를 AWS CloudHSM 클러스터와 통합하려면 다음 항목을 참조하십시오.

1. Oracle 데이터베이스 구성 업데이트하여 클러스터의 HSM을 외부 보안 모듈로 사용합니다. 외부 보안 모듈에 대한 자세한 내용은 오라클 데이터베이스 고급 보안 가이드의 투명한 데이터 암호화 소개를 참조하십시오.
2. 클러스터의 HSM에서 Oracle TDE 마스터 암호화 키 생성.

Oracle 데이터베이스 구성 업데이트

클러스터의 HSM을 외부 보안 모듈로 사용하도록 Oracle 데이터베이스 구성을 업데이트하려면 다음 단계를 수행합니다. 외부 보안 모듈에 대한 자세한 내용은 오라클 데이터베이스 고급 보안 가이드의 투명한 데이터 암호화 소개를 참조하십시오.

Oracle 구성을 업데이트하려면

1. Amazon EC2 클라이언트 인스턴스에 연결합니다. 이 인스턴스는 Oracle 데이터베이스를 설치한 인스턴스입니다.
2. `sqlnet.ora` 파일의 백업 사본을 생성합니다. 이 파일의 위치는 Oracle 문서를 참조하십시오.
3. 텍스트 편집기를 사용하여 `sqlnet.ora` 파일을 편집합니다. 다음 행을 추가합니다. 파일에서 기존 행이 `encryption_wallet_location`으로 시작할 경우 기존 행을 다음과 같이 바꿉니다.

   ```
   encryption_wallet_location=(source=(method=hsm))
   ```

   파일을 저장합니다.
4. 다음 명령을 실행하여 Oracle 데이터베이스가 AWS CloudHSM PKCS #11 소프트웨어 라이브러리의 라이브러리 파일을 찾을 것으로 예상되는 디렉토리를 생성합니다.

   ```
   sudo mkdir -p /opt/oracle/extapi/64/hsm
   ```
5. 다음 명령을 실행하여 PKCS #11 파일용 AWS CloudHSM 소프트웨어 라이브러리를 이전 단계에서 만든 디렉터리로 복사합니다.

   ```
   sudo cp /opt/cloudhsm/lib/libcloudhsm_pkcs11.so /opt/oracle/extapi/64/hsm/
   ```

> **Note**
>
> `/opt/oracle/extapi/64/hsm` 디렉터리는 라이브러리 파일을 하나만 포함해야 합니다. 해당 디렉터리에 있는 다른 파일을 제거합니다.

6. 다음 명령을 실행하여 `/opt/oracle` 디렉터리와 그 안에 포함된 모든 항목의 소유권을 변경합니다.

```
sudo chown -R oracle:dba /opt/oracle
```

7. Oracle 데이터베이스를 시작합니다.

Oracle TDE 마스터 암호화 키 생성

클러스터의 HSM에서 Oracle TDE 마스터 키를 생성하려면 다음 절차의 단계를 수행합니다.

마스터 키를 생성하려면

1. 명령을 사용하여 Oracle SQL*Plus를 엽니다. 메시지가 표시되면 Oracle 데이터베이스를 설치할 때 설정한 시스템 암호를 입력합니다.

```
sqlplus / as sysdba
```

> **Note**
>
> Client SDK 3의 경우 마스터 키를 생성할 때마다 `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE` 환경 변수를 설정해야 합니다. 이 변수는 마스터 키 생성에만 필요합니다. 자세한 내용은 타사 애플리케이션 통합에 대해 알려진 문제에서 "문제: Oracle은 마스터 키 생성 중에 PCKS #11 속성 `CKA_MODIFIABLE`을 설정하지만 HSM은 이를 지원하지 않습니다"를 참조하십시오.

2. 다음 예제와 같이 마스터 암호화 키를 생성하는 SQL 문을 실행합니다. Oracle 데이터베이스의 버전에 해당하는 문을 사용하십시오. *<CU user name>*을 CU(Cryptographic User)의 사용자 이름으로 바꿉니다. *<password>*를 CU 암호로 바꿉니다.

> **Important**
>
> 다음 명령은 한 번만 실행하십시오. 명령이 실행될 때마다 새 마스터 암호화 키가 생성됩니다.

- Oracle 데이터베이스 버전 11의 경우, 다음 SQL 문을 실행합니다.

```
SQL> alter system set encryption key identified by "<CU user name>:<password>";
```

- Oracle 데이터베이스 12버전 및 19c버전의 경우 다음 SQL문을 실행합니다.

```
SQL> administer key management set key identified by "<CU user
 name>:<password>";
```

응답이 `System altered` 또는 `keystore altered`일 경우 Oracle TDE의 마스터 키가 성공적으로 생성 및 설정된 것입니다.

3. (선택 사항) 다음 명령을 실행하여 Oracle wallet의 상태를 확인합니다.

```
SQL> select * from v$encryption_wallet;
```

Oracle wallet이 열리지 않은 경우 다음 명령 중 하나를 실행하여 엽니다. *<CU user name>*을 CU(Cryptographic User)의 이름으로 바꿉니다. *<password>*를 CU 암호로 바꿉니다.

- Oracle 11의 경우, 다음 명령을 실행하여 Oracle Wallet을 엽니다.

```
SQL> alter system set encryption wallet open identified by "<CU user
 name>:<password>";
```

수동으로 Oracle wallet을 닫으려면 다음 명령을 실행합니다.

```
SQL> alter system set encryption wallet close identified by "<CU user
 name>:<password>";
```

- Oracle 12 및 Oracle 19c의 경우 지갑을 열기 위해 다음 명령을 실행합니다.

```
SQL> administer key management set keystore open identified by "<CU user
 name>:<password>";
```

수동으로 Oracle wallet을 닫으려면 다음 명령을 실행합니다.

```
SQL> administer key management set keystore close identified by "<CU user
 name>:<password>";
```

SignTool Microsoft와 함께 AWS CloudHSM 사용하여 파일에 서명하세요.

암호화 및 PKI(퍼블릭 키 인프라)에서는 신뢰할 수 있는 기관이 전송한 데이터임을 확인하기 위해 디지털 서명이 사용됩니다. 서명은 데이터가 전송 중에 변조되지 않았음을 나타내기도 합니다. 서명은 발신자의 프라이빗 키로 생성된 암호화된 해시입니다. 수신자는 발신자의 퍼블릭 키로 해시 서명 암호를 해독하여 데이터 무결성을 확인할 수 있습니다. 디지털 인증서를 유지 관리하는 것은 발신자의 책임입니다. 디지털 인증서는 발신자의 프라이빗 키 소유권을 나타내며 암호 해독에 필요한 퍼블릭 키를 수신자에게 제공합니다. 발신자가 개인 키를 소유하기만 하면 서명을 신뢰할 수 있습니다. AWS CloudHSM 독점적인 단일 테넌트 액세스로 이러한 키를 보호할 수 있는 안전한 FIPS 140-2 레벨 3 검증 하드웨어를 제공합니다.

많은 조직에서 파일에 서명하고 SignTool, 확인하고, 타임스탬프를 찍어 코드 서명 프로세스를 간소화하는 명령줄 도구인 Microsoft를 사용합니다. AWS CloudHSM 를 사용하면 필요할 때까지 키 페어를 안전하게 저장할 수 있으므로 데이터 서명을 위한 SignTool 워크플로를 쉽게 자동화할 수 있습니다.

다음 항목에서는 함께 사용하는 SignTool 방법에 대한 개요를 제공합니다. AWS CloudHSM

주제

- SignTool Microsoft의 AWS CloudHSM 1단계: 사전 요구 사항 설정
- Microsoft에서 SignTool 제공하는 AWS CloudHSM 2단계: 서명 인증서 만들기
- 마이크로소프트의 SignTool AWS CloudHSM 3단계: 파일에 서명

SignTool Microsoft의 AWS CloudHSM 1단계: 사전 요구 사항 설정

SignTool Microsoft와 함께 AWS CloudHSM사용하려면 다음이 필요합니다.

- Windows 운영 체제를 실행하는 Amazon EC2 클라이언트 인스턴스.
- 자체 관리하거나 타사 공급자가 설정한 CA(인증 기관)

- EC2 인스턴스와 동일한 가상 퍼블릭 클라우드 (VPC) 에 있는 활성 AWS CloudHSM 클러스터. 클러스터에 HSM이 하나 이상 있어야 합니다.
- 클러스터에서 키를 소유하고 관리하는 암호화 사용자 (CU). AWS CloudHSM
- 서명되지 않은 파일 또는 실행 파일
- Microsoft Windows 소프트웨어 개발 키트(SDK)

Windows와 함께 사용하기 AWS CloudHSM 위한 사전 요구 사항을 설정하려면 SignTool

1. 본 설명서의 시작하기 단원에 있는 지침에 따라 Windows EC2 인스턴스와 AWS CloudHSM 클러스터를 시작합니다.
2. 자체 Windows Server CA를 호스팅하려면 Windows Server를 인증 기관으로 구성의 1단계와 2단계를 따르십시오. AWS CloudHSM그렇지 않으면 계속해서 공신력 있는 타사 CA를 사용합니다.
3. 다음 Microsoft Windows SDK 버전 중 하나를 다운로드하여 Windows EC2 인스턴스에 설치합니다.

   - Microsoft Windows SDK 10
   - Microsoft Windows SDK 8.1
   - Microsoft Windows SDK 7

   `SignTool` 실행 파일은 데스크톱 앱을 위한 Windows SDK 서명 도구 설치 기능의 일부분입니다. 필요 없을 경우 다른 기능은 생략하고 설치해도 됩니다. 기본 설치 위치는 다음과 같습니다.

   ```
   C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\<CPU
    architecture>\signtool.exe
   ```

이제 Microsoft Windows SDK, AWS CloudHSM 클러스터 및 CA를 사용하여 서명 인증서를 만들 수 있습니다.

Microsoft에서 SignTool 제공하는 AWS CloudHSM 2단계: 서명 인증서 만들기

이제 Windows SDK를 EC2 인스턴스에 다운로드했습니다. SDK를 사용하여 CSR(인증서 서명 요청)을 생성할 수 있습니다. CSR은 최종적으로 서명을 위해 CA에 전달되는 서명되지 않은 인증서입니다. 이 예제에서는 Windows SDK에 포함된 `certreq` 실행 파일을 사용하여 CSR을 생성합니다.

certreq 실행 파일을 사용하여 CSR 생성

1. 아직 연결하지 않았다면 Windows EC2 인스턴스에 연결합니다. 자세한 내용은 Amazon EC2 사용 설명서의 인스턴스에 연결을 참조하십시오.
2. 아래의 행이 포함된 `request.inf` 파일을 생성합니다. `Subject` 정보를 조직에 맞게 바꿉니다. 각 파라미터에 대한 설명은 Microsoft 설명서를 참조하십시오.

```
[Version]
Signature= $Windows NT$
[NewRequest]
Subject = "C=<Country>,CN=<www.website.com>,O=<Organization>,OU=<Organizational-
Unit>,L=<City>,S=<State>"
RequestType=PKCS10
HashAlgorithm = SHA256
KeyAlgorithm = RSA
KeyLength = 2048
ProviderName = Cavium Key Storage Provider
KeyUsage = "CERT_DIGITAL_SIGNATURE_KEY_USAGE"
MachineKeySet = True
Exportable = False
```

3. `certreq.exe`를 실행합니다. 이 예제에서는 CSR을 `request.csr`로 저장합니다.

```
certreq.exe -new request.inf request.csr
```

내부적으로 AWS CloudHSM 클러스터에 새 키 쌍이 생성되고 이 쌍의 개인 키는 CSR을 생성하는 데 사용됩니다.

4. CA에 CSR을 제출합니다. Windows Server CA를 사용하는 경우 다음 단계를 따릅니다.

   a. 다음 명령을 입력하여 CA 도구를 엽니다.

   ```
   certsrv.msc
   ```

   b. 새 창에서 CA 서버 이름을 마우스 오른쪽 버튼으로 클릭합니다. 모든 작업을 선택한 후 새 요청 제출을 선택합니다.

   c. `request.csr`의 위치로 이동하여 열기를 선택합니다.

   d. 서버 CA 메뉴를 확장하여 대기 중인 요청 폴더로 이동합니다. 방금 만든 요청을 마우스 오른쪽 버튼으로 클릭하고 모든 작업에서 발행을 선택합니다.

   e. 이제 발행된 인증서 폴더(대기 중인 요청 폴더 위쪽)로 이동합니다.

f. 열기를 선택하여 인증서를 보고 세부 정보 탭을 선택합니다.

g. 파일에 복사를 선택하여 인증서 내보내기 마법사를 시작합니다. DER 인코딩 X.509 파일을 안전한 곳에 `signedCertificate.cer`로 저장합니다.

h. CA 도구를 끝내고 다음 명령을 사용하여 인증서 파일을 Windows의 개인 인증서 저장소로 이동합니다. 그러면 다른 애플리케이션에서 사용할 수 있습니다.

```
certreq.exe -accept signedCertificate.cer
```

이제 가져온 인증서를 사용하여 파일에 서명할 수 있습니다.

마이크로소프트의 SignTool AWS CloudHSM 3단계: 파일에 서명

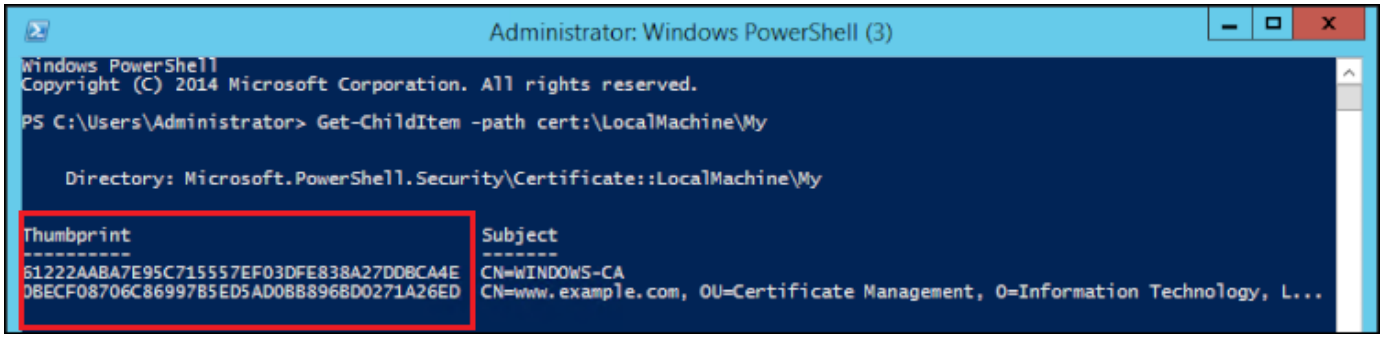
이제 가져온 인증서를 사용하여 SignTool 예제 파일에 서명할 준비가 되었습니다. 그러려면 인증서의 SHA-1 해시 또는 지문을 알아야 합니다. 지문은 에서 확인한 SignTool 인증서만 사용하도록 하는 데 사용됩니다. AWS CloudHSM이 예시에서는 인증서의 PowerShell 해시를 가져오는 데 사용합니다. CA의 GUI나 Windows SDK의 `certutil` 실행 파일을 사용할 수도 있습니다.

인증서 지문을 얻어 파일 서명에 사용하려면

1. 관리자 PowerShell 권한으로 열고 다음 명령을 실행합니다.

```
Get-ChildItem -path cert:\LocalMachine\My
```

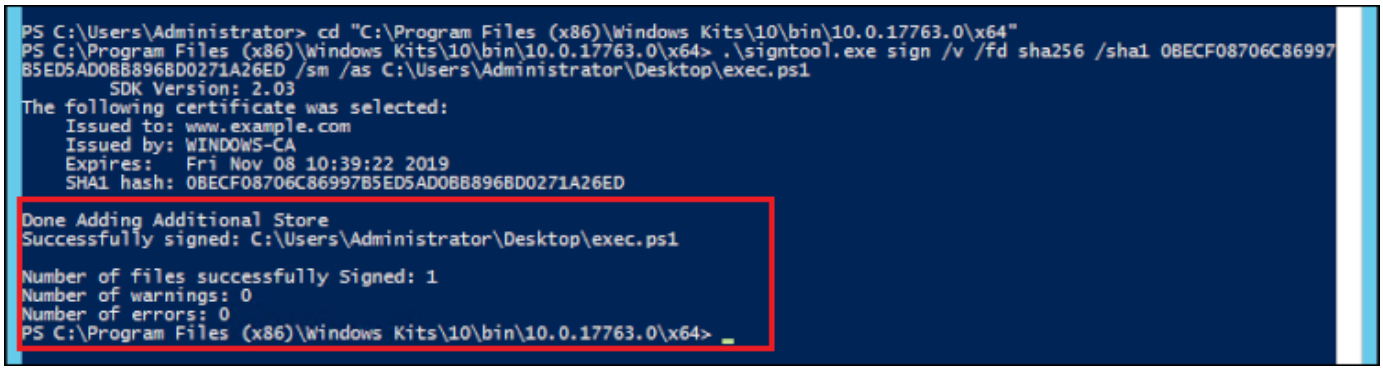
반환된 `Thumbprint`를 복사합니다.



2. PowerShell 포함된 디렉토리로 이동합니다`SignTool.exe`. 기본 위치는 `C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64`입니다.

3. 마지막으로 다음 명령을 실행하여 파일에 서명합니다. 명령이 성공하면 성공 메시지가 PowerShell 반환됩니다.

```
signtool.exe sign /v /fd sha256 /sha1 <thumbprint> /sm C:\Users\Administrator
\Desktop\<test>.ps1
```



4. (선택 사항) 파일의 서명을 확인하려면 다음 명령을 사용합니다.

```
signtool.exe verify /v /pa C:\Users\Administrator\Desktop\<test>.ps1
```

Java Keytool 및 Jarsigner

AWS CloudHSM 클라이언트 SDK 3 및 클라이언트 SDK 5를 통해 Java Keytool 및 Jarsigner 유틸리티와의 통합을 제공합니다. 이러한 도구를 사용하는 단계는 현재 다운로드한 클라이언트 SDK의 버전에 따라 달라집니다.

- 클라이언트 SDK 5를 사용하여 Java Keytool 및 Jarsigner와 통합
- 클라이언트 SDK 3을 사용하여 Java Keytool 및 Jarsigner와 통합

클라이언트 SDK 5를 사용하여 Java Keytool 및 Jarsigner와 통합

AWS CloudHSM 키 저장소는 및 와 같은 타사 도구를 통해 HSM의 키와 연결된 인증서를 활용하는 특수 목적의 JCE 키 저장소입니다. `keytool` `jarsigner` AWS CloudHSM 인증서는 공개 비기밀 데이터이므로 HSM에 인증서를 저장하지 않습니다. AWS CloudHSM 키 스토어는 인증서를 로컬 파일에 저장하고 인증서를 HSM의 해당 키에 매핑합니다.

AWS CloudHSM 키 스토어를 사용하여 새 키를 생성하면 로컬 키 스토어 파일에 항목이 생성되지 않고 HSM에 키가 생성됩니다. 이와 비슷하게, AWS CloudHSM 키 스토어를 사용하여 키를 검색할 때 검색이 HSM에 전달됩니다. 인증서를 AWS CloudHSM 키 저장소에 저장하면 공급자가 해당 별칭이 있는 키 쌍이 HSM에 있는지 확인한 다음 제공된 인증서를 해당 키 쌍과 연결합니다.

주제

필수 조건

AWS CloudHSM 키 스토어를 사용하려면 먼저 JCE SDK를 초기화하고 구성해야 합니다. AWS CloudHSM

1단계: JCE 설치

AWS CloudHSM 클라이언트 사전 요구 사항을 포함하여 JCE를 설치하려면 Java 라이브러리 설치 단계를 따르십시오.

2단계: 환경 변수에 HSM 로그인 자격 증명 추가

HSM 로그인 자격 증명을 포함하도록 환경 변수를 설정합니다.

Linux

```
$ export HSM_USER=<HSM user name>
```

```
$ export HSM_PASSWORD=<HSM password>
```

Windows

```
PS C:\> $Env:HSM_USER=<HSM user name>
```

```
PS C:\> $Env:HSM_PASSWORD=<HSM password>
```

> **Note**
>
> AWS CloudHSM JCE는 다양한 로그인 옵션을 제공합니다. 타사 애플리케이션에서 AWS CloudHSM 키 저장소를 사용하려면 환경 변수와 함께 암시적 로그인을 사용해야 합니다. 애플

리케이션 코드를 통한 명시적 로그인을 사용하려면 AWS CloudHSM 키 스토어를 사용하여 자체 애플리케이션을 빌드해야 합니다. 자세한 내용은 AWS CloudHSM 키 스토어 사용 문서를 참조하십시오.

3 단계: JCE 공급자 등록

Java CloudProvider 구성에서 JCE 제공자를 등록하려면 다음 단계를 따르십시오.

1. Java 설치에서 `java.security` 구성 파일을 열어 편집합니다.
2. `java.security` 구성 파일에서 `com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider`을 마지막 공급자로 추가합니다. 예를 들어, `java.security` 파일에 9개의 공급자가 있는 경우 섹션의 마지막 공급자로 다음 공급자를 추가합니다.

   ```
   security.provider.10=com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider
   ```

> **Note**
>
> AWS CloudHSM 제공자를 더 높은 우선 순위로 추가하면 소프트웨어로 안전하게 오프로드될 수 있는 작업에 대해 AWS CloudHSM 제공자의 우선 순위가 지정되므로 시스템 성능에 부정적인 영향을 미칠 수 있습니다. 가장 좋은 방법은 운영 공급자이든 소프트웨어 기반 공급자이든 관계없이 항상 작업에 사용할 공급자를 지정하는 것입니다. AWS CloudHSM

> **Note**
>
> AWS CloudHSM 키 스토어와 함께 keytool을 사용하여 키를 생성할 때 `-providerName`, `-providerclass` 및 `-providerpath` 명령줄 옵션을 지정하면 오류가 발생할 수 있습니다.

keytool과 함께 AWS CloudHSM 키 스토어 사용하기

Keytool은 공통 키 및 인증서 태스크를 위한 널리 사용되는 명령줄 유틸리티입니다. Keytool에 대한 전체 자습서는 AWS CloudHSM 설명서의 범위를 벗어납니다. 이 문서에서는 키 저장소를 통해 신뢰 AWS CloudHSM 루트로 활용할 때 다양한 키 도구 함수와 함께 사용해야 하는 특정 파라미터에 대해 AWS CloudHSM 설명합니다.

키 저장소와 함께 keytool을 AWS CloudHSM 사용하는 경우 모든 keytool 명령에 다음 인수를 지정하십시오.

Linux

```
-storetype CLOUDHSM -J-classpath< '-J/opt/cloudhsm/java/*'>
```

Windows

```
-storetype CLOUDHSM -J-classpath<'-J"C:\Program Files\Amazon\CloudHSM\java\*"'>
```

키 저장소를 사용하여 AWS CloudHSM 새 키 저장소 파일을 생성하려면 을 참조하십시오. 사용 AWS CloudHSM KeyStore 기존 키 스토어를 사용하려면 keytool에 대한 keystore 인수를 사용하여 이름(경로 포함)을 지정합니다. keytool 명령에 존재하지 않는 키 스토어 파일을 지정하면 키 스토어가 새 AWS CloudHSM 키 스토어 파일을 생성합니다.

Keytool을 사용하여 새 키 생성

Keytool을 사용하여 AWS CloudHSM의 JCE SDK에서 지원하는 RSA, AES 및 DESede 유형의 키를 생성할 수 있습니다.

> **Important**
>
> keytool을 통해 생성된 키는 소프트웨어에서 생성된 다음 추출 AWS CloudHSM 가능한 영구 키로 가져옵니다.

Keytool 외부에서 내보낼 수 없는 키를 생성한 다음 해당 인증서를 키 스토어로 가져오는 것이 좋습니다. keytool과 Jarsigner를 통해 추출 가능한 RSA 또는 EC 키를 사용하는 경우 제공자는 에서 키를 내보낸 다음 로컬에서 키를 서명 AWS CloudHSM 작업에 사용합니다.

AWS CloudHSM 클러스터에 연결된 클라이언트 인스턴스가 여러 개 있는 경우 한 클라이언트 인스턴스의 키 저장소에서 인증서를 가져와도 다른 클라이언트 인스턴스에서 인증서를 자동으로 사용할 수 있는 것은 아니라는 점에 유의하십시오. 각 클라이언트 인스턴스에서 키 및 관련 인증서를 등록하려면 the section called “Keytool을 사용하여 CSR 생성하기”에서 설명한 대로 Java 애플리케이션을 실행해야 합니다. 또는, 한 클라이언트에서 필요한 사항을 변경하고 결과 키 스토어 파일을 다른 모든 클라이언트 인스턴스에 복사할 수 있습니다.

예제 1: 레이블이 인 대칭 AES-256 키를 생성하고 작업 디렉터리에서 이름이 "my_keystore.store"인 키 스토어 파일에 저장하려면. *<secret label>*을 고유한 레이블로 바꾸십시오.

Linux

```
$ keytool -genseckey -alias <secret label> -keyalg aes \
 -keysize 256 -keystore my_keystore.store \
 -storetype CloudHSM -J-classpath '-J/opt/cloudhsm/java/*' \
```

Windows

```
PS C:\> keytool -genseckey -alias <secret label> -keyalg aes `
 -keysize 256 -keystore my_keystore.store `
 -storetype CloudHSM -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

예제 2: 레이블이 인 RSA 2048 키 페어를 생성하고 작업 디렉터리에서 이름이 “my_keystore.store”인 키 스토어 파일에 저장하려면. <RSA key pair label>고유 라벨로 교체하십시오.

Linux

```
$ keytool -genkeypair -alias <RSA key pair label> \
 -keyalg rsa -keysize 2048 \
 -sigalg sha512withrsa \
 -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -genkeypair -alias <RSA key pair label> `
 -keyalg rsa -keysize 2048 `
 -sigalg sha512withrsa `
 -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Java 라이브러리에서 지원되는 서명 알고리즘의 목록을 찾을 수 있습니다.

Keytool을 사용하여 키 삭제하기

AWS CloudHSM 키 스토어는 키 삭제를 지원하지 않습니다. Destroyable 인터페이스의 destroy 메서드를 사용하여 키를 삭제할 수 있습니다.

```
((Destroyable) key).destroy();
```

Keytool을 사용하여 CSR 생성하기

OpenSSL Dynamic Engine를 사용하는 경우 가장 유연하게 인증서 서명 요청(CSR)을 생성할 수 있습니다. 다음 명령은 keytool을 사용하여 별칭이 있는 키 페어, `my-key-pair`에 대한 CSR을 생성합니다.

Linux

```
$ keytool -certreq -alias <key pair label> \
 -file my_csr.csr \
 -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -certreq -alias <key pair label> `
 -file my_csr.csr `
 -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

> **Note**
>
> keytool의 키 페어를 사용하려면 해당 키 페어의 지정된 키 스토어 파일에 항목이 있어야 합니다. keytool 외부에서 생성된 키 페어를 사용하려면 키 및 인증서 메타데이터를 키 스토어로 가져와야 합니다. keystore 데이터를 가져오는 방법에 대한 지침은 the section called “keytool을 사용하여 중간 및 루트 인증서를 AWS CloudHSM 키 저장소로 가져오기 ”을 참조하십시오.

keytool을 사용하여 중간 및 루트 인증서를 AWS CloudHSM 키 저장소로 가져오기

CA 인증서를 가져오려면 새로 가져온 인증서에서 전체 인증서 체인의 검증을 활성화해야 합니다. 다음 명령은 예시를 나타냅니다.

Linux

```
$ keytool -import -trustcacerts -alias rootCAcert \
 -file rootCAcert.cert -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -import -trustcacerts -alias rootCAcert `
 -file rootCAcert.cert -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

AWS CloudHSM 클러스터에 여러 클라이언트 인스턴스를 연결하는 경우 한 클라이언트 인스턴스의 키 스토어에 있는 인증서를 가져와도 다른 클라이언트 인스턴스에서 인증서를 자동으로 사용할 수 없게 됩니다. 각 클라이언트 인스턴스에서 인증서를 가져와야 합니다.

keytool을 사용하여 키 저장소에서 AWS CloudHSM 인증서를 삭제합니다.

다음 명령은 Java keytool 키 스토어에서 인증서를 삭제하는 방법의 예를 보여줍니다.

Linux

```
$ keytool -delete -alias mydomain \
 -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -delete -alias mydomain `
 -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

AWS CloudHSM 클러스터에 여러 클라이언트 인스턴스를 연결하는 경우 한 클라이언트 인스턴스의 키 스토어에서 인증서를 삭제해도 다른 클라이언트 인스턴스에서 인증서가 자동으로 제거되지 않습니다. 각 클라이언트 인스턴스에서 인증서를 삭제해야 합니다.

keytool을 사용하여 작동 중인 인증서를 AWS CloudHSM 키 저장소로 가져오기

인증서 서명 요청(CSR)이 서명되면, 이를 AWS CloudHSM 키 스토어로 가져와서 적절한 키 페어와 연결할 수 있습니다. 다음 명령은 예시를 제공합니다.

Linux

```
$ keytool -importcert -noprompt -alias <key pair label> \
 -file my_certificate.crt \
 -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -importcert -noprompt -alias <key pair label> `
 -file my_certificate.crt `
 -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

별칭은 키 스토어에 연결된 인증서가 있는 키 페어여야 합니다. 키가 keytool 외부에서 생성되거나 다른 클라이언트 인스턴스에서 생성된 경우, 먼저 키 및 인증서 메타데이터를 키 스토어로 가져오기 해야 합니다.

인증서 체인은 검증 가능해야 합니다. 인증서를 검증할 수 없는 경우, 체인을 검증할 수 있도록 서명(인증 기관) 인증서를 키 스토어로 가져와야 할 수 있습니다.

Keytool을 사용하여 인증서 내보내기

다음 예에서는 이진수 X.509 형식으로 인증서를 생성합니다. 사람이 읽을 수 있는 인증서를 내보내려면 `-rfc`를 `-exportcert` 명령에 추가합니다.

Linux

```
$ keytool -exportcert -alias <key pair label> \
```

```
 -file my_exported_certificate.crt \
 -keystore my_keystore.store \
 -storetype CLOUDHSM \
 -J-classpath '-J/opt/cloudhsm/java/*'
```

Windows

```
PS C:\> keytool -exportcert -alias <key pair label> `
 -file my_exported_certificate.crt `
 -keystore my_keystore.store `
 -storetype CLOUDHSM `
 -J-classpath '-J"C:\Program Files\Amazon\CloudHSM\java\*"'
```

Jarsigner와 함께 AWS CloudHSM 키 스토어 사용

Jarsigner는 HSM에 안전하게 저장된 키를 사용하여 JAR 파일에 서명하는 데 널리 사용되는 명령줄 유틸리티입니다. Jarsigner에 대한 전체 자습서는 AWS CloudHSM 설명서의 범위를 벗어납니다. 이 섹션에서는 키 저장소를 통해 신뢰 AWS CloudHSM 루트로 사용하여 서명을 서명하고 확인하는 데 사용해야 하는 Jarsigner 매개 변수에 대해 설명합니다. AWS CloudHSM

키 및 인증서 설정

Jarsigner를 사용하여 JAR 파일에 서명하기 전에 다음 단계를 설정 또는 완료했는지 확인하십시오.

1. AWS CloudHSM 키 스토어 사전 조건의 지침을 따릅니다.
2. 현재 서버 또는 클라이언트 인스턴스의 키 저장소에 저장해야 하는 서명 AWS CloudHSM 키와 관련 인증서 및 인증서 체인을 설정합니다. 에서 키를 생성한 다음 관련 메타데이터를 AWS CloudHSM 키 스토어로 가져옵니다. AWS CloudHSM keytool을 사용하여 키와 인증서를 설정하려면 the section called “Keytool을 사용하여 새 키 생성” 단원을 참조하십시오. 여러 클라이언트 인스턴스를 사용하여 JAR에 서명하는 경우 키를 만들고 인증서 체인을 가져옵니다. 그런 다음 결과 키 스토어 파일을 각 클라이언트 인스턴스에 복사합니다. 새 키를 자주 생성하는 경우 인증서를 각 클라이언트 인스턴스에 개별적으로 가져오는 것이 더 쉬울 수 있습니다.
3. 전체 인증서 체인이 검증 가능해야 합니다. 인증서 체인을 확인할 수 있으려면 CA 인증서와 중간 인증서를 AWS CloudHSM 키 저장소에 추가해야 할 수 있습니다. Java 코드를 사용하여 인증서 체인을 확인하는 지침은 the section called “AWS CloudHSM 및 Jarsigner를 사용하여 JAR 파일에 서명”의 코드 조각을 참조하십시오. 원하는 경우 keytool을 사용하여 인증서를 가져올 수 있습니다. keytool을 사용하는 지침은 the section called “keytool을 사용하여 중간 및 루트 인증서를 AWS CloudHSM 키 저장소로 가져오기 ”을 참조하십시오.

AWS CloudHSM 및 Jarsigner를 사용하여 JAR 파일에 서명

JAR 파일에 서명하려면 다음 명령을 사용합니다.

Linux;

OpenJDK 8의 경우

```
jarsigner -keystore my_keystore.store \
 -signedjar signthisclass_signed.jar \
 -sigalg sha512withrsa \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass.jar <key pair label>
```

OpenJDK 11, OpenJDK 17 및 OpenJDK 21의 경우

```
jarsigner -keystore my_keystore.store \
 -signedjar signthisclass_signed.jar \
 -sigalg sha512withrsa \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass.jar <key pair label>
```

Windows

OpenJDK8의 경우

```
jarsigner -keystore my_keystore.store `
 -signedjar signthisclass_signed.jar `
 -sigalg sha512withrsa `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*;C:\Program Files\Java
\jdk1.8.0_331\lib\tools.jar' `
  "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass.jar <key pair label>
```

OpenJDK 11, OpenJDK 17 및 OpenJDK 21의 경우

```
jarsigner -keystore my_keystore.store `
 -signedjar signthisclass_signed.jar `
 -sigalg sha512withrsa `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*'`
  "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass.jar <key pair label>
```

서명된 JAR을 검증하려면 다음 명령을 사용합니다.

Linux

OpenJDK8의 경우

```
jarsigner -verify \
 -keystore my_keystore.store \
 -sigalg sha512withrsa \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass_signed.jar <key pair label>
```

OpenJDK 11, OpenJDK 17 및 OpenJDK 21의 경우

```
jarsigner -verify \
 -keystore my_keystore.store \
 -sigalg sha512withrsa \
 -storetype CloudHSM \
 -J-classpath '-J/opt/cloudhsm/java/*' \
 -J-Djava.library.path=/opt/cloudhsm/lib \
 signthisclass_signed.jar <key pair label>
```

Windows

OpenJDK 8의 경우

```
jarsigner -verify `
```

```
 -keystore my_keystore.store `
 -sigalg sha512withrsa `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*;C:\Program Files\Java
\jdk1.8.0_331\lib\tools.jar' `
 "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass_signed.jar <key pair label>
```

OpenJDK 11, OpenJDK 17 및 OpenJDK 21의 경우

```
jarsigner -verify `
 -keystore my_keystore.store `
 -sigalg sha512withrsa `
 -storetype CloudHSM `
 -J-classpath '-JC:\Program Files\Amazon\CloudHSM\java\*`
 "-J-Djava.library.path='C:\Program Files\Amazon\CloudHSM\lib\'" `
 signthisclass_signed.jar <key pair label>
```

알려진 문제

1. Keytool 및 Jarsigner에서는 EC 키를 지원하지 않습니다.

클라이언트 SDK 3을 사용하여 Java Keytool 및 Jarsigner와 통합

AWS CloudHSM 키 저장소는 및 와 같은 타사 도구를 통해 HSM의 키와 연결된 인증서를 활용하는 특수 목적의 JCE 키 저장소입니다. `keytool` `jarsigner` AWS CloudHSM 인증서는 기밀이 아닌 공개 데이터이므로 HSM에 인증서를 저장하지 않습니다. AWS CloudHSM 키 스토어는 인증서를 로컬 파일에 저장하고 인증서를 HSM의 해당 키에 매핑합니다.

AWS CloudHSM 키 스토어를 사용하여 새 키를 생성하면 로컬 키 스토어 파일에 항목이 생성되지 않고 HSM에 키가 생성됩니다. 이와 비슷하게, AWS CloudHSM 키 스토어를 사용하여 키를 검색할 때 검색이 HSM에 전달됩니다. 인증서를 AWS CloudHSM 키 저장소에 저장하면 공급자가 해당 별칭이 있는 키 쌍이 HSM에 있는지 확인한 다음 제공된 인증서를 해당 키 쌍과 연결합니다.

주제

- 필수 조건

- AWS CloudHSM keytool과 함께 키 스토어 사용
- jarsigner와 함께 AWS CloudHSM 키 스토어 사용
- 알려진 문제
- 키 스토어에 기존 키 등록 AWS CloudHSM

필수 조건

AWS CloudHSM 키 스토어를 사용하려면 먼저 JCE SDK를 초기화하고 구성해야 합니다. AWS CloudHSM

1단계: JCE 설치

AWS CloudHSM 클라이언트 사전 요구 사항을 포함하여 JCE를 설치하려면 Java 라이브러리 설치 단계를 따르십시오.

2단계: 환경 변수에 HSM 로그인 자격 증명 추가

HSM 로그인 자격 증명을 포함하도록 환경 변수를 설정합니다.

```
export HSM_PARTITION=PARTITION_1
export HSM_USER=<HSM user name>
export HSM_PASSWORD=<HSM password>
```

> **Note**
>
> CloudHSM JCE는 다양한 로그인 옵션을 제공합니다. 타사 애플리케이션에서 AWS CloudHSM 키 저장소를 사용하려면 환경 변수와 함께 암시적 로그인을 사용해야 합니다. 애플리케이션 코드를 통한 명시적 로그인을 사용하려면 AWS CloudHSM 키 스토어를 사용하여 자체 애플리케이션을 빌드해야 합니다. 자세한 내용은 AWS CloudHSM 키 스토어 사용 문서를 참조하십시오.

3 단계: JCE 공급자 등록

Java CloudProvider 구성에서 JCE 제공자를 등록하려면

1. 편집을 위해, Java 설치에서 java.security 구성 파일을 엽니다.

2. java.security 구성 파일에서 마지막 공급자로 `com.cavium.provider.CaviumProvider`를 추가합니다. 예를 들어, java.security 파일에 9개의 공급자가 있는 경우 섹션의 마지막 공급자로 다음 공급자를 추가합니다. Cavium 공급자를 더 높은 우선 순위로 추가하면 시스템의 성능에 부정적인 영향을 줄 수 있습니다.

   ```
   security.provider.10=com.cavium.provider.CaviumProvider
   ```

   > **Note**
   >
   > 파워 유저는 보안 구성 파일을 업데이트하는 대신, keytool 사용 시 `-providerName`, `-providerclass` 및 `-providerpath` 명령줄 옵션을 지정하는 데 익숙할 수 있습니다. 키 저장소로 AWS CloudHSM 키를 생성할 때 명령줄 옵션을 지정하려고 하면 오류가 발생합니다.

AWS CloudHSM keytool과 함께 키 스토어 사용

Keytool은 Linux 시스템에서 공통 키 및 인증서 작업을 위한 널리 사용되는 명령줄 유틸리티입니다. Keytool에 대한 전체 자습서는 AWS CloudHSM 설명서의 범위를 벗어납니다. 이 문서에서는 키 저장소를 통해 신뢰 AWS CloudHSM 루트로 활용할 때 다양한 키 도구 함수와 함께 사용해야 하는 특정 파라미터에 대해 AWS CloudHSM 설명합니다.

키 저장소와 함께 keytool을 AWS CloudHSM 사용하는 경우 모든 keytool 명령에 다음 인수를 지정하십시오.

```
-storetype CLOUDHSM \
  -J-classpath '-J/opt/cloudhsm/java/*' \
  -J-Djava.library.path=/opt/cloudhsm/lib
```

키 저장소를 사용하여 AWS CloudHSM 새 키 저장소 파일을 생성하려면 을 참조하십시오. 사용 AWS CloudHSM KeyStore 기존 키 스토어를 사용하려면 keytool에 대한 keystore 인수를 사용하여 이름(경로 포함)을 지정합니다. keytool 명령에 존재하지 않는 키 스토어 파일을 지정하면 키 스토어가 새 AWS CloudHSM 키 스토어 파일을 생성합니다.

Keytool을 사용하여 새 키 생성

keytool을 사용하여 의 JCE SDK가 지원하는 AWS CloudHSM모든 유형의 키를 생성할 수 있습니다. Java 라이브러리의 지원되는 키 문서에서 키 및 길이의 전체 목록을 참조하십시오.

> **Important**
>
> keytool을 통해 생성된 키는 소프트웨어에서 생성된 다음 추출 AWS CloudHSM 가능한 영구 키로 가져옵니다.

