aws

使用者指南

AWS CloudHSM

AWS CloudHSM: 使用者指南

Table of Contents

什麼是 AWS CloudHSM？

AWS CloudHSM 結合 AWS 雲端的優點與硬體安全模組 (HSM) 的安全性。硬體安全模組 (HSM) 是一種運算裝置，可處理密碼編譯操作，並為密碼編譯金鑰提供安全的儲存空間。您可以完全控制 AWS 雲端中的高可用性 HSM、具 AWS CloudHSM有低延遲存取，以及可自動化 HSM 管理的安全信任根 (包括備份、佈建、組態和維護)。

AWS CloudHSM 為客戶提供多種好處：

對 FIPS 和非 FIP 叢集的存取

AWS CloudHSM 提供兩種模式的叢集：FIPS 和非 FIP。在 FIPS 模式下，只能使用聯邦資訊處理標準 (FIPS) 核准的金鑰和演算法。非 FIPS 模式提供支援的所有金鑰和演算法 AWS CloudHSM，無論 FIPS 核准為何。如需詳細資訊，請參閱 AWS CloudHSM 叢集模式和 HSM 類型。

HSM 是一般用途、單一租用戶，以及通過 FIPS 140-2 層級 3 驗證的 FIPS 140-2 層級 3，適用於 FIPS 模式下的叢集

AWS CloudHSM 使用一般用途 HSM，相較於具有針對應用程式預先確定演算法和金鑰長度的全受管 AWS 服務，可提供更大的彈性。我們提供符合標準、單一租用戶的 HSM，並且通過 FIPS 140-2 層級 3 驗證，適用於 FIPS 模式的叢集。對於使用案例超出 FIPS 140-2 第 3 級驗證限制的客戶， AWS CloudHSM 也提供非 FIP 模式的叢集。如需詳細資訊，請參閱AWS CloudHSM 叢集。

AWS 看不到 E2E 加密

由於您的資料層已加密 end-to-end (E2E)，而且 AWS 看不到，因此您可以控制自己的使用者管理 (IAM 角色之外)。與使用受管 AWS 服務相比，此控制項的折衷之處在於您的責任更大。

完全控制金鑰、演算法和應用程式開發

AWS CloudHSM 讓您完全控制您使用的演算法和金鑰。您可以產生、存放、匯出、匯入、管理和使用密碼編譯金鑰 (包括工作階段金鑰、字符金鑰、對稱金鑰和非對稱金鑰對)。此外， AWS CloudHSM SDK 可讓您完全控制應用程式開發、應用程式語言、執行緒，以及應用程式實際存在的位置。

將加密編譯工作負載遷移到雲端

客戶移轉使用公開金鑰加密標準 #11 (PKCS #11)、Java 密碼編譯延伸模組 (JCE)、密碼編譯 API：下一代 (CNG) 或金鑰儲存區提供者 (KSP) 的公開金鑰基礎結構，可移轉至其應用程式的變更較少。 AWS CloudHSM

若要深入瞭解您可以使用哪些功能 AWS CloudHSM，請參閱下列主題。當您準備好開始使用時 AWS CloudHSM，請參閱開始使用。

> **Note**
>
> 如果您需要受管服務，用於建立和控制您的加密金鑰，但您不想要或不需要操作自己的 HSM，請考慮使用 AWS Key Management Service。
> 如果您正在尋找一種彈性服務來管理雲端支付 HSM 和付款處理應用程式的金鑰，請考慮使用 AWS 付款加密技術。

目錄

AWS CloudHSM 使用案例

AWS CloudHSM 可以用來實現各種目標。本主題中的內容概述了您可以執行的操作 AWS CloudHSM。

達成法規合規

需要符合企業安全標準的企業可以用 AWS CloudHSM 來管理私鑰，以保護高度機密的資料。所提供的 HSM 通過 AWS CloudHSM FIPS 140-2 第 3 級認證，並且符合 PCI DSS 的規範。此外，符 AWS CloudHSM 合 PCI 密碼相容且符合 PCI-3DS 標準。如需詳細資訊，請參閱 合規。

加密和解密資料

用 AWS CloudHSM 於管理保護高度機密資料、傳輸中加密和靜態加密的私密金鑰。此外，還 AWS CloudHSM 提供與多個密碼編譯 SDK 的符合標準的整合。

使用私有金鑰和公有金鑰簽署和驗證文件

在密碼編譯中，使用私有金鑰簽署文件可讓收件者使用公有金鑰來驗證您 (而非其他人) 是否確實傳送文件。用 AWS CloudHSM 於建立專為此目的而設計的非對稱公開和私密金鑰配對。

使用 HMAC 和 CMAC 進行訊息驗證

在密碼編譯中，加密式訊息驗證程式碼 (CMAC) 和雜湊訊息驗證碼 (HMAC) 用於驗證並確保透過不安全網路傳送訊息的完整性。使用 AWS CloudHSM，您可以安全地建立和管理支援 HMAC 和 CMACS 的對稱金鑰。

充分利用 AWS CloudHSM 和 AWS Key Management Service

客戶可以在獲AWS KMS得 FIPS 140-2 Level 3 認證的單一租用戶環境中結合 AWS CloudHSM 並儲存關鍵資料，同時還可獲得的金鑰管理、擴展和雲端整合優勢。 AWS KMS如需如何執行此操作的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的 AWS CloudHSM 金鑰存放區。

卸載 Web 伺服器的 SSL/TLS 處理

為了透過網際網路安全地傳送資料，Web 伺服器會使用公有金鑰和私有金鑰對和 SSL/TLS 公有金鑰憑證來建立 HTTPS 工作階段。此過程涉及大量的 Web 服務器計算，但是您可以通過將其中的一些卸載到集群來減輕計算負擔，同時提供額外的安全性。 AWS CloudHSM 如需使 AWS CloudHSM用設定 SSL/TLS 卸載的相關資訊，請參閱。SSL/TLS 卸載

啟用透明資料加密 (TDE)

透明資料加密 (TDE) 可用來加密資料庫檔案。使用 TDE，資料庫軟體將資料存儲在磁盤上之前對其進行加密。您可以將 TDE 主加密金鑰儲存在 AWS CloudHSM叢集的 HSM 中，以達到更高的安全性。如需使用設定 Oracle TDE 的詳細資訊 AWS CloudHSM，請參閱Oracle 資料庫加密。

管理發行憑證授權機構 (CA) 的私有金鑰

憑證授權機構 (CA) 是一種受信任的實體，會發出將公有金鑰繫結至身分 (個人或組織) 的數位憑證。若要操作 CA，您必須保護簽署 CA 所發行憑證的私有金鑰，以維護信任。您可以將這類私密金鑰儲存在 AWS CloudHSM 叢集中，然後使用 HSM 執行加密簽署作業。