추출할 수 없는 키를 HSM에서 직접 생성한 다음 keytool 또는 Jarsigner와 함께 사용하는 방법에 대한 지침은 키 스토어에 기존 키 등록의 코드 샘플에 나와 있습니다. AWS CloudHSM Keytool 외부에서 내보낼 수 없는 키를 생성한 다음 해당 인증서를 키 스토어로 가져오는 것이 좋습니다. keytool과 jarsigner를 통해 추출 가능한 RSA 또는 EC 키를 사용하는 경우 제공자는 에서 키를 내보낸 다음 로컬에서 이 키를 서명 작업에 사용합니다. AWS CloudHSM

CloudHSM 클러스터에 연결된 클라이언트 인스턴스가 여러 개 있는 경우, 한 클라이언트 인스턴스의 키 스토어에서 인증서를 가져오더라도 다른 클라이언트 인스턴스에서 자동으로 인증서를 사용할 수 없다는 점을 알아두십시오. 각 클라이언트 인스턴스에서 키 및 연결된 인증서를 등록하려면 Keytool을 사용하여 CSR 생성에 설명된 대로 Java 애플리케이션을 실행해야 합니다. 또는, 한 클라이언트에서 필요한 사항을 변경하고 결과 키 스토어 파일을 다른 모든 클라이언트 인스턴스에 복사할 수 있습니다.

예제 1: 레이블이 인 대칭 AES-256 키를 생성하고 작업 디렉터리에서 이름이 "my_keystore.store"인 키 스토어 파일에 저장하려면. *<secret label>*을 고유한 레이블로 바꾸십시오.

```
keytool -genseckey -alias <secret label> -keyalg aes \
  -keysize 256 -keystore my_keystore.store \
  -storetype CloudHSM -J-classpath '-J/opt/cloudhsm/java/*' \
  -J-Djava.library.path=/opt/cloudhsm/lib/
```

예제 2: 레이블이 인 RSA 2048 키 페어를 생성하고 작업 디렉터리에서 이름이 “my_keystore.store”인 키 스토어 파일에 저장하려면. <RSA key pair label>고유 라벨로 교체하십시오.

```
keytool -genkeypair -alias <RSA key pair label> \
        -keyalg rsa -keysize 2048 \
        -sigalg sha512withrsa \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

예제 3: 레이블이 인 p256 ED 키를 생성하고 작업 디렉터리에서 이름이 “my_keystore.store”인 키 스토어 파일에 저장하려면. <ec key pair label>고유 라벨로 교체하십시오.

```
keytool -genkeypair -alias <ec key pair label> \
        -keyalg ec -keysize 256 \
        -sigalg SHA512withECDSA \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Java 라이브러리에서 지원되는 서명 알고리즘의 목록을 찾을 수 있습니다.

Keytool을 사용하여 키 삭제하기

AWS CloudHSM 키 스토어는 키 삭제를 지원하지 않습니다. 키를 삭제하려면 AWS CloudHSM의 명령줄 도구인 `deleteKey` 기능을 사용해야 deleteKey 합니다.

Keytool을 사용하여 CSR 생성하기

OpenSSL Dynamic Engine를 사용하는 경우 가장 유연하게 인증서 서명 요청(CSR)을 생성할 수 있습니다. 다음 명령은 keytool을 사용하여 별칭이 있는 키 페어, `my-key-pair`에 대한 CSR을 생성합니다.

```
keytool -certreq -alias <key pair label> \
        -file my_csr.csr \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

> **Note**
>
> keytool의 키 페어를 사용하려면 해당 키 페어의 지정된 키 스토어 파일에 항목이 있어야 합니다. keytool 외부에서 생성된 키 페어를 사용하려면 키 및 인증서 메타데이터를 키 스토어로 가져와야 합니다. 키스토어 데이터를 가져오는 방법에 대한 지침은 Keytool을 사용하여 중간 및 루트 인증서를 AWS CloudHSM 키 스토어로 가져오기를 참조하십시오.

keytool을 사용하여 중간 및 루트 인증서를 키 저장소로 가져오기 AWS CloudHSM

CA 인증서를 가져오려면 새로 가져온 인증서에서 전체 인증서 체인의 검증을 활성화해야 합니다. 다음 명령은 예시를 나타냅니다.

```
keytool -import -trustcacerts -alias rootCAcert \
        -file rootCAcert.cert -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

AWS CloudHSM 클러스터에 여러 클라이언트 인스턴스를 연결하는 경우 한 클라이언트 인스턴스의 키 스토어에 있는 인증서를 가져와도 다른 클라이언트 인스턴스에서 인증서를 자동으로 사용할 수 없게 됩니다. 각 클라이언트 인스턴스에서 인증서를 가져와야 합니다.

keytool을 사용하여 키 저장소에서 AWS CloudHSM 인증서를 삭제합니다.

다음 명령은 Java keytool 키 스토어에서 인증서를 삭제하는 방법의 예를 보여줍니다.

```
keytool -delete -alias mydomain -keystore \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

AWS CloudHSM 클러스터에 여러 클라이언트 인스턴스를 연결하는 경우 한 클라이언트 인스턴스의 키 스토어에서 인증서를 삭제해도 다른 클라이언트 인스턴스에서 인증서가 자동으로 제거되지 않습니다. 각 클라이언트 인스턴스에서 인증서를 삭제해야 합니다.

keytool을 사용하여 작동 중인 인증서를 AWS CloudHSM 키 저장소로 가져오기

인증서 서명 요청(CSR)이 서명되면, 이를 AWS CloudHSM 키 스토어로 가져와서 적절한 키 페어와 연결할 수 있습니다. 다음 명령은 예시를 제공합니다.

```
keytool -importcert -noprompt -alias <key pair label> \
        -file my_certificate.crt \
        -keystore my_keystore.store
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

별칭은 키 스토어에 연결된 인증서가 있는 키 페어여야 합니다. 키가 keytool 외부에서 생성되거나 다른 클라이언트 인스턴스에서 생성된 경우, 먼저 키 및 인증서 메타데이터를 키 스토어로 가져오기 해야 합니다. 인증서 메타데이터를 가져오는 방법에 대한 지침은 키 스토어에 기존 키 등록의 코드 샘플을 참조하십시오. AWS CloudHSM

인증서 체인은 검증 가능해야 합니다. 인증서를 검증할 수 없는 경우, 체인을 검증할 수 있도록 서명(인증 기관) 인증서를 키 스토어로 가져와야 할 수 있습니다.

Keytool을 사용하여 인증서 내보내기

다음 예에서는 이진수 X.509 형식으로 인증서를 생성합니다. 사람이 읽을 수 있는 인증서를 내보내려면 `-rfc`를 `-exportcert` 명령에 추가합니다.

```
keytool -exportcert -alias <key pair label> \
        -file my_exported_certificate.crt \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

jarsigner와 함께 AWS CloudHSM 키 스토어 사용

Jarsigner는 HSM에 안전하게 저장된 키를 사용하여 JAR 파일에 서명하는 데 널리 사용되는 명령줄 유틸리티입니다. Jarsigner에 대한 전체 자습서는 AWS CloudHSM 설명서의 범위를 벗어납니다. 이 섹션에서는 키 저장소를 통해 신뢰 AWS CloudHSM 루트로 사용하여 서명을 서명하고 확인하는 데 사용해야 하는 Jarsigner 매개 변수에 대해 설명합니다. AWS CloudHSM

키 및 인증서 설정

Jarsigner를 사용하여 JAR 파일에 서명하기 전에 다음 단계를 설정 또는 완료했는지 확인하십시오.

1. AWS CloudHSM 키 스토어 사전 조건의 지침을 따릅니다.
2. 현재 서버 또는 클라이언트 인스턴스의 키 저장소에 저장해야 하는 서명 AWS CloudHSM 키와 관련 인증서 및 인증서 체인을 설정합니다. 에서 키를 생성한 다음 관련 메타데이터를 AWS CloudHSM 키 스토어로 가져옵니다. AWS CloudHSM 키 스토어에 기존 키 등록의 코드 샘플을 사용하여 메타데이터를 AWS CloudHSM 키 스토어로 가져올 수 있습니다. keytool을 사용하여 키와 인증서를 설정하려면 Keytool을 사용하여 새 키 생성 단원을 참조하십시오. 여러 클라이언트 인스턴스를 사용하여 JAR에 서명하는 경우 키를 만들고 인증서 체인을 가져옵니다. 그런 다음 결과 키 스토어 파일을 각 클라이언트 인스턴스에 복사합니다. 새 키를 자주 생성하는 경우 인증서를 각 클라이언트 인스턴스에 개별적으로 가져오는 것이 더 쉬울 수 있습니다.
3. 전체 인증서 체인이 검증 가능해야 합니다. 인증서 체인을 확인할 수 있으려면 CA 인증서와 중간 인증서를 키 저장소에 추가해야 할 수 있습니다. AWS CloudHSM Java 코드를 사용하여 인증서 체인을 확인하는 방법에 대한 지침은 AWS CloudHSM 및 Jarsigner를 사용하여 JAR 파일에 서명하기의 코드 스니펫을 참조하십시오. 원하는 경우 keytool을 사용하여 인증서를 가져올 수 있습니다.

keytool을 사용하는 방법에 대한 지침은 Keytool을 사용하여 중간 및 루트 인증서를 키 저장소로 가져오기를 참조하십시오. AWS CloudHSM

AWS CloudHSM 및 jarsigner를 사용하여 JAR 파일에 서명

JAR 파일에 서명하려면 다음 명령을 사용합니다.

```
jarsigner -keystore my_keystore.store \
        -signedjar signthisclass_signed.jar \
        -sigalg sha512withrsa \
        -storetype CloudHSM \
        -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \
        -J-Djava.library.path=/opt/cloudhsm/lib \
        signthisclass.jar <key pair label>
```

서명된 JAR을 검증하려면 다음 명령을 사용합니다.

```
jarsigner -verify \
        -keystore my_keystore.store \
        -sigalg sha512withrsa \
        -storetype CloudHSM \
        -J-classpath '-J/opt/cloudhsm/java/*:/usr/lib/jvm/java-1.8.0/lib/tools.jar' \
        -J-Djava.library.path=/opt/cloudhsm/lib \
        signthisclass_signed.jar <key pair label>
```

알려진 문제

다음 목록은 알려진 문제의 현재 목록을 제공합니다.

- keytool을 사용하여 키를 생성할 때 제공자 구성의 첫 번째 제공자는 생성할 수 없습니다. CaviumProvider
- keytool을 사용하여 키를 생성할 때, 보안 구성 파일의 첫 번째 (지원되는) 공급자가 키를 생성하는 데 사용됩니다. 이러한 공급자는 일반적으로 소프트웨어 공급자입니다. 그러면 생성된 키에 별칭이 지정되고 키 추가 프로세스 중에 영구 (토큰) 키로 AWS CloudHSM HSM에 가져옵니다.
- 키 저장소와 함께 AWS CloudHSM keytool을 사용하는 경우 명령줄에 `-providerName-providerclass`, 또는 `-providerpath` 옵션을 지정하지 마십시오. 키 스토어 사전 조건에 설명된 대로 보안 공급자 파일에서 이러한 옵션을 지정합니다.
- keytool 및 Jarsigner를 통해 추출할 수 없는 EC 키를 사용하는 경우 SunEC 공급자를 java.security 파일의 공급자 목록에서 제거/비활성화해야합니다. keytool과 Jarsigner를 통해 추출 가능한 EC 키를

사용하는 경우 공급자는 AWS CloudHSM HSM에서 키 비트를 내보내고 서명 작업에 로컬에서 키를 사용합니다. keytool 또는 Jarsigner와 함께 내보낼 수 있는 키를 사용하지 않는 것이 좋습니다.

키 스토어에 기존 키 등록 AWS CloudHSM

속성 및 레이블 지정에서 최대의 보안성 및 유연성을 위해 key_mgmt_util을 사용하여 서명 키를 생성하는 것이 좋습니다. 또한 Java 애플리케이션을 사용하여 AWS CloudHSM에서 키를 생성할 수도 있습니다.

다음 섹션에서는 HSM에서 새 키 쌍을 생성하고 키 스토어로 가져온 기존 키를 사용하여 키 쌍을 등록하는 방법을 보여주는 코드 샘플을 제공합니다. AWS CloudHSM 가져온 키는 keytool 및 Jarsigner와 같은 타사 도구와 함께 사용할 수 있습니다.

기존 키를 사용하려면, 새 키를 생성하는 대신 레이블별로 키를 조회하도록 코드 샘플을 수정합니다. 레이블별로 키를 조회하기 위한 샘플 KeyUtilitiesRunner코드는.java 샘플에서 확인할 수 있습니다. GitHub

> **Important**
>
> 저장된 키를 로컬 키 스토어에 등록해도 키는 내보내지지 않습니다. AWS CloudHSM 키가 등록되면, 키 스토어는 키의 별칭(또는 레이블)을 등록하고 스토어 인증서 객체를 AWS CloudHSM의 키 페어와 로컬로 상호 연관시킵니다. 키 페어가 내보낼 수 없도록 생성되는 한, 키 비트는 HSM을 떠나지 않습니다.

```
                    //
// Copyright 2018 Amazon.com, Inc. or its affiliates. All Rights Reserved.
//
// Permission is hereby granted, free of charge, to any person obtaining a copy of
this
// software and associated documentation files (the "Software"), to deal in the
Software
// without restriction, including without limitation the rights to use, copy, modify,
// merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
// permit persons to whom the Software is furnished to do so.
//
// THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
```

```
 // INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
 // PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
 // HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
 // OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
 // SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
 //

package com.amazonaws.cloudhsm.examples;

import com.cavium.key.CaviumKey;
import com.cavium.key.parameter.CaviumAESKeyGenParameterSpec;
import com.cavium.key.parameter.CaviumRSAKeyGenParameterSpec;
import com.cavium.asn1.Encoder;
import com.cavium.cfm2.Util;

import javax.crypto.KeyGenerator;

import java.io.ByteArrayInputStream;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;

import java.math.BigInteger;

import java.security.*;
import java.security.cert.Certificate;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;
import java.security.interfaces.RSAPrivateKey;
import java.security.interfaces.RSAPublicKey;
import java.security.KeyStore.PasswordProtection;
import java.security.KeyStore.PrivateKeyEntry;
import java.security.KeyStore.Entry;

import java.util.Calendar;
import java.util.Date;
import java.util.Enumeration;

//
// KeyStoreExampleRunner demonstrates how to load a keystore, and associate a
 certificate with a
// key in that keystore.
//
```

```
// This example relies on implicit credentials, so you must setup your environment
 correctly.
//
// https://docs.aws.amazon.com/cloudhsm/latest/userguide/java-library-
install.html#java-library-credentials
//

public class KeyStoreExampleRunner {

    private static byte[] COMMON_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
 (byte) 0x03 };
    private static byte[] COUNTRY_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
 (byte) 0x06 };
    private static byte[] LOCALITY_NAME_OID = new byte[] { (byte) 0x55, (byte) 0x04,
 (byte) 0x07 };
    private static byte[] STATE_OR_PROVINCE_NAME_OID = new byte[] { (byte) 0x55,
 (byte) 0x04, (byte) 0x08 };
    private static byte[] ORGANIZATION_NAME_OID = new byte[] { (byte) 0x55, (byte)
 0x04, (byte) 0x0A };
    private static byte[] ORGANIZATION_UNIT_OID = new byte[] { (byte) 0x55, (byte)
 0x04, (byte) 0x0B };

    private static String helpString = "KeyStoreExampleRunner%n" +
           "This sample demonstrates how to load and store keys using a keystore.%n%n"
 +
           "Options%n" +
           "\t--help\t\t\tDisplay this message.%n" +
           "\t--store <filename>\t\tPath of the keystore.%n" +
           "\t--password <password>\t\tPassword for the keystore (not your CU
 password).%n" +
           "\t--label <label>\t\t\tLabel to store the key and certificate under.%n" +
           "\t--list\t\t\tList all the keys in the keystore.%n%n";

   public static void main(String[] args) throws Exception {
       Security.addProvider(new com.cavium.provider.CaviumProvider());
       KeyStore keyStore = KeyStore.getInstance("CloudHSM");

       String keystoreFile = null;
       String password = null;
       String label = null;
       boolean list = false;
       for (int i = 0; i < args.length; i++) {
           String arg = args[i];
           switch (args[i]) {
```

```
                case "--store":
                    keystoreFile = args[++i];
                    break;
                case "--password":
                    password = args[++i];
                    break;
                case "--label":
                    label = args[++i];
                    break;
                case "--list":
                    list = true;
                    break;
                case "--help":
                    help();
                    return;
            }
        }

        if (null == keystoreFile || null == password) {
            help();
            return;
        }

        if (list) {
            listKeys(keystoreFile, password);
            return;
        }

        if (null == label) {
            label = "Keystore Example Keypair";
        }

        //
        // This call to keyStore.load() will open the pkcs12 keystore with the supplied
        // password and connect to the HSM. The CU credentials must be specified using
        // standard CloudHSM login methods.
        //
        try {
            FileInputStream instream = new FileInputStream(keystoreFile);
            keyStore.load(instream, password.toCharArray());
        } catch (FileNotFoundException ex) {
            System.err.println("Keystore not found, loading an empty store");
            keyStore.load(null, null);
        }
```

```
        PasswordProtection passwd = new PasswordProtection(password.toCharArray());
        System.out.println("Searching for example key and certificate...");

        PrivateKeyEntry keyEntry = (PrivateKeyEntry) keyStore.getEntry(label, passwd);
        if (null == keyEntry) {
            //
            // No entry was found, so we need to create a key pair and associate a
certificate.
            // The private key will get the label passed on the command line. The
keystore alias
            // needs to be the same as the private key label. The public key will have
":public"
            // appended to it. The alias used in the keystore will We associate the
certificate
            // with the private key.
            //
            System.out.println("No entry found, creating...");
            KeyPair kp = generateRSAKeyPair(2048, label + ":public", label);
            System.out.printf("Created a key pair with the handles %d/%d%n",
((CaviumKey) kp.getPrivate()).getHandle(), ((CaviumKey) kp.getPublic()).getHandle());

            //
            // Generate a certificate and associate the chain with the private key.
            //
            Certificate self_signed_cert = generateCert(kp);
            Certificate[] chain = new Certificate[1];
            chain[0] = self_signed_cert;
            PrivateKeyEntry entry = new PrivateKeyEntry(kp.getPrivate(), chain);

            //
            // Set the entry using the label as the alias and save the store.
            // The alias must match the private key label.
            //
            keyStore.setEntry(label, entry, passwd);

            FileOutputStream outstream = new FileOutputStream(keystoreFile);
            keyStore.store(outstream, password.toCharArray());
            outstream.close();

            keyEntry = (PrivateKeyEntry) keyStore.getEntry(label, passwd);
        }

        long handle = ((CaviumKey) keyEntry.getPrivateKey()).getHandle();
```

```
        String name = keyEntry.getCertificate().toString();
        System.out.printf("Found private key %d with certificate %s%n", handle, name);
    }

    private static void help() {
        System.out.println(helpString);
    }

    //
    // Generate a non-extractable / non-persistent RSA keypair.
    // This method allows us to specify the public and private labels, which
    // will make KeyStore alises easier to understand.
    //
    public static KeyPair generateRSAKeyPair(int keySizeInBits, String publicLabel,
String privateLabel)
            throws InvalidAlgorithmParameterException, NoSuchAlgorithmException,
NoSuchProviderException {

        boolean isExtractable = false;
        boolean isPersistent = false;
        KeyPairGenerator keyPairGen = KeyPairGenerator.getInstance("rsa", "Cavium");
        CaviumRSAKeyGenParameterSpec spec = new
CaviumRSAKeyGenParameterSpec(keySizeInBits, new BigInteger("65537"), publicLabel,
privateLabel, isExtractable, isPersistent);

        keyPairGen.initialize(spec);

        return keyPairGen.generateKeyPair();
    }

    //
    // Generate a certificate signed by a given keypair.
    //
    private static Certificate generateCert(KeyPair kp) throws CertificateException {
        CertificateFactory cf = CertificateFactory.getInstance("X509");
        PublicKey publicKey = kp.getPublic();
        PrivateKey privateKey = kp.getPrivate();
        byte[] version = Encoder.encodeConstructed((byte) 0,
Encoder.encodePositiveBigInteger(new BigInteger("2"))); // version 1
        byte[] serialNo = Encoder.encodePositiveBigInteger(new BigInteger(1,
Util.computeKCV(publicKey.getEncoded())));

        // Use the SHA512 OID and algorithm.
        byte[] signatureOid = new byte[] {
```

```
            (byte) 0x2A, (byte) 0x86, (byte) 0x48, (byte) 0x86, (byte) 0xF7, (byte)
0x0D, (byte) 0x01, (byte) 0x01, (byte) 0x0D };
        String sigAlgoName = "SHA512WithRSA";

         byte[] signatureId = Encoder.encodeSequence(
                                        Encoder.encodeOid(signatureOid),
                                        Encoder.encodeNull());

         byte[] issuer = Encoder.encodeSequence(
                                    encodeName(COUNTRY_NAME_OID, "<Country>"),
                                    encodeName(STATE_OR_PROVINCE_NAME_OID, "<State>"),
                                    encodeName(LOCALITY_NAME_OID, "<City>"),
                                    encodeName(ORGANIZATION_NAME_OID,
"<Organization>"),
                                    encodeName(ORGANIZATION_UNIT_OID, "<Unit>"),
                                    encodeName(COMMON_NAME_OID, "<CN>")
                                );

         Calendar c = Calendar.getInstance();
         c.add(Calendar.DAY_OF_YEAR, -1);
         Date notBefore = c.getTime();
         c.add(Calendar.YEAR, 1);
         Date notAfter = c.getTime();
         byte[] validity = Encoder.encodeSequence(
                                        Encoder.encodeUTCTime(notBefore),
                                        Encoder.encodeUTCTime(notAfter)
                                    );
         byte[] key = publicKey.getEncoded();

         byte[] certificate = Encoder.encodeSequence(
                                        version,
                                        serialNo,
                                        signatureId,
                                        issuer,
                                        validity,
                                        issuer,
                                        key);
         Signature sig;
         byte[] signature = null;
         try {
             sig = Signature.getInstance(sigAlgoName, "Cavium");
             sig.initSign(privateKey);
             sig.update(certificate);
             signature = Encoder.encodeBitstring(sig.sign());
```

```
        } catch (Exception e) {
            System.err.println(e.getMessage());
            return null;
        }

        byte [] x509 = Encoder.encodeSequence(
                        certificate,
                        signatureId,
                        signature
                        );
        return cf.generateCertificate(new ByteArrayInputStream(x509));
    }

     //
     // Simple OID encoder.
     // Encode a value with OID in ASN.1 format
     //
     private static byte[] encodeName(byte[] nameOid, String value) {
         byte[] name = null;
         name = Encoder.encodeSet(
                     Encoder.encodeSequence(
                             Encoder.encodeOid(nameOid),
                             Encoder.encodePrintableString(value)
                     )
                 );
         return name;
     }

    //
    // List all the keys in the keystore.
    //
    private static void listKeys(String keystoreFile, String password) throws Exception
{
        KeyStore keyStore = KeyStore.getInstance("CloudHSM");

        try {
            FileInputStream instream = new FileInputStream(keystoreFile);
            keyStore.load(instream, password.toCharArray());
        } catch (FileNotFoundException ex) {
            System.err.println("Keystore not found, loading an empty store");
            keyStore.load(null, null);
        }
```

```
        for(Enumeration<String> entry = keyStore.aliases(); entry.hasMoreElements();) {
            System.out.println(entry.nextElement());
        }
    }

}
```

기타 타사 벤더 통합

여러 타사 공급업체가 신뢰의 AWS CloudHSM 근원으로 지원합니다. 즉, CloudHSM 클러스터에 기본 키를 생성하고 저장하는 동안 원하는 소프트웨어 솔루션을 활용할 수 있습니다. 따라서 워크로드는 CloudHSM의 지연 시간, 가용성, 안정성 및 탄력성 이점을 활용할 AWS 수 있습니다. 다음 목록에는 CloudHSM을 지원하는 타사 벤더가 포함됩니다.

> **Note**
>
> AWS 타사 공급업체를 보증하거나 보증하지 않습니다.

- Hashicorp Vault는 조직 전체에서 협업과 거버넌스를 가능하게 하기 위해 고안된 비밀 관리 도구입니다. 이는 추가적인 보호를 AWS Key Management Service 위한 신뢰의 AWS CloudHSM 근원으로서 지원합니다.
- Thycotic Secrets Server는 고객이 권한이 있는 계정에서 민감한 자격 증명을 관리하도록 도와줍니다. 신뢰의 근원 AWS CloudHSM 역할을 합니다.
- P6R의 KMIP 어댑터를 사용하면 표준 KMIP 인터페이스를 통해 AWS CloudHSM 인스턴스를 활용할 수 있습니다.
- PrimeKey EJBCA는 널리 사용되는 PKI용 오픈 소스 솔루션입니다. 이를 통해 키 페어를 안전하게 생성하고 저장할 수 있습니다. AWS CloudHSM
- KeySafeBox는 엄격한 보안, 개인 정보 보호 및 규정 준수 요구 사항을 준수하는 많은 조직에 클라우드 콘텐츠에 대한 암호화 키 관리를 제공합니다. 또한 고객은 AWS KMS Custom Key AWS CloudHSM Store를 통해 직접 AWS Key Management Service 또는 간접적으로 KeySafe 키를 보호할 수 있습니다.
- Insyde Software는 펌웨어 서명을 위한 신뢰할 수 있는 루트 AWS CloudHSM 역할을 지원합니다.
- F5 BIG-IP LTM은 트러스트 루트 역할을 지원합니다 AWS CloudHSM .

- Cloudera Navigator Key HSM을 사용하면 CloudHSM 클러스터를 사용하여 Cloudera Navigator Key Trustee Server의 키를 작성하고 저장할 수 있습니다.
- Venafi 신뢰 보호 플랫폼은 AWS CloudHSM 키 생성 및 보호를 통해 TLS, SSH 및 코드 서명을 위한 포괄적인 시스템 ID 관리를 제공합니다.

모니터링 AWS CloudHSM

클라이언트 SDK에 내장된 로깅 기능 외에도 AWS CloudTrail, Amazon CloudWatch Logs 및 Amazon 을 사용하여 CloudWatch AWS CloudHSM모니터링할 수 있습니다.

클라이언트 SDK 로그

클라이언트 SDK 로깅을 사용하여 생성한 애플리케이션의 진단 및 문제 해결 정보를 모니터링할 수 있습니다.

CloudTrail

클러스터 생성 및 삭제 호출, 하드웨어 보안 모듈 (HSM), 리소스 태그를 포함하여 AWS 계정의 모든 API 호출을 모니터링하는 데 사용합니다 CloudTrail .

CloudWatch 로그

CloudWatch 로그를 사용하여 HSM 인스턴스의 로그를 모니터링할 수 있습니다. 여기에는 HSM 사용자 생성 및 삭제, 사용자 암호 변경, 키 생성 및 삭제 등에 대한 이벤트가 포함됩니다.

CloudWatch

클러스터 상태를 실시간으로 모니터링하는 CloudWatch 데 사용합니다.

주제

클라이언트 SDK 로그로 작업

클라이언트 SDK에서 생성된 로그를 검색할 수 있습니다. AWS CloudHSM 클라이언트 SDK 3 및 클라이언트 SDK 5를 사용한 로깅 구현을 제공합니다.

주제

클라이언트 SDK 5 로깅

클라이언트 SDK 5 로그에는 구성 요소에서 이름이 지정된 파일의 각 구성 요소에 대한 정보가 포함됩니다. 클라이언트 SDK 5용 구성 도구를 사용하여 각 구성 요소에 대한 로깅을 구성할 수 있습니다.

파일 위치를 지정하지 않으면 시스템은 기본 위치에 로그를 기록합니다.

PKCS #11 library

- Linux

  ```
  /opt/cloudhsm/run/cloudhsm-pkcs11.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
  ```

OpenSSL Dynamic Engine

- Linux

  ```
  stderr
  ```

JCE provider

- Linux

  ```
  /opt/cloudhsm/run/cloudhsm-jce.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
  ```

클라이언트 SDK 5의 로깅을 구성하는 방법에 대한 자세한 내용은 클라이언트 SDK 5 구성 도구를 참조하십시오.

클라이언트 SDK 3 로깅

클라이언트 SDK 3 로그에는 클라이언트 데몬의 AWS CloudHSM 세부 정보가 포함됩니다. 로그의 위치는 클라이언트 데몬을 실행하는 Amazon EC2 클라이언트 인스턴스의 운영 체제에 따라 달라집니다.

Amazon Linux

Amazon Linux에서는 AWS CloudHSM 클라이언트 로그가 이름이 지정된 파일에 기록됩니다/opt/cloudhsm/run/cloudhsm_client.log. logrotate 또는 유사한 도구를 사용하여 이러한 로그를 교체하고 관리할 수 있습니다.

Amazon Linux 2

Amazon Linux 2에서는 AWS CloudHSM 클라이언트 로그가 수집되어 저널에 저장됩니다. journalctl을 사용하여 이러한 로그를 보고 관리할 수 있습니다. 예를 들어, 다음 명령을 사용하여 AWS CloudHSM 클라이언트 로그를 볼 수 있습니다.

```
journalctl -f -u cloudhsm-client
```

CentOS 7

CentOS 7에서는 AWS CloudHSM 클라이언트 로그가 수집되어 저널에 저장됩니다. journalctl을 사용하여 이러한 로그를 보고 관리할 수 있습니다. 예를 들어, 다음 명령을 사용하여 AWS CloudHSM 클라이언트 로그를 볼 수 있습니다.

```
journalctl -f -u cloudhsm-client
```

CentOS 8

CentOS 8에서는 AWS CloudHSM 클라이언트 로그가 수집되어 저널에 저장됩니다. journalctl을 사용하여 이러한 로그를 보고 관리할 수 있습니다. 예를 들어, 다음 명령을 사용하여 AWS CloudHSM 클라이언트 로그를 볼 수 있습니다.

```
journalctl -f -u cloudhsm-client
```

RHEL 7

Red Hat 엔터프라이즈 리눅스 7에서는 AWS CloudHSM 클라이언트 로그가 수집되어 저널에 저장됩니다. journalctl을 사용하여 이러한 로그를 보고 관리할 수 있습니다. 예를 들어, 다음 명령을 사용하여 AWS CloudHSM 클라이언트 로그를 볼 수 있습니다.

```
journalctl -f -u cloudhsm-client
```

RHEL 8

Red Hat 엔터프라이즈 리눅스 8에서는 AWS CloudHSM 클라이언트 로그가 수집되어 저널에 저장됩니다. journalctl을 사용하여 이러한 로그를 보고 관리할 수 있습니다. 예를 들어, 다음 명령을 사용하여 AWS CloudHSM 클라이언트 로그를 볼 수 있습니다.

```
journalctl -f -u cloudhsm-client
```

Ubuntu 16.04

Ubuntu 16.04에서는 AWS CloudHSM 클라이언트 로그가 수집되어 저널에 저장됩니다. journalctl을 사용하여 이러한 로그를 보고 관리할 수 있습니다. 예를 들어, 다음 명령을 사용하여 클라이언트 로그를 볼 수 있습니다. AWS CloudHSM

```
journalctl -f -u cloudhsm-client
```

Ubuntu 18.04

Ubuntu 18.04에서는 AWS CloudHSM 클라이언트 로그가 수집되어 저널에 저장됩니다. journalctl을 사용하여 이러한 로그를 보고 관리할 수 있습니다. 예를 들어, 다음 명령을 사용하여 클라이언트 로그를 볼 수 있습니다. AWS CloudHSM

```
journalctl -f -u cloudhsm-client
```

Windows

- Windows 클라이언트 1.1.2+의 경우:

  AWS CloudHSM 클라이언트 로그는 AWS CloudHSM 프로그램 `cloudhsm.log` 파일 폴더 (`C:\Program Files\Amazon\CloudHSM\`) 의 파일에 기록됩니다. 각 로그 파일 이름에는 AWS CloudHSM 클라이언트가 시작된 시간을 나타내는 타임스탬프가 접미사로 붙습니다.

- Windows 클라이언트 1.1.1 이상의 경우:

  클라이언트 로그가 파일에 작성되지 않습니다. 로그는 명령 프롬프트 또는 클라이언트를 시작한 PowerShell 창에 표시됩니다. AWS CloudHSM

AWS CloudTrail 와 함께 일하기 AWS CloudHSM

AWS CloudHSM 사용자 AWS CloudTrail, 역할 또는 서비스가 수행한 작업의 기록을 제공하는 AWS 서비스와 통합되어 AWS CloudHSM있습니다. CloudTrail 모든 API 호출을 AWS CloudHSM 이벤트로 캡처합니다. 캡처된 호출에는 AWS CloudHSM 콘솔에서의 호출 및 AWS CloudHSM API 작업에 대한 코드 호출이 포함됩니다. 트레일을 생성하면 에 대한 이벤트를 포함하여 Amazon S3 버킷으로 CloudTrail 이벤트를 지속적으로 전송할 수 AWS CloudHSM있습니다. 트레일을 구성하지 않아도 CloudTrail 콘솔의 이벤트 기록에서 가장 최근 이벤트를 계속 볼 수 있습니다. 에서 수집한 CloudTrail 정보를 사용하여 요청을 받은 사람 AWS CloudHSM, 요청한 IP 주소, 요청한 사람, 요청 시기 및 추가 세부 정보를 확인할 수 있습니다.

자세한 CloudTrail 내용은 AWS CloudTrail 사용 설명서를 참조하십시오. AWS CloudHSM API 작업의 전체 목록은 AWS CloudHSM API 참조에서의 작업을 참조하십시오.

AWS CloudHSM 정보는 다음을 참조하십시오. CloudTrail

CloudTrail 계정을 만들 때 AWS 계정에서 활성화됩니다. 에서 AWS CloudHSM활동이 발생하면 해당 활동이 CloudTrail 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 내용은 이벤트 기록으로 CloudTrail 이벤트 보기를 참조하십시오.

에 대한 이벤트를 포함하여 AWS 계정의 진행 중인 이벤트 기록을 보려면 트레일을 생성하세요. AWS CloudHSM트레일을 사용하면 CloudTrail Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS Regions에 추적이 적용됩니다. 트레일은 AWS 파티션에 있는 모든 지역의 이벤트를 기록하고 지정한 Amazon S3 버킷으로 로그 파일을 전송합니다. 또한 CloudTrail 로그에서 수집된 이벤트 데이터를 추가로 분석하고 이에 따라 조치를 취하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음을 참조하십시오.

- 추적 생성 개요
- CloudTrail 지원되는 서비스 및 통합
- 에 대한 Amazon SNS 알림 구성 CloudTrail
- 여러 지역에서 CloudTrail 로그 파일 수신 및 여러 계정으로부터 CloudTrail 로그 파일 수신

CloudTrail 읽기 전용 AWS CloudHSM 작업 (예: 및) 과 관리 작업 (예:`ListTags`, `DescribeClusters` 및) 을 포함한 모든 작업을 기록합니다. `InitializeCluster` `CreatHsm` `DeleteBackup`

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 신원 정보를 이용하면 다음을 쉽게 알아볼 수 있습니다.

- 요청이 루트 또는 AWS Identity and Access Management (IAM) 사용자 자격 증명으로 이루어졌는지 여부.
- 역할 또는 연동 사용자를 위한 임시 보안 인증으로 요청을 생성하였는지.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail 사용자 ID 요소를 참조하십시오.

로그 파일 항목 이해 AWS CloudHSM

트레일은 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 전송할 수 있는 구성입니다. CloudTrail 로그 파일은 하나 이상의 로그 항목을 포함합니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜 및 시간, 요청 매개 변수 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 공개 API 호출의 정렬된 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음 예제는 AWS CloudHSM `CreateHsm` 작업을 보여주는 CloudTrail 로그 항목을 보여줍니다.

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAJZVM5NEGZSTCITAMM:ExampleSession",
        "arn": "arn:aws:sts::111122223333:assumed-role/AdminRole/ExampleSession",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIY22AX6VRYNBGJSA",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2017-07-11T03:48:44Z"
            },
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROAJZVM5NEGZSTCITAMM",
                "arn": "arn:aws:iam::111122223333:role/AdminRole",
                "accountId": "111122223333",
                "userName": "AdminRole"
            }
        }
    },
```

```
    "eventTime": "2017-07-11T03:50:45Z",
    "eventSource": "cloudhsm.amazonaws.com",
    "eventName": "CreateHsm",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "205.251.233.179",
    "userAgent": "aws-internal/3",
    "requestParameters": {
        "availabilityZone": "us-west-2b",
        "clusterId": "cluster-fw7mh6mayb5"
    },
    "responseElements": {
        "hsm": {
            "eniId": "eni-65338b5a",
            "clusterId": "cluster-fw7mh6mayb5",
            "state": "CREATE_IN_PROGRESS",
            "eniIp": "10.0.2.7",
            "hsmId": "hsm-6lz2hfmnzbx",
            "subnetId": "subnet-02c28c4b",
            "availabilityZone": "us-west-2b"
        }
    },
    "requestID": "1dae0370-65ec-11e7-a770-6578d63de907",
    "eventID": "b73a5617-8508-4c3d-900d-aa8ac9b31d08",
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```


Amazon CloudWatch 로그 및 AWS CloudHSM 감사 로그 사용

계정의 HSM이 명령줄 도구 또는 소프트웨어 라이브러리로부터 AWS CloudHSM 명령을 받으면 해당 명령의 실행을 감사 로그 형식으로 기록합니다. HSM을 만들고 삭제하며 HSM에 로그인 및 로그아웃하고 사용자와 키를 관리하는 명령을 포함하여 클라이언트가 시작한 모든 관리 명령이 HSM 감사 로그에 포함됩니다. 이 로그는 HSM 상태를 변경한 작업에 대해 신뢰할 수 있는 기록을 제공합니다.

AWS CloudHSM HSM 감사 로그를 수집하여 사용자를 대신하여 Amazon CloudWatch Logs로 전송합니다. CloudWatch 로그 검색 및 필터링, Amazon S3로 로그 데이터 내보내기 등 로그 기능을 사용하여 AWS CloudHSM 감사 로그를 관리할 수 있습니다. Amazon CloudWatch 콘솔에서 HSM 감사 로그로 작업하거나 AWS CLI및 CloudWatch Logs SDK에서 CloudWatch Logs 명령을 사용할 수 있습니다.

주제

- HSM 감사 로깅 작동 방식

- 로그에서 CloudWatch HSM 감사 로그 보기
- HSM 감사 로그 해석
- HSM 감사 로그 참조

HSM 감사 로깅 작동 방식

감사 로깅은 모든 AWS CloudHSM 클러스터에서 자동으로 활성화됩니다. 비활성화하거나 끌 수 없으며 로그를 로그로 내보내는 것을 AWS CloudHSM 막는 설정도 없습니다. CloudWatch 각 로그 이벤트에는 이벤트 순서를 나타내며 로그 훼손을 감지할 수 있도록 하는 타임스탬프와 시퀀스 번호가 있습니다.

각 HSM 인스턴스는 자체 로그를 생성합니다. 다양한 HSM의 감사 로그는 동일한 클러스터에 있는 경우에도 다를 수 있습니다. 예를 들어, 각 클러스터의 첫 번째 HSM만 HSM의 초기화를 기록합니다. 백업에서 복제된 HSM의 로그에는 초기화 이벤트가 나타나지 않습니다. 마찬가지로, 키를 생성할 때 키를 생성하는 HSM이 키 생성 이벤트를 기록합니다. 클러스터의 다른 HSM은 동기화를 통해 키를 수신할 때 이벤트를 기록합니다.

AWS CloudHSM 로그를 수집하여 계정의 CloudWatch 로그에 게시합니다. 사용자를 대신하여 CloudWatch 로그 서비스와 통신하려면 서비스 연결 역할을 AWS CloudHSM 사용합니다. 역할과 관련된 IAM 정책은 감사 로그를 Logs로 보내는 데 필요한 작업만 AWS CloudHSM 수행하도록 허용합니다. CloudWatch

> **Important**
>
> 2018년 1월 20일 이전에 클러스터를 생성했으며 연결된 서비스 연결 역할을 아직 생성하지 않은 경우 해당 역할을 수동으로 생성해야 합니다. 이는 AWS CloudHSM 클러스터로부터 감사 로그를 수신하는 CloudWatch 데 필요합니다. 서비스 연결 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 이해 및 서비스 연결 역할 생성을 참조하십시오.

로그에서 CloudWatch HSM 감사 로그 보기

Amazon CloudWatch Logs는 감사 로그를 로그 그룹으로, 로그 그룹 내에서 로그 스트림으로 구성합니다. 각 로그 항목은 이벤트입니다. AWS CloudHSM 각 클러스터에 대해 하나의 로그 그룹을 생성하고 클러스터의 각 HSM에 대해 하나의 로그 스트림을 생성합니다. CloudWatch 로그 구성 요소를 생성하거나 설정을 변경할 필요가 없습니다.

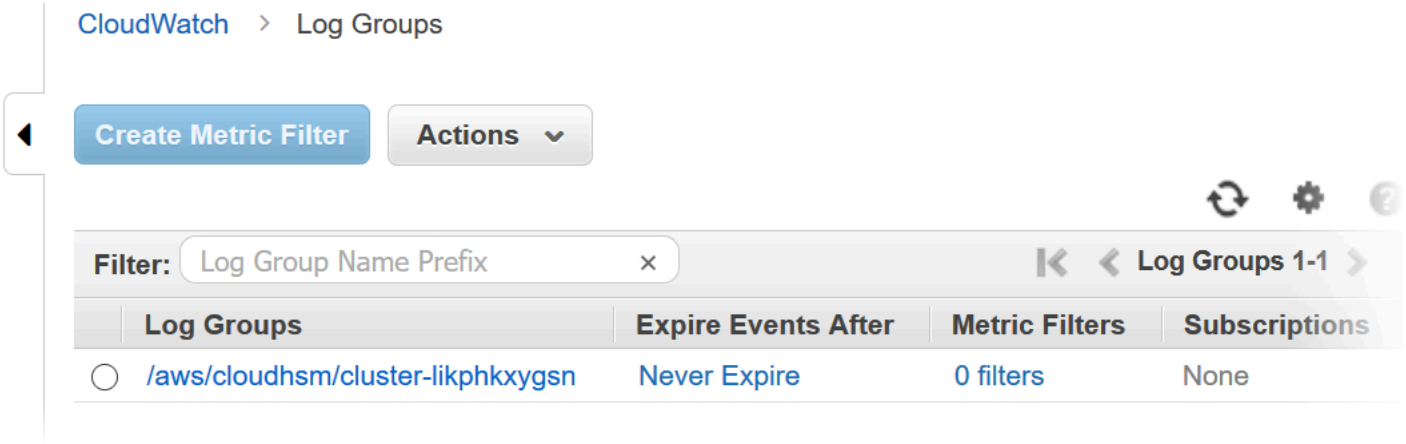
- 로그 그룹 이름은 `/aws/cloudhsm/<cluster ID>`입니다(예: `/aws/cloudhsm/cluster-likphkxygsn`). AWS CLI 또는 PowerShell 명령에 로그 그룹 이름을 사용할 때는 반드시 큰따옴표로 묶어야 합니다.
- 로그 스트림 이름은 HSM ID입니다(예: `hsm-nwbbiqbj4jk`).

  일반적으로 HSM마다 로그 스트림이 하나씩 있습니다. 하지만 HSM이 실패하고 교체되는 경우와 같이 HSM ID를 변경하는 작업은 새 로그 스트림을 생성합니다.

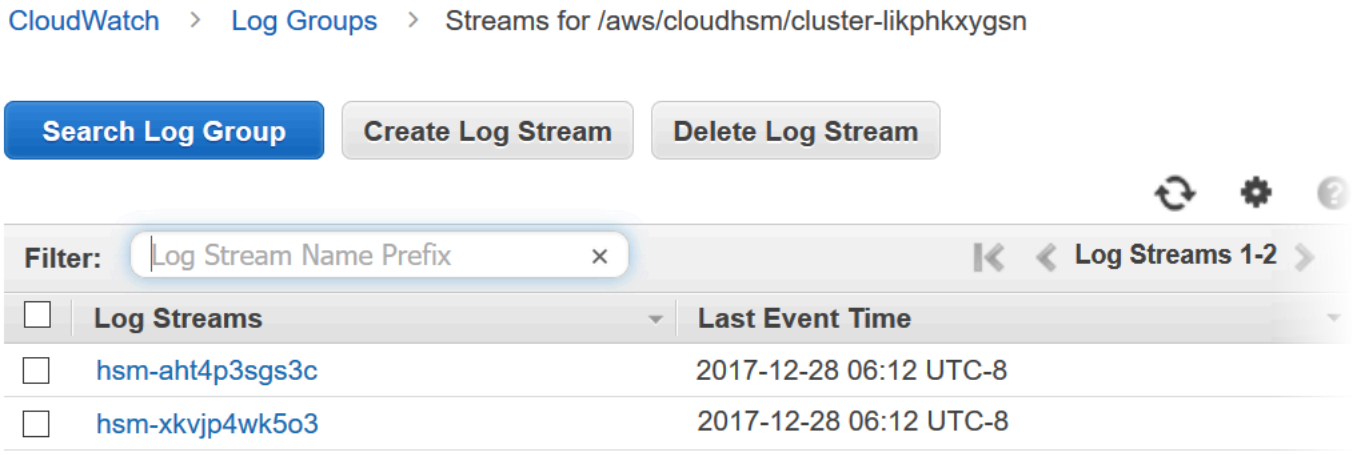
CloudWatch 로그 개념에 대한 자세한 내용은 Amazon CloudWatch Logs 사용 설명서의 개념을 참조하십시오.

의 로그 페이지, 의 로그 명령 AWS Management Console, CloudWatch Logs PowerShell cmdlet 또는 CloudWatch LogsSDK에서 HSM에 대한 감사 CloudWatch 로그를 볼 수 있습니다. AWS CLI CloudWatch 지침은 Amazon CloudWatch Logs 사용 설명서의 로그 데이터 보기를 참조하십시오.

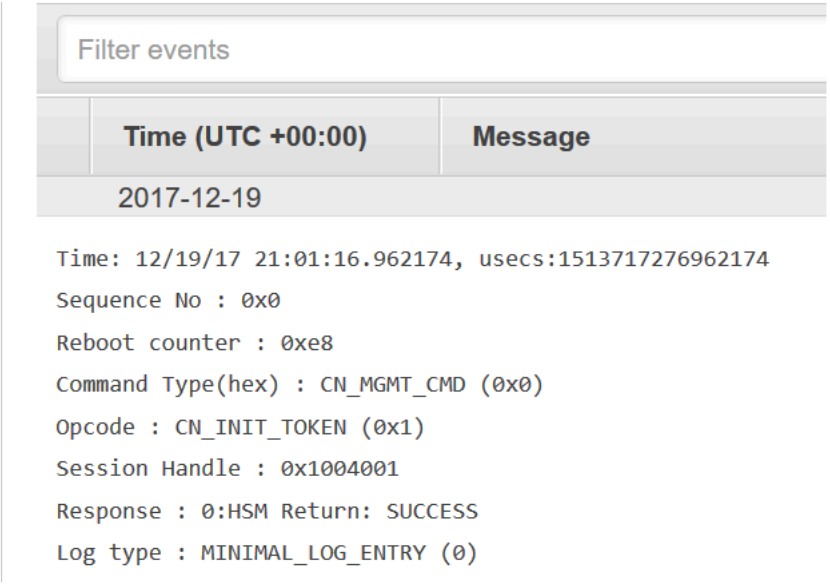
예를 들어 다음 이미지에서는 AWS Management Console에 있는 `cluster-likphkxygsn` 클러스터의 로그 그룹을 보여 줍니다.



클러스터 로그 그룹 이름을 선택할 때 클러스터에 있는 각 HSM의 로그 스트림을 볼 수 있습니다. 다음 이미지에서는 `cluster-likphkxygsn` 클러스터에 있는 HSM의 로그 스트림을 보여 줍니다.

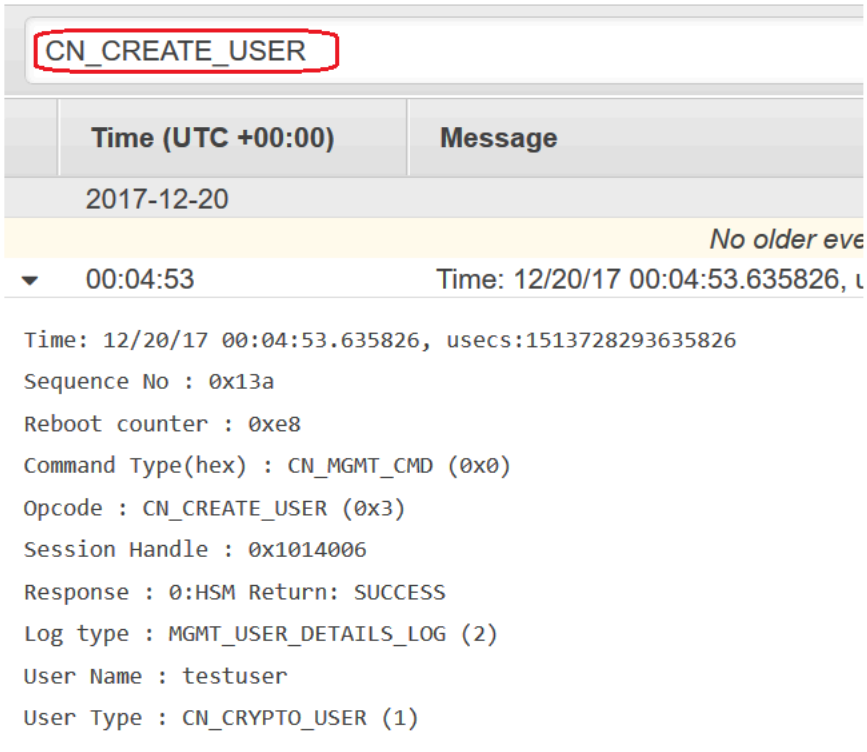


HSM 로그 스트림 이름을 선택할 때 감사 로그의 이벤트를 볼 수 있습니다. 예를 들어 시퀀스 번호가 0x0이고 Opcode가 CN_INIT_TOKEN인 이 이벤트는 일반적으로 각 클러스터에 있는 첫 번째 HSM의 첫 번째 이벤트입니다. 이 이벤트는 클러스터에 있는 HSM의 초기화를 기록합니다.



로그의 다양한 기능을 모두 사용하여 감사 CloudWatch 로그를 관리할 수 있습니다. 예를 들어 이벤트 필터링 기능을 사용하여 이벤트에서 CN_CREATE_USER Opcode와 같은 특정 텍스트를 찾을 수 있습니다.

지정된 텍스트를 포함하지 않는 모든 이벤트를 찾으려면 텍스트 앞에 빼기 기호(-)를 추가하십시오. 예를 들어 CN_CREATE_USER가 포함되지 않은 이벤트를 찾으려면 -CN_CREATE_USER를 입력합니다.



HSM 감사 로그 해석

HSM 감사 로그의 이벤트에는 표준 필드가 있습니다. 일부 이벤트 유형에는 이벤트에 대한 유용한 정보를 캡처하는 추가 필드가 있습니다. 예를 들어, 사용자 로그인 및 사용자 관리 이벤트에는 사용자 이름과 해당 사용자의 사용자 유형이 포함되어 있습니다. 키 관리 명령에는 key handle이 포함되어 있습니다.

일부 필드는 특별히 중요한 정보를 제공합니다. Opcode는 기록되는 관리 명령을 식별합니다. Sequence No는 로그 스트림의 이벤트를 식별하고 해당 이벤트가 기록된 순서를 나타냅니다.

예를 들어, 다음 예제 이벤트는 HSM에 대한 로그 스트림의 두 번째 이벤트(Sequence No: 0x1)입니다. 이 이벤트는 스타트업 루틴의 일부인 암호 암호화 키를 생성하는 HSM을 보여 줍니다.

```
Time: 12/19/17 21:01:17.140812, usecs:1513717277140812
Sequence No : 0x1
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_GEN_PSWD_ENC_KEY (0x1d)
Session Handle : 0x1004001
```

```
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

다음 필드는 감사 로그의 모든 AWS CloudHSM 이벤트에 공통적으로 사용됩니다.

Time

이벤트가 발생한 시간으로 UTC 시간대로 설정됩니다. 시간은 마이크로초의 Unix 시간과 사람이 읽을 수 있는 시간으로 표시됩니다.

재부팅 카운터

HSM 하드웨어가 재부팅될 때 증가하는 32비트의 영구 서수 카운터입니다.

로그 스트림의 모든 이벤트는 재부팅 카운터 값이 동일합니다. 재부팅 카운터는 동일한 클러스터에 있는 여러 HSM 인스턴스 간에 다를 수 있으므로 로그 스크림에 대해 고유하지 않을 수 있습니다.

시퀀스 번호

각 로그 이벤트에 대해 증가하는 64비트 서수 카운터입니다. 각 로그 스트림에 있는 첫 번째 이벤트의 시퀀스 번호는 0x0입니다. Sequence No 값에는 간격이 없어야 합니다. 시퀀스 번호는 로그 스트림 내에서만 고유합니다.

명령 유형

명령 범주를 나타내는 16진수 값입니다. AWS CloudHSM 로그 스트림의 명령에는 CN_MGMT_CMD(0x0) 또는 CN_CERT_AUTH_CMD(0x9)의 유형이 있습니다.

Opcode

실행된 관리 명령을 식별합니다. AWS CloudHSM 감사 로그의 Opcode 값 목록은 을 참조하십시오HSM 감사 로그 참조.

세션 핸들

명령이 실행되고 이벤트가 기록된 세션을 식별합니다.

응답

관리 명령에 대한 응답을 기록합니다. SUCCESS 및 ERROR 값에 대해 Response 필드를 검색할 수 있습니다.

로그 유형

명령을 기록한 AWS CloudHSM 로그의 로그 유형을 나타냅니다.

- MINIMAL_LOG_ENTRY (0)

- MGMT_KEY_DETAILS_LOG (1)
- MGMT_USER_DETAILS_LOG (2)
- GENERIC_LOG

감사 로그 이벤트 예제

로그 스트림의 이벤트는 생성부터 삭제까지의 HSM 기록을 기록합니다. 로그를 통해 HSM의 수명 주기를 검토하고 해당 작업에 대한 통찰을 얻을 수 있습니다. 이벤트를 해석할 때 관리 명령이나 작업을 나타내는 Opcode 및 이벤트 순서를 나타내는 Sequence No를 기록해 두십시오.

주제

- 예제: 클러스터의 첫 번째 HSM 초기화
- 로그인 및 로그아웃 이벤트
- 예제: 사용자 생성 및 삭제
- 예제: 키 페어 생성 및 삭제
- 예제: 키 생성 및 동기화
- 예제: 키 내보내기
- 예제: 키 가져오기
- 예제: 키 공유 및 공유 해제

예제: 클러스터의 첫 번째 HSM 초기화

각 클러스터의 첫 번째 HSM에 대한 감사 로그 스트림은 클러스터의 다른 HSM에 대한 로그 스트림과 매우 다릅니다. 각 클러스터의 첫 번째 HSM에 대한 감사 로그는 생성 및 초기화를 기록합니다. 백업에서 생성되는 클러스터의 추가 HSM 로그는 로그인 이벤트로 시작합니다.

> **Important**
>
> 다음 초기화 항목은 CloudHSM 감사 로깅 기능 출시 (2018년 8월 30일) 이전에 초기화된 클러스터의 CloudWatch 로그에 표시되지 않습니다. 자세한 내용은 문서 기록을 참조하십시오.

다음 예제 이벤트는 클러스터의 첫 번째 HSM에 대한 로그 스트림에 나타납니다. 로그의 첫 번째 이벤트인 Sequence No 0x0 이벤트는 HSM(CN_INIT_TOKEN)을 초기화하는 명령을 나타냅니다. 응답은 명령이 성공했음을 나타냅니다(Response : 0: HSM Return: SUCCESS).

```
Time: 12/19/17 21:01:16.962174, usecs:1513717276962174
Sequence No : 0x0
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INIT_TOKEN (0x1)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

이 예제 로그 스트림의 두 번째 이벤트(Sequence No 0x1)는 HSM에서 사용되는 암호 암호화 키를 생성하는 명령(CN_GEN_PSWD_ENC_KEY)을 기록합니다.

이는 각 클러스터의 첫 번째 HSM에 대한 일반적인 스타트업 시퀀스입니다. 동일한 클러스터에 있는 후속 HSM은 첫 번째의 복제본이므로 동일한 암호 암호화 키를 사용합니다.

```
Time: 12/19/17 21:01:17.140812, usecs:1513717277140812
Sequence No : 0x1
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_GEN_PSWD_ENC_KEY (0x1d)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

이 예제 로그 스트림(Sequence No 0x2)의 세 번째 이벤트는 AWS CloudHSM 서비스인 AU(Appliance User)의 생성입니다. HSM 사용자와 관련된 이벤트에는 사용자 이름과 사용자 유형에 대한 추가 필드가 포함되어 있습니다.

```
Time: 12/19/17 21:01:17.174902, usecs:1513717277174902
Sequence No : 0x2
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_APPLIANCE_USER (0xfc)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : app_user
User Type : CN_APPLIANCE_USER (5)
```

이 예제 로그 스트림의 네 번째 이벤트(Sequence No 0x3)는 HSM의 초기화를 완료하는 CN_INIT_DONE 이벤트를 기록합니다.

```
Time: 12/19/17 21:01:17.298914, usecs:1513717277298914
Sequence No : 0x3
Reboot counter : 0xe8
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INIT_DONE (0x95)
Session Handle : 0x1004001
Response : 0:HSM Return: SUCCESS
Log type : MINIMAL_LOG_ENTRY (0)
```

스타트업 시퀀스에서 나머지 이벤트를 따를 수 있습니다. 이 이벤트에는 여러 가지 로그인 및 로그아웃 이벤트와 KEK(키 암호화 키) 생성이 포함될 수 있습니다. 다음 이벤트는 PRECO(Precrypto Officer)의 암호를 변경하는 명령을 기록합니다. 이 명령은 클러스터를 활성화합니다.

```
Time: 12/13/17 23:04:33.846554, usecs:1513206273846554
Sequence No: 0x1d
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_CHANGE_PSWD (0x9)
Session Handle: 0x2010003
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: admin
User Type: CN_CRYPTO_PRE_OFFICER (6)
```

로그인 및 로그아웃 이벤트

감사 로그를 해석할 때 HSM에 대한 사용자의 로그인 및 로그아웃을 기록하는 이벤트를 기록해 두십시오. 이러한 이벤트를 통해 로그인 및 로그아웃 명령 간의 시퀀스에 나타나는 관리 명령에 대한 책임이 있는 사용자를 결정할 수 있습니다.

예를 들어, 이 로그 항목은 이름이 admin인 Crypto Officer의 로그인을 기록합니다. 0x0의 시퀀스 번호는 이 로그 스트림에서 첫 번째 이벤트임을 나타냅니다.

사용자가 HSM에 로그인할 때 클러스터의 다른 HSM도 사용자의 로그인 이벤트를 기록합니다. 최초 로그인 이벤트 직후 클러스터에 있는 다른 HSM의 로그 스트림에서 해당 로그인 이벤트를 찾을 수 있습니다.

```
Time: 01/16/18 01:48:49.824999, usecs:1516067329824999
Sequence No : 0x0
Reboot counter : 0x107
```

```
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
User Type : CN_CRYPTO_OFFICER (2)
```

다음 예제 이벤트는 admin CO(Crypto Officer) 로그아웃을 기록합니다. 시퀀스 번호 0x2는 이 항목이 로그 스트림의 세 번째 이벤트임을 나타냅니다.

로그인한 사용자가 로그아웃하지 않고 세션을 닫으면 로그 스트림이 CN_APP_FINALIZE를 포함하거나 CN_LOGOUT 이벤트 대신 세션 이벤트(CN_SESSION_CLOSE)를 닫습니다. 로그인 이벤트와 달리 이 로그아웃 이벤트는 일반적으로 명령을 실행하는 HSM에 의해서만 기록됩니다.

```
Time: 01/16/18 01:49:55.993404, usecs:1516067395993404
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGOUT (0xe)
Session Handle : 0x7014000
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
User Type : CN_CRYPTO_OFFICER (2)
```

사용자 이름이 잘못되어 로그인 시도가 실패하면 HSM이 로그인 명령에 제공된 사용자 이름과 유형을 사용하여 CN_LOGIN 이벤트를 기록합니다. 사용자 이름이 없다고 설명하는 오류 메시지 157이 응답에 표시됩니다.

```
Time: 01/24/18 17:41:39.037255, usecs:1516815699037255
Sequence No : 0x4
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 157:HSM Error: user isn't initialized or user with this name doesn't exist
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : ExampleUser
User Type : CN_CRYPTO_USER (1)
```

암호가 잘못되어 로그인 시도가 실패하면 HSM이 로그인 명령에 제공된 사용자 이름과 유형을 사용하여 CN_LOGIN 이벤트를 기록합니다. RET_USER_LOGIN_FAILURE 오류 코드와 함께 오류 메시지가 응답에 표시됩니다.

```
Time: 01/24/18 17:44:25.013218, usecs:1516815865013218
Sequence No : 0x5
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 163:HSM Error: RET_USER_LOGIN_FAILURE
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

예제: 사용자 생성 및 삭제

이 예제에서는 CO(Crypto Officer)가 사용자를 생성하고 삭제할 때 기록되는 로그 이벤트를 보여 줍니다.

첫 번째 이벤트는 HSM에 로그인하는 CO를 나타내는 admin을 기록합니다. 시퀀스 번호 0x0은 이 항목이 로그 스트림의 첫 번째 이벤트임을 나타냅니다. 로그인한 사용자의 이름과 유형이 이벤트에 포함됩니다.

```
Time: 01/16/18 01:48:49.824999, usecs:1516067329824999
Sequence No : 0x0
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : admin
User Type : CN_CRYPTO_OFFICER (2)
```

로그 스트림의 다음 이벤트(시퀀스 0x1)가 새 CU(Crypto User)를 만드는 CO를 기록합니다. 새 사용자의 이름과 유형이 이벤트에 포함됩니다.

```
Time: 01/16/18 01:49:39.437708, usecs:1516067379437708
Sequence No : 0x1
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
```

```
Opcode : CN_CREATE_USER (0x3)
Session Handle : 0x7014006
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : bob
User Type : CN_CRYPTO_USER (1)
```

그런 다음 CO가 또 다른 CO(Crypto Officer)인 `alice`를 생성합니다. 시퀀스 번호는 이 작업이 이전 작업에 곧장 이어진다는 것을 나타냅니다.

```
Time: 01/16/18 01:49:55.993404, usecs:1516067395993404
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_CREATE_CO (0x4)
Session Handle : 0x7014007
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : alice
User Type : CN_CRYPTO_OFFICER (2)
```

나중에 이름이 `admin`인 CO가 로그인하여 이름이 `alice`인 CO(Crypto Officer)를 삭제합니다. HSM가 `CN_DELETE_USER` 이벤트를 기록합니다. 삭제된 사용자의 이름과 유형이 이벤트에 포함됩니다.

```
Time: 01/23/18 19:58:23.451420, usecs:1516737503451420
Sequence No : 0xb
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_DELETE_USER (0xa1)
Session Handle : 0x7014007
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : alice
User Type : CN_CRYPTO_OFFICER (2)
```

예제: 키 페어 생성 및 삭제

이 예제에서는 키 페어를 생성하고 삭제할 때 HSM 감사에 기록되는 이벤트를 보여 줍니다.

다음 이벤트는 HSM에 로그인하는 `crypto_user`라는 CU(Crypto User)를 기록합니다.

```
Time: 12/13/17 23:09:04.648952, usecs:1513206544648952
```

```
Sequence No: 0x28
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_LOGIN (0xd)
Session Handle: 0x2014005
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: crypto_user
User Type: CN_CRYPTO_USER (1)
```

그런 다음 CU가 키 페어(CN_GENERATE_KEY_PAIR)를 생성합니다. 프라이빗 키에는 키 핸들 131079이 있습니다. 퍼블릭 키에는 키 핸들 131078이 있습니다.

```
Time: 12/13/17 23:09:04.761594, usecs:1513206544761594
Sequence No: 0x29
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_GENERATE_KEY_PAIR (0x19)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131079
Public Key Handle: 131078
```

CU가 키 페어를 즉시 삭제합니다. CN_DESTROY_OBJECT 이벤트가 퍼블릭 키(131078)의 삭제를 기록합니다.

```
Time: 12/13/17 23:09:04.813977, usecs:1513206544813977
Sequence No: 0x2a
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_DESTROY_OBJECT (0x11)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131078
Public Key Handle: 0
```

그런 다음 두 번째 CN_DESTROY_OBJECT 이벤트가 프라이빗 키(131079)의 삭제를 기록합니다.

```
Time: 12/13/17 23:09:04.815530, usecs:1513206544815530
```

```
Sequence No: 0x2b
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_DESTROY_OBJECT (0x11)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle: 131079
Public Key Handle: 0
```

마지막으로 CU가 로그아웃합니다.

```
Time: 12/13/17 23:09:04.817222, usecs:1513206544817222
Sequence No: 0x2c
Reboot counter: 0xe8
Command Type(hex): CN_MGMT_CMD (0x0)
Opcode: CN_LOGOUT (0xe)
Session Handle: 0x2014004
Response: 0:HSM Return: SUCCESS
Log type: MGMT_USER_DETAILS_LOG (2)
User Name: crypto_user
User Type: CN_CRYPTO_USER (1)
```

예제: 키 생성 및 동기화

이 예제에서는 HSM이 여러 개인 클러스터에서 키를 생성할 경우의 효과를 보여 줍니다. 키는 HSM 하나에 생성되고 마스킹 처리된 객체로 이 HSM에서 추출되며, 마스킹 처리된 객체로 다른 HSM에 삽입됩니다.

> **Note**
>
> 클라이언트 도구가 키 동기화에 실패할 수 있습니다. 또는 지정된 수의 HSM에만 키를 동기화하는 min_srv 파라미터가 명령에 포함될 수 있습니다. 어느 경우든 AWS CloudHSM 서비스는 키를 클러스터의 다른 HSM과 동기화합니다. HSM은 클라이언트 측 관리 명령만 로그에 기록하므로 서버 측 동기화는 HSM 로그에 기록되지 않습니다.

명령을 수신하고 실행하는 HSM의 로그 스트림을 먼저 고려하십시오. 로그 스트림은 HSM ID hsm-abcde123456을 따라 명명되지만 HSM ID는 로그 이벤트에 표시되지 않습니다.

먼저 testuser CU(Crypto User)가 hsm-abcde123456 HSM에 로그인합니다.

```
Time: 01/24/18 00:39:23.172777, usecs:1516754363172777
Sequence No : 0x0
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0xc008002
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

CU는 exSymKey명령을 실행하여 대칭 키를 생성합니다. hsm-abcde123456 HSM은 키 핸들이 262152인 키 핸들로 대칭 키를 생성합니다. HSM이 로그에 CN_GENERATE_KEY 이벤트를 기록합니다.

```
Time: 01/24/18 00:39:30.328334, usecs:1516754370328334
Sequence No : 0x1
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_GENERATE_KEY (0x17)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

hsm-abcde123456의 로그 스트림에 있는 다음 이벤트가 키 동기화 프로세스의 첫 단계를 기록합니다. 새 키(키 핸들 262152)가 마스킹 처리된 객체로 HSM에서 추출됩니다.

```
Time: 01/24/18 00:39:30.330956, usecs:1516754370330956
Sequence No : 0x2
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_EXTRACT_MASKED_OBJECT_USER (0xf0)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

이제 같은 클러스터에 있는 또 다른 HSM인 HSM hsm-zyxwv987654의 로그 스트림을 고려해 보십시오. 이 로그 스트림에는 testuser CU의 로그인 이벤트도 포함됩니다. 시간 값은 사용자가 hsm-abcde123456 HSM에 로그인한 직후 이 이벤트가 발생한다는 것을 보여 줍니다.

```
Time: 01/24/18 00:39:23.199740, usecs:1516754363199740
Sequence No : 0xd
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7004004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

이 HSM의 이 로그 스트림에는 CN_GENERATE_KEY 이벤트가 없지만 이 HSM에 대한 키 동기화를 기록하는 이벤트는 있습니다. CN_INSERT_MASKED_OBJECT_USER 이벤트는 키 262152의 수신을 마스킹 처리된 객체로 기록합니다. 이제 클러스터의 두 HSM에 모두 262152 키가 있습니다.

```
Time: 01/24/18 00:39:30.408950, usecs:1516754370408950
Sequence No : 0xe
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INSERT_MASKED_OBJECT_USER (0xf1)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 262152
Public Key Handle : 0
```

CU 사용자가 로그아웃할 때 명령을 수신한 HSM의 로그 스트림에만 이 CN_LOGOUT 이벤트가 나타납니다.

예제: 키 내보내기

이 예제에서는 CU(Crypto User)가 HSM이 여러 개인 클러스터에서 키를 내보낼 때 기록되는 감사 로그 이벤트를 보여 줍니다.

다음 이벤트는 key_mgmt_util에 로그인하는 CU(testuser)를 기록합니다.

```
Time: 01/24/18 19:42:22.695884, usecs:1516822942695884
```

```
Sequence No : 0x26
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7004004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

CU는 exSymKey명령을 실행하여 256비트 7 AES 키인 키를 내보냅니다. 명령에서 HSM의 256 비트 AES 키인 6키가 래핑 키로 사용됩니다.

명령을 수신하는 HSM이 내보낼 키인 7 키의 CN_WRAP_KEY 이벤트를 기록합니다.

```
Time: 01/24/18 19:51:12.860123, usecs:1516823472860123
Sequence No : 0x27
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_WRAP_KEY (0x1a)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 7
Public Key Handle : 0
```

그런 다음 HSM이 래핑 키인 키 6에 대해 CN_NIST_AES_WRAP 이벤트를 기록합니다. 키는 래핑되고 곧장 언래핑되지만 HSM은 한 가지 이벤트만 기록합니다.

```
Time: 01/24/18 19:51:12.905257, usecs:1516823472905257
Sequence No : 0x28
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_NIST_AES_WRAP (0x1e)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 6
Public Key Handle : 0
```

exSymKey 명령은 내보낸 키를 파일에 쓰지만 HSM에서 키를 변경하지는 않습니다. 그 결과, 클러스터에 있는 다른 HSM의 로그에는 해당하는 이벤트가 없습니다.

예제: 키 가져오기

이 예제에서는 클러스터의 HSM으로 키를 가져올 때 기록되는 감사 로그 이벤트를 보여 줍니다. 이 예제에서 암호화 사용자 (CU) 는 imSymKey명령을 사용하여 AES 키를 HSM으로 가져옵니다. 6 키가 래핑 키로 명령에 사용됩니다.

명령을 수신하는 HSM이 먼저 래핑 키인 6 키의 CN_NIST_AES_WRAP 이벤트를 기록합니다.

```
Time: 01/24/18 19:58:23.170518, usecs:1516823903170518
Sequence No : 0x29
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_NIST_AES_WRAP (0x1e)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 6
Public Key Handle : 0
```

그런 다음 HSM은 가져오기 작업을 나타내는 CN_UNWRAP_KEY 이벤트를 기록합니다. 가져온 키에 키 핸들 11이 할당됩니다.

```
Time: 01/24/18 19:58:23.200711, usecs:1516823903200711
Sequence No : 0x2a
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_UNWRAP_KEY (0x1b)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0
```

새로운 키를 생성하거나 가져오면 클라이언트 도구가 자동으로 클러스터의 다른 HSM에 새 키를 동기화하려고 시도합니다. 이 경우 HSM은 HSM에서 11 키가 마스킹 처리된 객체로 추출될 때 CN_EXTRACT_MASKED_OBJECT_USER 이벤트를 기록합니다.

```
Time: 01/24/18 19:58:23.203350, usecs:1516823903203350
Sequence No : 0x2b
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
```

```
Opcode : CN_EXTRACT_MASKED_OBJECT_USER (0xf0)
Session Handle : 0x7004003
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0
```

새로 가져온 키의 도착이 클러스터에 있는 다른 HSM의 로그 스트림에 반영됩니다.

예를 들어 이 이벤트는 같은 클러스터에 있는 다른 HSM의 로그 스트림에 기록되었습니다. 이 CN_INSERT_MASKED_OBJECT_USER 이벤트는 11 키를 나타내는 마스킹 처리된 객체의 도착을 기록합니다.

```
Time: 01/24/18 19:58:23.286793, usecs:1516823903286793
Sequence No : 0xb
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_INSERT_MASKED_OBJECT_USER (0xf1)
Session Handle : 0xc008004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_KEY_DETAILS_LOG (1)
Priv/Secret Key Handle : 11
Public Key Handle : 0
```

예제: 키 공유 및 공유 해제

이 예제에서는 CU(Crypto User)가 다른 Crypto User와 ECC 프라이빗 키를 공유하거나 공유 해제할 때 기록되는 감사 로그 이벤트를 보여줍니다. CU는 shareKey 명령을 사용하고 키 핸들, 사용자 ID 및 값 1을 제공하여 키를 공유하고 값 0을 제공하여 키를 공유 해제합니다.

다음 예제에서는 명령을 수신하는 HSM이 공유 작업을 나타내는 CM_SHARE_OBJECT 이벤트를 기록합니다.

```
Time: 02/08/19 19:35:39.480168, usecs:1549654539480168
Sequence No : 0x3f
Reboot counter : 0x38
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_SHARE_OBJECT (0x12)
Session Handle : 0x3014007
Response : 0:HSM Return: SUCCESS
Log type : UNKNOWN_LOG_TYPE (5)
```

HSM 감사 로그 참조

AWS CloudHSM 감사 로그 이벤트에 HSM 관리 명령을 기록합니다. 발생한 작업과 그에 대한 반응을 식별하는 작업 코드(0pcode) 값이 이벤트마다 있습니다. 0pcode 값을 사용하여 로그를 검색, 정렬 및 필터링할 수 있습니다.

다음 표는 AWS CloudHSM 감사 로그의 0pcode 값을 정의합니다.

작업 코드(Opcode)	설명
사용자 로그인: 사용자 이름과 사용자 유형이 이 이벤트에 포함됩니다.	
CN_LOGIN (0xd)	사용자 로그인
CN_LOGOUT (0xe)	사용자 로그아웃
CN_APP_FINALIZE	HSM과의 연결이 끊겼습니다. 이 연결의 모든 세션 키 또는 쿼럼 토큰이 삭제되었습니다.
CN_CLOSE_SESSION	HSM과의 세션이 종료되었습니다. 이 세션의 모든 세션 키 또는 쿼럼 토큰이 삭제되었습니다.
사용자 관리: 사용자 이름과 사용자 유형이 이 이벤트에 포함됩니다.	
CN_CREATE_USER (0x3)	CU(Crypto User) 생성
CN_CREATE_CO	CO(Crypto Officer)
CN_DELETE_USER	사용자 삭제
CN_CHANGE_PSWD	사용자의 암호 변경
CN_SET_M_VALUE	사용자 작업에 대한 쿼럼 인증 (M/N) 을 설정합니다.
CN_APPROVE_TOKEN	사용자 작업에 대한 쿼럼 인증 토큰 승인
CN_DELETE_TOKEN	쿼럼 토큰을 하나 이상 삭제합니다.
CN_GET_TOKEN	서명 토큰을 요청하여 쿼럼 작업을 시작하십시오.

작업 코드(Opcode)	설명
키 관리: 키 핸들이 이 이벤트에 포함됩니다.	
CN_GENERATE_KEY	대칭 키 생성
CN_GENERATE_KEY_PAIR (0x19)	비대칭 키 페어 생성
CN_CREATE_OBJECT	퍼블릭 키 가져오기(언래핑 안 함)
CN_MODIFY_OBJECT	키 속성 설정
CN_DESTROY_OBJECT (0x11)	세션 키 삭제
CN_TOMBSTONE_OBJECT	토큰 키 삭제
CN_SHARE_OBJECT	키 공유 또는 공유 해제
CN_WRAP_KEY	암호화된 키 복사본 내보내기(wrapKey)
CN_UNWRAP_KEY	암호화된 키 복사본 가져오기(unwrapKey)
CN_DERIVE_KEY	기존 키에서 대칭 키 도출
CN_NIST_AES_WRAP	AES 키를 사용하여 키를 암호화 또는 복호화합니다.
CN_INSERT_MASKED_OBJECT_USER	다른 HSM의 속성이 포함된 암호화된 키를 클러스터에 삽입합니다.
CN_EXTRACT_MASKED_OBJECT_USER	HSM의 속성으로 키를 래핑/암호화하여 클러스터의 다른 HSM으로 보냅니다.
Back up HSMs	
CN_BACKUP_BEGIN	백업 프로세스 시작
CN_BACKUP_END	백업 프로세스를 완료했습니다.
CN_RESTORE_BEGIN	백업에서 복원 시작
CN_RESTORE_END	백업에서 복원 프로세스를 완료했습니다.

작업 코드(Opcode)	설명
Certificate-Based Authentication	
`CN_CERT_AUTH_STORE_CERT`	클러스터 인증서를 저장합니다.
HSM Instance Commands	
`CN_INIT_TOKEN (0x1)`	HSM 초기화 프로세스 시작
`CN_INIT_DONE`	HSM 초기화 프로세스가 완료되었습니다.
`CN_GEN_KEY_ENC_KEY`	KEK(키 암호화 키) 생성
`CN_GEN_PSWD_ENC_KEY (0x1d)`	PEK(암호 암호화 키) 생성
HSM crypto commands	
`CN_FIPS_RAND`	FIPS 호환 난수 생성

에 대한 CloudWatch 지표 가져오기 AWS CloudHSM

AWS CloudHSM 클러스터를 실시간으로 모니터링하는 CloudWatch 데 사용합니다. 리전별, 클러스터 ID별 및 HSM ID별로 측정치를 그룹화할 수 있습니다.

`AWS/CloudHSM` 네임스페이스에는 다음 지표가 포함되어 있습니다.

지표	설명
HsmUnhealthy	HSM 인스턴스가 제대로 작동하지 않습니다. AWS CloudHSM 비정상 인스턴스를 자동으로 교체합니다. 사전에 클러스터 크기를 확장하도록 선택하여 HSM을 교체하는 동안 성능에 미치는 영향을 줄일 수 있습니다.
HsmTemperature [1]	하드웨어 프로세서의 접합 온도. 온도가 섭씨 110도에 도달하면 시스템이 종료됩니다.
HsmKeysSessionOccupied	HSM 인스턴스에서 사용되는 세션 키의 수.

지표	설명
HsmKeysTokenOccupied	HSM 인스턴스와 클러스터에서 사용되는 토큰 키의 수.
HsmSslCtxsOccupied [1]	HSM 인스턴스에 현재 설정된 end-to-end 암호화된 채널의 수. 최대 2048개의 채널이 허용됩니다.
HsmSessionCount	HSM 인스턴스에 열려 있는 연결의 수입니다. 최대 2048개가 허용됩니다. 기본적으로 클라이언트 데몬은 암호화된 채널 하나에서 각 HSM 인스턴스와 함께 두 개의 세션을 열도록 구성되어 있습니다. end-to-end AWS CloudHSM 또한 HSM의 상태를 모니터링하기 위해 HSM과 최대 2개의 연결을 열 수 있습니다.
HsmUsersAvailable	생성할 수 있는 추가 사용자의 수입니다. 이는 목록에 있는 최대 사용자 수 (목록 HsmUsersMax) 에서 현재까지 생성된 사용자를 뺀 값입니다.
HsmUsersMax [1]	HSM 인스턴스에서 생성할 수 있는 최대 사용자 수. 현재 이 값은 1024입니다.
InterfaceEth2OctetsInput [1]	현재까지 HSM으로 들어오는 트래픽의 누적 합계.
InterfaceEth2OctetsOutput [1]	현재까지 HSM으로 나가는 트래픽의 누적 합계.

- [1] hsm2m.medium에는 이 메트릭을 사용할 수 없습니다.

AWS CloudHSM 성능

프로덕션 클러스터의 경우 한 리전의 다양한 가용성 영역에 분산된 HSM 인스턴스가 두 개 이상 있어야 합니다. 클러스터의 부하 테스트를 통해 예상해야 하는 최대 부하를 확인한 다음 HSM을 하나 더 추가하여 고가용성을 보장하는 것이 좋습니다. 새로 생성된 키의 내구성이 필요한 애플리케이션의 경우 한 리전의 다양한 가용성 영역에 분산된 3개 이상의 HSM 인스턴스를 권장합니다.

성능 데이터

AWS CloudHSM 클러스터의 성능은 특정 워크로드에 따라 달라집니다. 성능을 높이기 위해 클러스터에 HSM 인스턴스를 추가할 수 있습니다. 성능은 EC2 인스턴스의 구성, 데이터 크기, 추가 애플리케이션 로드에 따라 달라질 수 있습니다. 확장 요구 사항을 결정하기 위해 애플리케이션 로드 테스트를 권장합니다.

다음 표는 hsm1.medium 인스턴스가 포함된 EC2 인스턴스에서 실행되는 일반적인 암호화 알고리즘의 대략적인 성능을 보여줍니다.

hsm1.medium의 성능 데이터

Operation	2-HSM 클러스터 [1]	3-HSM 클러스터 [2]	6-HSM 클러스터 [3]
RSA 2048-비트 기호	초당 2,000회	초당 3,000회	초당 5,000회
EC P256 사인	초당 500회	초당 750회	초당 1,500회

- [1] EC2 인스턴스와 동일한 AZ에 하나의 HSM이 있는 하나의 c4.large EC2 인스턴스에서 실행되는 Java 다중 스레드 애플리케이션이 포함된 2-HSM 클러스터.
- [2] EC2 인스턴스와 동일한 AZ에 하나의 HSM이 있는 하나의 c4.large EC2 인스턴스에서 실행되는 Java 다중 스레드 애플리케이션이 포함된 3-HSM 클러스터.
- [3] EC2 인스턴스와 동일한 AZ에 하나의 HSM이 있는 하나의 c4.large EC2 인스턴스에서 실행되는 Java 다중 스레드 애플리케이션이 포함된 6-HSM 클러스터.