生成隨機數字

產生隨機數字以建立加密金鑰是線上安全的核心。 AWS CloudHSM 可用於在您控制的 HSM 中安全地產生隨機數，而且只有您可以看到。

如何 AWS CloudHSM 工作

本主題提供您用來在 HSM 中安全加密資料及執行密碼編譯作業的基本概念和架構的概觀。 AWS CloudHSM 在您自己的 Amazon Virtual Private Cloud (VPC) 中運行。在您可以使用之前 AWS CloudHSM，請先建立叢集、將 HSM 新增至叢集、建立使用者和金鑰，然後使用用戶端 SDK 將 HSM 與應用程式整合。完成此操作後，您可以使用用戶端 SDK 日誌 AWS CloudTrail、稽核日誌和 Amazon CloudWatch 進行監控 AWS CloudHSM。

了解基 AWS CloudHSM本概念以及它們如何協同合作以協助保護您的資料。

主題

- AWS CloudHSM 叢集
- HSM 使用者
- HSM 金鑰
- 用戶端 SDK
- AWS CloudHSM 叢集備份
- 區域

AWS CloudHSM 叢集

讓個別 HSM 以同步、備援且高可用性的方式協同運作可能很困難，但是 AWS CloudHSM 透過在叢集中提供硬體安全性模組 (HSM)，為您提供繁重的工作。叢集是 AWS CloudHSM 保持同步的個別 HSM 集合。當您在叢集的一個 HSM 上執行任務或操作時，該叢集的其他 HSM 會自動保持在最新狀態。

AWS CloudHSM 提供兩種模式的叢集：FIPS 和非 FIP。在 FIPS 模式中，只能使用聯邦資訊處理標準 (FIPS) 核准的金鑰和演算法。非 FIPS 模式提供支援的所有金鑰和演算法 AWS CloudHSM，無論 FIPS 核准為何。 AWS CloudHSM 也提供兩種類型的 HSM：h sm1. 中型和 hsm2m。如需每個 HSM 類型和叢集模式之間差異的詳細資訊，請參閱AWS CloudHSM 叢集模式和 HSM 類型。

為了滿足可用性、耐久性和可擴展性目標，您可以在多個可用區域中設定叢集中的 HSM 數量。您可以建立具有 1 到 28 個 HSM 的叢集 (預設限制為每個AWS 區域每個 AWS 帳戶 6 個 HSM)。您可以將 HSM 放置在區域中的不同可用區 AWS 域中。將更多 HSM 新增到叢集可提高效能。將叢集分散於可用區域可提供備援和高可用性。

如需叢集的詳細資訊，請參閱 管理 AWS CloudHSM 叢集。

若要建立叢集，請參閱開始使用。

HSM 使用者

與大多數 AWS 服務和資源不同，您不使用 AWS Identity and Access Management (IAM) 使用者或 IAM 政策來存取叢集內的資源。而是直接在叢集中的 AWS CloudHSM H SM 上使用 HSM 使用者。

HSM 使用者與 IAM 使用者不同。擁有正確憑證的 IAM 使用者可利用 AWS API 與資源進行互動來建立 HSM。由於 AWS 看不到 E2E 加密，因此您必須使用 HSM 使用者憑證來驗證 HSM 上的操作，因為憑證會直接在 HSM 上進行。HSM 會透過您定義和管理的憑證來驗證每個 HSM 使用者。每個 HSM 使用者都有類型，可決定使用者在 HSM 上執行的操作。每個 HSM 都會透過您使用 CloudHSM CLI 定義的憑證來驗證每個 HSM 使用者。

如果您使用的是舊版 SDK 系列，則將使用 CloudHSM 管理公用程式 (CMU)。

HSM 金鑰

AWS CloudHSM 可讓您在 AWS CloudHSM 叢集中的單一租戶 HSM 中安全地產生、存放和管理加密金鑰。金鑰可以是對稱或非對稱的，可以是單一工作階段的工作階段金鑰 (暫時金鑰)、長期使用的權杖金鑰 (持續性金鑰)，也可以是從 AWS CloudHSM 匯出和匯入。金鑰也可以用來完成常見的密碼編譯任務和函數：

- 使用對稱和非對稱加密演算法執行加密資料簽署和簽章驗證。
- 使用雜湊函數來計算訊息摘要和雜湊訊息驗證碼 (HMAC)。
- 包裝並保護其他金鑰。
- 訪問以密碼編譯方式保護的隨機資料。

叢集可以擁有的金鑰上限取決於叢集中的 HSM 類型。例如，hsm2m。介質存儲比 hsm1，媒體更多的密鑰。如需比較，請參閱AWS CloudHSM 配額。

此外， AWS CloudHSM 遵循金鑰使用和管理的一些基本原則：

許多金鑰類型和演算法可供選擇

> 為了允許您自定義自己的解決方案， AWS CloudHSM 提供了許多密鑰類型和算法可供選擇。算法支持多種密鑰大小。如需詳細資訊，請參閱每個 AWS CloudHSM 用戶端開發套件 的屬性和機制頁面。

如何管理金鑰

AWS CloudHSM 金鑰是透過 SDK 和命令列工具管理的。如需如何使用這些工具管理金鑰的詳細資訊，請參閱 管理金鑰 AWS CloudHSM 和 的最佳做法 AWS CloudHSM。

誰擁有金鑰

在中 AWS CloudHSM，建立金鑰的加密使用者 (CU) 擁有該金鑰。擁有者可以使用 key share 和 key unshare 命令來與其他 CU 共用和取消共用金鑰。如需詳細資訊，請參閱 使用 CloudHSM CLI 共用和取消共用金鑰。

存取和使用可以通過基於屬性的加密來控制

AWS CloudHSM 允許您使用基於屬性的加密，這是一種加密形式，可讓您使用金鑰屬性來控制誰可以根據策略解密資料。

用戶端 SDK

使用時 AWS CloudHSM，您可以使用用AWS CloudHSM 戶端軟體開發套件 (SDK) 執行密碼編譯作業。 AWS CloudHSM 用戶端 SDK 包括：

- 公有金鑰密碼編譯標準 #11 (PKCS #11)
- JCE 提供者
- OpenSSL 動態引擎
- 密碼編譯 API：適用於 Microsoft Windows 的下一代 (CNG) 和金鑰儲存提供者 (KSP)