HSM 스로틀링

워크로드가 클러스터의 HSM 용량을 초과하면 HSM이 사용 중이거나 병목 현상이 발생했다는 오류 메시지를 받을 겁니다. 이 경우 취해야 할 조치에 대한 자세한 내용은 HSM 스로틀링 단원을 참조하십시오.

보안 입력 AWS CloudHSM

클라우드 AWS 보안이 최우선 과제입니다. AWS 고객은 가장 보안에 민감한 조직의 요구 사항을 충족하도록 구축된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 기업과 기업 간의 AWS 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드 보안 - AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호하는 역할을 합니다. AWS AWS 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 적용되는 규정 준수 프로그램에 대해 자세히 알아보려면 규정 준수 프로그램별 AWS 범위 내 서비스 규정 준수 프로그램별 참조하십시오. AWS CloudHSM
- 클라우드에서의 보안 — 귀하의 책임은 사용하는 AWS 서비스에 따라 결정됩니다. 또한 귀하는 귀사의 데이터의 민감도, 귀사의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 공동 책임 모델을 사용할 때 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 AWS CloudHSM됩니다. 다음 항목에서는 보안 및 규정 준수 목표를 AWS CloudHSM 충족하도록 구성하는 방법을 보여줍니다. 또한 AWS CloudHSM 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스를 사용하는 방법도 배웁니다.

내용

데이터 보호: AWS CloudHSM

AWS 공동 책임 모델 의 데이터 보호에 적용됩니다 AWS CloudHSM. 이 모델에 설명된 대로 AWS 는 모든 데이터를 실행하는 글로벌 인프라를 보호하는 역할을 AWS 클라우드합니다. 사용자는 인프라에

서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스 의 보안 구성과 관리 작업에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하십시오.

데이터 보호를 위해 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center OR AWS Identity and Access Management (IAM) 을 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 멀티 팩터 인증 설정(MFA)을 사용하세요.
- SSL/TLS를 사용하여 리소스와 통신할 수 있습니다. AWS TLS 1.2는 필수이며 TLS 1.3를 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다. AWS CloudTrail
- 포함된 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용하십시오 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 AWS 통해 액세스할 때 FIPS 140-2로 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용하십시오. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하십시오.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔, API AWS CloudHSM 또는 AWS 서비스 SDK를 사용하거나 다른 방법으로 작업하는 경우가 포함됩니다. AWS CLI AWS 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 보안 인증 정보를 URL에 포함시켜서는 안 됩니다.

저장된 데이터 암호화

HSM에서 백업을 AWS CloudHSM 만들면 HSM은 데이터를 전송하기 전에 데이터를 암호화합니다. AWS CloudHSM데이터는 고유한 임시 암호화 키를 사용하여 암호화됩니다. 자세한 정보는 AWS CloudHSM 클러스터 백업을 참조하세요.

전송 중 데이터 암호화

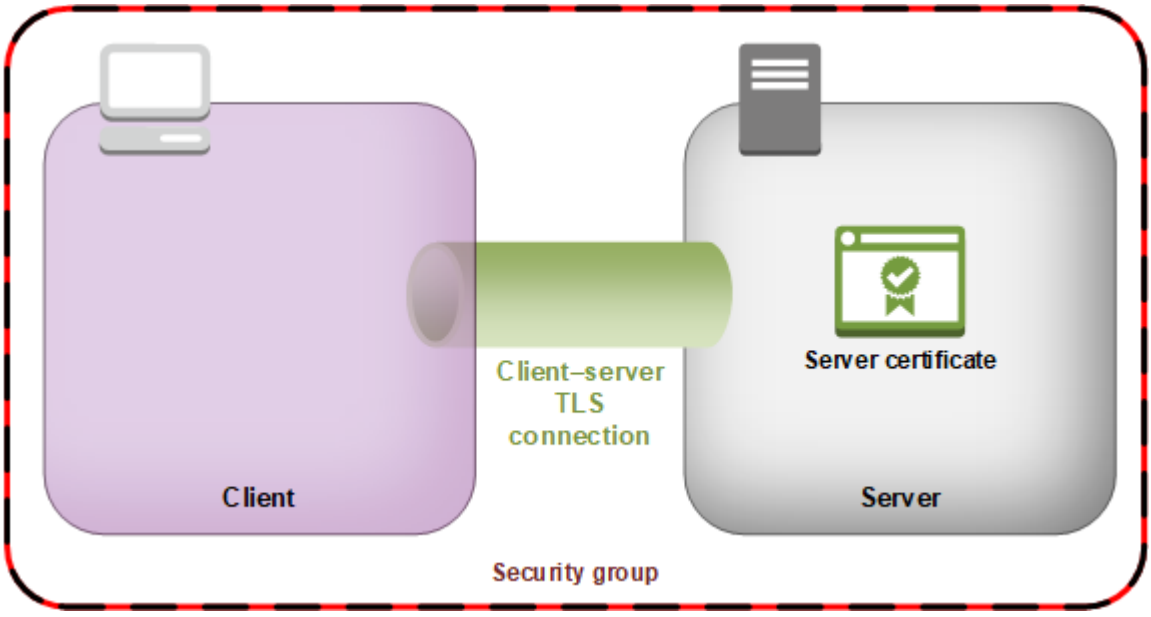
클러스터의 AWS CloudHSM 클라이언트와 HSM 간의 통신은 처음부터 끝까지 암호화됩니다. 이 통신은 클라이언트와 HSM에 의해서만 해독될 수 있습니다. 자세한 정보는 E nd-to-end 암호화을 참조하세요.

AWS CloudHSM 클라이언트 암호화 end-to-end

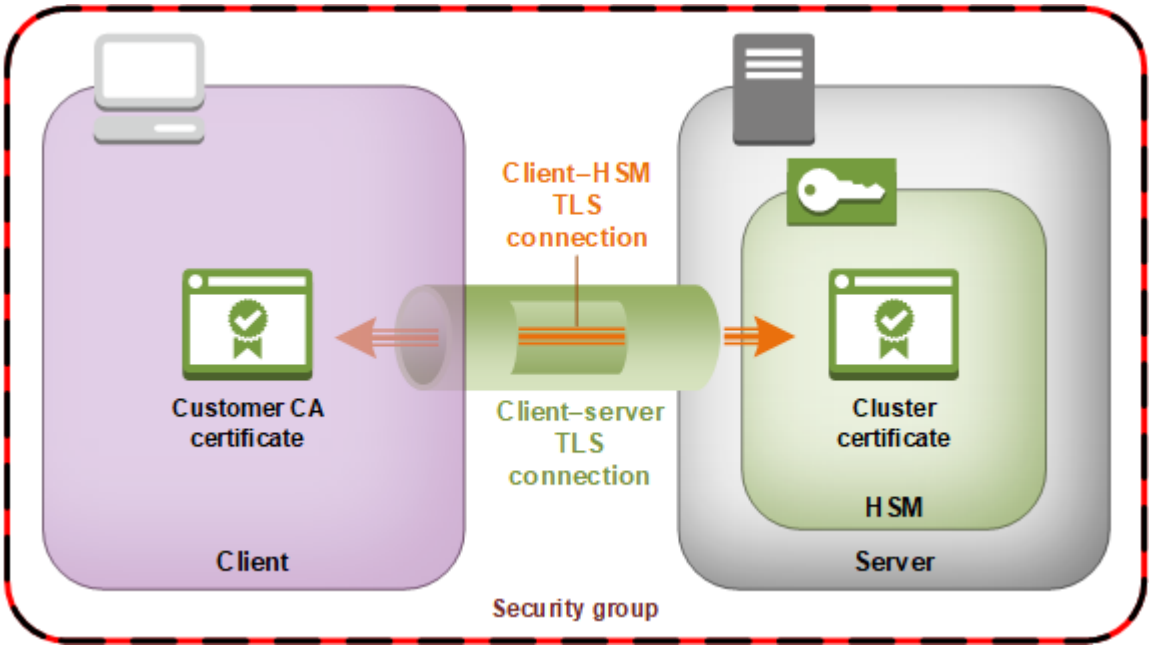
클러스터의 클라이언트 인스턴스와 HSM 간의 통신은 처음부터 끝까지 암호화됩니다. 해당 클라이언트와 HSM만 통신을 해독할 수 있습니다.

다음 프로세스는 클라이언트가 HSM과 end-to-end 암호화된 통신을 설정하는 방법을 설명합니다.

1. 클라이언트는 HSM 하드웨어를 호스팅하는 서버와 TLS(전송 계층 보안) 연결을 설정합니다. 클러스터 보안 그룹은 서버에 대해 보안 그룹의 클라이언트 인스턴스로부터의 인바운드 트래픽만 허용합니다. 클라이언트는 서버를 신뢰할 수 있는지 확인하기 위해 서버의 인증서도 확인합니다.



2. 그런 다음 클라이언트는 HSM 하드웨어와 암호화된 연결을 설정합니다. HSM에는 해당 인증 기관(CA)을 통해 서명한 클러스터 인증서가 있고, 클라이언트에는 CA의 루트 인증서가 있습니다. 클라이언트-HSM 암호화 연결이 설정되기 전에 클라이언트는 루트 인증서와 비교하여 HSM의 클러스터 인증서를 확인합니다. 연결은 클라이언트에서 HSM을 신뢰할 수 있다고 확인한 경우에만 이루어집니다.



클러스터 백업의 보안

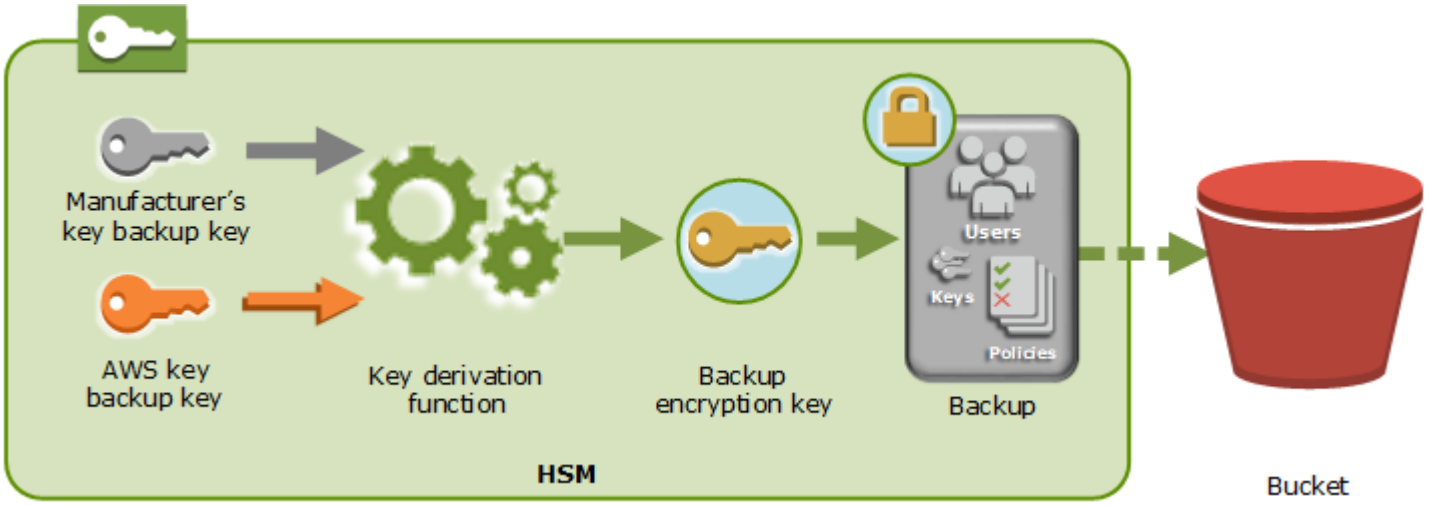
HSM에서 백업을 AWS CloudHSM 만들 때 HSM은 데이터를 보내기 전에 모든 데이터를 암호화합니다. AWS CloudHSM해당 데이터는 HSM을 일반 텍스트 형식으로 두지 않습니다. 또한 백업은 백업을 해독하는 데 사용된 키에 액세스할 수 AWS 없으므로 AWS 복호화할 수 없습니다.

데이터를 암호화하기 위해 HSM은 EBK(임시 백업 키)라는 고유한 임시 암호화 키를 사용합니다. EBK는 백업 시 HSM 내에서 생성되는 AES 256비트 암호화 키입니다. AWS CloudHSM HSM은 EBK를 생성한 다음 이를 사용하여 NIST Special Publication 800-38F를 준수하는 FIPS 승인 AES 키 래핑 방법으로 HSM의 데이터를 암호화합니다. 그러면 HSM이 암호화된 데이터를 에 제공합니다. AWS CloudHSM암호화된 데이터에는 EBK의 암호화된 복사본이 포함됩니다.

EBK를 암호화하기 위해 HSM은 PBK(영구 백업 키)라는 다른 암호화 키를 사용합니다. PBK도 AES 256비트 암호화 키입니다. PBK를 생성하기 위해 HSM은 NIST Special Publication 800-108을 준수하는 카운터 모드에서 FIPS 승인 키 추출 함수(KDF)를 사용합니다. 이 KDF의 입력 내용은 다음과 같습니다.

- 하드웨어 제조업체가 HSM 하드웨어에 영구적으로 삽입한 제조업체 키 백업 키(MKBK).
- 에 의해 처음 구성될 때 HSM에 안전하게 설치되는 AWS 키 백업 키 (AKBK). AWS CloudHSM

다음 그림에는 암호화 프로세스가 요약되어 있습니다. 백업 암호화 키는 PBK(영구 백업 키) 및 EBK(임시 백업 키)를 나타냅니다.



AWS CloudHSM 동일한 제조업체에서 AWS만든 소유의 HSM에만 백업을 복원할 수 있습니다. 각 백업에는 원본 HSM의 모든 사용자, 키 및 구성이 포함되므로 복원된 HSM에는 원본과 동일한 보호 및 액세스 제어가 포함됩니다. 복원된 데이터는 복원 전에 HSM에 있던 다른 모든 데이터를 덮어씁니다.

백업은 암호화된 데이터로만 구성됩니다. 서비스가 Amazon S3에 백업을 저장하기 전에 서비스는 AWS Key Management Service (AWS KMS) 를 사용하여 백업을 다시 암호화합니다.

자격 증명 및 액세스 관리: AWS CloudHSM

AWS는 보안 자격 증명을 사용하여 사용자를 식별하고 AWS 리소스에 대한 액세스 권한을 부여합니다. AWS Identity and Access Management (IAM) 기능을 사용하여 다른 사용자, 서비스 및 애플리케이션이 AWS 리소스를 완전히 또는 제한적으로 사용하도록 허용할 수 있습니다. 이를 위해 보안 자격 증명을 공유하지 않아도 됩니다.

기본적으로 IAM 사용자는 AWS 리소스를 생성, 확인 또는 수정할 수 있는 권한이 없습니다. IAM 사용자가 로드 밸런서와 같은 리소스에 액세스하여 작업을 수행하도록 허용하려면 다음을 수행하십시오.

1. IAM 사용자에게 필요한 API 작업 및 특정 리소스를 사용할 권한을 부여하는 IAM 정책을 생성합니다.
2. 정책을 IAM 사용자 또는 IAM 사용자가 속한 그룹에 연결합니다.

사용자 또는 사용자 그룹에 정책을 연결하면 지정된 리소스에 대해 지정된 작업을 수행할 권한이 허용되거나 거부됩니다.

예를 들어 IAM을 사용하여 AWS 계정 아래에 사용자 및 그룹을 생성할 수 있습니다. IAM 사용자는 사용자, 시스템 또는 애플리케이션입니다. 그런 다음 IAM 정책을 사용하여 지정된 리소스에 대한 특정 작업을 수행할 수 있도록 사용자 및 그룹에 권한을 부여합니다.

IAM 정책을 사용하여 권한 부여

사용자 또는 사용자 그룹에 정책을 연결하면 지정된 리소스에 대해 지정된 작업을 수행할 권한이 허용되거나 거부됩니다.

IAM 정책은 하나 이상의 문으로 구성된 JSON 문서입니다. 각 문은 다음 예와 같이 구성됩니다.

```
{
  "Version": "2012-10-17",
  "Statement":[{
    "Effect": "effect",
    "Action": "action",
    "Resource": "resource-arn",
    "Condition": {
      "condition": {
        "key":"value"
      }
    }
  }]
}
```

- Effect - effect는 `Allow` 또는 `Deny`일 수 있습니다. 기본적으로 IAM 사용자에게는 리소스 및 API 작업을 사용할 권한이 없으므로 모든 요청이 거부됩니다. 명시적 허용은 기본 설정을 무시합니다. 명시적 거부는 모든 허용을 무시합니다.
- 작업: 작업은 권한을 부여하거나 거부할 특정 API 작업입니다. 작업 지정에 대한 자세한 내용은 에 대한 API 작업 AWS CloudHSM 단원을 참조하십시오.
- 리소스 — 작업의 영향을 받는 리소스입니다. AWS CloudHSM 리소스 수준 권한을 지원하지 않습니다. * 와일드카드를 사용하여 모든 리소스를 지정해야 합니다. AWS CloudHSM
- 조건 - 정책이 시행 중일 때 관리하기 위해 조건을 선택적으로 사용할 수 있습니다. 자세한 내용은 에 대한 조건 키 AWS CloudHSM 단원을 참조하십시오.

자세한 내용은 IAM 사용 설명서를 참조하십시오.

에 대한 API 작업 AWS CloudHSM

IAM 정책 설명의 Action 요소에서 AWS CloudHSM 제공하는 모든 API 작업을 지정할 수 있습니다. 다음 예와 같이 작업 이름 앞에 접두사로 소문자 문자열 `cloudhsm:`을 붙여야 합니다.

```
"Action": "cloudhsm:DescribeClusters"
```

명령문 하나에 여러 작업을 지정하려면 다음 예제와 같이 대괄호로 묶은 후 각 작업을 쉼표로 구분합니다.

```
"Action": [
    "cloudhsm:DescribeClusters",
    "cloudhsm:DescribeHsm"
]
```

* 와일드카드를 사용하여 여러 작업을 지정할 수도 있습니다. 다음 예시에서는 로 `List` 시작하는 모든 API 작업 이름을 지정합니다 AWS CloudHSM .

```
"Action": "cloudhsm:List*"
```

에 대한 AWS CloudHSM모든 API 작업을 지정하려면 다음 예와 같이* 와일드카드를 사용합니다.

```
"Action": "cloudhsm:*"
```

에 대한 API 작업 목록은 AWS CloudHSM 작업을 참조하십시오. AWS CloudHSM

에 대한 조건 키 AWS CloudHSM

정책을 만들 때 정책 적용 시기를 제어하는 조건을 지정할 수 있습니다. 각 조건에는 하나 이상의 키-값 쌍이 포함됩니다. 조건 키에는 전역 조건 키와 서비스별 조건 키가 있습니다.

AWS CloudHSM 서비스별 컨텍스트 키가 없습니다.

전역 조건 키에 대한 자세한 내용은 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하십시오.

에 대한 사전 정의된 AWS 관리형 정책 AWS CloudHSM

AWS가 생성한 관리형 정책은 일반 사용 사례에서 필요한 권한을 부여합니다. AWS CloudHSM 에 필요한 액세스를 기반으로 IAM 사용자에게 이러한 정책을 연결할 수 있습니다.

- AWSCloudHSMFullAccess— AWS CloudHSM 기능을 사용하는 데 필요한 전체 액세스 권한을 부여합니다.
- AWSCloudHSMReadOnlyAccess— AWS CloudHSM 기능에 대한 읽기 전용 액세스 권한을 부여합니다.

고객 관리형 정책은 다음과 같습니다. AWS CloudHSM

AWS CloudHSM실행에 필요한 권한만 AWS CloudHSM 포함하는 IAM 관리자 그룹을 생성하는 것이 좋습니다. 적절한 권한이 있는 정책을 이 그룹에 연결합니다. 필요에 따라 그룹에 IAM 사용자를 추가합니다. 추가한 각 사용자는 관리자 그룹에서 정책을 상속합니다.

또한 사용자에게 필요한 권한에 따라 추가 사용자 그룹을 생성하는 것이 좋습니다. 이렇게 하면 신뢰할 수 있는 사용자만 중요한 API 작업에 액세스할 수 있습니다. 예를 들어 사용자 그룹을 생성하고 이를 사용하여 클러스터 및 HSM에 대한 읽기 전용 액세스 권한을 부여할 수 있습니다. 이 그룹에서는 사용자에게 클러스터 또는 HSM을 삭제할 수 있도록 허용하지 않으므로 신뢰할 수 없는 사용자가 프로덕션 워크로드의 가용성에 영향을 줄 수 없습니다.

시간이 지남에 따라 새로운 AWS CloudHSM 관리 기능이 추가되므로 신뢰할 수 있는 사용자에게만 즉시 액세스 권한을 부여할 수 있습니다. 나중에 생성할 때 제한된 권한을 정책에 할당하여 새 기능 권한을 이들 신뢰할 수 있는 사용자에게 수동으로 할당할 수 있습니다.

다음은 에 대한 예제 AWS CloudHSM정책입니다. 정책을 생성하고 IAM 사용자 그룹에 연결하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 JSON 탭에서 정책 생성을 참조하십시오.

예

- 읽기 전용 권한
- 파워 유저 권한
- 관리자 권한

Example 예: 읽기 전용 권한

이 정책은 `DescribeClusters` 및 `DescribeBackups` API 작업에 대한 액세스를 허용합니다. 또한 특정 Amazon EC2 API 작업에 대한 추가 권한도 포함됩니다. 하지만 사용자에게 클러스터 또는 HSM을 삭제하도록 허용하지는 않습니다.

```
{
```

```
   "Version": "2012-10-17",
   "Statement": {
      "Effect": "Allow",
      "Action": [
         "cloudhsm:DescribeClusters",
         "cloudhsm:DescribeBackups",
         "cloudhsm:ListTags"
      ],
      "Resource": "*"
   }
}
```

Example 예: 고급 사용자 권한

이 정책은 AWS CloudHSM API 작업의 하위 집합에 대한 액세스를 허용합니다. 또한 특정 Amazon EC2 작업에 대한 추가 권한도 포함됩니다. 하지만 사용자에게 클러스터 또는 HSM을 삭제하도록 허용하지는 않습니다. 계정에 AWSServiceRoleForCloudHSM서비스 연결 역할을 자동으로 생성할 수 AWS CloudHSM 있는 `iam:CreateServiceLinkedRole` 작업을 포함해야 합니다. 이 역할을 통해 이벤트를 AWS CloudHSM 기록할 수 있습니다. 자세한 정보는 서비스 연결 역할은 다음과 같습니다. AWS CloudHSM을 참조하세요.

> **Note**
>
> 각 API의 특정 권한을 보려면 서비스 승인 참조의 작업, 리소스 및 조건 키 AWS CloudHSM를 참조하십시오.

```
{
   "Version": "2012-10-17",
   "Statement": {
      "Effect": "Allow",
      "Action": [
         "cloudhsm:DescribeClusters",
         "cloudhsm:DescribeBackups",
         "cloudhsm:CreateCluster",
         "cloudhsm:CreateHsm",
         "cloudhsm:RestoreBackup",
         "cloudhsm:CopyBackupToRegion",
         "cloudhsm:InitializeCluster",
         "cloudhsm:ListTags",
         "cloudhsm:TagResource",
```

```
            "cloudhsm:UntagResource",
            "ec2:CreateNetworkInterface",
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeNetworkInterfaceAttribute",
            "ec2:DetachNetworkInterface",
            "ec2:DeleteNetworkInterface",
            "ec2:CreateSecurityGroup",
            "ec2:AuthorizeSecurityGroupIngress",
            "ec2:AuthorizeSecurityGroupEgress",
            "ec2:RevokeSecurityGroupEgress",
            "ec2:DescribeSecurityGroups",
            "ec2:DeleteSecurityGroup",
            "ec2:CreateTags",
            "ec2:DescribeVpcs",
            "ec2:DescribeSubnets",
            "iam:CreateServiceLinkedRole"
        ],
        "Resource": "*"
    }
}
```

Example 예: 관리자 권한

이 정책은 HSM 및 클러스터를 삭제하는 작업을 포함한 모든 AWS CloudHSM API 작업에 대한 액세스를 허용합니다. 또한 특정 Amazon EC2 작업에 대한 추가 권한도 포함됩니다. 계정에서 AWSServiceRoleForCloudHSM서비스 연결 역할을 자동으로 생성할 수 있도록 AWS CloudHSM 하는 `iam:CreateServiceLinkedRole` 작업을 포함해야 합니다. 이 역할을 통해 이벤트를 AWS CloudHSM 기록할 수 있습니다. 자세한 정보는 서비스 연결 역할은 다음과 같습니다. AWS CloudHSM을 참조하세요.

```
{
   "Version":"2012-10-17",
   "Statement":{
      "Effect":"Allow",
      "Action":[
         "cloudhsm:*",
         "ec2:CreateNetworkInterface",
         "ec2:DescribeNetworkInterfaces",
         "ec2:DescribeNetworkInterfaceAttribute",
         "ec2:DetachNetworkInterface",
         "ec2:DeleteNetworkInterface",
         "ec2:CreateSecurityGroup",
         "ec2:AuthorizeSecurityGroupIngress",
```

```
          "ec2:AuthorizeSecurityGroupEgress",
          "ec2:RevokeSecurityGroupEgress",
          "ec2:DescribeSecurityGroups",
          "ec2:DeleteSecurityGroup",
          "ec2:CreateTags",
          "ec2:DescribeVpcs",
          "ec2:DescribeSubnets",
          "iam:CreateServiceLinkedRole"
      ],
      "Resource":"*"
   }
}
```

서비스 연결 역할은 다음과 같습니다. AWS CloudHSM

작업을 고객 관리형 정책은 다음과 같습니다. AWS CloudHSM 포함하기 위해 이전에 만든 IAM 정책입니다. `iam:CreateServiceLinkedRole` AWS CloudHSM 라는 서비스 연결 역할을 정의합니다. AWSServiceRoleForCloudHSM 역할은 미리 AWS CloudHSM 정의되며 사용자를 대신하여 다른 AWS 서비스를 AWS CloudHSM 호출해야 하는 권한을 포함합니다. 이 역할을 사용하면 수동으로 역할 정책 및 신뢰 정책 권한을 추가할 필요가 없으므로 서비스를 더 쉽게 설정할 수 있습니다.

역할 정책을 통해 AWS CloudHSM Amazon CloudWatch Logs 로그 그룹과 로그 스트림을 생성하고 사용자 대신 로그 이벤트를 작성할 수 있습니다. 아래 및 IAM 콘솔에서 볼 수 있습니다.

```
{
    "Version": "2018-06-12",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

역할에 대한 신뢰 정책에 따라 AWSServiceRoleForCloudHSM역할을 AWS CloudHSM 맡을 수 있습니다.

```
{
  "Version": "2018-06-12",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudhsm.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

서비스 연결 역할 생성(자동)

AWS CloudHSM AWS CloudHSM 관리자 그룹을 만들 때 정의한 권한에 `iam:CreateServiceLinkedRole` 작업을 포함하는 경우 클러스터를 만들 때 AWSServiceRoleForCloudHSM역할을 생성합니다. 고객 관리형 정책은 다음과 같습니다. AWS CloudHSM 섹션을 참조하십시오.

이미 하나 이상의 클러스터가 있는데 AWSServiceRoleForCloudHSM역할만 추가하려는 경우 콘솔, create-cluster 명령 또는 CreateClusterAPI 작업을 사용하여 클러스터를 생성할 수 있습니다. 그런 다음 콘솔, 클러스터 삭제 명령 또는 DeleteClusterAPI 작업을 사용하여 삭제합니다. 새로운 클러스터를 만들면 서비스 연결 역할이 생성되며 계정의 모든 클러스터에 이 역할이 적용됩니다. 또는 수동으로 역할을 만들 수도 있습니다. 자세한 내용은 다음 단원을 참조하십시오.

> **Note**
>
> 역할을 추가하기 위한 클러스터만 생성하는 경우에는 클러스터를 시작하기 AWS CloudHSM 생성하기 위해 설명된 모든 단계를 수행할 필요가 없습니다. AWSServiceRoleForCloudHSM

서비스 연결 역할 생성(수동)

IAM 콘솔 또는 API를 사용하여 AWS CLI역할을 생성할 수 있습니다. AWSServiceRoleForCloudHSM 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 생성 단원을 참조하십시오.

서비스 연결 역할 편집

AWS CloudHSM AWSServiceRoleForCloudHSM역할을 편집할 수 없습니다. 예를 들어, 역할이 생성된 후에는 여러 엔터티에서 이 역할을 이름으로 참조할 수 있으므로 이름을 변경할 수 없습니다. 역할 정책도 변경할 수 없습니다. 그러나 IAM을 사용하여 역할 설명을 편집할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 편집을 참조하십시오.

서비스 연결 역할 삭제

서비스 연결 역할이 적용되는 클러스터가 존재하는 동안에는 이 역할을 삭제할 수 없습니다. 이 역할을 삭제하려면 먼저 클러스터의 각 HSM을 삭제한 후 클러스터를 삭제해야 합니다. 계정의 모든 클러스터를 삭제해야 합니다. 그런 다음 IAM 콘솔 또는 API를 사용하여 역할을 삭제할 수 있습니다. AWS CLI 클러스터 삭제에 대한 자세한 내용은 AWS CloudHSM 클러스터 삭제를 참조하십시오. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 삭제 단원을 참조하십시오.

규정 준수

FIPS 모드의 클러스터의 경우 PCI-PIN, PCI-3DS 및 AWS CloudHSM SOC2 규정 준수 요구 사항을 충족하는 FIPS 승인 HSM을 제공합니다. AWS CloudHSM 또한 고객에게 비 FIPS 모드인 클러스터를 선택할 수 있는 옵션도 제공합니다. 각각에 적용되는 인증 및 규정 준수 요구 사항에 대한 자세한 내용은 을 참조하십시오. AWS CloudHSM 클러스터 모드 및 HSM 유형

FIPS 검증을 거친 HSM을 사용하면 클라우드의 데이터 보안에 대한 기업, 계약 및 규제 준수 요구 사항을 충족하는 데 도움이 될 수 있습니다. AWS

FIPS 140-2 규정 준수

Federal Information Processing Standard(FIPS) Publication 140-2는 미국 정부 보안 표준으로서, 기밀 정보를 보호하는 암호 모듈의 보안 요건을 규정하고 있습니다. 에서 제공하는 hsm1.medium HSM 유형은 FIPS 140-2 레벨 3 인증을 받았습니다 (인증서 #4218). AWS CloudHSM 자세한 내용은 하드웨어에 대한 FIPS 검증을 참조하십시오.

PCI DSS 준수

PCI DSS(신용카드 업계 데이터 보안 표준)는 PCI 보안 표준 위원회에서 관리하는 독점적인 정보 보안 표준입니다. 에서 제공하는 HSM은 PCI AWS CloudHSM DSS를 준수합니다.

PCI PIN 준수

PCI PIN은 ATM 및 (POS) 터미널에서 거래에 사용되는 정보인 PIN (개인 식별 번호) 데이터를 전송, 처리 및 관리하기 위한 보안 요구 사항 및 point-of-sale 평가 표준을 제공합니다. 에서 제공하는

hsm1.medium HSM은 2023년 1월부터 PCI AWS CloudHSM PIN을 준수하고 있습니다. 자세한 내용은 AWS CloudHSM이 이제 PCI PIN 인증 문서를 참조하십시오

PCI-3DS 준수

PCI 3DS(또는 쓰리 도메인 보안, 3D 보안)는 EMV 3D 보안 전자 상거래 결제를 위한 데이터 보안을 제공합니다. PCI 3DS는 온라인 쇼핑을 위한 또 다른 보안 계층을 제공합니다. 에서 제공하는 hsm1.medium HSM 유형은 PCI-3DS 규격을 준수합니다. AWS CloudHSM

SOC2

SOC2는 서비스 조직이 클라우드 및 데이터센터 보안 제어를 시연할 수 있도록 지원하는 프레임워크입니다. AWS CloudHSM은 신뢰할 수 있는 서비스 원칙을 준수하기 위해 중요 영역에 SOC2 제어를 구현했습니다. 자세한 내용은 AWS SOC FAQ 페이지를 참조하십시오.

AWS CloudHSM PCI-PIN 규정 준수 관련 자주 묻는 질문

PCI PIN은 ATM 및 (POS) 터미널에서 거래에 사용되는 정보인 PIN (개인 식별 번호) 데이터를 전송, 처리 및 관리하기 위한 보안 요구 사항 및 point-of-sale 평가 표준을 제공합니다.

PCI-PIN 규정 준수 증명(AOC) 및 책임 요약은 AWS 규정 준수 보고서에 대한 온디맨드 액세스를 위한 셀프 서비스 포털인 AWS Artifact를 통해 고객에게 제공됩니다. 자세한 내용은 AWS Management Console에서 AWS Artifact에 로그인하거나 AWS Artifact로 시작하기에서 자세히 알아보십시오.

FAQ

Q: 규정 준수 및 책임 증명 요약이란 무엇입니까?

규정 준수 증명 (AOC) 은 PCI-PIN 표준의 해당 규제 AWS CloudHSM 기준을 충족하는지 검증하는 공인 PIN 평가자 (QPA) 가 작성합니다. 책임 요약 매트릭스는 각 책임과 고객이 각각 담당하는 규제 항목을 설명합니다. AWS CloudHSM

Q: 규정 준수 증명을 받으려면 어떻게 해야 합니까? AWS CloudHSM

PCI-PIN 규정 준수 증명(AOC)은 AWS 규정 준수 보고서에 대한 온디맨드 액세스를 위한 셀프 서비스 포털인 AWS Artifact를 통해 고객에게 제공됩니다. 자세한 내용은 AWS Management Console에서 AWS Artifact에 로그인하거나 AWS Artifact로 시작하기에서 자세히 알아보십시오.

Q: 어떤 PCI PIN 제어를 담당하는지 어떻게 알 수 있습니까?

자세한 내용은 AWS PCI PIN 규정 준수 패키지의 “AWS CloudHSM PCI PIN 책임 요약”을 참조하십시오. 이 패키지는 고객이 AWS 규정 준수 보고서에 온디맨드로 액세스할 수 있는 셀프 서비스 포털인

AWS Artifact를 통해 제공됩니다. 자세한 내용은 AWS Management Console에서 AWS Artifact에 로그인하거나 AWS Artifact로 시작하기에서 자세히 알아보십시오.

Q: AWS CloudHSM 고객으로서 PCI-PIN 규정 준수 증명 (AOC) 을 신뢰할 수 있습니까?

고객은 자체 PCI-PIN 규정 준수를 관리해야 합니다. 귀하의 지불 워크로드가 모든 PCI-PIN 제어/요구 사항을 충족하는지 확인하려면 QPA(Qualified PIN Assessor)를 통해 공식 PCI-PIN 증명 프로세스를 거쳐야 합니다. 하지만 AWS가 담당하는 제어의 경우 QPA는 추가 테스트 없이 규정 준수 AWS CloudHSM 증명 (AOC) 을 신뢰할 수 있습니다.

Q: 키 관리 수명 AWS CloudHSM 주기와 관련된 PCI-PIN 요구 사항에 대한 책임이 있습니까?

AWS CloudHSM HSM의 물리적 디바이스 수명 주기를 담당합니다. 고객은 PCI-PIN 표준의 주요 관리 수명주기 요구 사항에 대한 책임이 있습니다.

Q: AWS CloudHSM PCI-PIN을 준수하는 컨트롤은 무엇입니까?

AOC에는 QPA가 평가하는 AWS CloudHSM 컨트롤이 요약되어 있습니다. PCI-PIN 책임 요약은 AWS 규정 준수 보고서에 대한 온디맨드 액세스를 위한 셀프 서비스 포털인 AWS Artifact를 통해 고객에게 제공됩니다.

Q: PIN 번역 및 DUKPT와 같은 결제 기능을 AWS CloudHSM 지원합니까?

아니요. 범용 AWS CloudHSM HSM을 제공합니다. 시간이 지나면 결제 기능을 제공할 수 있습니다. 서비스가 직접 결제 기능을 수행하지는 않지만 AWS CloudHSM PCI PIN 규정 준수 증명을 통해 고객은 실행 중인 서비스에 대해 자체적으로 PCI 규정을 준수할 수 있습니다. AWS CloudHSM워크로드에 AWS Payment Cryptography 서비스를 사용하는 데 관심이 있는 경우 “AWS Payment Cryptography를 사용하여 결제 처리를 클라우드로 이동” 블로그를 참조하십시오.

지원 중단 알림

FIPS 140, PCI-DSS, PCI-PIN, PCI-3DS 및 SOC2의 요구 사항을 계속 준수하기 위해 때때로 기능이 더 이상 사용되지 AWS CloudHSM 않을 수 있습니다. 이 페이지에는 현재 적용되는 변경 사항이 나열되어 있습니다.

FIPS 140 규정 준수: 2024 메커니즘 지원 중단

미국 국립표준기술연구소(NIST)[1]는 2023년 12월 31일 이후에는 Triple DES(DESede, 3DES, DES3) 암호화 및 PKCS#1 v1.5 패딩을 사용한 RSA 키 래핑 및 언래핑 지원이 허용되지 않을 것을 권고합니다. 따라서 연방 정보 처리 표준 (FIPS) 모드 클러스터에서는 2024년 1월 1일에 지원이 종료됩니다. 비 FIPS 모드의 클러스터에 대한 지원은 그대로 유지됩니다.

이 지침은 다음 암호화 작업에 적용됩니다.

- 트리플 DES 키 생성
  - PKCS #11 라이브러리용 CKM_DES3_KEY_GEN
  - JCE 공급자를 위한 DESede 키젠
  - KMU용 -t=21을 포함한 genSymKey
- 트리플 DES 키를 사용한 암호화(참고: 복호화 작업 허용)
  - PKCS #11 라이브러리용: CKM_DES3_CBC 암호화, CKM_DES3_CBC_PAD 암호화 및 CKM_DES3_ECB 암호화
  - JCE 공급자의 경우: DESede/CBC/PKCS5Padding 암호화, DESede/CBC/NoPadding 암호화, DESede/ECB/Padding 암호화 및 DESede/ECB/NoPadding 암호화
- PKCS #1 v1.5 패딩을 사용한 RSA 키 랩, 언래핑, 암호화 및 복호화
  - PKCS #11 SDK용 CKM_RSA_PKCS 래핑, 언래핑, 암호화 및 복호화
  - JCE SDK용 RSA/ECB/PKCS1Padding 래핑, 언래핑, 암호화 및 복호화
  - KMU의 경우 wrapKey와 unWrapKey, -m 12 (참고 12는 메커니즘 RSA_PKCS의 값입니다)

[1] 이 변경에 대한 자세한 내용은 암호화 알고리즘 및 키 길이 사용 전환의 표 1 및 표 5를 참조하십시오.

의 레질리언스 AWS CloudHSM

AWS 글로벌 인프라는 AWS 지역 및 가용 영역을 중심으로 구축됩니다. AWS 지역은 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며, 이러한 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹으로 연결됩니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 지역 및 가용 영역에 대한 자세한 내용은 글로벌 인프라를 참조하십시오AWS . 복원성을 지원하는 AWS CloudHSM 기능에 대한 자세한 내용은 클러스터 고가용성 및 로드 밸런싱 단원을 참조하십시오.

의 인프라 보안 AWS CloudHSM

관리형 서비스로서 Amazon Web Services: 보안 프로세스 개요 백서에 설명된 AWS 글로벌 네트워크 보안 절차에 따라 보호됩니다. AWS CloudHSM

AWS 게시된 API 호출을 사용하여 네트워크를 AWS CloudHSM 통해 액세스할 수 있습니다. 또한 요청은 액세스 키 ID 및 IAM 주체와 관련된 비밀 액세스 키를 사용하여 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)을 사용하여 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

네트워크 격리

Virtual Private Cloud(VPC)는 AWS 클라우드에서 논리적으로 격리된 고유한 영역의 가상 네트워크입니다. VPC의 프라이빗 서브넷에 클러스터를 생성할 수 있습니다. VPC를 생성할 때 프라이빗 서브넷을 생성할 수 있습니다. 자세한 정보는 Virtual Private Cloud(VPC) 생성을 참조하세요.

HSM을 생성할 때는 HSM과 상호 작용할 수 있도록 서브넷에 ENI (엘라스틱 네트워크 인터페이스) 를 추가하십시오. AWS CloudHSM 자세한 정보는 클러스터 아키텍처을 참조하세요.

AWS CloudHSM 클러스터 내 HSM 간의 인바운드 및 아웃바운드 통신을 허용하는 보안 그룹을 생성합니다. 이 보안 그룹을 사용하여 EC2 인스턴스가 클러스터의 HSM과 통신하도록 할 수 있습니다. 자세한 정보는 클라이언트 Amazon EC2 인스턴스 보안 그룹 구성을 참조하세요.

사용자 승인

를 사용하면 AWS CloudHSM HSM에서 작업을 수행하려면 인증된 HSM 사용자의 자격 증명이 필요합니다. 자세한 정보는 the section called “HSM 사용자 이해”을 참조하세요.

AWS CloudHSM 및 VPC 엔드포인트

인터페이스 VPC 엔드포인트를 AWS CloudHSM 생성하여 VPC 간에 프라이빗 연결을 설정할 수 있습니다. 인터페이스 엔드포인트는 인터넷 게이트웨이 AWS PrivateLink, NAT 디바이스, VPN 연결 또는 AWS Direct Connect 연결 없이 비공개로 AWS CloudHSM API에 액세스할 수 있는 기술인 에 의해 구동됩니다. VPC의 인스턴스는 API와 AWS CloudHSM 통신하는 데 퍼블릭 IP 주소가 필요하지 않습니다. VPC와 AWS CloudHSM 간의 트래픽은 Amazon 네트워크를 벗어나지 않습니다.

각 인터페이스 엔드포인트는 서브넷에서 하나 이상의 탄력적 네트워크 인터페이스로 표현됩니다.

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 VPC 엔드포인트 (AWS PrivateLink) 를 참조하십시오.

AWS CloudHSM VPC 엔드포인트 고려 사항

에 대한 AWS CloudHSM인터페이스 VPC 엔드포인트를 설정하기 전에 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 속성 및 제한을 검토하십시오.

- AWS CloudHSM VPC에서 모든 API 작업에 대한 호출을 지원합니다.

AWS CloudHSM에 대한 인터페이스 VPC 엔드포인트 생성

Amazon VPC 콘솔 또는 () 를 사용하여 AWS CloudHSM 서비스에 대한 VPC 엔드포인트를 생성할 수 있습니다. AWS Command Line Interface AWS CLI자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하세요.

에 대한 AWS CloudHSM VPC 엔드포인트를 만들려면 다음 서비스 이름을 사용합니다.

```
com.amazonaws.region.cloudhsmv2
```

예를 들어 미국 서부(오레곤) 리전(us-west-2)에서 서비스 이름은 다음과 같습니다.

```
com.amazonaws.us-west-2.cloudhsmv2
```

VPC 엔드포인트를 더 쉽게 사용하려면 VPC 엔드포인트에 프라이빗 DNS 호스트 이름을 사용하도록 설정합니다. 프라이빗 DNS 이름 활성화 옵션을 선택하면 표준 AWS CloudHSM DNS 호스트 이름 (https://cloudhsmv2.*<region>*.amazonaws.com) 이 VPC 엔드포인트로 확인됩니다.

이 옵션을 선택하면 VPC 엔드포인트를 더 쉽게 사용할 수 있습니다. AWS SDK는 기본적으로 표준 AWS CloudHSM DNS 호스트 이름을 AWS CLI 사용하므로 애플리케이션 및 명령에서 VPC 엔드포인트 URL을 지정할 필요가 없습니다.

자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트를 통해 서비스 액세스를 참조하세요.

에 대한 VPC 엔드포인트 정책 생성 AWS CloudHSM

AWS CloudHSM에 대한 액세스를 제어하는 VPC 엔드포인트에 엔드포인트 정책을 연결할 수 있습니다. 이 정책은 다음 정보를 지정합니다.

- 작업을 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업.
- 작업을 수행할 수 있는 리소스.

자세한 정보는 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하십시오.

예: 작업에 대한 VPC 엔드포인트 정책 AWS CloudHSM

다음은 에 대한 AWS CloudHSM엔드포인트 정책의 예입니다. 엔드포인트에 연결할 경우 이 정책은 모든 리소스의 모든 보안 주체에 대해 나열된 AWS CloudHSM 작업에 대한 액세스 권한을 부여합니다. 기타 AWS CloudHSM 작업 및 해당 IAM 권한은 을 자격 증명 및 액세스 관리: AWS CloudHSM 참조하십시오.

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "cloudhsm:DescribeBackups",
            "cloudhsm:DescribeClusters",
            "cloudhsm:ListTags",
         ],
         "Resource":"*"
      }
   ]
}
```

의 업데이트 관리 AWS CloudHSM

AWS에서 펌웨어를 관리합니다. 펌웨어는 타사에서 유지 관리하며, FIPS 140-2 레벨 3 규정을 준수하는지 NIST의 검증을 받아야 합니다. FIPS 키에 의해 암호화 방식으로 서명된 펌웨어만 설치할 수 있으며, AWS는 이 키에 대한 액세스 권한이 없습니다.

문제 해결 AWS CloudHSM

에서 문제가 발생하는 AWS CloudHSM경우 다음 항목을 참조하여 문제를 해결할 수 있습니다.

주제

알려진 문제

AWS CloudHSM 에는 다음과 같은 알려진 문제가 있습니다. 주제 영역을 선택하여 자세히 알아보십시오.

주제

모든 HSM 인스턴스의 알려진 문제

다음 문제는 key_mgmt_util 명령줄 도구를 사용하든, PKCS #11 SDK를 사용하든, JCE SDK를 사용하든, OpenSSL SDK를 사용하든 관계없이 모든 AWS CloudHSM 사용자에게 영향을 미칩니다.

주제

문제: 제로 패딩으로 키 래핑의 표준 호환 구현을 제공하는 대신 AES 키 래핑에 PKCS#5 패딩이 사용됩니다.

또한 패딩이 없는 키 래핑 및 제로 패딩이 지원되지 않습니다.

- 영향: 내에서 이 알고리즘을 사용하여 래핑하고 래핑을 해제해도 아무런 영향이 없습니다. AWS CloudHSM그러나 로 래핑된 키는 패딩 없음 사양을 준수할 것으로 예상되는 다른 HSM 또는 소프트웨어 내에서 래핑을 해제할 수 AWS CloudHSM 없습니다. 표준 호환 언래핑 중에 8바이트의 패딩 데이터가 키 데이터 끝에 추가될 수 있기 때문입니다. 외부에서 래핑된 키는 인스턴스로 제대로 언래핑할 수 없습니다. AWS CloudHSM

- 해결 방법: AWS CloudHSM 인스턴스에서 PKCS #5 패딩이 있는 AES 키 래핑으로 래핑된 키를 외부에서 언래핑하려는 경우 키를 사용하려고 시도하기 전에 추가 패딩을 제거합니다. 파일 편집기에서 추가 바이트를 잘라내거나 키 바이트만 코드의 새 버퍼로 복사하여 추가 패딩을 제거할 수 있습니다.
- 해결 상태: 3.1.0 클라이언트 및 소프트웨어 릴리스에서 AWS CloudHSM 은 AES 키 래핑을 위한 표준 호환 옵션을 제공합니다. 자세한 내용은 AES 키 래핑을 참조하세요.

문제: 클라이언트 데몬이 클러스터에 성공적으로 연결하려면 구성 파일에서 최소 하나의 유효한 IP 주소가 필요합니다.

- 영향: 클러스터에서 모든 HSM을 삭제한 다음 새 IP 주소를 가져오는 다른 HSM을 추가하면 클라이언트 데몬은 원래의 IP 주소에서 HSM을 계속 검색합니다.
- 해결 방법: 간헐적인 워크로드를 실행하는 경우 `IpAddress` CreateHsm함수의 인수를 사용하여 Elastic network interface (ENI) 를 원래 값으로 설정하는 것이 좋습니다. ENI는 가용 영역(AZ)에 특정합니다. 대체 방법은 `/opt/cloudhsm/daemon/1/cluster.info` 파일을 삭제한 다음 클라이언트 구성을 새 HSM의 IP 주소로 재설정하는 것입니다. `client -a <IP address>` 명령을 사용할 수 있습니다. 자세한 내용은 클라이언트 설치 및 구성 (Linux) 또는 AWS CloudHSM 클라이언트 설치 및 구성 (Windows) 을 AWS CloudHSM 참조하십시오.

문제: 클라이언트 SDK 3을 AWS CloudHSM 사용하여 해시하고 서명할 수 있는 데이터의 상한은 16KB였습니다.

- 해결 상태: 크기가 16KB 미만인 데이터는 해싱을 위해 계속해서 HSM에 전송됩니다. 소프트웨어에서 크기가 16KB ~ 64KB인 데이터를 로컬 해싱하는 기능이 추가되었습니다. 데이터 버퍼가 64KB보다 크면 클라이언트 SDK 5가 명시적으로 실패합니다. 수정 사항을 적용하려면 클라이언트와 SDK를 5.0.0 이상 버전으로 업데이트해야 합니다.

문제: 가져온 키를 내보낼 수 없도록 지정할 수 없습니다.

- 해결 상태: 이 문제가 수정되었습니다. 수정의 이점을 누리기 위해 사용자가 특별히 할 일은 없습니다.

문제: key_mgmt_util의 WrapKey unWrapKey 및 명령에 대한 기본 메커니즘이 제거되었습니다.

- 해결 방법: WrapKey unWrapKey 또는 명령을 사용하는 경우 옵션을 사용하여 메커니즘을 `-m` 지정해야 합니다. 자세한 내용은 WrapKey의 예제 unWrapKey또는 문서를 참조하십시오.

문제: 클러스터에 HSM이 하나 있는 경우, HSM 장애 조치가 올바르게 작동하지 않습니다.

- 영향: 클러스터의 단일 HSM 인스턴스가 연결이 끊어지면, HSM 인스턴스가 나중에 복원되더라도 클라이언트가 인스턴스와 다시 연결되지 않습니다.
- 해결 방법: 프로덕션 클러스터에 HSM 인스턴스가 두 개 이상 있는 것이 좋습니다. 이 구성을 사용하면 이 문제의 영향을 받지 않습니다. 단일 HSM 클러스터의 경우, 클라이언트 데몬을 바운스하여 연결을 복원합니다.
- 해결 상태: 이 문제는 AWS CloudHSM 클라이언트 1.1.2 릴리스에서 해결되었습니다. 수정의 이점을 누리려면 이 클라이언트로 업그레이드해야 합니다.

문제: 짧은 시간 내에 클러스터에 있는 HSM의 키 용량이 초과하면, 클라이언트가 처리되지 않은 오류 상태로 전환됩니다.

- 영향: 클라이언트에 처리되지 않은 오류 상태가 발생하면 중지되므로 다시 시작해야 합니다.
- 해결 방법: 처리량을 테스트하여 클라이언트가 처리할 수 없는 속도로 세션 키를 생성하고 있지 않은지 확인하십시오. 클러스터에 HSM을 추가하거나 세션 키 생성 속도를 줄여 속도를 낮출 수 있습니다.
- 해결 상태: 이 문제는 AWS CloudHSM 클라이언트 1.1.2 릴리스에서 해결되었습니다. 수정의 이점을 누리려면 이 클라이언트로 업그레이드해야 합니다.

문제: HMAC 키 크기가 800바이트보다 큰 다이제스트 작업은 지원되지 않습니다.

- 영향: 800바이트보다 큰 HMAC 키를 HSM에 생성하거나 가져올 수 있습니다. 그러나 JCE 또는 key_mgmt_util을 통해 다이제스트 작업에서 이보다 더 큰 키를 사용하면, 작업이 실패합니다. PKCS11을 사용하는 경우, HMAC 키가 64바이트 크기로 제한됩니다.
- 해결 방법: HSM에서 다이제스트 작업에 HMAC 키를 사용할 예정인 경우, 크기가 800바이트보다 작은지 확인하십시오.

- 해결 상태: 현재는 없습니다.

문제: Client SDK 3과 함께 배포된 client_info 도구가 선택적 출력 인수로 지정된 경로의 내용을 삭제합니다.

- 영향: 지정된 출력 경로에 있는 모든 기존 파일 및 하위 디렉터리가 영구적으로 손실될 수 있습니다.
- 해결 방법: `client_info`도구를 사용할 때 `-output` *`path`*선택적 인수를 사용하지 마십시오.
- 해결 상태: 이 문제는 Client SDK 3.3.2 릴리스에서 해결되었습니다. 수정의 이점을 누리려면 이 클라이언트로 업그레이드해야 합니다.

문제: 컨테이너화된 환경에서 **`--cluster-id`** 인수를 사용하여 SDK 5 구성 도구를 실행할 때 오류가 발생합니다.

구성 도구와 함께 --cluster-id 인수를 사용할 때 다음 오류가 발생합니다.

```
No credentials in the property bag
```

이 오류는 인스턴스 메타데이터 서비스 버전 2(IMDSv2) 업데이트로 인해 발생합니다. 자세한 내용은 IMDSv2 설명서를 참조하십시오.

- 영향: 이 문제는 컨테이너화된 환경에서 SDK 버전 5.5.0 이상에서 구성 도구를 실행하고 EC2 인스턴스 메타데이터를 활용하여 자격 증명을 제공하는 사용자에게 영향을 미칩니다.
- 해결 방법: PUT 응답 홉 제한을 두개 이상으로 설정합니다. 이를 수행하는 방법에 대한 지침은 인스턴스 메타데이터 옵션 구성'을 참조하십시오.

문제: “제공된 pfx 파일에서 인증서/키를 만들지 못했습니다.” 라는 오류 메시지가 나타납니다. NotPkcs오류: 8"

- 영향: 개인 키가 PKCS8 형식이 아닌 경우 인증서와 개인 키로 SSL을 재구성하는 SDK 5.11.0 사용자가 실패합니다.
- 해결 방법: openssl 명령을 사용하여 사용자 지정 SSL 개인 키를 PKCS8 형식으로 변환할 수 있습니다. `openssl pkcs8 -topk8 -inform PEM -outform PEM -in` *`ssl_private_key`* `-out` *`ssl_private_key_pkcs8`*
- 해결 상태: 이 문제는 클라이언트 SDK 5.12.0 릴리스에서 해결되었습니다. 수정 사항을 적용하려면 이 클라이언트 버전 이상으로 업그레이드해야 합니다.

hsm2m.medium 인스턴스의 알려진 문제

다음 문제는 모든 hsm2m.medium 인스턴스에 영향을 미칩니다.

주제

문제: PBKDF2 반복 횟수 증가로 인한 로그인 지연 시간 증가

- 영향: 보안 강화를 위해 hsm2m.medium은 로그인 요청 시 암호 기반 키 도출 함수 2 (PBKDF2) 를 60,000회 반복하는데 반해 hsm1.medium에서는 1,000회 반복합니다. 이렇게 증가하면 로그인 요청당 대기 시간이 최대 2초 (2초) 까지 증가할 수 있습니다.

  AWS CloudHSM 클라이언트 SDK의 기본 제한 시간은 20초입니다. 로그인 요청 시간이 초과되어 오류가 발생할 수 있습니다.
- 해결 방법: 가능하면 동일한 애플리케이션에서 로그인 요청을 직렬화하여 로그인 중 지연 시간이 길어지지 않도록 하십시오.
- 해결 상태: 이러한 지연 시간 증가를 고려하여 향후 버전의 클라이언트 SDK에서는 로그인 요청의 기본 제한 시간을 늘릴 예정입니다.

문제: Client SDK 5.12.0 이전 버전에서는 키의 신뢰할 수 있는 속성을 설정하려고 시도하는 CO가 실패합니다.

- 영향: 키의 신뢰할 수 있는 특성을 설정하려고 시도하는 모든 CO 사용자에게 이를 나타내는 오류 메시지가 표시됩니다. `User type should be CO or CU`
- 해결 방법: 향후 버전의 클라이언트 SDK에서 이 문제가 해결될 예정입니다. 업데이트는 사용 설명서에 발표될 예정입니다. 문서 기록

PKCS#11 SDK의 알려진 문제

주제

문제: PKCS #11 라이브러리 버전 3.0.0의 AES 키 래핑은 사용 전에 IV의 유효성을 검사하지 않습니다.

길이가 8바이트보다 짧은 IV를 지정하면 사용하기 전에 예측할 수 없는 바이트로 패딩됩니다.

> **Note**
>
> 이는 `CKM_AES_KEY_WRAP` 메커니즘을 사용하는 `C_WrapKey`에만 영향을 미칩니다.

- 영향: PKCS #11 SDK 3.0.0에서 8바이트보다 짧은 IV를 제공하는 경우 키를 언래핑할 수 있습니다.
- 해결 방법:
  - AES 키 래핑 중에 IV 길이를 올바르게 적용하는 PKCS #11 라이브러리 3.0.1 버전 이상으로 업그레이드하는 것이 좋습니다. NULL IV를 전달하도록 래핑 코드를 수정하거나 `0xA6A6A6A6A6A6A6A6`의 기본 IV를 지정하십시오. 자세한 내용은 AES 키 래핑에 대한 규정 미준수 길이의 사용자 지정 IV를 참조하십시오.
  - 8바이트보다 짧은 IV를 사용하여 PKCS #11 라이브러리 3.0.0 버전으로 키를 래핑한 경우 지원을 요청하십시오.

- 해결 상태: 이 문제는 PKCS #11 라이브러리 3.0.1 버전에서 해결되었습니다. AES 키 래핑을 사용하여 키를 래핑하려면 NULL 또는 8바이트 길이의 IV를 지정합니다.

문제: PKCS#11 SDK 2.0.4 및 이전 버전에서는 AES 키 래핑 및 래핑 해제에 항상 기본 IV인 **0xA6A6A6A6A6A6A6A6**를 사용했습니다.

사용자 제공 IV는 자동으로 무시되었습니다.

> **Note**
>
> 이는 CKM_AES_KEY_WRAP 메커니즘을 사용하는 C_WrapKey에만 영향을 미칩니다.

- 영향:
  - PKCS#11 SDK 2.0.4 이전 버전과 사용자 제공 IV를 사용한 경우 키가 0xA6A6A6A6A6A6A6A6의 기본 IV로 래핑됩니다.
  - PKCS#11 SDK 3.0.0 이상 버전과 사용자 제공 IV를 사용한 경우 키가 사용자 제공 IV로 래핑됩니다.
- 해결 방법:
  - PKCS#11 SDK 2.0.4 이전 버전으로 래핑된 키의 언래핑하려면 0xA6A6A6A6A6A6A6A6의 기본 IV를 사용합니다.
  - PKCS #11 SDK 3.0.0 이상 버전으로 래핑된 키를 언래핑하려면 사용자 제공 IV를 사용합니다.
- 해결 상태: NULL IV를 전달하도록 래핑 및 언래핑 코드를 수정하거나 0xA6A6A6A6A6A6A6A6의 기본 IV를 지정하는 것이 좋습니다.

문제: **CKA_DERIVE** 속성이 지원 및 처리되지 않았습니다.

- 해결 상태: FALSE로 설정될 경우 CKA_DERIVE를 수락하도록 수정했습니다. 키 추출 함수를 AWS CloudHSM에 추가하기 전에는 CKA_DERIVE를 TRUE로 설정할 수 없습니다. 수정의 이점을 누리기 위해서는 클라이언트 및 SDK를 버전 1.1.1 이상으로 업데이트해야 합니다.

문제: **CKA_SENSITIVE** 속성이 지원 및 처리되지 않았습니다.

- Resolution status(해결 상태): CKA_SENSITIVE 속성을 수락하고 적절하게 인식하도록 수정을 했습니다. 수정의 이점을 누리기 위해서는 클라이언트 및 SDK를 버전 1.1.1 이상으로 업데이트해야 합니다.

문제: 멀티파트 해시 및 서명이 지원되지 않습니다.

- 영향: C_DigestUpdate 및 C_DigestFinal이 구현되지 않습니다. C_SignFinal도 구현되지 않으며 NULL이 아닌 버퍼에 대한 CKR_ARGUMENTS_BAD로 실패합니다.
- 해결 방법: 응용 프로그램 내에서 데이터를 해시하고 해시 AWS CloudHSM 서명에만 사용하십시오.
- 해결 상태: 멀티파트 해시를 올바르게 구현하도록 클라이언트와 SDK를 수정하고 있습니다. AWS CloudHSM 포럼 및 버전 기록 페이지에 업데이트가 발표됩니다.

문제: **C_GenerateKeyPair**는 표준을 준수하는 방식으로 개인 템플릿의 **CKA_MODULUS_BITS** 또는 **CKA_PUBLIC_EXPONENT**를 처리하지 않습니다.

- 영향: 프라이빗 템플릿에 or C_GenerateKeyPair가 포함되는 경우 CKA_TEMPLATE_INCONSISTENTCKA_MODULUS_BITS는 CKA_PUBLIC_EXPONENT를 반환해야 합니다. 그 대신 모든 사용 필드가 FALSE로 설정된 프라이빗 키를 생성합니다. 이 키는 사용할 수 없습니다.
- 해결 방법: 애플리케이션에서 오류 코드 외에도 사용 필드 값을 확인하는 것이 좋습니다.
- 해결 상태: 잘못된 프라이빗 키 템플릿이 사용되는 경우 적절한 오류 메시지를 반환하도록 수정을 구현하고 있습니다. 업데이트된 PKCS#11 라이브러리가 버전 기록 페이지에 발표됩니다.

문제: **CKM_AES_GCM** 메커니즘을 사용할 때 **C_Encrypt** 및 **C_Decrypt** API 작업에 대한 버퍼는 16KB를 초과할 수 없습니다.

AWS CloudHSM 멀티파트 AES-GCM 암호화는 지원하지 않습니다.

- 영향: CKM_AES_GCM 메커니즘을 사용하여 16KB보다 큰 데이터를 암호화할 수 없습니다.
- 해결 방법: CKM_AES_CBC, CKM_AES_CBC_PAD 등의 대체 메커니즘을 사용하거나 데이터를 여러 조각으로 나누고 AES_GCM을 사용하여 각 조각을 개별적으로 암호화할 수 있습니다. 를 사용하는 AES_GCM 경우 데이터 분할 및 후속 암호화를 관리해야 합니다. AWS CloudHSM 멀티파트 AES-

GCM 암호화를 대신 수행하지 않습니다. FIPS에서는 AES-GCM에 대한 초기화 벡터(IV)가 HSM에서 생성되어야 합니다. 따라서 AES-GCM 암호화 데이터의 각 부분에 대한 IV가 다릅니다.

- 해결 상태: 데이터 버퍼가 너무 큰 경우 명시적으로 실패하도록 SDK를 수정하고 있습니다. C_EncryptUpdate 및 C_DecryptUpdate API 작업에 대해 CKR_MECHANISM_INVALID가 반환됩니다. 멀티파트 암호화에 의존하지 않고 더 큰 버퍼를 지원하기 위한 대체 방법을 평가하고 있습니다. 업데이트는 AWS CloudHSM 포럼과 버전 기록 페이지에 발표될 예정입니다.

문제: ECDH(Elliptic-curve Diffie-Hellman) 키 파생은 HSM 내에서 부분적으로 실행됩니다.

EC 프라이빗 키는 항상 HSM 내에 있지만 키 추출 프로세스는 여러 단계로 수행됩니다. 결과적으로 각 단계의 중간 결과를 클라이언트에서 사용할 수 있습니다.

- 영향: 클라이언트 SDK 3에서는 CKM_ECDH1_DERIVE 메커니즘을 사용하여 파생된 키를 먼저 클라이언트에서 사용할 수 있게 된 다음 HSM으로 가져옵니다. 그러면 키 핸들이 애플리케이션에 반환됩니다.
- 해결 방법: AWS CloudHSM에서 SSL/TLS 오프로드를 구현하는 경우 이 제한 사항은 문제가 되지 않습니다. 애플리케이션에서 항상 키가 FIPS 경계 내에 있어야 하는 경우 ECDH 키 파생에 독립적인 대체 프로토콜을 사용하는 것이 좋습니다.
- 해결 상태: HSM 내에서 ECDH 키 파생을 완전히 수행할 수 있는 옵션을 개발하고 있습니다. 업데이트가 구현되면 버전 기록 페이지에 발표됩니다.

문제: CentOS6 및 RHEL 6과 같은 EL6 플랫폼에서 secp256k1 서명 확인이 실패합니다.

CloudHSM PKCS#11 라이브러리가 OpenSSL을 사용하여 EC 곡선 데이터를 확인함으로써 확인 작업을 초기화하는 동안 네트워크 호출을 피하기 때문입니다. Secp256k1은 EL6 플랫폼의 기본 OpenSSL 패키지에서 지원되지 않으므로 초기화가 실패합니다.

- 영향: Secp256k1 서명 확인이 EL6 플랫폼에서 실패합니다. 확인 호출이 CKR_HOST_MEMORY 오류로 실패합니다.
- 해결 방법: PKCS#11 애플리케이션이 secp256k1 서명을 확인해야 하는 경우 Amazon Linux 1 또는 EL7 플랫폼을 사용하는 것이 좋습니다. 또는 secp256k1 곡선을 지원하는 OpenSSL 패키지 버전으로 업그레이드하십시오.
- 해결 상태: 로컬 곡선 검증을 사용할 수 없는 경우 HSM으로 대체하기 위한 수정을 구현하고 있습니다. 업데이트된 PKCS#11 라이브러리가 버전 기록 페이지에 발표됩니다.

문제: 잘못된 함수 호출 순서로 인해 실패하는 대신 정의되지 않은 결과가 발생합니다.

- 영향: 잘못된 함수 시퀀스를 호출하면 개별 함수 호출이 성공을 반환하더라도 최종 결과가 올바르지 않습니다. 예를 들어, 해독된 데이터는 원본 일반 텍스트와 일치하지 않거나 서명 확인에 실패할 수 있습니다. 이 문제는 단일 부품 및 다중 부품 작업 모두에 영향을 미칩니다.

  잘못된 함수 시퀀스의 예시:

  - `C_EncryptInit`/`C_EncryptUpdate` 다음에 `C_Encrypt`가 옵니다
  - `C_DecryptInit`/`C_DecryptUpdate` 다음에 `C_Decrypt`가 옵니다
  - `C_SignInit`/`C_SignUpdate` 다음에 `C_Sign`가 옵니다
  - `C_VerifyInit`/`C_VerifyUpdate` 다음에 `C_Verify`가 옵니다
  - `C_FindObjectsInit` 다음에 `C_FindObjectsInit`가 옵니다
- 해결 방법: 애플리케이션은 PKCS #11 사양을 준수하여 단일 및 다중 부분 작업 모두에 대해 올바른 함수 호출 순서를 사용해야 합니다. 이러한 상황에서 애플리케이션은 오류를 반환하기 위해 CloudHSM PKCS #11 라이브러리에 의존해서는 안 됩니다.

문제: SDK 5에서는 읽기 전용 세션이 지원되지 않습니다.

- 문제: SDK 5는 `C_OpenSession`을 사용하여 읽기 전용 세션을 여는 것을 지원하지 않습니다.
- 영향: `CKF_RW_SESSION`을 제공하지 않고 `C_OpenSession`을 호출하려고 하면 `CKR_FUNCTION_FAILED` 오류와 함께 호출이 실패합니다.
- 해결 방법: 세션을 열 때 `CKF_SERIAL_SESSION | CKF_RW_SESSION` 플래그를 `C_OpenSession` 함수 호출에 전달해야 합니다.

문제: **`cryptoki.h`** 헤더 파일이 Windows 전용입니다.

- 문제: Linux에서 AWS CloudHSM 클라이언트 SDK 5 버전 5.0.0부터 5.4.0까지 사용할 경우 헤더 파일은 Windows 운영 체제와만 호환됩니다. `/opt/cloudhsm/include/pkcs11/cryptoki.h`
- 영향: Linux 기반 운영 체제에서 애플리케이션에 이 헤더 파일을 포함하려고 하면 문제가 발생할 수 있습니다.
- 해결 상태: 이 헤더 파일의 Linux 호환 버전을 포함하는 AWS CloudHSM 클라이언트 SDK 5 버전 5.4.1 이상으로 업그레이드하십시오.

JCE SDK의 알려진 문제

주제

문제: 비동기 키 페어로 작업할 때 키를 명시적으로 생성하거나 가져오지 않더라도 점유된 키 용량이 보입니다.

- 영향: 이 문제로 인해 HSM에서 예기치 않게 키 공간이 부족해질 수 있으며 이 문제는 애플리케이션이 `CaviumKey` 객체 대신 표준 JCE 키 객체를 암호화 작업에 사용할 때 발생합니다. 표준 JCE 키 객체를 사용할 때는 애플리케이션이 종료될 때까지 세션 키가 이 키를 삭제하지 않으므로 `CaviumProvider`가 해당 키를 HSM으로 묵시적으로 가져옵니다. 결과적으로 애플리케이션이 실행되는 동안 키가 증가하여 HSM에서 사용 가능한 키 공간이 부족하게 되고 따라서 애플리케이션이 중단됩니다.
- 해결 방법: `CaviumSignature` 클래스, `CaviumCipher` 클래스, `CaviumMac` 클래스 또는 `CaviumKeyAgreement` 클래스를 사용할 때는 표준 JCE 키 객체 대신 `CaviumKey`로 키를 제공해야 합니다.

  `ImportKey` 클래스를 사용하여 일반 키를 `CaviumKey`로 수동으로 변환한 다음 작업이 완료된 후 수동으로 키를 삭제할 수 있습니다.
- 해결 상태: 묵시적 가져오기를 적절하게 관리할 수 있도록 `CaviumProvider`를 업데이트하고 있습니다. 수정 사항이 제공되면 버전 기록 페이지에 발표됩니다.

문제: JCE는 읽기 전용입니다. KeyStore

- 영향: 현재 HSM에서 지원하지 않는 객체 유형을 JCE 키 스토어에 저장할 수 없습니다. 특히 인증서를 키 스토어에 저장할 수 없습니다. 따라서 키 스토어에서 인증서를 찾으려는 jarsigner와 같은 도구와의 상호 운용성이 방해를 받습니다.
- 해결 방법: 키 스토어가 아닌 로컬 파일이나 S3 버킷 위치에서 인증서를 로드하도록 코드를 고칠 수 있습니다.
- 해결 상태: 키 스토어에 인증서 스토리지 지원을 추가하고 있습니다. 기능이 제공되면 버전 기록 페이지에 발표됩니다.

문제: AES-GCM 암호화를 위한 버퍼가 16,000바이트를 초과할 수 없습니다.

또한 멀티파트 AES-GCM 암호화가 지원되지 않습니다.

- 영향: AES-GCM을 사용하여 16,000바이트보다 큰 데이터를 암호화할 수 없습니다.
- 해결 방법: AES-CBC와 같은 대체 메커니즘을 사용하거나, 데이터를 여러 개로 분리하고 각 부분을 개별적으로 암호화할 수 있습니다. 데이터를 나누는 경우 분할된 암호문과 암호 해독을 관리해야 합니다. FIPS에서는 AES-GCM의 IV(초기화 벡터)를 HSM에서 생성해야 하므로 AES-GCM으로 암호화된 각 데이터의 IV가 다릅니다.
- 해결 상태: 데이터 버퍼가 너무 큰 경우 명시적으로 실패하도록 SDK를 수정하고 있습니다. 멀티파트 암호화에 의존하지 않고 더 큰 버퍼를 지원하기 위한 대체 방법을 평가하고 있습니다. AWS CloudHSM 포럼 및 버전 기록 페이지에 업데이트가 발표됩니다.

문제: ECDH(Elliptic-curve Diffie-Hellman) 키 파생은 HSM 내에서 부분적으로 실행됩니다.

EC 프라이빗 키는 항상 HSM 내에 있지만 키 추출 프로세스는 여러 단계로 수행됩니다. 결과적으로 각 단계의 중간 결과를 클라이언트에서 사용할 수 있습니다. ECDH 키 파생 샘플은 Java 코드 샘플에서 사용 가능합니다.

- 영향: 클라이언트 SDK 3이 JCE에 ECDH 기능을 추가합니다. `KeyAgreement`클래스를 사용하여 SecretKey a를 파생하면 먼저 클라이언트에서 클래스를 사용할 수 있게 되고 HSM으로 가져옵니다. 그러면 키 핸들이 애플리케이션에 반환됩니다.
- 해결 방법: 에서 AWS CloudHSM SSL/TLS 오프로드를 구현하는 경우에는 이 제한이 문제가 되지 않을 수 있습니다. 애플리케이션에서 항상 키가 FIPS 경계 내에 있어야 하는 경우 ECDH 키 파생에 독립적인 대체 프로토콜을 사용하는 것이 좋습니다.

- 해결 상태: HSM 내에서 ECDH 키 파생을 완전히 수행할 수 있는 옵션을 개발하고 있습니다. 가능한 경우, 버전 기록 페이지에 업데이트된 구현을 발표할 것입니다.

문제: 키 크기 KeyGenerator 매개 KeyAttribute 변수를 비트 대신 바이트 수로 잘못 해석합니다.

KeyGenerator 클래스의 `init` 함수나 AWS CloudHSM KeyAttribute 열거형의 `SIZE` 속성을 사용하여 키를 생성할 때 API는 인수가 키 바이트 수가 아니라 키 비트 수가 되어야 한다고 잘못 예상합니다.

- 영향: Client SDK 버전 5.4.0~5.4.2에서는 지정된 API에 키 크기가 바이트로 제공될 것으로 잘못 예상합니다.
- 해결 방법: 클라이언트 SDK 버전 5.4.0~5.4.2를 사용하는 경우 AWS CloudHSM JCE 공급자를 사용하여 KeyGenerator 클래스 또는 KeyAttribute 열거형을 사용하여 키를 생성하기 전에 키 크기를 비트에서 바이트로 변환하십시오.
- 해결 상태: 클라이언트 SDK 버전을 5.5.0 이상으로 업그레이드합니다. 여기에는 클래스 또는 열거형을 사용하여 키를 생성할 때 키 크기 (비트) 를 올바르게 예상하도록 수정된 사항이 포함됩니다. KeyGenerator KeyAttribute

문제: Client SDK 5에서 "불법적인 반사 액세스 작업이 발생했습니다"라는 경고가 표시됩니다.

Java 11과 함께 Client SDK 5를 사용하는 경우 CloudHSM은 다음 Java 경고를 표시합니다:

```
```
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by
 com.amazonaws.cloudhsm.jce.provider.CloudHsmKeyStore (file:/opt/cloudhsm/java/
cloudhsm-jce-5.6.0.jar) to field java.security .KeyStore.keyStoreSpi
WARNING: Please consider reporting this to the maintainers of
 com.amazonaws.cloudhsm.jce.provider.CloudHsmKeyStore
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective
 access operations
WARNING: All illegal access operations will be denied in a future release
```
```

이러한 경고는 영향을 주지 않습니다. 우리는 이 문제를 인지하고 있으며 해결하기 위해 노력하고 있습니다. 해결 방법이나 해결 방법이 필요하지 않습니다.

문제: JCE 세션 풀이 소진되었습니다.

영향: 다음 메시지가 표시된 후 JCE에서 작업을 수행하지 못할 수도 있습니다.

```
com.amazonaws.cloudhsm.jce.jni.exception.InternalException: There are too many
 operations
happening at the same time: Reached max number of sessions in session pool: 1000
```

해결 방법:

- 영향을 받은 경우 JCE 애플리케이션을 다시 시작하십시오.
- 작업을 수행할 때 작업에 대한 참조를 잃기 전에 JCE 작업을 완료해야 할 수도 있습니다.

> **Note**
>
> 작업에 따라 완료 메서드가 필요할 수 있습니다.

Operation	완료 메서드(들)
암호	암호화 또는 복호화 모드에서 `doFinal()` 랩 모드에서 `wrap()` 언랩 모드에서 `unwrap()`
KeyAgreement	`generateSecret()` 또는 `generateSecret(String)`
KeyPairGenerator	`generateKeyPair()` ,`genKeyPair()` 또는 `reset()`
KeyStore	메서드가 필요 없음
Mac	`doFinal()` 또는 `reset()`
MessageDigest	`digest()` 또는 `reset()`
SecretKeyFactory	메서드가 필요 없음

Operation	완료 메서드(들)
SecureRandom	메서드가 필요 없음
Signature	서명 모드에서 `sign()` 검증 모드에서 `verify()`

해결 상태: Client SDK 5.9.0 이상에서 이 문제를 해결했습니다. 이 문제를 해결하려면 Client SDK를 다음 버전 중 하나로 업그레이드하십시오.

문제: GetKey 작업 시 클라이언트 SDK 5 메모리 누수

- 영향: 클라이언트 SDK 버전 5.10.0 및 이전 버전의 JCE에서 API `getKey` 작업의 메모리 누수가 발생했습니다. 애플리케이션에서 `getKey` API를 여러 번 사용하면 메모리 증가가 증가하여 결과적으로 애플리케이션의 메모리 사용량이 증가합니다. 시간이 지남에 따라 병목 오류가 발생하거나 애플리케이션을 다시 시작해야 할 수 있습니다.
- 해결 방법: 클라이언트 SDK 5.11.0으로 업그레이드하는 것이 좋습니다. 이렇게 할 수 없는 경우 애플리케이션에서 `getKey` API를 여러 번 호출하지 않는 것이 좋습니다. 그보다는 이전 `getKey` 작업에서 이전에 반환된 키를 가능한 한 많이 재사용하십시오.
- 해결 상태: 클라이언트 SDK 버전을 5.11.0 이상으로 업그레이드하십시오. 이 버전에는 이 문제에 대한 수정이 포함됩니다.

OpenSSL Dynamic Engine의 알려진 문제

이는 OpenSSL Dynamic Engine의 알려진 문제입니다.

주제

- 문제: AWS CloudHSM OpenSSL 다이내믹 엔진이 OpenSSL v3.x용 FIPS 공급자와 호환되지 않습니다.

문제: RHEL 6 및 CentOS6에는 AWS CloudHSM OpenSSL 동적 엔진을 설치할 수 없습니다.

- 영향: OpenSSL Dynamic Engine은 OpenSSL 1.0.2[f+]만 지원합니다. 기본적으로 RHEL 6 및 CentOS 6은 OpenSSL 1.0.1과 함께 제공됩니다.
- 해결 방법: RHEL 6 및 CentOS 6의 OpenSSL 라이브러리를 버전 1.0.2[f+]로 업그레이드합니다.

문제: 기본적으로 HSM에 대한 RSA 오프로드만 지원됩니다.

- 영향: 성능을 극대화하기 위해 난수 생성이나 EC-DH 작업과 같은 추가 기능을 오프로드하도록 SDK가 구성되지 않습니다.
- 해결 방법: 추가 작업을 오프로드해야 할 경우 지원 사례를 통해 문의해 주십시오.
- 해결 상태: 구성 파일을 통해 오프로드 옵션을 구성하도록 SDK에 지원을 추가하고 있습니다. 업데이트가 제공되면 버전 기록 페이지에 발표됩니다.

문제: HSM에서 키를 사용하여 OAEP 패딩의 RSA 암호화 및 암호화 해제가 지원되지 않습니다.

- 영향: OAEP 패딩을 사용한 RSA 암호화 및 복호화 호출이 오류와 함께 실패합니다. divide-by-zero 이 결과는 OpenSSL Dynamic Engine이 작업을 HSM에 오프로드하는 대신 가짜 PEM 파일을 사용하여 작업을 로컬로 호출하기 때문에 발생합니다.
- 해결 방법: PKCS #11 라이브러리 또는 JCE 공급자를 사용하여 이 절차를 수행할 수 있습니다.
- 해결 상태: 이 작업을 올바르게 오프로드하도록 SDK에 지원을 추가하고 있습니다. 업데이트가 제공되면 버전 기록 페이지에 발표됩니다.

문제: RSA 및 ECC 키의 프라이빗 키 생성만 HSM으로 오프로드됩니다.

다른 키 유형의 경우 OpenSSL AWS CloudHSM 엔진은 호출 처리에 사용되지 않습니다. 로컬 OpenSSL 엔진이 대신 사용됩니다. 이 엔진은 소프트웨어에서 로컬로 키를 생성합니다.

- 영향: 장애 조치가 작동하지 않기 때문에 HSM에서 안전하게 생성된 키를 받지 못했다는 표시가 없습니다. 키가 소프트웨어에서 OpenSSL에 의해 로컬로 생성된 경우 "`.....++++++`" 문자

열을 포함하는 추적 출력이 표시됩니다. 작업이 HSM으로 오프로드되면 이 추적이 없습니다. 키가 HSM에 생성되거나 저장되어 있지 않으므로 나중에 사용할 수 없습니다.

- 해결 방법: OpenSSL 엔진을 지원되는 키 유형에 대해서만 사용하십시오. 다른 모든 키 유형의 경우 애플리케이션에서 PKCS #11 또는 JCE를 사용하거나 CLI에서 사용하십시오`key_mgmt_util`.

문제: RHEL 8, CentOS 8 또는 Ubuntu 18.04 LTS에 Client SDK 3용 OpenSSL 동적 엔진을 설치할 수 없습니다.

- 영향: 기본적으로 RHEL 8, CentOS 8 및 Ubuntu 18.04 LTS는 Client SDK 3용 OpenSSL Dynamic Engine과 호환되지 않는 OpenSSL 버전을 제공합니다.
- 해결 방법: OpenSSL Dynamic Engine을 지원하는 Linux 플랫폼을 사용하십시오. 지원되는 플랫폼에 대한 자세한 내용은 지원되는 플랫폼을 참조하십시오.
- 해결 상태: 클라이언트 SDK 5용 OpenSSL 동적 엔진과 함께 이러한 플랫폼을 AWS CloudHSM 지원합니다. 자세한 내용은 지원되는 플랫폼 및 OpenSSL 동적 엔진을 참조하십시오.

문제: SHA-1 RHEL 9에 대한 서명 및 확인 지원 중단 (9.2+)

- 영향: 암호화를 위한 SHA-1 메시지 다이제스트의 사용은 RHEL 9 (9.2+) 에서 더 이상 사용되지 않습니다. 따라서 OpenSSL 동적 엔진을 사용하는 SHA-1 서명 및 확인 작업은 실패합니다.
- 해결 방법: 기존 또는 타사 암호화 서명의 서명/확인을 위해 SHA-1 사용을 요구하는 시나리오의 경우, 자세한 내용은 RHEL 보안 강화: RHEL 9 (9.2+) 및 RHEL 9 (9.2+) 릴리스 노트의 SHA-1 지원 중단에 대한 이해를 참조하십시오.

문제: AWS CloudHSM OpenSSL 다이내믹 엔진이 OpenSSL v3.x용 FIPS 공급자와 호환되지 않습니다.

- 영향: OpenSSL 버전 3.x에 대해 FIPS 공급자가 활성화되어 있을 때 AWS CloudHSM OpenSSL 동적 엔진을 활용하려고 하면 오류가 발생합니다.
- 해결 방법: AWS CloudHSM OpenSSL 버전 3.x에서 OpenSSL 동적 엔진을 사용하려면 “기본” 공급자가 구성되어 있는지 확인하십시오. OpenSSL 웹 사이트에서 기본 공급자에 대해 자세히 읽어보십시오.

Amazon Linux 2를 실행하는 Amazon EC2 인스턴스에 대한 알려진 문제

문제: Amazon Linux 2 버전 2018.07은 현재 SDK와 호환되지 않는 업데이트된 **ncurses** 패키지 (버전 6) 를 사용합니다. AWS CloudHSM

cloudhsm_mgmt_util 또는 key_mgmt_util을 실행하면 다음 오류가 반환되는 것을 볼 수 있습니다. AWS CloudHSM

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util: error while loading shared libraries:
 libncurses.so.5: cannot open shared object file: No such file or directory
```

- 영향: Amazon Linux 2 버전 2018.07에서 실행되는 인스턴스는 모든 유틸리티를 사용할 수 없게 됩니다. AWS CloudHSM
- 해결 방법: 지원되는 `ncurses` 패키지(버전 5)를 설치하려면 Amazon Linux 2 EC2 인스턴스에서 다음 명령을 실행하십시오.

  ```
  sudo yum update && yum install ncurses-compat-libs
  ```
- 해결 상태: 이 문제는 AWS CloudHSM 클라이언트 1.1.2 릴리스에서 해결되었습니다. 수정의 이점을 누리려면 이 클라이언트로 업그레이드해야 합니다.