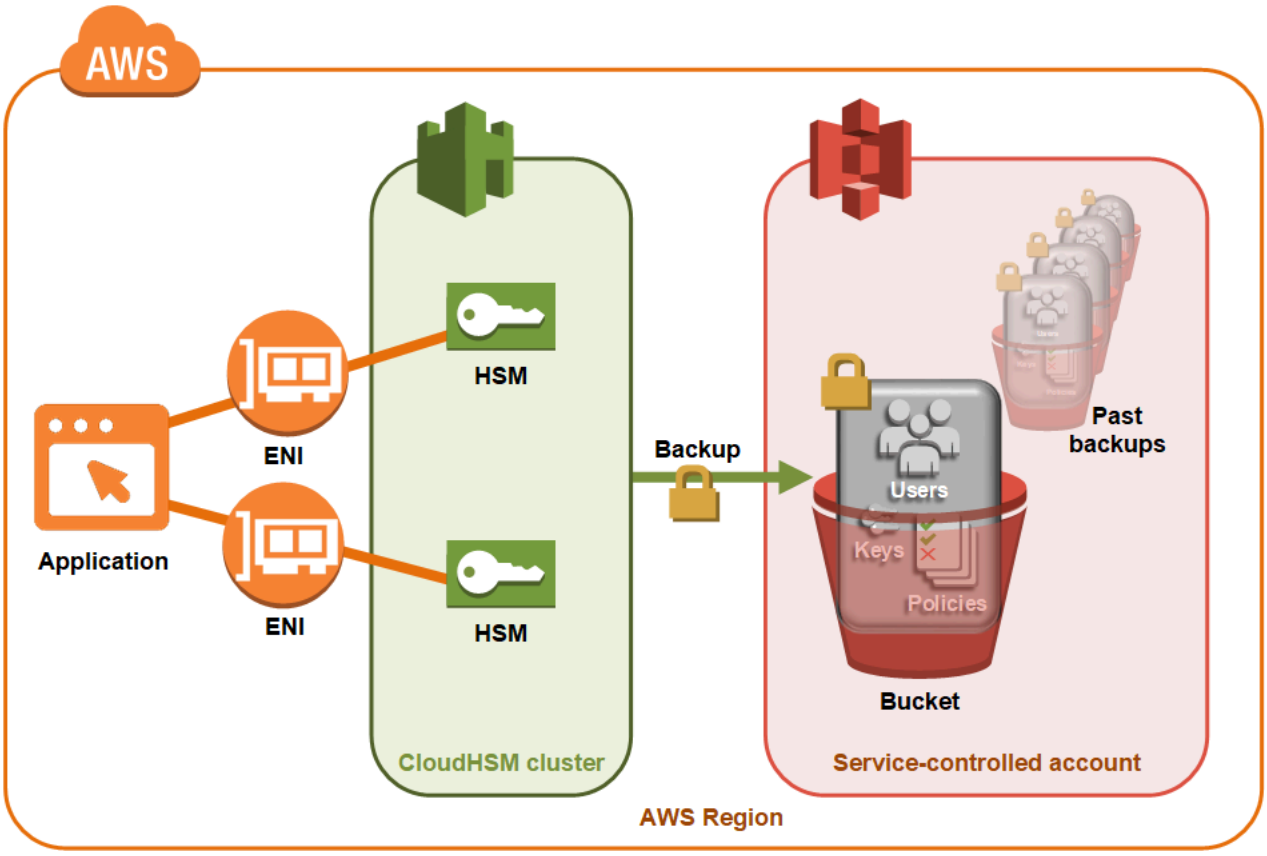
您可以在 AWS CloudHSM 叢集中使用任何或所有這些 SDK。撰寫您的應用程式碼，以使用這些 SDK 在 HSM 中執行密碼編譯作業。若要查看哪些平台和 HSM 類型支援每個 SDK，請參閱 用戶端 SDK 5 支援的平台

公用程式和命令列工具不僅需要使用 SDK，還需要設定應用程式的憑證、政策和設定。如需詳細資訊，請參閱 AWS CloudHSM 命令行工具。

如需關於安裝和使用客戶端 SDK 或用戶端連線安全性的詳細資訊，請參閱 用戶端 SDK 和 E nd-to-end 加密。

AWS CloudHSM 叢集備份

AWS CloudHSM 定期備份叢集中的使用者、金鑰和原則。備份既安全又耐用，並按可預測的排程進行更新。下圖顯示備份與叢集的關係。



如需使用備份的詳細資訊，請參閱 管理備份。

安全性

從 HSM AWS CloudHSM 進行備份時，HSM 會先加密其所有資料，然後再傳送到該資料。 AWS CloudHSM資料永遠不會以純文字格式離開 HSM。此外，無法解密備份， AWS 因為 AWS 無法存取用於解密備份的金鑰。如需更多資訊，請參閱叢集備份的安全性

耐久性

AWS CloudHSM 將備份存放在服務控制的 Amazon Simple Storage Service (Amazon S3) 儲存貯體中，與叢集位於相同的區域。備份具有 99.999999999% 的耐久性層級，與存放在 Amazon S3 中的任何物件相同。

區域

如需有關的支援區域的資訊 AWS CloudHSM，請參閱AWS 一般參考、或區域表中的區AWS CloudHSM域和端點。

AWS CloudHSM 指定區域中的所有可用區域可能無法使用。但是，這不會影響效能，因為叢集中所有 HSM 之間 AWS CloudHSM 自動進行負載平衡。

與大多數 AWS 資源一樣，叢集和 HSM 都是區域資源。您不能跨區域重複使用或擴展叢集。您必須執行開始使用 AWS CloudHSM中列出的所有必要步驟，才能在新區域中建立叢集。

基於災難復原的目的， AWS CloudHSM 可讓您將 AWS CloudHSM 叢集的備份從一個區域複製到另一個區域。如需更多詳細資訊，請參閱 AWS CloudHSM 叢集備份。

定價

使用 AWS CloudHSM，您可以按小時付款，無需長期合約或預付款。如需詳細資訊，請參閱 AWS 網站上的AWS CloudHSM 定價。

開始使用 AWS CloudHSM

下列主題可協助您建立、初始化及啟動 AWS CloudHSM 叢集。完成這些程序之後，您就能管理使用者、管理叢集，以及使用隨附的軟體程式庫執行密碼編譯操作。

目錄

建立 IAM 管理群組

作為最佳實踐，不要使用您與之 AWS 帳戶根使用者 進行交互 AWS，包括 AWS CloudHSM. 而是使用 AWS Identity and Access Management (IAM) 建立 IAM 使用者、IAM 角色或聯合身分使用者。請依照本節建立 IAM 使用者和管理員群組中的步驟建立系統管理員群組並將AdministratorAccess原則附加至該群組。然後建立新管理使用者，並將該使用者新增到群組。視需要將其他使用者新增至該群組。您新增的每個使用者都會繼承群組的AdministratorAccess原則。

另一個最佳作法是建立僅具有執行所需權限的 AWS CloudHSM 管理員群組 AWS CloudHSM。視需要將個別使用者新增至此群組。每位使用者都會繼承已連接至群組的有限許可，而不是完整的 AWS 存取權。下一客戶管理的政策 AWS CloudHSM節包含您應附加至 AWS CloudHSM 系統管理員群組的原則。

AWS CloudHSM 定義您 AWS 帳戶的服務連結角色。服務連結角色目前定義了允許您的帳戶記錄 AWS CloudHSM 事件的權限。您可以自動建立角色，也可 AWS CloudHSM 以由您手動建立。您無法編輯此角色，但可以刪除。如需詳細資訊，請參閱 服務連結角色 AWS CloudHSM。

建立 IAM 使用者和管理員群組

首先，建立 IAM 使用者以及該使用者的管理員群組。

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時登錄 https://aws.amazon.com/ 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。
2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

  如需使用 IAM 身分中心使用者登入的說明，請參閱使用AWS 登入 者指南中的登入 AWS 存取入口網站。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

如需可附加至 IAM 使用者群組的政策範例，請參閱的身分識別與存取管理 AWS CloudHSM。 AWS CloudHSM

建立虛擬私有雲端 (VPC)