서드 파티 애플리케이션 통합에 대한 알려진 문제

문제: Client SDK 3은 마스터 키 생성 중에 Oracle 설정 PKCS #11 속성 **CKA_MODIFIABLE**을 지원하지 않습니다.

이 제한은 PKCS #11 라이브러리에 정의되어 있습니다. 자세한 내용은 지원되는 PKCS #11 속성의 주석 1을 참조하십시오.

- 영향: Oracle 마스터 키 생성이 실패합니다.
- 해결 방법: 새 마스터 키 생성 시 특별한 환경 변수 `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE`를 TRUE로 설정합니다. 이 환경 변수는 마스터 키 생성에만 필요하므로 이 환경 변수를 다른 용도로 사용할 필요가 없습니다. 예를 들어 만든 첫 번째 마스터 키에 이 변수를 사용한 다음 마스터 키 버전을 교체하려는 경우에만 이 환경 변수를 다시 사용합니다. 자세한 내용은 Oracle TDE 마스터 암호화 키 생성을 참조하십시오.
- 해결 상태: CKA_MODIFIABLE 속성을 완전히 지원하도록 HSM 펌웨어를 개선하고 있습니다. 업데이트는 AWS CloudHSM 포럼과 버전 기록 페이지에 발표될 예정입니다.

클라이언트 SDK 3 키 동기화 실패

클라이언트 SDK 3에서는 클라이언트측 동기화가 실패할 경우 생성되었을 수 있는 (지금은 원치 않는) 원치 않는 키를 모두 정리하기 위해 최선의 노력을 다합니다. AWS CloudHSM 이 프로세스에는 불필요한 키 자료를 즉시 제거하거나 나중에 제거할 수 있도록 불필요한 자료를 표시하는 작업이 포함됩니다. 이 두 경우 모두 문제 해결을 위해 별도의 작업을 취하지 않아도 됩니다. 드문 경우지만 불필요한 키 구성 AWS CloudHSM 요소를 제거할 수 없고 표시할 수 없는 경우에는 키 구성 요소를 삭제해야 합니다.

문제: 토큰 키 생성, 가져오기 또는 래핑 해제 작업을 시도하면 tombstone 실패를 지정하는 오류가 나타납니다.

```
2018-12-24T18:28:54Z liquidSecurity ERR: print_node_ts_status:
[create_object_min_nodes]Key: 264617 failed to tombstone on node:1
```

원인: 불필요한 키 자료를 제거하고 표시하는 데 AWS CloudHSM 실패했습니다.

해결 방법: 클러스터의 HSM에 불필요한 것으로 표시되지 않은 불필요한 키 자료가 포함되어 있습니다. 키 자료를 수동으로 제거해야 합니다. 불필요한 키 자료를 수동으로 삭제하려면 PKCS #11 라이브러리 또는 JCE 공급자의 key_mgmt_util(KMU) 또는 API를 사용하십시오. 자세한 내용은 deleteKey 또는 클라이언트 SDK를 참조하십시오.

토큰 키의 내구성을 높이기 위해 클라이언트측 동기화 설정에 지정된 최소 수의 HSM에서 성공하지 AWS CloudHSM 못한 키 생성 작업이 실패합니다. 자세한 내용은 AWS CloudHSM의 키 동기화를 참조하세요.

클라이언트 SDK 3: pkpspeed 도구를 사용하여 HSM 성능 검증

이 주제에서는 클라이언트 SDK 3에서 HSM 성능을 확인하는 방법에 대해 설명합니다.

AWS CloudHSM 클러스터 내 HSM의 성능을 확인하려면 클라이언트 SDK 3에 포함된 pkpspeed (리눅스) 또는 pkpspeed_blocking (윈도우) 도구를 사용할 수 있습니다. pkpspeed 도구는 이상적인 조건에서 실행되며 PKCS11 같은 SDK를 거치지 않고 HSM을 직접 호출하여 작업을 실행합니다. 애플리케이션을 독립적으로 로드 테스트하여 규모 조정 요구 사항을 파악하는 것이 좋습니다. 랜덤 (I), (R) 및 EC 포인트 mul ModExp (Y) 테스트는 실행하지 않는 것이 좋습니다.

Linux EC2 인스턴스에 에이전트를 설치하는 자세한 내용은 AWS CloudHSM 클라이언트 설치 및 구성 (Linux) 단원을 참조하세요. Windows 인스턴스에 에이전트를 설치하는 자세한 내용은 AWS CloudHSM 클라이언트 설치 및 구성 (Windows) 단원을 참조하세요.

AWS CloudHSM 클라이언트를 설치하고 구성한 후 다음 명령을 실행하여 시작합니다.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Windows 클라이언트 1.1.2+의 경우:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows 클라이언트 1.1.1 이상의 경우:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

이미 클라이언트 소프트웨어를 설치했다면 최신 버전을 다운로드 및 설치하여 pkpspeed를 가져와야 할 수 있습니다. pkpspeed 도구는(/opt/cloudhsm/bin/pkpspeedLinux) 또는(C:\Program Files\Amazon\CloudHSM\Windows)에 있습니다.

pkpspeed를 사용하려면 pkpspeed 명령 또는 pkpspeed_blocking.exe를 실행하고 HSM에 있는 CU(Crypto User)의 사용자 이름 및 암호를 지정합니다. 다음 권장 사항을 고려하여 사용할 옵션을 설정합니다.

권장 테스트

- RSA 서명의 성능을 테스트하고 작업을 확인하려면 RSA_CRT 암호(Linux) 또는 B 옵션(Windows)을 선택합니다. RSA(Windows에서 A 옵션)를 선택하지 마십시오. 암호는 동등하지만 RSA_CRT가 성능에 최적화되어 있습니다.
- 먼저 소수의 스레드로 시작합니다. AES 성능 테스트에는 일반적으로 하나의 스레드도 최대 성능을 표시하는 데 충분합니다. RSA 성능(RSA_CRT) 테스트에는 일반적으로 3~4개의 스레드로 충분합니다.

pkpspeed 도구를 위한 구성 가능한 옵션

- FIPS 모드: 항상 FIPS AWS CloudHSM 모드입니다 (자세한 내용은 AWS CloudHSM FAQ 참조). 이는 사용 설명서에 설명된 CLI 도구를 사용하고 FIPS 모드 상태를 나타내는 getHSMInfo 명령을 실행하여 확인할 수 있습니다. AWS CloudHSM
- 테스트 유형(차단 및 비차단): 스레드 방식으로 작업이 수행되는 방식을 지정합니다. 비 차단을 사용하면 더 나은 수치를 얻을 가능성이 높습니다. 스레드와 동시성을 활용하기 때문입니다.
- 스레드 수: 테스트를 실행하는 데 사용할 스레드 수입니다.
- 테스트 실행 시간(초)(최대 = 600): pkpspeed는 “초당 작업 수”로 측정된 결과를 생성하고 테스트가 실행되는 매 초마다 이 값을 보고합니다. 예를 들어 테스트를 5초 동안 실행하면 다음과 같은 샘플 값이 출력될 수 있습니다.
  - OPERATIONS/second 821/1
  - OPERATIONS/second 833/1

- OPERATIONS/second 845/1
- OPERATIONS/second 835/1
- OPERATIONS/second 837/1

pkpspeed 도구를 사용하여 실행할 수 있는 테스트

- AES GCM: AES GCM 모드 암호화를 테스트합니다.
- 기본 3DES CBC: 3DES CBC 모드 암호화를 테스트합니다. 예정된 변경 사항은 1 아래 참고를 참조하세요.
- 기본 AES: AES CBC/ECB 암호화를 테스트합니다.
- 다이제스트.: 해시 다이제스트를 테스트합니다
- ECDSA 서명: ECDSA 기호를 테스트합니다.
- ECDSA 검증: ECDSA 검증을 테스트합니다.
- FIPS 랜덤: FIPS 호환 난수 생성을 테스트합니다(참고: 차단 모드에서만 사용할 수 있음).
- HMAC: HMAC를 테스트합니다.
- 임의: FIPS 140-2 HSM을 사용하고 있기 때문에 이 테스트는 관련이 없습니다.
- 비 CRT RSA 대 RSA_CRT 비교: RSA 서명을 테스트하고 작업을 확인합니다.
- RSA OAEP Enc: RSA OAEP 암호화를 테스트합니다.
- RSA OAEP Dec: RSA OAEP 암호 해독을 테스트합니다.
- RSA 프라이빗 덱 비 CRT: RSA 프라이빗 키 암호화(최적화되지 않음) 를 테스트합니다.
- RSA 프라이빗 키 dec CRT: RSA 프라이빗 키 암호화를 테스트합니다(최적화).
- RSA PSS 서명: RSA PSS 서명을 테스트합니다.
- RSA PSS 검증: 테스트: RSA PSS 검증을 테스트합니다.
- RSA 퍼블릭 키 enc: RSA 퍼블릭 키 암호화를 테스트합니다.

RSA 퍼블릭 키 암호화, RSA 개인 암호 해독 비 CRT 및 RSA 프라이빗 키 암호 해독 CRT도 사용자에게 다음 질문에 답하라는 메시지를 표시합니다.

```
Do you want to use static key [y/n]
```

를 입력하면 미리 y 계산된 키를 HSM으로 가져옵니다.

를 입력하면 새 키가 생성됩니다. n

[1] NIST 지침에 따라 2023년 이후 FIPS 모드의 클러스터에서는 이 기능이 허용되지 않습니다. 비 FIPS 모드의 클러스터의 경우 2023년 이후에도 여전히 허용됩니다. 세부 정보는 FIPS 140 규정 준수: 2024 메커니즘 지원 중단을 참조하세요.

예

다음 예제는 RSA 및 AES 작업에 대해 HSM 성능을 테스트하기 위해 pkpspeed(Linux) 또는 pkpspeed_blocking.exe(Windows)에서 선택할 수 있는 옵션을 보여 줍니다.

Example – pkpspeed를 사용하여 RSA 성능 테스트

이 예제는 Windows, Linux 및 호환되는 운영 체제에서 실행할 수 있습니다.

Linux

Linux 및 호환되는 운영 체제에의 경우 다음 지침을 사용하십시오.

```
/opt/cloudhsm/bin/pkpspeed -s CU user name -p password

SDK Version: 2.03

        Available Ciphers:
                AES_128
                AES_256
                3DES
                RSA  (non-CRT. modulus size can be 2048/3072)
                RSA_CRT (same as RSA)
For RSA, Exponent will be 65537

Current FIPS mode is: 00002
Enter the number of thread [1-10]: 3
Enter the cipher: RSA_CRT
Enter modulus length: 2048
Enter time duration in Secs: 60
Starting non-blocking speed test using data length of 245 bytes...
[Test duration is 60 seconds]

Do you want to use static key[y/n] (Make sure that KEK is available)?n
```

Windows

```
c:\Program Files\Amazon\CloudHSM>pkpspeed_blocking.exe -s CU user name -p password
```

```
Please select the test you want to run

RSA non-CRT------------------->A
RSA CRT----------------------->B
Basic 3DES CBC---------------->C
Basic AES--------------------->D
FIPS Random------------------->H
Random------------------------>I
AES GCM ---------------------->K

eXit------------------------->X
B

Running 4 threads for 25 sec

Enter mod size(2048/3072):2048
Do you want to use Token key[y/n]n
Do you want to use static key[y/n] (Make sure that KEK is available)?  n
OPERATIONS/second                821/1
OPERATIONS/second                833/1
OPERATIONS/second                845/1
OPERATIONS/second                835/1
OPERATIONS/second                837/1
OPERATIONS/second                836/1
OPERATIONS/second                837/1
OPERATIONS/second                849/1
OPERATIONS/second                841/1
OPERATIONS/second                856/1
OPERATIONS/second                841/1
OPERATIONS/second                847/1
OPERATIONS/second                838/1
OPERATIONS/second                843/1
OPERATIONS/second                852/1
OPERATIONS/second                837/
```

Example – pkpspeed를 사용하여 AES 성능 테스트

Linux

Linux 및 호환되는 운영 체제에의 경우 다음 지침을 사용하십시오.

```
/opt/cloudhsm/bin/pkpspeed -s <CU user name> -p <password>
```

```
SDK Version: 2.03

        Available Ciphers:
                AES_128
                AES_256
                3DES
                RSA  (non-CRT. modulus size can be 2048/3072)
                RSA_CRT (same as RSA)
For RSA, Exponent will be 65537

Current FIPS mode is: 00000002
Enter the number of thread [1-10]: 1
Enter the cipher: AES_256
Enter the data size [1-16200]: 8192
Enter time duration in Secs: 60
Starting non-blocking speed test using data length of 8192 bytes...
```

Windows

```
c:\Program Files\Amazon\CloudHSM>pkpspeed_blocking.exe -s CU user name -p password
login as USER
Initializing Cfm2 library
        SDK Version: 2.03

 Current FIPS mode is: 00000002
Please enter the number of threads [MAX=400] : 1
Please enter the time in seconds to run the test [MAX=600]: 20


Please select the test you want to run

RSA non-CRT------------------->A
RSA CRT----------------------->B
Basic 3DES CBC---------------->C
Basic AES--------------------->D
FIPS Random------------------->H
Random------------------------>I
AES GCM ---------------------->K

eXit-------------------------->X
D
```

```
Running 1 threads for 20 sec

Enter the key size(128/192/256):256
Enter the size of the packet in bytes[1-16200]:8192
OPERATIONS/second                9/1
OPERATIONS/second                10/1
OPERATIONS/second                11/1
OPERATIONS/second                10/1
OPERATIONS/second                10/1
OPERATIONS/second                10/...
```

클라이언트 SDK 5 사용자가 일치하지 않는 값을 포함함

이 `user list` 명령은 클러스터의 모든 사용자 및 사용자 속성 목록을 반환합니다. 사용자 속성 중 값이 "일관되지 않음"이 있는 경우 이 사용자는 클러스터 전체에서 동기화되지 않습니다. 즉, 클러스터의 HSM마다 다른 속성을 가진 사용자가 존재합니다. 어떤 속성이 일치하지 않는지에 따라 다른 복구 단계를 수행할 수 있습니다.

다음 표에는 단일 사용자의 불일치를 해결하는 단계가 포함되어 있습니다. 한 명의 사용자에게 불일치가 여러 개 있는 경우 다음 단계를 처음부터 끝까지 따라 문제를 해결하세요. 불일치를 겪는 사용자가 여러 명 있는 경우 다음 단계로 넘어가기 전에 각 사용자에 대해 이 목록을 검토하여 해당 사용자에 대한 불일치를 완전히 해결하세요.

> **Note**
>
> 이 단계를 수행하려면 관리자로 로그인하는 것이 가장 좋습니다. 관리자 계정이 일관되지 않은 경우 다음 단계를 수행하여 관리자로 로그인하고 모든 속성이 일치할 때까지 단계를 반복하세요. 관리자 계정이 일관되면 해당 관리자를 사용하여 클러스터의 다른 사용자를 동기화할 수 있습니다.

속성이 일치하지 않습니다.	사용자 목록 출력 예시	함축	복구 방법
사용자 "역할"이 "일치하지 않음"	`{ "username": "test_user",`	이 사용자는 일부 CryptoUser HSM에서는 A이고 다른 HSM에서는 관리자입니다. 이	1. 관리자로 로그인하세요. 2. 모든 HSM에서 사용자 삭제:

속성이 일치하지 않습니다.	사용자 목록 출력 예시	함축	복구 방법

	<pre>"role":
	"**inconsistent** ",
	
	"locked":
	"false",
	"mfa": [],
	"cluster-
	coverage":
	"full"
	}</pre>
		는 두 SDK가 동시에 다른 역할을 가진 동일한 사용자를 생성하려고 시도하는 경우 발생할 수 있습니다. 이 사용자를 제거하고 원하는 역할로 다시 생성해야 합니다.	user delete --username **<user's name>** --role admin
			
			user delete --username **<user's name>** --role crypto-user
			3. 원하는 역할을 가진 사용자 생성:
			
			user create --username **<user's name>** --role **<desired role>**

속성이 일치하지 않습니다.	사용자 목록 출력 예시	함축	복구 방법

사용자 “클러스터 범위”가 “일관되지 않음”	{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [],
	"cluster-coverage":
	"**inconsistent** "
	}
		이 사용자는 클러스터에 있는 HSM의 하위 집합에 존재합니다. 이는 user create에서 부분적으로 성공한 경우 또는 user delete에서 부분적으로 성공한 경우에 발생할 수 있습니다.
		
		클러스터에서 이 사용자를 생성하거나 제거하는 등 이전 작업을 완료해야 합니다.
			사용자가 존재하지 않아야 하는 경우 다음 단계를 따르십시오.
			
			1. 관리자로 로그인하세요.
			2. 다음 명령을 실행합니다.
			
			user delete --username**<user's name>** --role admin
			3. 이제 다음 명령을 실행합니다.
			
			user delete --username**<user's name>** --role crypto-user
			
			사용자가 존재해야 하는 경우 다음 단계를 따르세요.
			
			1. 관리자로 로그인하세요.
			2. 다음 명령을 실행합니다:
			
			user create --username **<user's name>** --role **<desired role>**

속성이 일치하지 않습니다.	사용자 목록 출력 예시	함축	복구 방법

사용자 “잠금” 매개변수가 “일관성 없음” 또는 “true”입니다.	<pre>{
	"username":
	"test_user",
	"role": "crypto-user",
	"locked"
	: **inconsistent** ,
	
	"mfa": [],
	"cluster-coverage":
	"full"
	}</pre>
		이 사용자는 일부 HSM에서 차단되었습니다.
		
		사용자가 잘못된 암호를 사용하고 클러스터에 있는 HSM의 일부에만 연결하는 경우 이 문제가 발생할 수 있습니다.
		
		클러스터 전체에서 일관성을 유지하려면 사용자 자격 증명을 변경해야 합니다.
			사용자가 MFA를 활성화한 경우 다음 단계를 따르십시오.
			
			1. 관리자로 로그인하세요.
			2. MFA를 일시적으로 비활성화하려면 다음 명령을 실행합니다.
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			
			3. 모든 HSM에 로그인할 수 있도록 사용자 암호를 변경하세요.
			
			user change-password --username **<user's name>** --role **<desired role>**
			
			사용자에 대해 MFA를 활성화해야 하는 경우 다음 단계를 따르십시오.

속성이 일치하지 않습니다.	사용자 목록 출력 예시	함축	복구 방법
			1. 사용자가 로그인하고 MFA를 다시 활성화하도록 합니다(이렇게 하려면 토큰에 서명하고 PEM 파일에 공개 키를 제공해야 함). user change-mfa token-sign --username **<user's name>** --role **<desired role>** —token <File>

속성이 일치하지 않습니다.	사용자 목록 출력 예시	함축	복구 방법

MFA 상태가 “일관되지 않음”	<pre>{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [
	{
	"strategy":
	"token-sign",
	"status":
	"**inconsistent** "
	}
	],
	"cluster-coverage":
	"full"
	}</pre>
		이 사용자는 클러스터의 HSM마다 다른 MFA 플래그를 사용합니다.
		
		이는 MFA 작업이 일부 HSM에서만 완료된 경우 발생할 수 있습니다.
		
		사용자 암호를 재설정하고 사용자가 MFA를 다시 활성화하도록 허용해야 합니다.
			사용자가 MFA를 활성화한 경우 다음 단계를 따르십시오.
			
			1. 관리자로 로그인하세요.
			2. MFA를 일시적으로 비활성화하려면 다음 명령을 실행합니다.
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			
			3. 그런 다음 모든 HSM에 로그인할 수 있도록 사용자 암호도 변경해야 합니다.
			
			user change-password --username **<user's name>** --role **<desired role>**
			
			사용자에 대해 MFA를 활성화해야 하는 경우 다음 단계를 따르십시오.

속성이 일치하지 않습니다.	사용자 목록 출력 예시	함축	복구 방법
			1. 사용자가 로그인하고 MFA를 다시 활성화하도록 합니다(이렇게 하려면 토큰에 서명하고 PEM 파일에 공개 키를 제공해야 함). user change-mfa token-sign --username **<user's name>** --role **<desired role>** —token <File>

키 가용성 검사 중에 오류가 표시됨

문제: HSM에서 다음 오류가 반환됩니다.

```
Key <KEY HANDLE> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

원인: 키 가용성 검사는 드물지만 가능한 상황에서는 손실될 수 있는 키를 찾습니다. 이 오류는 일반적으로 HSM이 하나뿐인 클러스터 또는 HSM이 두 개 있는 클러스터에서 둘 중 하나가 교체되는 기간 동안 발생합니다. 이러한 상황에서는 다음과 같은 고객 작업으로 인해 위와 같은 오류가 발생했을 수 있습니다.

- 키 생성-대칭 또는 와 같은 명령을 사용하여 새 키가 생성되었습니다키 generate-asymmetric-pair .
- 키 목록 작업이 시작되었습니다.
- SDK의 새 인스턴스가 시작되었습니다.

> **Note**
>
> OpenSSL은 SDK의 새 인스턴스를 자주 포크합니다.

해결책/권장 사항: 이 오류가 발생하지 않도록 하려면 다음 작업 중에서 선택하십시오.

- --disable-key-availability-check 파라미터를 사용하여 구성 도구의 구성 파일에서 키 가용성을 false로 설정합니다. 자세한 내용은 구성 도구의 파라미터 섹션을 참조하십시오.
- HSM이 두 개 있는 클러스터를 사용하는 경우 초기화 코드를 제외하고는 오류가 발생한 작업을 사용하지 마십시오.
- 클러스터의 HSM 수를 3개 이상으로 늘리십시오.

JCE를 사용하여 키 추출

GetEncoded 또는 Get은 null을 getPrivateExponent 반환합니다.

`getEncoded`,`getPrivateExponent` 및 `getS`은 기본적으로 비활성화되어 있으므로 null을 반환합니다. 활성화하려면 JCE를 사용한 키 추출을 참조하십시오.

활성화된 후 `getEncoded`, `getPrivateExponent` 및 `getS`이 null을 반환하면 키가 올바른 필수 조건을 충족하지 못합니다. 자세한 정보는 JCE를 사용한 키 추출 단원을 참조하세요.

GetEncoded 또는 GET은 getPrivateExponent HSM 외부에서 키 바이트를 반환합니다.

사용자 또는 시스템에 액세스할 수 있는 다른 사람이 클리어 키 추출을 활성화했습니다. 이 구성을 기본 비활성화 상태로 재설정하는 방법을 비롯한 자세한 내용은 다음 페이지를 참조하십시오.

- JCE를 사용한 키 추출
- HSM에서 키 보호 및 추출

HSM 스로틀링

워크로드가 클러스터의 HSM 용량을 초과하면 HSM이 사용 중이거나 병목 현상이 발생했다는 오류 메시지를 받을 겁니다. 이 경우 처리량이 감소하거나 HSM의 거부 요청 비율이 증가할 수 있습니다. 또한 HSM은 다음과 같은 사용 중 오류를 보낼 수 있습니다.

클라이언트 SDK 5의 경우

- PKCS11에서 사용 중인 오류는 CKR_FUNCTION_FAILED로 매핑됩니다. 이 오류는 여러 가지 이유로 발생할 수 있지만 HSM 제한으로 인해 이 오류가 발생하는 경우 로그에 다음과 같은 로그 줄이 나타납니다.
  - `[cloudhsm_provider::hsm1::hsm_connection::e2e_encryption::error] Failed to prepare E2E response. Error: Received error response code from Server. Response Code: 187`
  - `[cloudhsm_pkcs11::decryption::aes_gcm] Received error from the server. Error: This operation is already in progress. Internal error code: 0x000000BB`
- JCE에서 사용 중인 오류는 `com.amazonaws.cloudhsm.jce.jni.exception.InternalException: Unexpected error with the Provider: The HSM could not queue the request for processing.`로 매핑됩니다.
- 다른 SDK의 사용 중 오류가 발생하면 `Received error response code from Server. Response Code: 187`이라는 메시지가 출력됩니다.

클라이언트 SDK 3의 경우

- PKCS11에서 사용 중인 오류는 CKR_OPERATION_ACTIVE 오류로 매핑됩니다.
- JCE에서 사용 중인 오류는 `0xBB (187)`의 상태로 CFM2Exception로 매핑됩니다. 애플리케이션은 HSM에 의해 반환되는 상태를 확인하기 위해 `getStatus()` 기능을 CFM2Exception에 사용할 수 있다.
- 다른 SDK의 사용 중인 오류는 `HSM Error: HSM is already busy generating the keys(or random bytes) for another request.`이라는 메시지를 출력합니다.

해결 방법

다음 작업 중 하나 이상을 수행하여 이러한 문제를 해결할 수 있습니다.

- 애플리케이션 계층에서 거부된 HSM 작업에 대한 재시도 명령을 추가합니다. 재시도 명령을 활성화하기 전에 클러스터의 크기가 최대 부하를 충족할 수 있도록 적절한지 확인하십시오.

  > **Note**
  >
  > 클라이언트 SDK 5.8.0 이상의 경우 재시도 명령이 기본적으로 켜져 있습니다. 각 SDK의 재시도 명령 구성에 대한 자세한 내용은 클라이언트 SDK 5 구성 도구의 고급 구성을 참조하십시오.

- 클러스터에서 HSM 추가 또는 제거 AWS CloudHSM의 지침에 따라 클러스터에 HSM을 더 추가하세요.

  > **Important**
  >
  > 클러스터의 부하 테스트를 통해 예상해야 하는 최대 부하를 확인한 다음 HSM을 하나 더 추가하여 고가용성을 보장하는 것이 좋습니다.

클러스터에서 HSM 간에 HSM 사용자의 동기화 유지

HSM 사용자를 관리하려면 cloudhsm_mgmt_util이라는 명령줄 도구를 사용합니다. AWS CloudHSM 이는 도구의 구성 파일에 있는 HSM하고만 통신합니다. 구성 파일에 없는 클러스터의 다른 HSM을 인식하지 못합니다.

AWS CloudHSM HSM의 키를 클러스터의 다른 모든 HSM과 동기화하지만 HSM의 사용자 또는 정책을 동기화하지는 않습니다. cloudhsm_mgmt_util을 사용하여 HSM 사용자를 관리하는 경우 이러한 사용자 변경 사항은 cloudhsm_mgmt_util 구성 파일에 있는 클러스터의 일부 HSM에만 영향을 미칠 수 있습니다. 이렇게 하면 클러스터의 HSM 간에 키를 AWS CloudHSM 동기화할 때 문제가 발생할 수 있습니다. 키를 소유한 사용자가 클러스터의 모든 HSM에 존재하지 않을 수 있기 때문입니다.

이 문제를 방지하려면 사용자 관리에 앞서 cloudhsm_mgmt_util 구성 파일을 편집해야 합니다. 자세한 내용은 ???을(를) 참조하세요.

클러스터에 대한 연결 끊김

AWS CloudHSM 클라이언트를 구성할 때 클러스터의 첫 번째 HSM의 IP 주소를 제공했습니다. 이 IP 주소는 AWS CloudHSM 클라이언트의 구성 파일에 저장됩니다. 클라이언트가 시작되면 이 IP 주소에 대한 연결이 시도됩니다. 연결이 불가능할 경우, 예를 들어 HSM이 실패했거나 사용자가 삭제한 경우 다음과 같은 오류가 나타날 수 있습니다.

```
LIQUIDSECURITY: Daemon socket connection error
```

```
LIQUIDSECURITY: Invalid Operation
```

이러한 오류를 해결하려면 구성 파일을 클러스터 내 활성 상태의 연결 가능한 HSM의 IP 주소로 업데이트합니다.

AWS CloudHSM 클라이언트의 구성 파일을 업데이트하려면

1. 다음 방법 중 하나를 사용하여 클러스터 내 활성 HSM의 IP 주소를 찾습니다.
   - AWS CloudHSM 콘솔의 클러스터 세부 정보 페이지에서 HSM 탭을 확인합니다.
   - AWS Command Line Interface (AWS CLI) 를 사용하여 describe-clusters명령을 실행합니다.

   추후 단계에서 이 IP 주소가 필요합니다.
2. 다음 명령을 사용하여 클라이언트를 중지합니다.

   Amazon Linux

   ```
   $ sudo stop cloudhsm-client
   ```

   Amazon Linux 2

   ```
   $ sudo service cloudhsm-client stop
   ```

   CentOS 7

   ```
   $ sudo service cloudhsm-client stop
   ```

CentOS 8

```
$ sudo service cloudhsm-client stop
```

RHEL 7

```
$ sudo service cloudhsm-client stop
```

RHEL 8

```
$ sudo service cloudhsm-client stop
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client stop
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client stop
```

Windows

- Windows 클라이언트 1.1.2+의 경우:

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Windows 클라이언트 1.1.1 이상의 경우:

  클라이언트를 시작한 명령 창에서 Ctrl + C를 사용합니다. AWS CloudHSM

3. 아래 명령을 사용하여 클라이언트의 구성 파일을 업데이트하고 이전 단계에서 확인한 IP 주소를 제공합니다.

```
$ sudo /opt/cloudhsm/bin/configure -a <IP address>
```

4. 다음 명령을 사용하여 클라이언트를 시작합니다.

Amazon Linux

```
$ sudo start cloudhsm-client
```

Amazon Linux 2

```
$ sudo service cloudhsm-client start
```

CentOS 7

```
$ sudo service cloudhsm-client start
```

CentOS 8

```
$ sudo service cloudhsm-client start
```

RHEL 7

```
$ sudo service cloudhsm-client start
```

RHEL 8

```
$ sudo service cloudhsm-client start
```

Ubuntu 16.04 LTS

```
$ sudo service cloudhsm-client start
```

Ubuntu 18.04 LTS

```
$ sudo service cloudhsm-client start
```

Windows

- Windows 클라이언트 1.1.2+의 경우:

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Windows 클라이언트 1.1.1 이상의 경우:

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

AWS CloudHSM 감사 로그인 누락 CloudWatch

2018년 1월 20일 이전에 클러스터를 생성한 경우 해당 클러스터의 감사 로그 전달을 활성화하기 위해 서비스 연결 역할을 수동으로 구성해야 합니다. HSM 클러스터에서 서비스 연결 역할을 활성화하는 방법에 대한 지침은 IAM 사용 설명서의 서비스 연결 역할 이해 및 서비스 연결 역할 생성을 참조하십시오.

AES 키 래핑용 길이를 준수하지 않는 맞춤형 IV

이 문제 해결 항목은 애플리케이션에서 복구할 수 없는 래핑된 키를 생성하는지 확인하는 데 도움이 됩니다. 이 문제의 영향을 받는 경우 이 주제를 사용하여 문제를 해결하세요.

주제

- 코드에서 복구할 수 없는 래핑된 키가 생성되는지 확인하세요.
- 코드에서 복구할 수 없는 래핑된 키가 생성되는 경우 취해야 할 조치

코드에서 복구할 수 없는 래핑된 키가 생성되는지 확인하세요.

아래 조건을 모두 충족하는 경우에만 영향을 받습니다.

Condition	방법
애플리케이션은 PKCS #11 라이브러리를 사용합니다.	PKCS #11 라이브러리는 `/opt/cloudhsm/lib` 폴더에 `libpkcs11.so` 파일로 설치됩니다. C 언어로 작성된 애플리케이션은 일반적으로 PKCS #11 라이브러리를 직접 사용하는 반면, Java로 작성된 애플리케이션은 Java 추상화 계층을 통해 라이브러리를 간접적으로 사용할 수 있습니다. 현재 Windows에서는 PKCS #11

Condition	방법
	라이브러리를 사용할 수 없으므로 Windows를 사용하는 경우에는 영향을 받지 않습니다.
애플리케이션은 특히 PKCS #11 라이브러리 버전 3.0.0 을 사용합니다.	AWS CloudHSM 팀으로부터 이메일을 받았다면 PKCS #11 라이브러리 버전 3.0.0을 사용하고 있을 가능성이 높습니다. 애플리케이션 인스턴스의 소프트웨어 버전을 확인하려면 다음 명령어를 사용하세요. `rpm -qa | grep ^cloudhsm`
AES 키 래핑을 사용하여 키를 래핑합니다.	AES 키 래핑이란 AES 키를 사용하여 다른 키를 래핑하는 것을 의미합니다. 해당 메커니즘 이름은 입니다`CKM_AES_KEY_WRAP` . 함수와 함께 사용됩니다`C_WrapKey` . 초기화 벡터(IV)를 사용하는 기타 AES 기반 래핑 메커니즘(예: `CKM_AES_GCM` 및 `CKM_CLOUDHSM_AES_GCM`)은 이 문제의 영향을 받지 않습니다. 함수 및 메커니즘에 대해 자세히 알아보십시오.
AES 키 래핑을 호출할 때 사용자 지정 IV를 지정하는데 이 IV의 길이는 8보다 짧습니다.	AES 키 래핑은 일반적으로 다음과 같은 `CK_MECHANISM` 구조를 사용하여 초기화됩니다. `CK_MECHANISM mech = {CKM_AES_KEY_WRAP, IV_POINTER, IV_LENGTH};` 다음과 같은 경우에만 이 문제가 적용됩니다. • IV_포인터는 널이 아닙니다. • IV_길이가 8바이트 미만입니다.

위의 모든 조건을 충족하지 못하면 지금 읽기를 중단할 수 있습니다. 래핑된 키는 제대로 래핑을 풀 수 있으며 이 문제는 영향을 받지 않습니다. 그렇지 않으면 the section called “코드에서 복구할 수 없는 래핑된 키가 생성되는 경우 취해야 할 조치” 단원을 참조하세요.

코드에서 복구할 수 없는 래핑된 키가 생성되는 경우 취해야 할 조치

다음 세 단계를 수행해야 합니다.

1. PKCS #11 라이브러리를 최신 버전으로 즉시 업그레이드
   - Amazon Linux, CentOS 6 및 RHEL 6용 최신 PKCS #11 라이브러리
   - Amazon Linux 2, CentOS 7 및 RHEL 7용 최신 PKCS #11 라이브러리
   - Ubuntu 16.04 LTS용 최신 PKCS #11 라이브러리
2. 표준 호환 IV를 사용하도록 소프트웨어 업데이트

   샘플 코드를 따르고 NULL IV를 지정하기만 하면 HSM이 표준을 준수하는 기본 IV를 활용하도록 하는 것이 좋습니다. 또는 IV를 해당되는 8의 IV 길이를 가진 0xA6A6A6A6A6A6A6A6로 명시적으로 지정할 수도 있습니다. AES 키 래핑에는 다른 IV를 사용하지 않는 것이 좋으며, 향후 버전의 PKCS #11 라이브러리에서는 AES 키 래핑을 위한 사용자 지정 IV를 명시적으로 비활성화할 예정입니다.

   IV를 올바르게 지정하기 위한 샘플 코드는 aes_wrapping.c에 나와 있습니다. GitHub
3. 기존의 래핑된 키 식별 및 복구

   PKCS #11 라이브러리 버전 3.0.0을 사용하여 래핑한 모든 키를 식별한 다음, 이러한 키를 복구하는 데 필요한 지원을 받으려면 지원팀(https://aws.amazon.com/support) 에 문의해야 합니다.

> **Important**
>
> 이 문제는 PKCS #11 라이브러리 버전 3.0.0으로 래핑된 키에만 영향을 미칩니다. PKCS #11 라이브러리의 이전 버전(2.0.4 이하 패키지) 또는 이후 버전(3.0.1 이상 번호 패키지)을 사용하여 키를 래핑할 수 있습니다.

클러스터 생성 실패 해결

클러스터를 생성할 때 AWSServiceRoleForCloudHSM 서비스 연결 역할을 AWS CloudHSM 생성합니다 (해당 역할이 아직 없는 경우). 서비스 연결 역할을 생성할 AWS CloudHSM 수 없는 경우 클러스터를 만들려는 시도가 실패할 수 있습니다.

이 주제에서는 클러스터를 성공적으로 생성할 수 있도록 가장 일반적인 문제를 해결하는 방법을 설명합니다. 이 역할은 한 번만 생성하면 됩니다. 계정에서 서비스 연결 역할이 생성되면 지원되는 모든 방법을 사용하여 추가 클러스터를 생성하고 관리할 수 있습니다.

다음 단원에서는 서비스 연결 역할과 관련된 클러스터 생성 실패 문제를 해결하기 위한 제안을 제공합니다. 이 제안대로 시도해도 클러스터를 생성할 수 없으면 AWS Support에 문의하십시오. AWSServiceRoleForCloudHSM 서비스 연결 역할에 대한 자세한 내용은 을 참조하십시오. 서비스 연결 역할은 다음과 같습니다. AWS CloudHSM

주제

- 누락된 권한 추가
- 수동으로 서비스 연결 역할 생성
- 비 연합 사용자 사용

누락된 권한 추가

서비스 연결 역할을 생성하려면 사용자에게 `iam:CreateServiceLinkedRole` 권한이 있어야 합니다. 클러스터를 생성하는 IAM 사용자에게 이 권한이 없는 경우, 계정에서 서비스 연결 역할을 생성하려고 하면 클러스터 생성 프로세스가 실패합니다. AWS

누락된 권한 때문에 실패가 발생할 경우 오류 메시지에 다음 텍스트가 포함됩니다.