如果沒有虛擬私有雲端 (VPC)，請按照本主題中的步驟建立一個。

> **Note**
>
> 按照這些步驟將建立公有和私有子網路。

建立 VPC

1. 前往 https://console.aws.amazon.com/vpc/ 開啟 Amazon VPC 主控台。
2. 在導覽列上，使用區AWS 域選擇器選擇目前支援的其 AWS CloudHSM中一個區域。
3. 選取建立 VPC 按鈕。
4. 針對 Resources to create (建立資源)，選擇 VPC and more (VPC 等)。
5. 針對名稱標籤自動產生，請輸入可識別的名稱，例如 **`CloudHSM`**。
6. 所有其他選項保持設定為預設值。
7. 選擇建立 VPC。
8. 建立 VPC 之後，選取檢視 VPC 以檢視您剛建立的 VPC。

建立叢集

叢集是個別 HSM 的集合。 AWS CloudHSM 同步化每個叢集中的 HSM，以便它們作為邏輯單元運作。 AWS CloudHSM 提供兩種類型的 HSM：h sm1. 中型和 hsm2m。建立叢集時，您可以選擇兩個叢集中的哪一個。如需每個 HSM 類型和叢集模式之間差異的詳細資訊，請參閱AWS CloudHSM 叢集模式和 HSM 類型。

建立叢集時，請代表您為叢集建 AWS CloudHSM 立安全性群組。此安全群組會控制對叢集中 HSM 的網路存取。該群組僅允許來自安全群組中 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體的傳入連接。在預設情況下，此安全群組不包含任何執行個體。您之後會啟動用戶端執行個體並設定叢集的安全群組，以允許與 HSM 進行通訊並連線。

> **Important**
>
> 建立叢集時， AWS CloudHSM 會建立名為 AWSServiceRoleForCloudHSM的服務連結角色。如果 AWS CloudHSM 無法建立角色或角色不存在，您可能無法建立叢集。如需詳細資訊，請參閱 解決叢集建立失敗的問題。如需服務連結角色的詳細資訊，請參閱服務連結角色 AWS CloudHSM。

您可以從 AWS CloudHSM 主控台、AWS Command Line Interface (AWS CLI) 或 AWS CloudHSM API 建立叢集。

> **Note**
>
> 如需叢集引數和 API 的詳細資訊，請參閱 AWS CLI 命令參考create-cluster中的。

建立叢集 (主控台)

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 在導覽列上，使用區AWS 域選擇器選擇目前支援的其 AWS CloudHSM中一個區域。
3. 選擇建立叢集。
4. 在叢集組態區段中，執行下列操作：
   a. 對於 VPC，請選取您在 建立虛擬私有雲端 (VPC) 建立的 VPC。
   b. 對於可用區域，請在每個可用區域旁選擇您建立的私有子網路。

      > **Note**
      >
      > 即使指定 AWS CloudHSM 的可用區域不受支援，效能也不會受到影響，因為叢集中所有 HSM 之間 AWS CloudHSM 會自動進行負載平衡。如需瞭解的可用區域支援 AWS 一般參考，請參閱中的區AWS CloudHSM 域和端點 AWS CloudHSM。

   c. 對於 HSM 類型，請選取可在叢集中建立的 HSM 類型，以及所需的叢集模式。若要查看每個區域支援哪些 HSM 類型，請參閱定AWS CloudHSM 價計算器。

      > **Important**
      >
      > 建立叢集之後，無法變更 HSM 類型和叢集模式。如需適合您使用案例之類型和模式的資訊，請參閱AWS CloudHSM 叢集模式和 HSM 類型。

   d. 針對叢集來源，指定是要建立新叢集還是從現有備份還原叢集。

      - 非 FIP 模式下的叢集備份只能用於還原處於非 FIP 模式的叢集。
      - FIPS 模式下的叢集備份只能用於還原處於 FIPS 模式的叢集。
5. 選擇下一步。
6. 指定服務應保留備份的時間長度。

> **Note**
>
> 接受預設保留期 90 天，或輸入介於 7 到 379 天之間的新值。服務會自動刪除此叢集中超過您在此指定值的備份。您之後可以變更這個設定。如需詳細資訊，請參閱 設定備份保留原則。

7. 選擇下一步。
8. (選用) 輸入標籤索引鍵和選用標籤值。若要將多個標籤新增至叢集，請選擇新增標籤。
9. 選擇檢閱。
10. 檢閱您的叢集組態，然後選擇建立叢集。

建立叢集 (AWS CLI)

- 在命令提示中，執行 create-cluster 命令。指定您計劃建立 HSM 所在子網路的 HSM 執行個體類型、備份保留時長和子網路 ID。使用您建立之私有子網路的子網路 ID。僅能對每個可用區域指定一個子網路。

```
$ 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`如果未指定，則預設為 FIPS 模式。若要建立非 FIPS 叢集，您必須包含以下參數：`--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
> ```

若要建立叢集 (AWS CloudHSM API)

- 傳送 CreateCluster 要求。指定您計劃建立 HSM 所在子網路的 HSM 執行個體類型、備份保留政策和子網路 ID。使用您建立之私有子網路的子網路 ID。僅能對每個可用區域指定一個子網路。

如果您嘗試建立叢集失敗，則可能與 AWS CloudHSM 服務連結角色的問題相關。如需解決失敗的協助，請參閱 解決叢集建立失敗的問題。

檢閱叢集安全群組

建立叢集時，請使用名稱建 AWS CloudHSM 立安全群組`cloudhsm-cluster-`*`clusterID`*`-sg`。此安全群組包含預先設定的 TCP 規則，其允許叢集安全群組內透過連接埠 2223-2225 的傳入和傳出通訊。此 SG 可讓您的 EC2 執行個體使用 VPC 與叢集中的 HSM 進行通訊。

> **Warning**
>
> - 請勿刪除或修改預先設定的 TCP 規則 (已填入叢集安全群組中)。這個規則可以防止連線問題和對 HSM 未經授權的存取。
> - 叢集安全群組可防止對 HSM 未經授權的存取。可以存取安全群組中執行個體的任何人都可以存取您的 HSM。大多數操作需要使用者登入 HSM，但可能會在未經驗證的情況下就將

HSM 歸零，而將金鑰資料、憑證和其他資料銷毀。如果發生這種情況，在最新的備份之後建立或修改的資料會遺失且無法復原。為了避免未經授權的存取，請確保只有受信任的管理員可以修改或存取預設安全群組中的執行個體。

在下一個步驟中，您可以啟動 Amazon EC2 執行個體，並將其連接叢集安全群組以將它連接到您的 HSM。

啟動 Amazon EC2 用戶端執行個體

若要與 AWS CloudHSM 叢集和 HSM 執行個體互動並管理，您必須能夠與 HSM 的彈性網路介面進行通訊。最簡單的方式是使用與您的叢集位於相同 VPC 的 EC2 執行個體。您也可以使用下列 AWS 資源來連線至叢集：

- Amazon VPC 對等互連
- AWS Direct Connect
- VPN 連接

> **Note**
>
> 本指南提供如何將 EC2 執行個體連接到 AWS CloudHSM 叢集的簡化範例。如需有關安全網路組態的最佳作法，請參閱安全存取您的叢集。

AWS CloudHSM 文件通常假設您在建立叢集的相同 VPC 和可用區域 (AZ) 中使用 EC2 執行個體。

建立 EC2 執行個體

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 EC2 儀表板。
2. 選取啟動執行個體。從下拉式選單中，選擇啟動執行個體。
3. 在名稱欄位中，輸入 EC2 執行個體的名稱。
4. 在應用程式和 OS 影像 (Amazon Machine Image) 區段，選擇與 CloudHSM 支援平台相對應的 Amazon Machine Image (AMI)。如需詳細資訊，請參閱 用戶端 SDK 5 支援的平台。
5. 在執行個體類型區段中，選取執行個體類型。
6. 在金鑰對區段中，使用現有金鑰對，或選取建立新金鑰配對並完成下列步驟：

a. 在金鑰對名稱中，輸入新金鑰對的名稱。
b. 在金鑰對類型中，選擇一個金鑰對類型。
c. 在私有金鑰檔案格式中，選擇私有金鑰的儲存格式。
d. 選取建立金鑰對。
e. 下載並存儲私有金鑰檔案。

> **Important**
>
> 這是您儲存私有金鑰檔案的唯一機會。將檔案下載並儲存於安全處。當您啟動執行個體時，必須提供金鑰對的名稱。此外，每次連接至執行個體，您都必須提供對應的私有金鑰，同時要選擇您在設定時建立的金鑰對。

7. 在網路設定中選取編輯。
8. 在 VPC 中，選擇先前為叢集建立的 VPC。
9. 在子網路中，選擇先前為 VPC 建立的公有子網路。
10. 在 Auto-assign Public IP (自動指派公有 IP) 中，選擇 Enable (啟用)。
11. 選擇 Select an existing security group (選取現有的安全群組)。
12. 在一般安全群組中，從下拉式選單中選取預設安全群組。
13. 在設定儲存中，使用下拉式功能表選擇儲存組態。
14. 在彙總視窗中，選取啟動執行個體。

> **Note**
>
> 完成此步驟後，將開始建立 EC2 執行個體的執行程序。

如需建立 Linux Amazon EC2 用戶端的詳細資訊，請參閱 Amazon EC2 Linux 執行個體入門。如需連接到執行中用戶端的詳細資訊，請參閱下列主題：

- 使用 SSH 連接至您的 Linux 執行個體
- 使用 PuTTY 從 Windows 連接至您的 Linux 執行個體

Amazon EC2 使用者指南包含設定和使用 Amazon EC2 執行個體的詳細說明。以下清單提供適用於 Linux 和 Windows Amazon EC2 用戶端的可用文件概觀：

- 若要建立 Linux Amazon EC2 用戶端，請參閱 Amazon EC2 Linux 執行個體入門。

  如需連接到執行中用戶端的詳細資訊，請參閱下列主題：

  - 使用 SSH 連接至您的 Linux 執行個體
  - 使用 PuTTY 從 Windows 連接至您的 Linux 執行個體
- 若要建立 Windows Amazon EC2 用戶端，請參閱 Amazon EC2 Windows 執行個體入門。如需連接到您的 Windows 用戶端的詳細資訊，請參閱 連接到您的 Windows 執行個體。

> **Note**
>
> 您的 EC2 執行個體可以執行本指南中包含的所有 AWS CLI 命令。如果未安裝 AWS CLI ，您可以從 AWS Command Line Interface 下載。如果您使用 Windows，可以下載及執行 64 位元或 32 位元 Windows 安裝程式。如果您使用 Linux 或 macOS，可以使用 pip 安裝 CLI。

設定用戶端 Amazon EC2 執行個體安全群組

當您啟動 Amazon EC2 執行個體時，可將其與預設的 Amazon VPC 安全群組建立關聯。本主題說明如何將叢集安全群組與 EC2 執行個體建立關聯。此關聯可讓 EC2 執行個體上執行的 AWS CloudHSM 用戶端與您的 HSM 通訊。若要將 EC2 執行個體連線到 AWS CloudHSM 叢集，您必須正確設定 VPC 預設安全群組，並將叢集安全群組與執行個體建立關聯。

修改預設安全群組

您需要修改預設安全群組，允許 SSH 或 RDP 連線，以便下載及安裝用戶端軟體，並與您的 HSM 互動。

修改預設安全群組

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 EC2 儀表板。
2. 選取執行個體 (執行中)，然後選取要安裝 AWS CloudHSM 用戶端之 EC2 執行個體旁邊的核取方塊。
3. 在安全標籤下方，選擇名稱為預設的安全群組。
4. 在頁面頂端，依序選擇 Actions (動作) 和 Edit Inbound Rules (編輯傳入規則)。
5. 選取 Add Rule (新增規則)。
6. 針對 Type (類型)，執行下列其中一項操作：

- 若是 Windows Server Amazon EC2 執行個體，請選擇 RDP。即會自動填入連接埠 3389。
- 若是 Linux EC2 執行個體，請選擇 SSH。即會自動填入連接埠範圍 22。

7. 對於任意一個選項，請將來源設定為我的 IP，以便與 Amazon EC2 執行個體進行通訊。

> **Important**
>
> 請不要將 0.0.0.0/0 指定為 CIDR 範圍，避免其他人存取您的執行個體。

8. 選擇儲存。

將亞馬遜 EC2 執行個體 Connect 到 AWS CloudHSM 叢集

您必須將叢集安全群組連接到 EC2 執行個體，以便 EC2 執行個體可以與叢集中的 HSM 通訊。叢集安全群組包含預先設定的規則，其允許透過連接埠 2223-2225 的傳入通訊。

若要將 EC2 執行個體連線到 AWS CloudHSM 叢集

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 EC2 儀表板。
2. 選取執行個體 (執行中)，然後選取要安裝 AWS CloudHSM 用戶端之 EC2 執行個體的核取方塊。
3. 在頁面頂端，依序選擇動作、聯網和變更安全群組。
4. 選取群組名稱符合您叢集 ID 的安全群組，例如 `cloudhsm-cluster-`*`clusterID`*`-sg`。
5. 選擇新增安全群組。
6. 選取 Save (儲存)。

> **Note**
>
> 您最多可以指派 5 個安全群組給 Amazon EC2 執行個體。如果您已達到上限，則必須修改 Amazon EC2 執行個體的預設安全群組和叢集安全群組：
>
> 在預設安全群組中，執行下列動作：
>
> - 新增傳入規則，允許使用 TCP 通訊協定、來自叢集安全群組，且透過連接埠 2223-2225 的流量。
>
> 在叢集安全群組中，執行下列動作：

- 新增傳入規則，允許使用 TCP 通訊協定、來自預設安全群組，且透過連接埠 2223-2225 的流量。

建立 HSM

建立叢集後，您可以建立 HSM。不過，叢集必須在未初始化狀態，您才能在叢集建立 HSM。若要判斷叢集的狀態，請在AWS CloudHSM 主控台中檢視叢集頁面、使用執行describe-clusters命令，或在 AWS CloudHSM API 中傳送DescribeClusters要求。 AWS CLI 您可以從 AWS CloudHSM 主控台、AWS CLI 或 AWS CloudHSM API 建立 HSM。

建立 HSM (主控台)

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 在您要為其建立 HSM 的叢集 ID 旁邊，選取選項按鈕。
3. 選取動作。從下拉式功能表中，選擇初始化。
4. 針對您要建立的 HSM 選擇可用區域 (AZ)。
5. 選取建立。

建立 HSM (AWS CLI)

- 在命令提示中，執行 create-hsm 命令。指定您先前建立之叢集的叢集 ID，以及 HSM 的可用區域。指定可用區域，格式為 us-west-2a、us-west-2b 等。