```
This operation requires that the caller have permission to call
 iam:CreateServiceLinkedRole to create the CloudHSM Service Linked Role.
```

이 오류를 해결하려면 클러스터를 생성하는 IAM 사용자에게 `AdministratorAccess` 권한을 부여하거나 사용자의 IAM 정책에 `iam:CreateServiceLinkedRole` 권한을 추가합니다. 지침은 신규 또는 기존 사용자에게 권한 추가를 참조하십시오.

그런 다음 다시 클러스터를 생성해 봅니다.

수동으로 서비스 연결 역할 생성

IAM 콘솔, CLI 또는 API를 사용하여 서비스 연결 역할을 생성할 수 있습니다 AWSServiceRoleForCloudHSM. 자세한 내용은 IAM 사용 설명서의 서비스 연결 역할 생성 단원을 참조하세요.

비 연합 사용자 사용

외부에서 자격 증명을 가져온 연동 사용자는 비연동 사용자의 많은 작업을 수행할 수 있습니다. AWS 하지만 AWS는 사용자가 연합된 엔드포인트에서 API 호출을 사용하여 서비스 연결 역할을 생성하도록 허용하지 않습니다.

이 문제를 해결하려면 권한을 가진 비연동 사용자를 만들거나 기존 비연동 사용자에게 `iam:CreateServiceLinkedRole` 권한을 부여하십시오. `iam:CreateServiceLinkedRole` 그런 다음 해당 사용자에게 에서 클러스터를 생성하도록 하십시오. AWS CLI그러면 계정에서 서비스 연결 역할이 생성됩니다.

일단 서비스 연결 역할이 생성되면, 원할 경우 비 연합 사용자가 생성한 클러스터를 삭제할 수 있습니다. 클러스터를 삭제해도 역할에는 영향을 미치지 않습니다. 이후에는 페더레이션 사용자를 포함하여 필요한 권한을 가진 모든 사용자가 계정에서 클러스터를 만들 수 있습니다. AWS CloudHSM

역할이 생성되었는지 확인하려면 https://console.aws.amazon.com/iam/ 에서 IAM 콘솔을 열고 역할을 선택합니다. 또는 에서 IAM get-role 명령을 사용할 수도 있습니다. AWS CLI

```
$  aws iam get-role --role-name AWSServiceRoleForCloudHSM
{
    "Role": {
        "Description": "Role for CloudHSM service operations",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
                {
                    "Action": "sts:AssumeRole",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "cloudhsm.amazonaws.com"
                    }
                }
            ]
        },
        "RoleId": "AROAJ4I6WN5QVGG5G7CBY",
        "CreateDate": "2017-12-19T20:53:12Z",
        "RoleName": "AWSServiceRoleForCloudHSM",
        "Path": "/aws-service-role/cloudhsm.amazonaws.com/",
        "Arn": "arn:aws:iam::111122223333:role/aws-service-role/cloudhsm.amazonaws.com/
AWSServiceRoleForCloudHSM"
    }
}
```

클라이언트 구성 로그 검색

AWS CloudHSM AWS Support에서 문제를 해결할 수 있도록 사용자 환경에 대한 정보를 수집하는 Client SDK 3 및 Client SDK 5용 도구를 제공합니다.

주제

클라이언트 SDK 5 지원 도구

스크립트는 다음 정보를 추출합니다.

- 클라이언트 SDK 5 구성 요소의 구성 파일
- 사용 가능한 로그 파일
- 운영 체제의 현재 버전
- 패키지 정보

클라이언트 SDK 5용 정보 도구 실행

클라이언트 SDK 5에는 각 구성 요소에 대한 클라이언트 지원 도구가 포함되어 있지만 모든 도구가 동일하게 작동합니다. 도구를 실행하여 수집된 모든 정보로 출력 파일을 생성합니다.

도구는 다음과 같은 구문을 사용합니다.

```
[ pkcs11 | dyn | jce ]_info
```

예를 들어, PKCS #11 라이브러리를 실행하는 Linux 호스트에서 지원 정보를 수집하고 시스템이 기본 디렉터리에 기록하도록 하려면 다음 명령을 실행합니다.

```
/opt/cloudhsm/bin/pkcs11_info
```

도구는 /tmp 디렉터리 내에 출력 파일을 생성합니다.

PKCS #11 library

Linux의 PKCS #11 라이브러리에 대한 지원 데이터를 수집하려면

- 지원 도구를 사용하여 데이터를 수집하십시오.

```
/opt/cloudhsm/bin/pkcs11_info
```

Windows용 PKCS #11 라이브러리에 대한 지원 데이터를 수집하려면

- 지원 도구를 사용하여 데이터를 수집하십시오.

```
C:\Program Files\Amazon\CloudHSM\bin\pkcs11_info.exe
```

OpenSSL Dynamic Engine

Linux의 OpenSSL Dynamic Engine에 대한 지원 데이터를 수집하려면

- 지원 도구를 사용하여 데이터를 수집하십시오.

```
/opt/cloudhsm/bin/dyn_info
```

JCE provider

Linux의 JCE 공급자에 대한 지원 데이터를 수집하려면

- 지원 도구를 사용하여 데이터를 수집하십시오.

```
/opt/cloudhsm/bin/jce_info
```

Windows에서 JCE 공급자에 대한 지원 데이터를 수집하려면

- 지원 도구를 사용하여 데이터를 수집하십시오.

```
C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe
```

서버리스 환경에서 로그 검색

Fargate 또는 Lambda와 같은 서버리스 환경을 구성하려면 로그 유형을 로 구성하는 것이 좋습니다. AWS CloudHSM `term` 로 `term` 구성한 후에는 서버리스 환경에서 출력을 수행할 수 있습니다. CloudWatch

클라이언트 로그를 CloudWatch 가져오려면 Amazon CloudWatch Logs 사용 설명서의 로그 그룹 및 로그 스트림 작업을 참조하십시오.

클라이언트 SDK 3 지원 도구

스크립트는 다음 정보를 추출합니다.

- 운영 체제 및 현재 버전
- `cloudhsm_client.cfg`, `cloudhsm_mgmt_util.cfg`, 및 `application.cfg` 파일의 클라이언트 구성 정보
- 플랫폼에 특정한 위치에서의 클라이언트 로그
- cloudhsm_mgmt_util을 사용한 클러스터 및 HSM 정보
- OpenSSL 정보
- 현재 클라이언트 및 빌드 버전
- 설치 프로그램 버전

클라이언트 SDK 3용 정보 도구 실행

이 스크립트는 수집된 모든 정보가 포함된 출력 파일을 생성합니다. 스크립트는 `/tmp` 디렉터리 내에 출력 파일을 생성합니다.

Linux: `/opt/cloudhsm/bin/client_info`

Windows: `C:\Program Files\Amazon\CloudHSM\client_info`

> **Warning**
>
> 이 스크립트에는 클라이언트 SDK 3 버전 3.1.0~3.3.1에서 알려진 문제가 있습니다. 이 문제에 대한 수정 사항이 포함된 버전 3.3.2로 업그레이드하는 것이 좋습니다. 이 도구를 사용하기 전에 알려진 문제 페이지에서 자세한 내용을 참조하십시오.

AWS CloudHSM 할당량

할당량 (이전 명칭: 한도) 은 리소스에 할당된 값입니다. AWS 다음 할당량은 지역 및 계정별로 리소스에 적용됩니다. AWS CloudHSM AWS AWS 기본 할당량은 에서 적용한 초기 값이며 AWS, 이 값은 아래 표에 나열되어 있습니다. 조정 가능한 할당량은 기본 할당량보다 늘릴 수 있습니다.

Service quotas

Resource	기본 할당량	조정 가능?
클러스터	4	예
HSM	6	예
클러스터당 HSM	28	아니요

할당량 증가를 요청할 때 권장되는 방법은 서비스 할당량 콘솔을 여는 것입니다. 콘솔에서 서비스 및 할당량을 선택하고 요청을 제출합니다. 자세한 내용은 서비스 할당량 설명서를 참조하십시오.

다음 시스템 할당량 표의 할당량은 조정할 수 없습니다.

시스템 할당량

Resource	hsm1.medium에 대한 할당량	hsm2m.medium의 할당량
클러스터당 최대 키	3,300	총 16,666개의 키, 비대칭 키는 최대 3,333개입니다.
클러스터당 최대 사용자	1,024	1,024
최대 사용자 이름 길이	31자	31자
필수 암호 길이	8~32자	8자에서 32자 사이입니다.
[1] 클러스터 당 최대 동시 클라이언트 연결 수	900	900
애플리케이션당 최대 PKCS #11 세션 수	1,024	1,024

[1] 클라이언트 SDK 3의 클라이언트 연결은 클라이언트 데몬입니다. 클라이언트 SDK 5의 경우 클라이언트 연결은 애플리케이션입니다.

자세한 내용은 시스템 리소스을(를) 참조하세요.

시스템 리소스

시스템 리소스 할당량은 AWS CloudHSM 클라이언트가 실행 시 사용할 수 있는 할당량입니다.

파일 설명자는 프로세스별로 열려 있는 파일을 식별하고 관리하기 위한 운영 체제의 메커니즘입니다.

CloudHSM 클라이언트 데몬은 파일 설명자를 활용하여 애플리케이션과 클라이언트 간의 연결뿐 아니라 클라이언트와 서버 간의 연결도 관리합니다.

기본적으로 CloudHSM 클라이언트 구성은 3000개의 파일 설명자를 할당합니다. 이 기본값은 클라이언트 데몬과 HSM 간에 최적의 세션 및 스레딩 용량을 산출하도록 설계되었습니다.

드물지만 일부 상황에서는 제한된 리소스 환경에서 클라이언트를 실행하는 경우 이러한 기본값을 변경해야 할 수 있습니다.

> **Note**
>
> 이러한 값을 변경하면 CloudHSM 클라이언트 성능이 저하되거나 애플리케이션이 작동 불가능하게 될 수 있습니다.

1. `/etc/security/limits.d/cloudhsm.conf` 파일을 편집합니다.