```
$ 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"
    }
```

```
}
```

若要建立 HSM (AWS CloudHSM API)

- 傳送 CreateHsm 要求。指定您先前建立之叢集的叢集 ID，以及 HSM 的可用區域。

在建立叢集和 HSM 之後，您可以選擇驗證 HSM 的身分，或直接開始初始化叢集。

驗證叢集 HSM 的身分和真偽 (選用)

若要初始化您的叢集，請簽署叢集的第一個 HSM 所產生的憑證簽署請求 (CSR)。在這樣做之前，您可能需要驗證 HSM 的身分和真偽。

> **Note**
>
> 此為選用程序。然而只在叢集初始化之前適用。叢集初始化之後，您就無法使用此程序來取得憑證或驗證 HSM。

主題

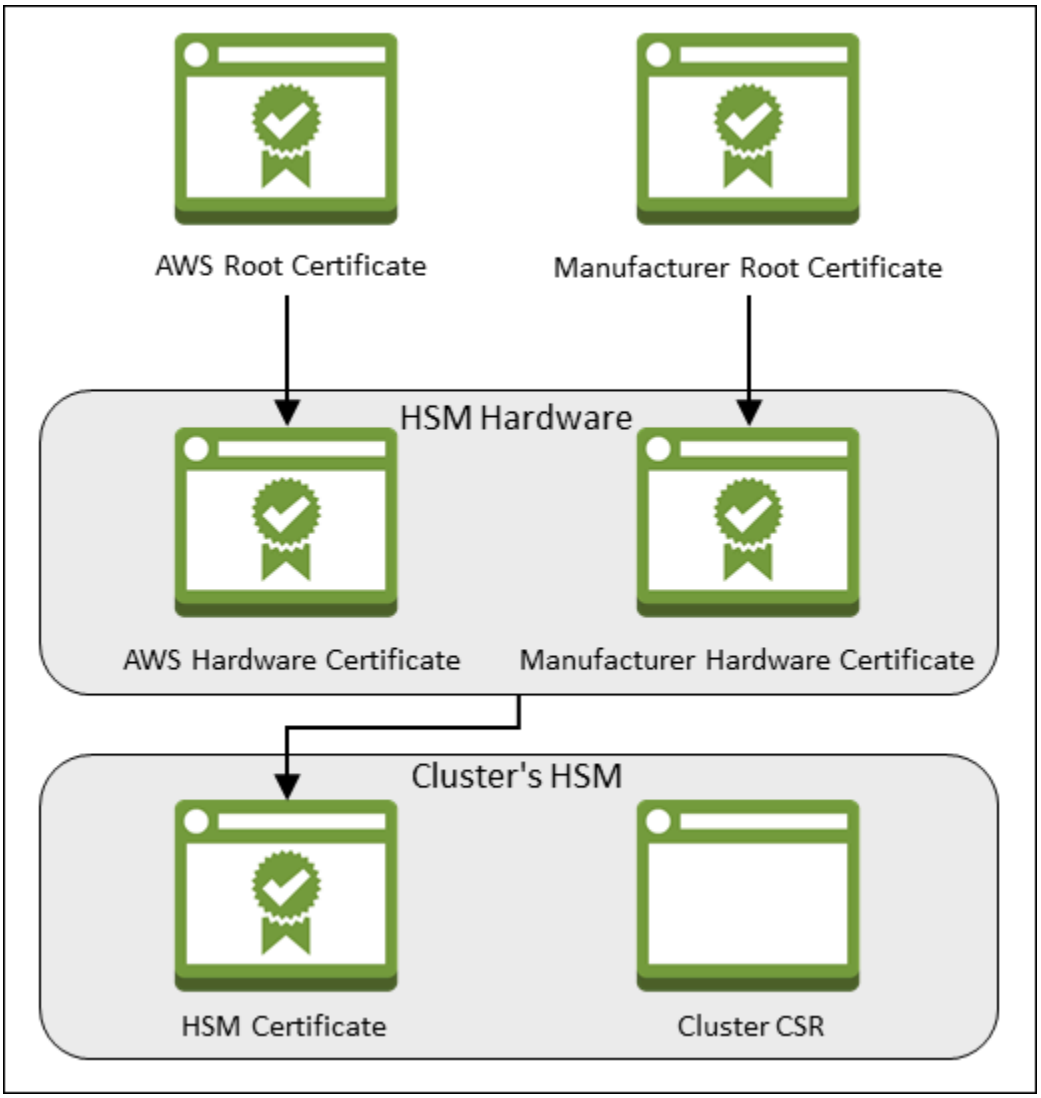
概觀

若要驗證叢集第一個 HSM 的身分，請完成以下步驟：

1. 取得憑證和 CSR：在此步驟中，您從 HSM 取得三個憑證和一個 CSR。您也會取得兩個根憑證，一個來自 HSM 硬體製造商 AWS CloudHSM ，另一個來自 HSM 硬體製造商。
2. 驗證憑證鏈結 — 在此步驟中，您會建構兩個憑證鏈結，一個連結至 AWS CloudHSM 根憑證，另一個是製造商根憑證。然後，您可以使用這些憑證鏈結來驗證 HSM 憑證， AWS CloudHSM 以判斷硬體製造商同時證明 HSM 的身分和真實性。

3. 比較公有金鑰：在此步驟中，您擷取並比較 HSM 憑證和叢集 CSR 中的公有金鑰，以確保金鑰相同。這樣可讓您確信 CSR 是由真實可信任的 HSM 所產生。

下圖顯示 CSR、憑證及其相互關係。後續清單定義每個憑證。



AWS 根憑證

這 AWS CloudHSM是根憑證。

製造商根憑證

這是硬體製造商的根憑證。

AWS 硬體憑證

AWS CloudHSM 將 HSM 硬體新增至叢集時建立此憑證。此憑證會宣告 AWS CloudHSM 擁有硬體。

製造商硬體憑證

HSM 硬體製造商在製造 HSM 硬體時建立此憑證。此憑證聲明製造商建立此硬體。

HSM 憑證

當您在叢集中建立第一個 HSM 時，通過 FIPS 驗證的硬體會產生 HSM 憑證。此憑證聲明 HSM 硬體建立此 HSM。

叢集 CSR

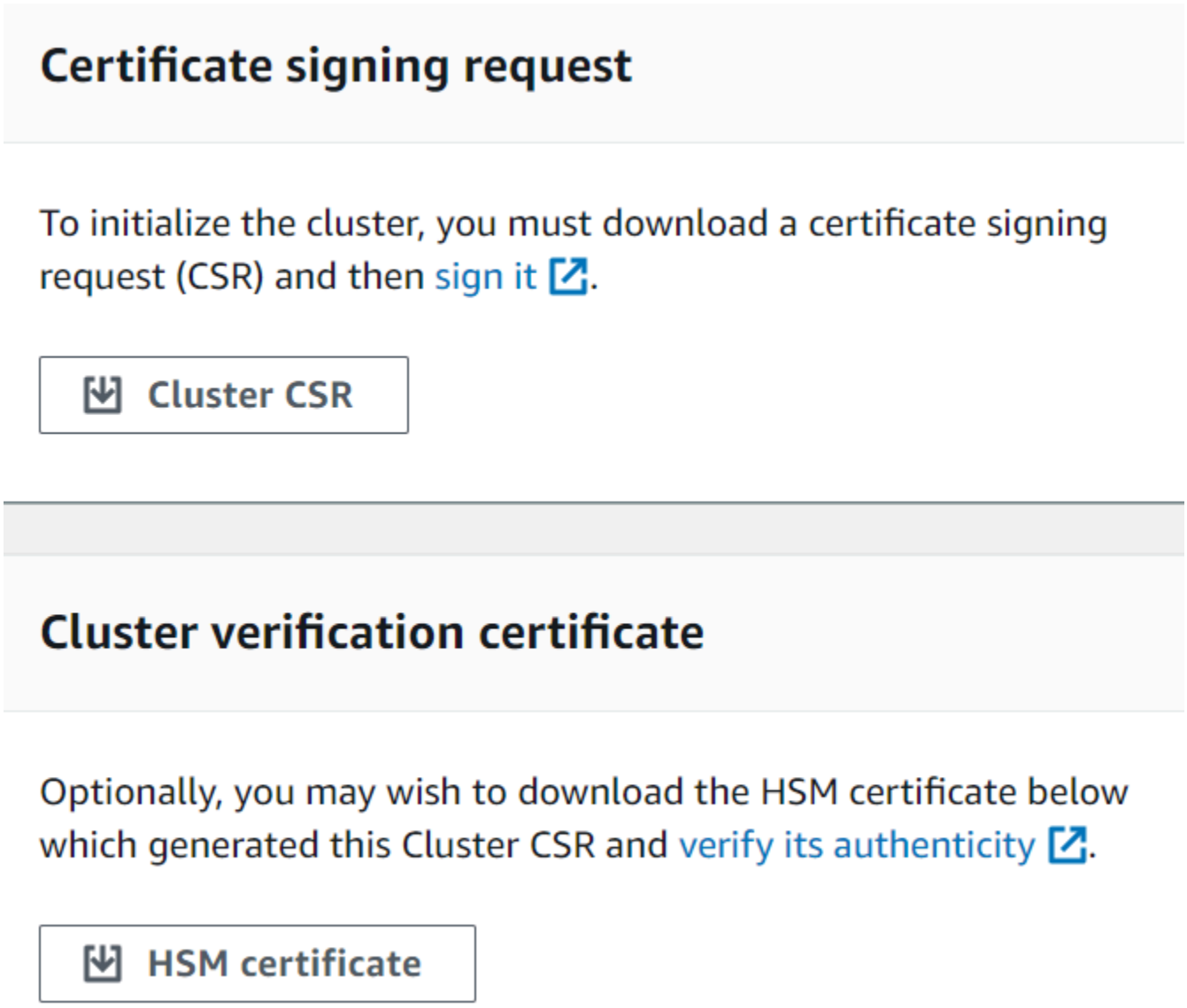
第一個 HSM 建立叢集 CSR。當您簽署叢集 CSR 時，您即宣稱擁有叢集。然後，您可以使用已簽署的 CSR 來初始化叢集。

從 HSM 取得憑證

若要驗證 HSM 的身分和真偽，請先取得一個 CSR 和 5 個 HSM。您可以從 HSM 取得三個憑證，您可以使用AWS CloudHSM 主控台、AWS Command Line Interface (AWS CLI) 或 AWS CloudHSM API 執行這些憑證。

取得 CSR 和 HSM 憑證 (主控台)

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 選取您要驗證的 HSM 叢集 ID 旁的選項按鈕。
3. 選取動作。從下拉式功能表中，選擇初始化。
4. 如果您未完成上一個步驟來建立 HSM，請為您正在建立的 HSM 選擇一個可用區域 (AZ)。然後選取建立。
5. 當憑證和 CSR 準備好時，您會看到下載連結。



6. 選擇每個連結來下載和儲存 CSR 和憑證。為了簡化後續步驟，請將所有檔案儲存到相同目錄，並使用預設的檔案名稱。

取得 CSR 和 HSM 憑證 (AWS CLI)

- 在命令提示字元中，執行 describe-clusters 命令四次，每次都擷取 CSR 和不同的憑證，並將其儲存到檔案。

  a. 發出下列命令來擷取叢集 CSR。將 *<cluster ID>* 取代為您先前建立的叢集 ID。

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

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

b. 發出下列命令來擷取 HSM 憑證。將 *<cluster ID>* 取代為您先前建立的叢集 ID。

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

c. 發出以下命令以解壓縮 AWS 硬體憑證。將 *<cluster ID>* 取代為您先前建立的叢集 ID。

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

d. 發出下列命令來擷取製造商硬體憑證。將 *<cluster ID>* 取代為您先前建立的叢集 ID。

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

若要取得企業社會責任與 HSM 憑證 (AWS CloudHSM API)

- 傳送 DescribeClusters 請求，然後從回應中擷取並儲存 CSR 和憑證。

取得根憑證

請依照下列步驟取得 AWS CloudHSM 和製造商的根憑證。將根憑證檔案儲存到包含 CSR 和 HSM 憑證檔案的目錄。

取得 AWS CloudHSM 和製造商根憑證

1. 下載 AWS CloudHSM 根憑證：AWS_CloudHSM_Root-G1.zip

2. 為您的 HSM 類型下載正確的製造商根憑證：

   - 中型製造商根證書:liquid_security_certificate.zip
   - 中型製造商根憑證:liquid_security_certificate.zip

   > **Note**
   >
   > 若要從其登陸頁面下載每個憑證，請使用下列連結：
   >
   > - hsm1.medium 製造商根憑證的登陸頁面
   > - hsm2m.medium 製造商根憑證的登陸頁面
   >
   > 您可能需要用滑鼠右鍵按一下 Download Certificate (下載憑證) 連結，然後選擇 Save Link As...(另存連結為...) 以儲存憑證檔案。

3. 下載檔案之後，將其內容解壓縮 (unzip)。

驗證憑證鏈

在此步驟中，您會建構兩個憑證鏈結，一個是 AWS CloudHSM 根憑證，另一個是製造商根憑證。然後，使用 OpenSSL 根據每個憑證鏈來驗證 HSM 憑證。

若要建立憑證鏈，請開啟 Linux shell。您需要 OpenSSL (大多數的 Linux shell 中都有)，還需要有您下載的根憑證和 HSM 憑證檔案。但是，您不需要執 AWS CLI 行此步驟，並且 shell 不需要與您的 AWS 帳戶關聯。

使用 AWS CloudHSM 根憑證驗證 HSM 憑證

1. 導覽至您下載的根憑證和 HSM 憑證檔案的儲存目錄。以下命令假設所有憑證都在目前的目錄中，且使用預設的檔案名稱。

   使用下列命令建立憑證鏈結，其中包含 AWS 硬體憑證和 AWS CloudHSM 根憑證 (依該順序)。將 *<cluster ID>* 取代為您先前建立的叢集 ID。