```
#
# DO NOT EDIT THIS FILE
#
hsmuser soft nofile 3000
hsmuser hard nofile 3000
```

2. 필요에 따라 숫자 값을 변경합니다.

> **Note**
>
> `soft` 할당량은 `hard` 할당량보다 작거나 같아야 합니다.

3. CloudHSM 클라이언트 데몬 프로세스를 다시 시작합니다.

> **Note**
>
> Microsoft Windows 플랫폼에서는 이 구성 옵션을 사용할 수 없습니다.

AWS CloudHSM 클라이언트 SDK용 다운로드

다운로드

2021년 3월, 다양한 요구 사항, 기능 및 플랫폼 지원을 갖춘 완전히 새로운 클라이언트 SDK를 도입하는 클라이언트 SDK 버전 5.0.0이 AWS CloudHSM 출시되었습니다.

클라이언트 SDK 5는 프로덕션 환경에서 완벽하게 지원되며, CNG 및 KSP 제공자에 대한 지원을 제외하고 클라이언트 SDK 3과 동일한 구성 요소 및 지원 수준을 제공합니다. 자세한 정보는 클라이언트 SDK 구성 요소 비교을 참조하세요.

> **Note**
>
> 각 클라이언트 SDK에서 지원하는 플랫폼에 대한 자세한 내용은 및 을 참조하십시오. Client SDK 5 지원 플랫폼 Client SDK 3 지원 플랫폼

최신 릴리스

이 섹션에는 최신 버전의 클라이언트 SDK가 포함되어 있습니다.

클라이언트 SDK 5 출시: 버전 5.12.0

Amazon Linux 2

x86_64 아키텍처에서 아마존 리눅스 2용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL Dynamic Engine (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE 공급자 (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

ARM64 아키텍처 기반 아마존 리눅스 2용 버전 5.12.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum c28a1f27e23e6ab1550dab6a353c6c9338a391a84d57f4ac99a1a3a9810c753f)
- OpenSSL Dynamic Engine (SHA256 checksum 7d2e864c31c13f55443c1b1d04589fbdd4558fe103954de4384691e2c429a872)
- JCE 공급자 (SHA256 checksum e9a35eb87b2f257c47fb083d286deb835da45858b2d89759ca7d5bb4ef747b4b)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 28b6f918912b5c63bf10018824b642a805b309c21947a1d0ebbdc44647e80554)

Amazon Linux 2023

x86_64 아키텍처에서 아마존 리눅스 2023용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 02801365cba449c5238a4e5ad3df1ddf7edd00ade976f47e956e885286503f3f)
- OpenSSL Dynamic Engine (SHA256 checksum 0abed69a7c6acaafdaabdcc5fab7d56611ffd94f5480cade6f8beace9aeae056)
- JCE 공급자 (SHA256 checksum 3d5d9a903d3a216eca40f92dbb0b4030b7a86ad7ceee8d62241c97a6e1881e25)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum f96671d882b862033bba0b3633448dc6a26e45a25063e29b79a5cd4b7fc4945c)

ARM64 아키텍처의 아마존 리눅스 2023용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 53d05006b46bda8e9c1dd76e8307a780bfe0a67b10a9a87723c97f94e29f5b8e)
- OpenSSL Dynamic Engine (SHA256 checksum ec1cca8e01b3303ff9473eeef6b33dc85b6affac7a47387b098905f9f2fc85ba)
- JCE 공급자 (SHA256 checksum c828ae56f46233215b9f35798b5859ebdac962af442acbc457081c3baaa44f11)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum ddd5dcd68d01f4fafaf13dc0b4ddcf98e3731ed51bdd51f85535b29353644a9f)

CentOS 7 (7.8+)

x86_64 아키텍처의 CentOS 7용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL Dynamic Engine (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE 공급자 (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 7 (7.8+)

x86_64 아키텍처의 RHEL 7용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL Dynamic Engine (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE 공급자 (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 8 (8.3+)

x86_64 아키텍처의 RHEL 8용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 6e51e95122fd0991278888287f0c408808b26fb5f1196c46168477b9090fc478)
- OpenSSL Dynamic Engine (SHA256 checksum 1f1d52ff7af6c537d8cfeb5973c691a9d90a518accd685ff9b66cd78daf98928)
- JCE 공급자 (SHA256 checksum 156944607de987d6b39bd8a2d21ccd294c01377a9e35f9f15f8b0f4c8bb90033)

  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 351e802f79dd2d0b5f7d23bb74c146be05e5169b603c9aace24189094a45a35d)

RHEL 9 (9.2+)

x86_64 아키텍처의 RHEL 9용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum d1b2f4ac7e6e0c18e788512e7726bc68b571d99a1442ce2f2e80f4b0f9956266)
- OpenSSL Dynamic Engine (SHA256 checksum cf86a3f17cd6c51969d4ce80c1e3ea6513b995611be7e2e72e5e5233c71d6add)
- JCE 공급자 (SHA256 checksum ae89e256eb89ec6b4fa0f001e7a4e1d8f1c08530423e81aa74d69a17b25d9a99)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum dfe6fe5d890c33b2f5d38f906ade113b06c8c05f3427a327744c454e7302f1a5)

ARM64 아키텍처의 RHEL 9용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum cad72a6ab2232b4c38b90d7c62147520b975d646773dd90d7be897fa0a537d2d)
- OpenSSL Dynamic Engine (SHA256 checksum ad751f756530a2317c3c64380ea3a07865b13e1874fab0e61ac530b21487c7fb)
- JCE 공급자 (SHA256 checksum d204e69acfbb90996fb08ae3573607b65630b1124fb379e078c002d55ac07766)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum c0f412cc59bafd235e046cdc1a0c5d330f2d72f7d6434672e9522f86bc945090)

Ubuntu 20.04 LTS

x86_64 아키텍처의 우분투 20.04 LTS용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum d37b1f872eb2b1ab34303d5b8b803daa925902b645c57c6e15a28bb6321e0f42)

- OpenSSL Dynamic Engine (SHA256 checksum cdc6e737652556b57d26d8816b2bc9820128cb3919360660b6f7fe65f9d39e3f)
- JCE 공급자 (SHA256 checksum f567a08344414a4776e1c5a9715657476925ca32695c4c2dd84a4f3fc5dc1615)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum f2ee5ad01c5018fc3670f602228fd71087228cd3923bf5b9bc73e4d7084dac6c)

Ubuntu 22.04 LTS

x86_64 아키텍처의 우분투 22.04 LTS용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 0e78928acd7a1662e4b07b15d5c3ccb88714ff89e47b991c8ab6e4c2229ee5aa)
- OpenSSL Dynamic Engine (SHA256 checksum 4f3168745edc5592234891a7b1d82b179a4947e87c72fade1be3bad58b7ed1a3)
- JCE 공급자 (SHA256 checksum d4c3655cdc2b00d1ab5ceafac94dfbc5c5244ed20e10fdd9db9f4e741e013733)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum d00bbacb6f2e57bd92d832a2bd11cadede972f8e82cc402ec0684b9c6b23123c)

ARM64 아키텍처의 우분투 22.04LTS용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 0c1121535c523acb864215338292bab32acee438357878b5fc0b6d268713b86f)
- OpenSSL Dynamic Engine (SHA256 checksum dc7a219302021570bc8c36674d2bd33165557bb2f9a0af8fdf114f1b85a70d84)
- JCE 공급자 (SHA256 checksum af3834a10081f1e4e7894275c8b9c7b7649b8de3b6f0aeb0781a3358183a9046)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum baa253ac62c2fbcc5712561e0fb0feb25461efc3ce68cf86d4c7bf0af0f14a34)

Windows Server 2016

x86_64 아키텍처의 윈도우 서버 2016용 버전 5.12.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- JCE 공급자 (SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

Windows Server 2019

x86_64 아키텍처의 윈도우 서버 2019용 버전 5.12.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- JCE 공급자 (SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

클라이언트 SDK 5.12.0은 여러 플랫폼에 ARM 지원을 추가하고 모든 SDK의 성능을 개선합니다. CloudHSM CLI 및 JCE 공급자에 새로운 기능이 추가되었습니다.

플랫폼 지원

- 모든 SDK에 대해 ARM64 아키텍처의 아마존 리눅스 2023에 대한 지원이 추가되었습니다.
- 모든 SDK에 대해 ARM64 아키텍처의 레드햇 엔터프라이즈 리눅스 9 (9.2+) 에 대한 지원이 추가되었습니다.
- 모든 SDK에 대해 ARM64 아키텍처의 우분투 22.04 LTS에 대한 지원이 추가되었습니다.

CloudHSM CLI

- 다음 명령이 추가되었습니다.
  - 키 리플리케이트
- 여러 클러스터에 연결할 수 있는 지원이 추가되었습니다. 자세한 정보는 CloudHSM CLI를 사용하여 여러 클러스터에 연결을 참조하세요.

JCE 공급자

- 를 사용하여 KeyStoreWithAttributes 키를 KeyReferenceSpec 검색하기 위해 추가되었습니다.
- 를 사용하여 여러 키를 한 번에 getKeys 검색할 수 있도록 추가되었습니다. KeyStoreWithAttributes

성능 개선

- 모든 SDK의 AES CBC NoPadding 작업 성능이 개선되었습니다.

이전 클라이언트 SDK 릴리스

이 섹션에는 이전 클라이언트 SDK 릴리스가 나열되어 있습니다.

버전 5.11.0

Amazon Linux 2

x86_64 아키텍처에서 아마존 리눅스 2용 버전 5.11.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL Dynamic Engine (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE 공급자 (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - AWS CloudHSM용 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

ARM64 아키텍처 기반 아마존 리눅스 2용 버전 5.11.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum 5ac16449ec149c9b5e7776865803245ab17d0f1ad56df80173840c5e8d257b19)
- OpenSSL Dynamic Engine (SHA256 checksum 28c2eb7f3f60172b0186e5c25f71bb7341537058a71f288673936766048083c1)

- JCE 공급자 (SHA256 checksum 06c9d9d281c12b1d2bd9a7b601d6317e46cedf175706bbfa3e4dcaed6ba05448)
  - AWS CloudHSM용 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 218982bb17aa751969a7866b0a9ff27e7aa5007a07817627d9cc1f7d60a78160)

Amazon Linux 2023

x86_64 아키텍처에서 아마존 리눅스 2023용 버전 5.11.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 55310ab333d18bcfabdc4b74115b040386b4508934bdff93e1d054c4c4a6f9ea)
- OpenSSL Dynamic Engine (SHA256 checksum f3d4934dc872a9b5212a180b9814ca2af3eca01ee228a8725563f1770add0dce)
- JCE 공급자 (SHA256 checksum 757d3abb515aeb08f4b1c83970ee0979399efee00ee78c9a9dbec05f4ed9768d)
  - AWS CloudHSM용 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 22af8f0501ff9a45a9e0683a408a63771c2c06c66abf5478d310d6d32e013555)

CentOS 7 (7.8+)

x86_64 아키텍처의 CentOS 7용 버전 5.11.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL Dynamic Engine (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE 공급자 (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - AWS CloudHSM용 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 7 (7.8+)

x86_64 아키텍처의 RHEL 7용 버전 5.11.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL Dynamic Engine (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE 공급자 (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - AWS CloudHSM용 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 8 (8.3+)

x86_64 아키텍처의 RHEL 8용 버전 5.11.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum b95b9f588656fb14fd08bb66ce0e0da807b96daa38348dec07a508c9bef7403a)
- OpenSSL Dynamic Engine (SHA256 checksum 7bb437b91a52e863b2b00ff7f427ce22522026daf757be873ee031ec6ffffd88)
- JCE 공급자 (SHA256 checksum e0db887e05eb535314f4d99f21da12d87d35ebb8baf9726f4ce8f01d9df0ea01)
  - AWS CloudHSM용 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 8485b5a6d679767ca9b4f611718159a643cf3e85090a8e4d20fe53c3707e25c3)

RHEL 9 (9.2+)

x86_64 아키텍처의 RHEL 9용 버전 5.11.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum 87b56a20accf67df53a203b7f115655b2acfaec4516682d4976d9475b10bec8e)
- OpenSSL Dynamic Engine (SHA256 checksum 83a6b58572e985df937beede4b10e867b0ac6050ace8010dc8d535be365d2747)
- JCE 공급자 (SHA256 checksum ee95213d02d913250478d0793d6dd578e5c54d765e635c7468a49bdff4c2a6f3)
  - AWS CloudHSM용 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 7e168ed3bef8e9c5110645e9960680e9a57f7b94e16aec71422e3c67ebc58fb5)

Ubuntu 20.04 LTS

x86_64 아키텍처의 우분투 20.04 LTS용 버전 5.11.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum abc3a339d1fe5850db65620804e9a910f8b4f913624ef9b7189f2f0df1825c01)
- OpenSSL Dynamic Engine (SHA256 checksum 075fc3f9974d552f27ad67fa92c8abff31b756b9add875b8cd4957e6801583a4)
- JCE 공급자 (SHA256 checksum 5de45c519133a0dae8da3ac01809db7974be25c14c15eb773fc5c972c0178c13)
  - AWS CloudHSM용 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 83e0e4505a063792c19feb3d4cfd032b9089091916168d92b0f51a967a007734)

Ubuntu 22.04 LTS

x86_64 아키텍처의 우분투 22.04 LTS용 버전 5.11.0 소프트웨어를 다운로드하십시오.

- PKCS #11 라이브러리 (SHA256 checksum b8f20be125c8530b2a7bd945956e9c04296fba5634af408b40be4e03bdbad72a)
- OpenSSL Dynamic Engine (SHA256 checksum d728c156eb4ee5c67159e57d6b092785800baa5fb61c14d64f460a8b8f53a778)
- JCE 공급자 (SHA256 checksum 44e943b8cd1176ad666e249342687744a280c6222df58b5a9f084c932f628284)
  - AWS CloudHSM용 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 8ccf5389d459611be813e42d7f9d040090f94f3fe88f9d110bcfb25e9619e4a7)

Windows Server 2016

x86_64 아키텍처의 윈도우 서버 2016용 버전 5.11.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- JCE 공급자 (SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - AWS CloudHSM용 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

Windows Server 2019

x86_64 아키텍처의 윈도우 서버 2019용 버전 5.11.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- JCE 공급자 (SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - AWS CloudHSM용 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

클라이언트 SDK 5.11.0에는 새로운 기능이 추가되고 안정성이 향상되며 모든 SDK에 대한 버그 수정이 포함됩니다.

플랫폼 지원

- 모든 SDK에 대한 아마존 리눅스 2023과 RHEL 9 (9.2+) 에 대한 지원이 추가되었습니다.
- 우분투 18.04 LTS의 최근 수명 종료로 인해 지원이 제거되었습니다.
- Amazon Linux의 최근 수명 종료로 인해 지원이 제거되었습니다.

CloudHSM CLI

- 다음 명령이 추가되었습니다.
  - 크립토 사인
  - 크립토 검증
  - 키 임포트 펜
  - 키 언랩
  - 키 랩
- 키 생성-파일이제 퍼블릭 키 내보내기가 지원됩니다.

OpenSSL Dynamic Engine

- AWS CloudHSM OpenSSL 동적 엔진은 이제 OpenSSL 라이브러리 버전 3.x와 함께 설치된 플랫폼에서 지원됩니다. 여기에는 아마존 리눅스 2023, RHEL 9 (9.2+), 우분투 22.04가 포함됩니다.

JCE

- JDK 17 및 JDK 21에 대한 지원이 추가되었습니다.
- HMAC 작업에 사용할 AES 키에 대한 지원이 추가되었습니다.
- 새 키 속성이 `ID` 추가되었습니다.
- 키 소진을 위한 새로운 `DataExceptionCause` 변형을 도입했습니다:. `DataExceptionCause.KEY_EXHAUSTED`

버그 수정/개선 사항

- `label`속성의 최대 길이를 126자에서 127자로 늘렸습니다.
- 메커니즘으로 EC 키의 래핑을 풀 수 없었던 버그를 수정했습니다. RsaOaep
- JCE 공급자의 GetKey 작업과 관련하여 알려진 문제가 해결되었습니다. 자세한 내용은 문제: GetKey 작업 시 클라이언트 SDK 5 메모리 누수 섹션을 참조하세요.
- FIPS 140-2에 따라 최대 암호화 블록 한도에 도달한 트리플 DES 키에 대한 모든 SDK의 로깅을 개선했습니다.
- OpenSSL 동적 엔진에 대해 알려진 문제를 추가했습니다. 세부 정보는 OpenSSL Dynamic Engine의 알려진 문제를 참조하세요.

버전 5.10.0

Amazon Linux

x86_64 아키텍처의 Amazon Linux용 버전 5.10.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum d63adf3e96c19c2d894b2defcbadd916dbb0398993050b1358bd93a36aa5acab)
- OpenSSL Dynamic Engine (SHA256 checksum 4daa3e591ffd5f7ce8ef3759c41deaa38867f5e5d21f15927aea83afb1678ac5)
- JCE 공급자 (SHA256 checksum 6c1ac94d3080f1c609d9dafbcb14480911beef3a488c4ed6f2b11b377da9b477)
  - AWS CloudHSM용 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum c12617fcd7990ba53e96f477979b410e3a5f17842ca7a912861b8b820809b5b5)

Amazon Linux 2

x86_64 아키텍처의 Amazon Linux 2용 버전 5.10.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL Dynamic Engine (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE 공급자 (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - AWS CloudHSM용 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

ARM64 아키텍처의 Amazon Linux 2용 버전 5.10.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum 5d8dfd835f1ed5a7f5a4fcc8ecf81cfa29883aca7e2985de69b5db723ab663db)
- OpenSSL Dynamic Engine (SHA256 checksum 91fb8efe2646bf0dbd9087554baa09554714e9d56e9bfd5c0dc3023a9f485574)
- JCE 공급자 (SHA256 checksum 99f6e55c37fdf00085a816d46835aeff54470797b3b71f4d28a70dc79c9caf44)
  - AWS CloudHSM용 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 4a88ba9b4cf0dd5573f3dd88ab9dc257e4c486069cb529c5d554979ee2dd83af)

CentOS 7 (7.8+)

x86_64 아키텍처의 CentOS 7용 버전 5.10.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL Dynamic Engine (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE 공급자 (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - AWS CloudHSM용 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)

- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 7 (7.8+)

x86_64 아키텍처의 RHEL 7용 버전 5.10.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL Dynamic Engine (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE 공급자 (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - AWS CloudHSM용 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 8 (8.3+)

x86_64 아키텍처의 RHEL 8용 버전 5.10.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum 96afb7042a148ddc7a60ab6235b49e176d0460d1c2957bd76ca3d8406ac1cb03)
- OpenSSL Dynamic Engine (SHA256 checksum 2caad2bffea8aef73c91ad422d09772ef830fe7f80a7be19020e6a107eadfbe8)
- JCE 공급자 (SHA256 checksum 3543551f08fbe3900821ea2d4ea148b4e86e2334bc94d7ffef6f3b831457cd71)
  - AWS CloudHSM용 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 812eccaadfc490f13bcd0b0a835ef58f3a3d4344ad7e0a237de476dd24509525)

Ubuntu 18.04 LTS

x86_64 아키텍처의 Ubuntu 18.04 LTS용 버전 5.10.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum be4c61766b8b46e1f6c14c3dcf90aaab9f38240fcd9c68b4009704276c5f6f4a)
- OpenSSL Dynamic Engine (SHA256 checksum 64bd8af827b6dc3786e8ad28858cbc4ef6a0fd42164a0945f427eddcf5f02858)

- JCE 공급자 (SHA256 checksum 9fcbdf08e93641468588b608173f26f18781bbc029ed95b2e086da29a968cc00)
  - AWS CloudHSM용 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 13808bddddb7eedeb2b8486d23a9976c7fa8d9220149a6b9400626bcaff3b513)

> **Note**
>
> Ubuntu 18.04 LTS의 최근 수명이 다함에 따라 다음 AWS CloudHSM 릴리스에서는 이 플랫폼을 더 이상 지원할 수 없습니다.

Ubuntu 20.04 LTS

x86_64 아키텍처의 Ubuntu 20.04 LTS용 버전 5.10.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum 99ae96504580ff85ed4958a582903a847f666bdaafafbe887a5a76db58f24500)
- OpenSSL Dynamic Engine (SHA256 checksum 13e3f6fe086acf9617b163f66e3941f973daa583fb9322d16c396aa29fc3611d)
- JCE 공급자 (SHA256 checksum 44562cebd9af1aa965840cd9bcb237e518d24c715b3c8bca1405c9c1871835e2)
  - AWS CloudHSM용 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum ab71b4ec531c5e6d05c91539c7edc1c07e6c748052ebf6200f148cb6812538c5)

Ubuntu 22.04 LTS

x86_64 아키텍처의 Ubuntu 22.04 LTS용 버전 5.10.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum ee331a44fbe4936ec98a3ae55d58e67ed38e8bbff0a4f4ce8b1bd8239b75877b)
- 이 플랫폼에서는 OpenSSL Dynamic Engine을 아직 지원하지 않습니다.
- JCE 공급자 (SHA256 checksum 9e44d14dd33624f6fe36711633013e47e4a93f4d4635e08900546113ded56e3d)
  - AWS CloudHSM용 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 2df361546848cd3f8965b1007dca42a0c959eb10d9e3f4995e8e1c852406751d)

Windows Server 2016

x86_64 아키텍처의 Windows Server 2016용 버전 5.10.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- JCE 공급자 (SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)
  - AWS CloudHSM용 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

Windows Server 2019

x86_64 아키텍처의 Windows Server 2019용 버전 5.10.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- JCE 공급자 (SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)
  - AWS CloudHSM용 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

클라이언트 SDK 5.10.0은 안정성을 개선하고 모든 SDK에 대한 버그 수정을 포함합니다.

CloudHSM CLI

- 고객이 CloudHSM CLI를 사용하여 키를 관리할 수 있는 새 명령이 추가되었습니다. 여기에는 다음이 포함됩니다.
  - 대칭 키 및 비대칭 키 페어 생성
  - 키 공유 및 공유 해제
  - 키 속성을 사용하여 키를 나열하고 필터링합니다.
  - 키 속성 설정
  - 키 참조 파일 생성
  - 키 삭제
- 오류 로깅을 개선함.

- 대화형 모드에서 여러 줄 유니코드 명령에 대한 지원이 추가되었습니다.

버그 수정/개선 사항

- 모든 SDK의 세션 키 가져오기, 언래핑, 파생 및 생성 성능이 개선되었습니다.
- 종료 시 임시 파일이 제거되지 않던 JCE 공급자의 버그를 수정했습니다.
- 클러스터의 HSM이 교체된 후 특정 조건에서 연결 오류가 발생하는 버그를 수정했습니다.
- 큰 마이너 버전 번호를 처리하고 패치 번호를 포함하도록 JCE `getVersion` 출력 형식을 수정했습니다.

플랫폼 지원

- JCE, PKCS #11 및 CloudHSM CLI와 함께 우분투 22.04에 대한 지원이 추가되었습니다(OpenSSL 동적 엔진에 대한 지원은 아직 제공되지 않음).

버전 5.9.0

Amazon Linux

x86_64 아키텍처의 Amazon Linux용 버전 5.9.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum 4f368be41f006b751ac41b14e1435c27841f60bbde0f032ec02a359fea637dcf)
- OpenSSL Dynamic Engine (SHA256 checksum 81af0d34683825cd6ff844ccacf9c8f4842a4ba76e3875a89121d09a286b4490)
- JCE 공급자 (SHA256 checksum e8e5bc09d8e0b3cb24f30ab420fe08902a19073012335ac94382ec55fcc45abd)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 17284144b45043204ce012fe8b62b1973f10068950abedbd9c2c6172ed0979c6)

Amazon Linux 2

x86_64 아키텍처의 Amazon Linux 2용 버전 5.9.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)

- OpenSSL Dynamic Engine (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE 공급자 (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

ARM64 아키텍처의 Amazon Linux 2용 버전 5.9.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum 4337dca5a08c5194b1118fa197bb4a4f7988df4e1b961e6f2e367295ba99d61d)
- OpenSSL Dynamic Engine (SHA256 checksum 4f08689934e877662a7ce64554fb04eb4b2c213b936018609ff187d100e34a85)
- JCE 공급자 (SHA256 checksum b337b80271a2d308949d5911971fe6ad35df4e34876a481fcac347f1d897fe39)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum a4d466e6b5f74dcd283ba32c9dd87441941d5e5a05936b7c2b4cc7ef85eb1071)

CentOS 7 (7.8+)

x86_64 아키텍처의 CentOS 7용 버전 5.9.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL Dynamic Engine (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE 공급자 (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 7 (7.8+)

x86_64 아키텍처의 RHEL 7용 버전 5.9.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)

- OpenSSL Dynamic Engine (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE 공급자 (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 8 (8.3+)

x86_64 아키텍처의 RHEL 8용 버전 5.9.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum 081887f6ea1d9df9d1e409b2b5bde83e965c42229acbeb1f950c8fe478361edc)
- OpenSSL Dynamic Engine (SHA256 checksum 6b0500a42fd57c39f076f14e5079f80145b6ebd2c441395761eb04600c07bda5)
- JCE 공급자 (SHA256 checksum 2bc7ac26b259af92a65fbd5a30d5eb2a92ce0e70efe41feb53bf82f168aa90bb)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 79ecbe9b4c5316ccf447d8c59b76b5ac2cc854bd79cd50c1f29197aa8cb080db)

Ubuntu 18.04 LTS

x86_64 아키텍처의 Ubuntu 18.04 LTS용 버전 5.9.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum bc6d2227edd7b5a83fed32741fbacbb1756d5df89ebb3435d96f0609a180db65)
- OpenSSL Dynamic Engine (SHA256 checksum 2d6a26434fa6faf337f1dfb42de033220fa405a82d4540e279639a03b3ee6e9d)
- JCE 공급자 (SHA256 checksum e12aef122f490e9026452ce31c25625b1accb9a5866b3d470488f10f047f1873)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum f0bcabe594db3e8ff86cc0f65c2a10858d34452eb6b9fc33d7aac05c0f5f4f30)

Ubuntu 20.04 LTS

x86_64 아키텍처의 Ubuntu 20.04 LTS용 버전 5.9.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum 15dde8182f432de9e7d369b05e384e1f2d80dcca85db3b16ecc26cdef1a34bb9)
- OpenSSL Dynamic Engine (SHA256 checksum c8ba94a999038af87d4905b7c1feb4cc87e20d1776a32ef6f6d11ee000b5a896)
- JCE 공급자 (SHA256 checksum de33cd3e8130a06d9da5207079533aac8276a1319ac435a3737b4f65bd8fb972)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum cfa31535ad9a99a5113496c06fbace38e9593491aca9bb031a18b51075973e68)

Windows Server 2016

x86_64 아키텍처의 Windows Server 2016용 버전 5.9.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- JCE 공급자 (SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

Windows Server 2019

x86_64 아키텍처의 Windows Server 2019용 버전 5.9.0 소프트웨어 다운로드:

- PKCS #11 라이브러리 (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- JCE 공급자 (SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)
  - AWS CloudHSM용 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

클라이언트 SDK 5.9.0은 안정성을 개선하고 모든 SDK에 대한 버그 수정을 포함합니다. 모든 SDK는 HSM을 사용할 수 없는 것으로 판단되는 경우 애플리케이션에 즉시 작업 장애를 알릴 수 있도록 최적화되었습니다. 이번 릴리스에는 JCE의 성능 개선 사항이 포함되어 있습니다.

JCE 공급자

- 확장된 성능
- 세션 풀 부족과 관련된 알려진 문제가 수정되었습니다.

버전 3.4.4

Linux 플랫폼에서 클라이언트 SDK 3을 업그레이드하려면 클라이언트와 모든 라이브러리를 동시에 업그레이드하는 배치 명령을 사용해야 합니다. 업그레이드에 대한 자세한 내용은 클라이언트 SDK 3 업그레이드를 참조하세요.

> **Note**
>
> 클라이언트 SDK 3 및 관련 명령줄 도구 (키 관리 유틸리티 및 CloudHSM 관리 유틸리티) 는 HSM 유형 hsm1.medium에서만 사용할 수 있습니다. 세부 정보는 AWS CloudHSM 클러스터 모드 및 HSM 유형를 참조하세요.

소프트웨어를 다운로드하려면 해당 운영 체제의 탭을 선택한 후, 각 소프트웨어 패키지에 대한 링크를 선택합니다.

Amazon Linux

Amazon Linux용 3.4.4 버전 소프트웨어 다운로드:

- AWS CloudHSM 클라이언트 (SHA256 checksum 900de424d70f41e661aa636f256a6a79cc43bea6b0fe6eb95c2aaa63e5289505)
- PKCS #11 라이브러리 (SHA256 checksum a3f93f084d59fee5d7c859292bc02cb7e7f15fb06e971171ebf9b52bbd229c30)
- OpenSSL Dynamic Engine (SHA256 checksum 8db07b9843d49016b0b6fec46d39881d94e426fcaae1cee2747be14af9313bb0)
- JCE 공급자 (SHA256 checksum 360617c55bf4caa8e6e78ede079ca68cf9ef11473e7918154c22ba908a219843)
- AWS CloudHSM 관리 유틸리티 (SHA256 checksum c9961ffe38921131bd6f3702e10d73588e68b8ab10fbb241723e676f4fa8c4fa)

Amazon Linux 2

Amazon Linux 2용 3.4.4 버전 소프트웨어 다운로드:

- AWS CloudHSM 클라이언트 (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS #11 라이브러리 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL Dynamic Engine (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE 공급자 (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM 관리 유틸리티 (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 6

AWS CloudHSM 클라이언트 SDK 버전 3.4.4의 CentOS 6은 지원하지 않습니다.

CentOS 6에서 the section called “버전 3.2.1”을 사용하거나 지원되는 플랫폼을 선택하세요.

CentOS 7 (7.8+)

CentOS 7용 3.4.4 버전 소프트웨어 다운로드:

- AWS CloudHSM 클라이언트 (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS #11 라이브러리 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL Dynamic Engine (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE 공급자 (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM 관리 유틸리티 (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 8

CentOS 8용 3.4.4 버전 소프트웨어 다운로드:

- AWS CloudHSM 클라이언트 (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)
- PKCS #11 라이브러리 (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)
- JCE 공급자 (SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM 관리 유틸리티 (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

> **Note**
>
> 최근 CentOS 8의 수명 종료로 인해 다음 릴리스에서는 이 플랫폼을 더 이상 지원할 수 없습니다.

RHEL 6

AWS CloudHSM 클라이언트 SDK 버전 3.4.4가 있는 RedHat 엔터프라이즈 리눅스 6은 지원하지 않습니다.

RedHat 엔터프라이즈 리눅스 the section called “버전 3.2.1” 6용으로 사용하거나 지원되는 플랫폼을 선택하세요.

RHEL 7 (7.8+)

RedHat 엔터프라이즈 리눅스 7용 버전 3.4.4 소프트웨어 다운로드:

- AWS CloudHSM 클라이언트 (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS #11 라이브러리 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL Dynamic Engine (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE 공급자 (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM 관리 유틸리티 (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

RHEL 8 (8.3+)

RedHat 엔터프라이즈 리눅스 8용 버전 3.4.4 소프트웨어 다운로드:

- AWS CloudHSM 클라이언트 (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)
- PKCS #11 라이브러리 (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)
- JCE 공급자 (SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM 관리 유틸리티 (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

Ubuntu 16.04 LTS

Ubuntu 16.04 LTS용 3.4.4 버전 소프트웨어 다운로드:

- AWS CloudHSM 클라이언트 (SHA256 checksum 317c92c2e0b5d60afab1beb947f053d13ddaacb994cccc2c2b898e997ece29b9)
- PKCS #11 라이브러리 (SHA256 checksum 91451c420c51488a022569fd32f052a3b988a2883ea4c2ac952acb61a2fea37c)
- OpenSSL Dynamic Engine (SHA256 checksum 4098771ad0e38df9bf14d50520ca49b9395f819f0387e2bc3b0e61abb5888e66)
- JCE 공급자 (SHA256 checksum e136ff183271c2f9590a9fccb8261a7eb809506686b070e3854df1b8686c6641)
- AWS CloudHSM 관리 유틸리티 (SHA256 checksum cbf24a4032f393a913a9898b1b27036392104e8e05d911cab84049b2bcca2541)

> **Note**
>
> Ubuntu 16.04의 EOL이 임박함에 따라 다음 릴리스에서는 이 플랫폼에 대한 지원을 중단할 계획입니다.

Ubuntu 18.04 LTS

Ubuntu 18.04 LTS용 3.4.4 버전 소프트웨어 다운로드:

- AWS CloudHSM 클라이언트 (SHA256 checksum cf57d5e0e95efbf032aac8887aebd59ac8cc80e97c69e7c39fdad40873374fe8)
- PKCS #11 라이브러리 (SHA256 checksum 428f8bdad7925db5401112f707942ee8f3ca554f4ab53fa92237996e69144d2f)
- JCE 공급자 (SHA256 checksum 1ff17b8f7688e84f7f0bfc96383564dca598a1cab2f2c52c888d0361682f2b9e)

- AWS CloudHSM 관리 유틸리티 (SHA256 checksum afe253046146ed6177c520b681efc680dac1048c4a95b3d8ad0f305e79bbe93e)

Windows Server

AWS CloudHSM 64비트 버전의 윈도우 서버 2012, 윈도우 서버 2012 R2, 윈도우 서버 2016 및 윈도우 서버 2019를 지원합니다. 윈도우 서버용 AWS CloudHSM 3.4.4 클라이언트 소프트웨어에는 필수 CNG 및 KSP 공급자가 포함되어 있습니다. 자세한 내용은 AWS CloudHSM 클라이언트 설치 및 구성 (Windows) 을 참조하십시오. Windows Server용 최신 버전(3.4.4) 소프트웨어 다운로드:

- AWS CloudHSM 윈도우 서버용 (SHA256 checksum d51a7db588e9121d8f0b0351606bd986e1c4de6547f2c8235200dc8a5ffbe53e)
- AWS CloudHSM 관리 유틸리티 (SHA256 checksum 0c12d7da9086735cdf189535937a8e036163009c5018dcaf2ee9cdbb6bd4c06f)

버전 3.4.4에는 JCE 제공자에 대한 업데이트가 추가되었습니다.

AWS CloudHSM 클라이언트 소프트웨어

- 일관성을 위해 버전이 업데이트되었습니다.

PKCS #11 라이브러리

- 일관성을 위해 버전이 업데이트되었습니다.

OpenSSL Dynamic Engine

- 일관성을 위해 버전이 업데이트되었습니다.

JCE 공급자

- log4j를 버전 2.17.1로 업데이트합니다.

Windows(CNG 및 KSP 공급자)

- 일관성을 위해 버전이 업데이트되었습니다.

사용되지 않는 릴리스

버전 5.8.0 및 이전 버전은 더 이상 사용되지 않습니다. 사용되지 않는 릴리스를 프로덕션 워크로드에 사용하지 않는 것이 좋습니다. 더 이상 사용되지 않는 릴리스에 대해서는 이전 버전과 호환되는 업데이트를 제공하지 않으며 더 이상 사용되지 않는 릴리스를 다운로드용으로 호스팅하지도 않습니다. 더 이상 사용되지 않는 릴리스를 사용하는 동안 프로덕션에 영향을 미치는 경우 소프트웨어 수정을 위해 업그레이드해야 합니다.

더 이상 사용되지 않는 클라이언트 SDK 5 릴리스

이 섹션에는 더 이상 사용되지 않는 클라이언트 SDK 5 릴리스가 나열되어 있습니다.

버전 5.8.0

버전 5.8.0에는 CloudHSM CLI에 대한 쿼럼 인증, JSSE를 통한 SSL/TLS 오프로드, PKCS #11 다중 슬롯 지원, JCE를 통한 다중 클러스터/다중 사용자 지원, JCE를 통한 키 추출, JCE용 KeyFactory 지원, 비터미널 반환 코드에 대한 새로운 재시도 구성이 도입되었으며 모든 SDK에 대한 향상된 안정성과 버그 수정이 포함되어 있습니다.

PKCS #11 라이브러리

- 다중 슬롯 구성에 대한 지원이 추가되었습니다.

JCE 공급자

- 구성 기반 키 추출이 추가되었습니다.
- 다중 클러스터 및 다중 사용자 구성에 대한 지원이 추가되었습니다.
- JSSE를 통한 SSL 및 TLS 오프로드에 대한 지원이 추가되었습니다.
- AES/CBC/에 대한 언랩 지원이 추가되었습니다. NoPadding
- 새로운 유형의 키 팩토리가 추가되었습니다: 및. SecretKeyFactory KeyFactory

CloudHSM CLI

- 쿼럼 인증 지원 추가

버전 5.7.0

버전 5.7.0에는 CloudHSM CLI가 도입되었으며 새로운 암호 기반 메시지 인증 코드(CMAC) 알고리즘이 포함되어 있습니다. 이 릴리스에는 Amazon Linux 2에 ARM 아키텍처가 추가되었습니다. 이제 JCE 제공자인 Javadocs를 사용할 수 있습니다. AWS CloudHSM

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.
- 이제 Amazon Linux 2의 ARM 아키텍처에서 지원됩니다.
- 알고리즘
  - CKM_AES_CMAC(서명 및 확인)

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.
- 이제 Amazon Linux 2의 ARM 아키텍처에서 지원됩니다.

JCE 공급자

- 개선된 안정성 및 버그 수정.
- 알고리즘
  - AESCMAC

버전 5.6.0

버전 5.6.0에는 PKCS #11 라이브러리 및 JCE 공급자에 대한 새로운 메커니즘 지원이 포함되어 있습니다. 또한 버전 5.6은 Ubuntu 20.04를 지원합니다.

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.
- 메커니즘
  - CKM_RSA_X_509, 암호화, 암호 해독, 서명 및 검증 모드용

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.

JCE 공급자

- 개선된 안정성 및 버그 수정.
- 암호(Ciphers)
  - RSA/ECB/NoPadding, 암호화 및 복호화 모드용

지원되는 키

- 곡선이 있는 EC(secp224r1 및 secp521r1)

플랫폼 지원

- Ubuntu 20.04에 대한 지원이 지원됩니다.

버전 5.5.0

버전 5.5.0에는 OpenJDK 11, Keytool 및 Jarsigner 통합, JCE 제공자에 대한 추가 메커니즘에 대한 지원이 추가되었습니다. KeyGenerator 클래스가 키 크기 매개변수를 비트 대신 바이트 수로 잘못 해석하는 것과 관련된 알려진 문제를 해결합니다.

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.

JCE 공급자

- Keytool 및 Jarsigner 유틸리티 지원
- 모든 플랫폼에서 OpenJDK 11에 대한 지원
- 암호(Ciphers)

  - NoPadding AES/CBC/ 암호화 및 복호화 모드
  - AES/ECB/PKCS5Padding 암호화 및 암호 해독 모드
  - AES/CTR/ NoPadding 암호화 및 복호화 모드
  - NoPadding AES/GCM/ 랩 앤 언랩 모드
  - DESede/ECB/PKCS5Padding 암호화 및 암호 해독 모드
  - 디시드/CBC/ 암호화 및 복호화 모드 NoPadding
  - AESWrap/ECB/랩 앤 NoPadding 언랩 모드
  - AESWrap/ECB/PKCS5Padding 래핑 앤 언래핑 모드
  - ZeroPadding AESWrap/ECB/랩 앤 언랩 모드
  - RSA/ECB/PKCS1Padding 래핑 및 언래핑 모드
  - RSA/ECB/OAEPPadding 래핑 및 언래핑 모드
  - RSA/ECB/OAEPWithSHA-1ANDMGF1Padding 래핑 및 언래핑 모드
  - RSA/ECB/OAEPWithSHA-224ANDMGF1Padding 래핑 및 언래핑 모드
  - RSA/ECB/OAEPWithSHA-256ANDMGF1Padding 래핑 및 언래핑 모드
  - RSA/ECB/OAEPWithSHA-384ANDMGF1Padding 래핑 및 언래핑 모드
  - RSA/ECB/OAEPWithSHA-512ANDMGF1Padding 래핑 및 언래핑 모드
  - RSAAESWrap/ECB/OAEPPadding 래핑 및 언래핑 모드
  - RSAAESWrap/ECB/OAEPWithSHA-1ANDMGF1Padding 래핑 및 언래핑 모드
  - RSAAESWrap/ECB/OAEPWithSHA-224ANDMGF1Padding 래핑 및 언래핑 모드
  - RSAAESWrap/ECB/OAEPWithSHA-256ANDMGF1Padding 래핑 및 언래핑 모드
  - RSAAESWrap/ECB/OAEPWithSHA-384ANDMGF1Padding 래핑 및 언래핑 모드
  - RSAAESWrap/ECB/OAEPWithSHA-512ANDMGF1Padding 래핑 및 언래핑 모드
- KeyFactory 그리고 SecretKeyFactory
  - RSA 2048비트 – 4096비트의 RSA 키(256비트씩 증분)
  - AES – 128비트, 192비트 및 256비트 AES 키
  - NIST 곡선 secp256r1(P-256), secp384r1(P-384) 및 secp256k1용 EC 키 페어입니다.
  - DESede (3DES)
  - GenericSecret
  - HMAC — SHA1, SHA224, SHA256, SHA384, SHA512 해시 지원 포함
- 서명/확인

- RASASSA-PSS
- SHA1withRSA/PSS
- SHA224withRSA/PSS
- SHA256withRSA/PSS
- SHA384withRSA/PSS
- SHA512withRSA/PSS
- SHA1withRSAandMGF1
- SHA224withRSAandMGF1
- SHA256withRSAandMGF1
- SHA384withRSAandMGF1
- SHA512withRSAandMGF1

버전 5.4.2

버전 5.4.2에는 모든 SDK의 향상된 안정성과 버그 수정이 포함되어 있습니다. 이것은 CentOS 8 플랫폼의 마지막 릴리스이기도 합니다. 자세한 내용은 CentOS 웹 사이트를 참조하세요.

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.

JCE 공급자

- 개선된 안정성 및 버그 수정.

버전 5.4.1

버전 5.4.1은 PKCS #11 라이브러리의 알려진 문제를 해결합니다. 이것은 CentOS 8 플랫폼의 마지막 릴리스이기도 합니다. 자세한 내용은 CentOS 웹 사이트를 참조하세요.

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.

JCE 공급자

- 개선된 안정성 및 버그 수정.

버전 5.4.0

버전 5.4.0에는 모든 플랫폼에 대한 JCE 제공자에 대한 초기 지원이 추가되었습니다. JCE 공급자는 OpenJDK 8과 호환됩니다.

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.

JCE 공급자

- 키 유형
  - RSA - 2048비트 ~ 4096비트의 RSA 키입니다(256비트씩 증분).
  - AES - 128비트, 192비트 및 256비트 AES 키입니다.
  - NIST 곡선 secp256r1(P-256), secp384r1(P-384) 및 secp256k1용 ECC 키 페어입니다.
  - DESede (3DES)
  - HMAC — SHA1, SHA224, SHA256, SHA384, SHA512 해시 지원
- 암호(암호화 및 복호화만 해당)
  - AES/GCM/ NoPadding
  - AES/ECB/ NoPadding

- AES/CBC/PKCS5Padding
- 스웨덴/ECB/ NoPadding
- DESede/CBC/PKCS5Padding
- AES/CTR/ NoPadding
- RSA/ECB/PKCS1Padding
- RSA/ECB/OAEPPadding
- RSA/ECB/OAEPWithSHA-1ANDMGF1Padding
- RSA/ECB/OAEPWithSHA-224ANDMGF1Padding
- RSA/ECB/OAEPWithSHA-256ANDMGF1Padding
- RSA/ECB/OAEPWithSHA-384ANDMGF1Padding
- RSA/ECB/OAEPWithSHA-512ANDMGF1Padding

- 다이제스트
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- 서명/확인
  - NONEwithRSA
  - SHA1withRSA
  - SHA224(RSA 포함)
  - SHA256withRSA
  - SHA384withRSA
  - SHA512withRSA
  - NONEwithECDSA
  - SHA1(ECDSA 포함)
  - SHA224(ECDSA 포함)
  - SHA256(ECDSA 포함)
  - SHA384(ECDSA 포함)
  - SHA512(ECDSA 포함)

- 자바와의 통합 KeyStore

버전 5.3.0

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.

OpenSSL Dynamic Engine

- P-256, P-384 및 secp256k1 곡선을 사용하여 ECDSA 서명/검증에 대한 지원을 추가합니다.
- 플랫폼에 대한 지원을 추가합니다: 아마존 리눅스, 아마존 리눅스 2, 센토스 7.8+, RHEL 7 (7.8+).
- OpenSSL 버전 1.0.2에 대한 지원을 추가합니다.
- 개선된 안정성 및 버그 수정.

JCE 공급자

- 키 유형
  - RSA - 2048비트 ~ 4096비트의 RSA 키입니다(256비트씩 증분).
  - AES - 128비트, 192비트 및 256비트 AES 키입니다.
  - NIST 곡선 secp256r1(P-256), secp384r1(P-384) 및 secp256k1용 EC 키 페어입니다.
  - DESede (3DES)
  - HMAC — SHA1, SHA224, SHA256, SHA384, SHA512 해시 지원
- 암호(암호화 및 복호화만 해당)
  - AES/GCM/ NoPadding
  - AES/ECB/ NoPadding
  - AES/CBC/PKCS5Padding
  - 스웨덴/ECB/ NoPadding
  - DESede/CBC/PKCS5Padding
  - AES/CTR/ NoPadding
  - RSA/ECB/PKCS1Padding
  - RSA/ECB/OAEPPadding
  - RSA/ECB/OAEPWithSHA-1ANDMGF1Padding

  - RSA/ECB/OAEPWithSHA-224ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-256ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-384ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-512ANDMGF1Padding
- 다이제스트
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- 서명/확인
  - NONEwithRSA
  - SHA1withRSA
  - SHA224(RSA 포함)
  - SHA256withRSA
  - SHA384withRSA
  - SHA512withRSA
  - NONEwithECDSA
  - SHA1(ECDSA 포함)
  - SHA224(ECDSA 포함)
  - SHA256(ECDSA 포함)
  - SHA384(ECDSA 포함)
  - SHA512(ECDSA 포함)
- 자바와의 통합 KeyStore

버전 5.2.1

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.

버전 5.2.0

버전 5.2.0은 PKCS #11 라이브러리에 추가 키 유형 및 메커니즘 지원을 추가합니다.

PKCS #11 라이브러리

키 유형

- ECDSA — P-224, P-256, P-384, P-521 및 secp256k1 곡선
- Triple DES(3DES)

메커니즘

- CM_EC_KEY_PAIR_GEN
- CM_DES3_KEY_GEN
- CM_DES3_CBC
- CM_DES3_CBC_PAD
- CKM_DES3_ECB
- CM_ECDSA
- CM_ECDSA_SHA1
- CKM_ECDSA_SHA224
- CKM_ECDSA_SHA256
- CKM_ECDSA_SHA384
- CKM_ECDSA_SHA512
- 암호화/암호 해독을 위한 CKM_RSA_PKCS

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.

버전 5.1.0

버전 5.1.0은 PKCS #11 라이브러리에 추가 메커니즘에 대한 지원을 추가합니다.

PKCS #11 라이브러리

메커니즘

- 래핑/언래핑을 위한 CKM_RSA_PKCS
- CKM_RSA_PKCS_PSS
- CKM_SHA1_RSA_PKCS_PSS
- CKM_SHA224_RSA_PKCS_PSS
- CKM_SHA256_RSA_PKCS_PSS
- CKM_SHA384_RSA_PKCS_PSS
- CKM_SHA512_RSA_PKCS_PSS
- CM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CM_AES_CBC_PAD
- CKM_SP800_108_COUNTER_KDF
- CM_GENERIC_SECRET_KEY_GEN
- CM_SHA_1_HMAC
- CM_SHA224_HMAC
- CM_SHA256_HMAC
- CM_SHA384_HMAC
- CM_SHA512_HMAC
- CKM_RSA_PKCS_OAEP 래핑핑/언래핑핑 전용
- CM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
- CM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD

API 작업

- C_ CreateObject
- C_ DeriveKey
- C_ WrapKey
- C_ UnWrapKey

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.

버전 5.0.1

버전 5.0.1에는 OpenSSL Dynamic Engine에 대한 초기 지원이 추가되었습니다.

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.

OpenSSL Dynamic Engine

- OpenSSL Dynamic Engine의 초기 릴리스.
- 이 릴리스에서는 키 유형 및 OpenSSL API에 대한 기본 지원을 제공합니다.
  - 2048비트, 3072비트 및 4096비트 키를 위한 RSA 키 생성
  - OpenSSL API:
    - SHA1/224/256/384/512 및 RSA PSS와 함께 RSA PKCS를 사용한 RSA 서명
    - RSA 키 생성

  자세한 내용은 OpenSSL 동적 엔진을 참조하세요.
- 지원되는 플랫폼: CentOS 8.3+, 레드햇 엔터프라이즈 리눅스(RHEL) 8.3+, 우분투 18.04 LTS
  - 필요: OpenSSL 1.1.1

  자세한 내용은 지원되는 플랫폼 섹션을 참조하세요.
- NGINX 1.19를 포함한 CentOS 8.3+, 레드햇 엔터프라이즈 리눅스(RHEL) 8.3 및 우분투 18.04 LTS에서 SSL/TLS 오프로드를 지원합니다(일부 암호 제품군용).

  자세한 내용은 Linux에서의 SSL/TLS 오프로드를 참조하세요.

버전 5.0.0

버전 5.0.0이 첫 번째 릴리스입니다.

PKCS #11 라이브러리

- 이것은 최초 릴리스입니다.

PKCS #11 라이브러리 입문용 PKCS 라이브러리 버전 5.0.0

이 섹션에서는 키 유형, 메커니즘, API 작업 및 속성 클라이언트 SDK 버전 5.0.0에 대한 지원을 자세히 설명합니다.

키 유형:

- AES- 128비트, 192비트 및 256비트 AES 키
- RSA– 2048비트 ~ 4096비트의 RSA 키(256비트씩 증분)

메커니즘:

- CKM_AES_GCM
- CM_AES_KEY_GEN
- CKM_CLOUDHSM_AES_GCM
- CKM_RSA_PKCS
- CKM_RSA_X9_31_KEY_PAIR_GEN
- CM_SHA1
- CM_SHA1_RSA_PKCS
- CKM_SHA224
- CKM_SHA224_RSA_PKCS
- CKM_SHA256
- CKM_SHA256_RSA_PKCS
- CKM_SHA384
- CKM_SHA384_RSA_PKCS
- CKM_SHA512

- CM_SHA512_RSA_PKCS

API 연산:

- C_ CloseAllSessions
- C_ CloseSession
- C_Decrypt
- C_ DecryptFinal
- C_ DecryptInit
- C_ DecryptUpdate
- C_ DestroyObject
- C_Digest
- C_ DigestFinal
- C_ DigestInit
- C_ DigestUpdate
- C_Encrypt
- C_ EncryptFinal
- C_ EncryptInit
- C_ EncryptUpdate
- C_Finalize
- C_ FindObjects
- C_ FindObjectsFinal
- C_ FindObjectsInit
- C_ GenerateKey
- C_ GenerateKeyPair
- C_ GenerateRandom
- C_ GetAttributeValue
- C_ GetFunctionList
- C_ GetInfo
- C_ GetMechanismInfo

- C_ GetMechanismList
- C_ GetSessionInfo
- C_ GetSlotInfo
- C_ GetSlotList
- C_ GetTokenInfo
- C_Initialize
- C_Login
- C_Logout
- C_ OpenSession
- C_Sign
- C_ SignFinal
- C_ SignInit
- C_ SignUpdate
- C_Verify
- C_ VerifyFinal
- C_ VerifyInit
- C_ VerifyUpdate

속성:

- GenerateKeyPair
  - 모든 RSA 키 속성
- GenerateKey
  - 모든 AES 키 속성
- GetAttributeValue
  - 모든 RSA 키 속성
  - 모든 AES 키 속성

샘플:

- 키 생성(AES, RSA, EC)
- 키 속성 나열

- AES GCM을 사용하여 데이터 암호화 및 암호화 해제
- RSA를 사용하여 데이터 서명 및 확인

더 이상 사용되지 않는 클라이언트 SDK 3 릴리스

이 섹션에는 더 이상 사용되지 않는 클라이언트 SDK 3 릴리스가 나열되어 있습니다.

버전 3.4.3

버전 3.4.3에는 JCE 공급자에 대한 업데이트가 추가되었습니다.

AWS CloudHSM 클라이언트 소프트웨어

- 일관성을 위해 버전이 업데이트되었습니다.

PKCS #11 라이브러리

- 일관성을 위해 버전이 업데이트되었습니다.

OpenSSL Dynamic Engine

- 일관성을 위해 버전이 업데이트되었습니다.

JCE 공급자

- log4j를 버전 2.17.0으로 업데이트합니다.

Windows(CNG 및 KSP 공급자)

- 일관성을 위해 버전이 업데이트되었습니다.

버전 3.4.2

버전 3.4.2에는 JCE 공급자에 대한 업데이트가 추가되었습니다.

AWS CloudHSM 클라이언트 소프트웨어

- 일관성을 위해 버전이 업데이트되었습니다.

PKCS #11 라이브러리

- 일관성을 위해 버전이 업데이트되었습니다.

OpenSSL Dynamic Engine

- 일관성을 위해 버전이 업데이트되었습니다.

JCE 공급자

- log4j를 버전 2.16.0으로 업데이트합니다.

Windows(CNG 및 KSP 공급자)

- 일관성을 위해 버전이 업데이트되었습니다.

버전 3.4.1

버전 3.4.1에는 JCE 공급자에 대한 업데이트가 추가되었습니다.

AWS CloudHSM 클라이언트 소프트웨어

- 일관성을 위해 버전이 업데이트되었습니다.

PKCS #11 라이브러리

- 일관성을 위해 버전이 업데이트되었습니다.

OpenSSL Dynamic Engine

- 일관성을 위해 버전이 업데이트되었습니다.

JCE 공급자

- log4j를 버전 2.15.0으로 업데이트합니다.

Windows(CNG 및 KSP 공급자)

- 일관성을 위해 버전이 업데이트되었습니다.

버전 3.4.0

버전 3.4.0은 모든 구성 요소에 업데이트를 추가합니다.

AWS CloudHSM 클라이언트 소프트웨어

- 개선된 안정성 및 버그 수정.

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.

JCE 공급자

- 개선된 안정성 및 버그 수정.

Windows(CNG 및 KSP 공급자)

- 개선된 안정성 및 버그 수정.

버전 3.3.2

버전 3.3.2에서는 client_info 스크립트 관련 문제가 해결되었습니다.

AWS CloudHSM 클라이언트 소프트웨어

- 일관성을 위해 버전이 업데이트되었습니다.

PKCS #11 라이브러리

- 일관성을 위해 버전이 업데이트되었습니다.

OpenSSL Dynamic Engine

- 일관성을 위해 버전이 업데이트되었습니다.

JCE 공급자

- 일관성을 위해 버전이 업데이트되었습니다.

Windows(CNG 및 KSP 공급자)

- 일관성을 위해 버전이 업데이트되었습니다.

버전 3.3.1

버전 3.3.1에는 모든 구성 요소에 대한 업데이트가 추가되었습니다.

AWS CloudHSM 클라이언트 소프트웨어

- 개선된 안정성 및 버그 수정.

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.

JCE 공급자

- 개선된 안정성 및 버그 수정.

Windows(CNG 및 KSP 공급자)

- 개선된 안정성 및 버그 수정.

버전 3.3.0

버전 3.3.0에는 2단계 인증(2FA) 및 기타 개선 사항이 추가되었습니다.

AWS CloudHSM 클라이언트 소프트웨어

- 암호화폐 책임자(CO) 를 위한 2FA 인증을 추가했습니다. 자세한 내용은 Crypto Officer에 대한 2단계 인증 관리를 참조하세요.
- RedHat 엔터프라이즈 리눅스 6 및 CentOS 6에 대한 플랫폼 지원이 제거되었습니다. 자세한 내용은 OS X 지원을 참조하세요.
- 클라이언트 SDK 5 또는 클라이언트 SDK 3과 함께 사용할 수 있는 CMU의 독립형 버전을 추가했습니다. 이 CMU는 버전 3.3.0의 클라이언트 데몬에 포함된 것과 동일한 버전의 CMU이며, 이제 클라이언트 데몬을 다운로드하지 않고도 CMU를 다운로드할 수 있습니다.

PKCS #11 라이브러리

- 개선된 안정성 및 버그 수정.
- RedHat 엔터프라이즈 리눅스 6 및 CentOS 6에 대한 플랫폼 지원이 제거되었습니다. 자세한 내용은 OS X 지원을 참조하세요.

OpenSSL Dynamic Engine

- 일관성을 위해 버전이 업데이트되었습니다.
- RedHat 엔터프라이즈 리눅스 6 및 CentOS 6에 대한 플랫폼 지원이 제거되었습니다. 자세한 내용은 OS X 지원을 참조하세요.

JCE 공급자

- 개선된 안정성 및 버그 수정.
- RedHat 엔터프라이즈 리눅스 6 및 CentOS 6에 대한 플랫폼 지원이 제거되었습니다. 자세한 내용은 OS X 지원을 참조하세요.

Windows(CNG 및 KSP 공급자)

- 일관성을 위해 버전이 업데이트되었습니다.

버전 3.2.1

버전 3.2.1에는 PKCS #11 라이브러리 AWS CloudHSM 구현과 PKCS #11 표준, 새 플랫폼 및 기타 개선 사항 간의 규정 준수 분석이 추가되었습니다.

AWS CloudHSM 클라이언트 소프트웨어

- CentOS 8, RHEL 8 및 Ubuntu 18.04 LTS에 대한 플랫폼 지원을 추가합니다. 자세한 정보는 ???을 참조하세요.

PKCS #11 라이브러리

- 클라이언트 SDK 3.2.1에 대한 PKCS #11 라이브러리 규정 준수 보고서
- CentOS 8, RHEL 8 및 Ubuntu 18.04 LTS에 대한 플랫폼 지원을 추가합니다. 자세한 정보는 ???을 참조하세요.

OpenSSL Dynamic Engine

- CentOS 8, RHEL 8 및 Ubuntu 18.04 LTS는 지원되지 않습니다. 자세한 정보는 ???을 참조하세요.

JCE 공급자

- CentOS 8, RHEL 8 및 Ubuntu 18.04 LTS에 대한 플랫폼 지원을 추가합니다. 자세한 정보는 ???을 참조하세요.

Windows(CNG 및 KSP 공급자)

- 개선된 안정성 및 버그 수정.

버전 3.2.0

버전 3.2.0에는 암호 마스킹 및 기타 개선 사항에 대한 지원이 추가되었습니다.

AWS CloudHSM 클라이언트 소프트웨어

- 명령줄 도구를 사용할 때 암호를 숨기는 서포트를 추가합니다. 자세한 내용은 LoginHSM 및 LogouthSM(cloudhsm_mgmt_util) 및 LoginHSM 및 LogouthSM(key_mgmt_util)을 참조하세요.

PKCS #11 라이브러리

- 이전에는 지원되지 않았던 일부 PKCS #11 메커니즘에 대해 소프트웨어에서 대용량 데이터를 해싱할 수 있는 지원을 추가합니다. 자세한 내용은 지원되는 메커니즘을 참조하세요.

OpenSSL Dynamic Engine

- 개선된 안정성 및 버그 수정.

JCE 공급자

- 일관성을 위해 버전이 업데이트되었습니다.

Windows(CNG 및 KSP 공급자)

- 개선된 안정성 및 버그 수정.

버전 3.1.2

버전 3.1.2에는 JCE 공급자에 대한 업데이트가 추가되었습니다.

AWS CloudHSM 클라이언트 소프트웨어

- 일관성을 위해 버전이 업데이트되었습니다.

PKCS #11 라이브러리

- 일관성을 위해 버전이 업데이트되었습니다.

OpenSSL Dynamic Engine

- 일관성을 위해 버전이 업데이트되었습니다.

JCE 공급자

- log4j를 버전 2.13.3으로 업데이트

Windows(CNG 및 KSP 공급자)

- 일관성을 위해 버전이 업데이트되었습니다.

버전 3.1.1

AWS CloudHSM 클라이언트 소프트웨어

- 일관성을 위해 버전이 업데이트되었습니다.

PKCS #11 라이브러리

- 일관성을 위해 버전이 업데이트되었습니다.

OpenSSL Dynamic Engine

- 일관성을 위해 버전이 업데이트되었습니다.

JCE 공급자

- 버그 수정 및 성능 향상.

Windows(CNG, KSP)

- 일관성을 위해 버전이 업데이트되었습니다.

버전 3.1.0

버전 3.1은 표준 호환 AES 키 래핑을 추가합니다.

AWS CloudHSM 클라이언트 소프트웨어

- 업그레이드를 위한 새로운 요구 사항: 클라이언트 버전이 사용 중인 소프트웨어 라이브러리의 버전과 일치해야 합니다. 업그레이드하려면 클라이언트와 모든 라이브러리를 동시에 업그레이드하는 배치 명령을 사용해야 합니다. 자세한 내용은 클라이언트 SDK 3 업그레이드를 참조하세요.
- Key_mgmt_util(KMU)에는 다음 업데이트가 포함됩니다.
  - 제로 패딩을 사용하는 표준 호환 AES 키 래핑과 패딩이 없는 AES 키 래핑의 두 가지 새 AES 키 래핑이 추가되었습니다. 자세한 내용은 wrapKey 및 unwrapKey를 참조하세요

- AES_KEY_WRAP_PAD_PKCS5를 사용하여 키를 래핑할 때 사용자 지정 IV를 지정하는 기능이 비활성화되었습니다. 자세한 내용은 AES 키 래핑을 참조하세요.

PKCS #11 라이브러리

- 제로 패딩을 사용하는 표준 호환 AES 키 래핑과 패딩이 없는 AES 키 래핑의 두 가지 새 AES 키 래핑이 추가되었습니다. 자세한 내용은 AES 키 래핑을 참조하세요.
- RSA-PSS 서명에 대한 솔트 길이를 구성할 수 있습니다. 이 기능을 사용하는 방법을 알아보려면 RSA-PSS 서명의 구성 가능한 솔트 길이 설정을 참조하십시오. GitHub

OpenSSL Dynamic Engine

- 주요 변경 사항: SHA1을 사용한 TLS 1.0 및 1.2 암호 그룹은 OpenSSL Engine 3.1.0에서 사용할 수 없습니다. 이 문제는 곧 해결될 것입니다.
- RHEL 6 또는 CentOS 6에 OpenSSL Dynamic Engine 라이브러리를 설치하려는 경우 해당 운영 체제에 설치된 기본 OpenSSL 버전에 대한 알려진 문제를 참조하세요.
- 개선된 안정성 및 버그 수정

JCE 공급자

- 주요 변경 사항: JCE(Java Cryptography Extension) 규정 준수 문제를 해결하기 위해 이제 AES 래핑 및 언래핑에서 AES 알고리즘 대신 AESWrap 알고리즘을 사용합니다. 이는 `Cipher.WRAP_MODE` 및 `Cipher.UNWRAP_MODE`가 AES/ECB 및 AES/CBC 메커니즘에 대해 더 이상 성공할 수 없음을 의미합니다.

  클라이언트 버전 3.1.0으로 업그레이드하려면 코드를 업데이트해야 합니다. 기존 래핑된 키가 있는 경우, 언래핑하는 데 사용하는 메커니즘과 IV 기본값이 변경된 방식에 대해 특히 주의해야 합니다. 클라이언트 버전 3.0.0 이하에서 키를 래핑한 경우, 3.1.1에서 기존 키를 언래핑하려면 AESWrap/ECB/PKCS5Padding을 사용해야 합니다. 자세한 내용은 AES 키 래핑을 참조하세요.
- JCE 공급자에서 동일한 레이블로 여러 키를 나열할 수 있습니다. 사용 가능한 모든 키를 반복하는 방법을 알아보려면 모든 키 찾기를 참조하십시오. GitHub
- 퍼블릭 키와 프라이빗 키에 대해 서로 다른 레이블을 지정하는 등 키를 생성하는 동안 속성에 대해 더 제한적인 값을 설정할 수 있습니다. 자세한 내용은 지원되는 Java 속성을 참조하세요.

Windows(CNG, KSP)

- 개선된 안정성 및 버그 수정.

E 릴리스 nd-of-life

AWS CloudHSM 더 이상 서비스와 호환되지 않는 릴리스의 수명이 종료되었음을 알립니다. 애플리케이션의 안전을 유지하기 위해 당사는 end-of-life 릴리즈로부터의 연결을 적극적으로 거부할 권리를 보유합니다.

- 현재 end-of-life 출시된 클라이언트 SDK 버전은 없습니다.

문서 기록

이 주제에서는 AWS CloudHSM 사용 설명서에 대한 중요한 업데이트 사항에 대해 설명합니다.

주제

최신 업데이트

다음 표에서는 2018년 4월 이후 이 설명서의 중요한 변경 사항을 설명합니다. 또한 여기에 나열된 주요 변경 사항 외에도 설명 및 예시를 개선하고 보내주신 피드백을 해결하기 위해 문서를 자주 업데이트합니다. 중요한 변경 사항에 대해 알림을 받으려면 오른쪽 상단 모서리의 링크를 사용하여 RSS 피드를 구독합니다.

새 릴리스에 대한 자세한 내용은 AWS CloudHSM 클라이언트 SDK용 다운로드을 참조하십시오.

변경 사항	설명	날짜
새 HSM 유형 및 클러스터 모드	새 HSM 유형 (hsm2m.medium) 과 새 클러스터 모드 (비 FIPS) 를 출시했습니다.	2024년 6월 10일
새로 추가된 릴리스	클라이언트 버전 5.12.0이 AWS CloudHSM 출시되었습니다.	2024년 3월 20일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.11.0이 출시되었습니다.	2024년 1월 17일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.10.0이 출시되었습니다.	2023년 7월 28일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.9.0이 출시되었습니다.	2023년 5월 23일

새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.8.0이 출시되었습니다.	2023년 3월 16일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.7.0이 출시되었습니다.	2022년 11월 16일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.6.0이 출시되었습니다.	2022년 9월 1일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.5.0이 출시되었습니다.	2022년 5월 13일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.4.2가 출시되었습니다.	2022년 3월 18일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.4.1이 출시되었습니다.	2022년 2월 10일
새로 추가된 릴리스	Windows 플랫폼용 AWS CloudHSM JCE 프로바이더 버전 5.4.0을 출시했습니다.	2022년 2월 1일
새로 추가된 릴리스	모든 Linux 플랫폼에 대한 JCE 공급자에 대한 초기 지원을 추가하는 AWS CloudHSM 클라이언트 버전 5.4.0을 출시했습니다.	2022년 1월 28일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.3.0이 출시되었습니다.	2022년 1월 3일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.4.4가 출시되었습니다.	2022년 1월 3일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.4.3이 출시되었습니다.	2021년 12월 20일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.4.2가 출시되었습니다.	2021년 12월 15일

새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.4.1이 출시되었습니다.	2021년 12월 10일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.2.1이 출시되었습니다.	2021년 10월 4일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.4.0이 출시되었습니다.	2021년 8월 25일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.2.0이 출시되었습니다.	2021년 8월 3일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.3.2를 출시했습니다.	2021년 7월 2일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.1.0이 출시되었습니다.	2021년 6월 1일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.3.1이 출시되었습니다.	2021년 4월 26일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.0.1이 출시되었습니다.	2021년 4월 8일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 5.0.0이 출시되었습니다.	2021년 3월 12일
새로 추가된 내용	인터넷이나 NAT 디바이스, VPN 연결 또는 연결을 통해 AWS CloudHSM 액세스하지 않고도 VPC 간에 프라이빗 연결을 생성할 수 있는 AWS 기능인 인터페이스 VPC 엔드포인트가 추가되었습니다. AWS Direct Connect	2021년 2월 10일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.3.0이 출시되었습니다.	2021년 2월 3일

새로운 내용 추가하기	오래된 백업을 자동으로 삭제하는 기능인 관리형 백업 보존 기능이 추가되었습니다.	2020년 11월 18일
새로운 내용 추가하기	PKCS #11 표준을 사용하는 PKCS #11 라이브러리의 AWS CloudHSM 클라이언트 SDK 3.2.1 구현을 분석하는 규정 준수 보고서가 추가되었습니다.	2020년 10월 29일
새로 추가된 릴리스	클라이언트 버전 3.2.1이 출시되었습니다. AWS CloudHSM	2020년 10월 8일
새로 추가된 내용	AWS CloudHSM에 키 동기화 설정을 설명하는 설명서가 추가되었습니다.	2020년 9월 1일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.2.0이 출시되었습니다.	2020년 8월 31일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.1.2를 출시했습니다.	2020년 7월 30일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.1.1을 출시했습니다.	2020년 6월 3일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.1.0이 출시되었습니다.	2020년 5월 21일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 3.0.1을 출시했습니다.	2020년 4월 20일
새로 추가된 릴리스	윈도우 서버 플랫폼용 AWS CloudHSM 클라이언트 버전 3.0.0이 출시되었습니다.	2019년 10월 30일

새로 추가된 릴리스	Windows를 제외한 모든 플랫폼에서 사용할 수 있는 AWS CloudHSM 클라이언트 버전 3.0.0이 출시되었습니다.	2019년 10월 22일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 2.0.4가 출시되었습니다.	2019년 8월 26일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 2.0.3이 출시되었습니다.	2019년 5월 13일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 2.0.1이 출시되었습니다.	2019년 3월 21일
새로 추가된 릴리스	AWS CloudHSM 클라이언트 버전 2.0.0이 출시되었습니다.	2019년 2월 6일
리전 지원 추가	EU (스톡홀름) 및 AWS GovCloud (미국 동부) 지역에 대한 AWS CloudHSM 지원이 추가되었습니다.	2018년 12월 19일
새로 추가된 릴리스	Windows용 AWS CloudHSM 클라이언트 버전 1.1.2를 출시했습니다.	2018년 11월 20일
업데이트된 알려진 문제	새로운 내용이 문제 해결 안내서에 추가되었습니다.	2018년 11월 8일
새로 추가된 릴리스	Linux 플랫폼용 AWS CloudHSM 클라이언트 버전 1.1.2를 출시했습니다.	2018년 11월 8일
리전 지원 추가	EU (파리) 및 아시아 태평양 (서울) 지역에 대한 AWS CloudHSM 지원이 추가되었습니다.	2018년 10월 24일

새로 추가된 내용	AWS CloudHSM 백업 삭제 및 복원 기능이 추가되었습니다.	2018년 9월 10일
새로 추가된 내용	Amazon Logs에 자동 감사 CloudWatch 로그 전송을 추가했습니다.	2018년 8월 13일
새로 추가된 내용	지역 간에 AWS CloudHSM 클러스터 백업을 복사하는 기능이 추가되었습니다.	2018년 7월 30일
리전 지원 추가	EU (런던) 지역에 대한 AWS CloudHSM 지원이 추가되었습니다.	2018년 13월 6일
새로 추가된 내용	아마존 리눅스 2, 레드햇 엔터프라이즈 리눅스 (RHEL) 6, 레드햇 엔터프라이즈 리눅스 (RHEL) 7, CentOS 6, CentOS 7, 우분투 16.04 LTS에 대한 AWS CloudHSM 클라이언트 및 라이브러리 지원이 추가되었습니다.	2018년 5월 10일
새로 추가된 릴리스	AWS CloudHSM 윈도우 클라이언트가 추가되었습니다.	2018년 4월 30일

이전 업데이트

다음 표에는 2018년 AWS CloudHSM 이전의 중요한 변경 사항이 설명되어 있습니다.

변경 사항	설명	날짜
새로운 내용	CO(Crypto Officer)에 대한 쿼럼 인증(N 액세스 컨트롤 중 M)이 추가되었습니다. 자세한 정보는 CloudHSM 관리 유틸리티	2017년 11월 9일

변경 사항	설명	날짜
	(CMU)를 사용하여 쿼럼 인증 관리(M/N 액세스 제어)을 참조하세요.	
업데이트	key_mgmt_util 명령줄 도구 사용에 대한 설명서가 추가되었습니다. 자세한 정보는 key_mgmt_util 명령 참조을 참조하세요.	2017년 11월 9일
새로운 내용	Oracle Transparent Data Encryption이 추가되었습니다. 자세한 정보는 Oracle Database 암호화을 참조하세요.	2017년 10월 25일
새로운 내용	SSL 오프로드가 추가되었습니다. 자세한 정보는 SSL/TLS 오프로드을 참조하세요.	2017년 10월 12일
새 안내서	이 릴리스에는 다음이 포함됩니다. AWS CloudHSM	2017년 8월 14일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.