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

2. 使用下列 OpenSSL 命令，根據 AWS 憑證鏈來驗證 HSM 憑證。將 *<cluster ID>* 取代為您先前建立的叢集 ID。

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

根據製造商根憑證來驗證 HSM 憑證

1. 使用下列命令來建立憑證鏈，其中依序包含製造商硬體憑證和製造商根憑證。將 *<cluster ID>* 取代為您先前建立的叢集 ID。

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

2. 使用以下 OpenSSL 命令，根據製造商憑證鏈來驗證 HSM 憑證。將 *<cluster ID>* 取代為您先前建立的叢集 ID。

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

擷取和比較公有金鑰

使用 OpenSSL 來擷取並比較 HSM 憑證和叢集 CSR 中的公有金鑰，以確保金鑰相同。

若要比較公有金鑰，請使用 Linux shell。您需要 OpenSSL，它在大多數 Linux 命令介面中都可以使用，但您不需要執 AWS CLI 行此步驟。殼層不需要與您的 AWS 帳戶建立關聯。

擷取和比較公有金鑰

1. 使用下列命令從 HSM 憑證中擷取公有金鑰。

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

2. 使用下列命令從叢集 CSR 中擷取公有金鑰。

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

3. 使用下列命令來比較公有金鑰。如果公有金鑰相同，以下命令不會產生任何輸出。

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

驗證 HSM 的身分和真偽之後，請繼續初始化叢集。

初始化叢集

完成下列主題中的步驟，以初始化 AWS CloudHSM 叢集。

> **Note**
>
> 初始化叢集之前，請檢閱您可以用來驗證 HSM 的身分和真實性的程序。此程序為選擇性，且只能在叢集初始化前使用。將叢集初始化後，您無法使用此程序來取得您的憑證或驗證 HSM。

主題

- 取得叢集 CSR
- 簽署 CSR
- 初始化叢集

取得叢集 CSR

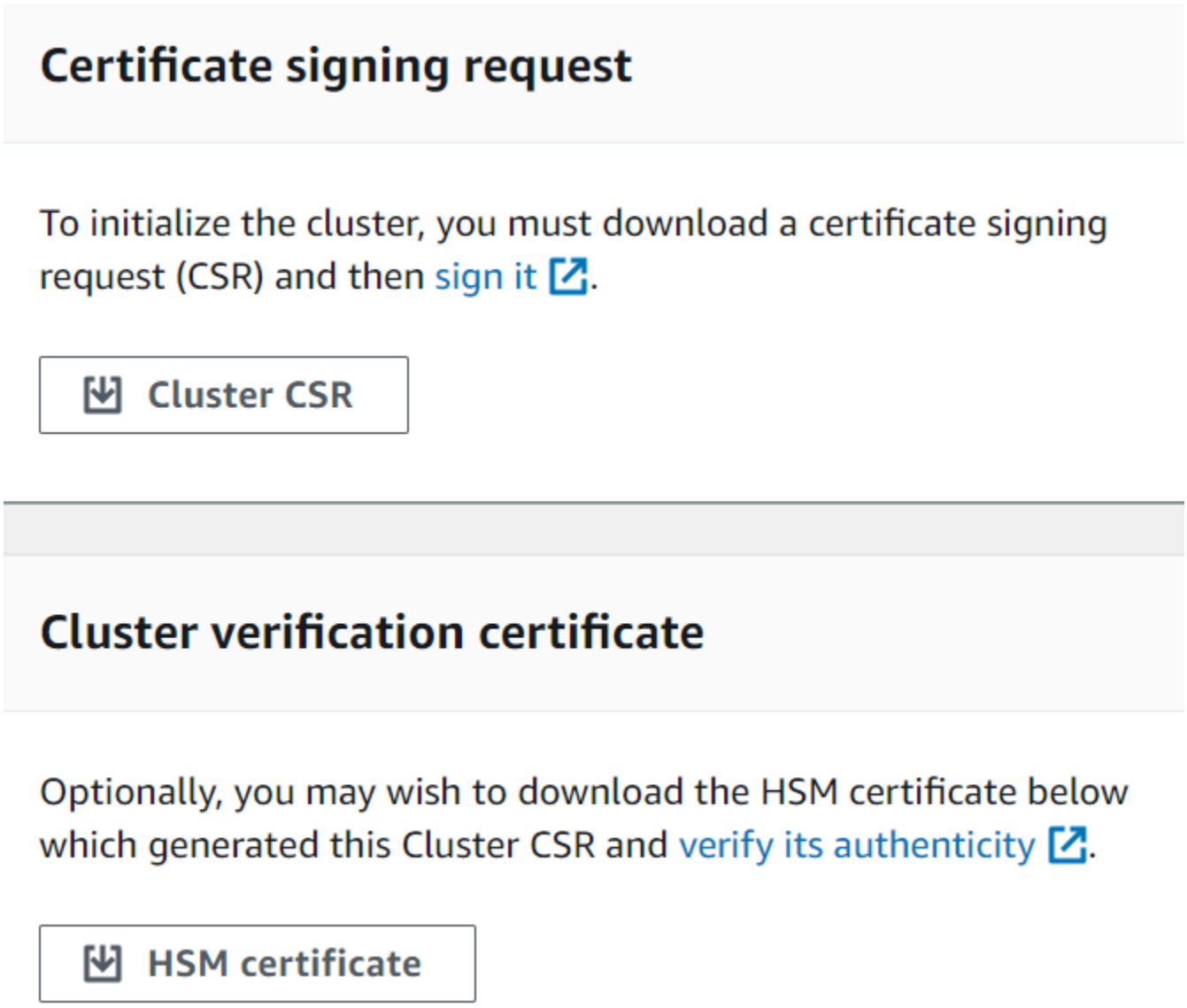
初始化叢集之前，您必須下載並簽署憑證簽署請求 (CSR)，CSR 是由叢集的第一個 HSM 產生。如果您之前已依照驗證叢集的 HSM 身分中的步驟操作，則您已有 CSR，可以直接簽署。否則，請立即使用主AWS CloudHSM 控台、AWS Command Line Interface (AWS CLI) 或 AWS CloudHSM API 取得 CSR。

> **Important**
>
> 如要初始化叢集，您的信任錨點必須符合 RFC 5280 並符合下列需求：
>
> - 如使用 X509v3 延伸功能，則必須存在 X509v3 基本限制條件延伸。

- 信任錨點必須是自我簽署的憑證。
- 延伸值不得相互衝突。

取得 CSR (主控台)

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 選取您要驗證的 HSM 叢集 ID 旁的選項按鈕。
3. 選取動作。從下拉式功能表中，選擇初始化。
4. 如果您未完成上一個步驟來建立 HSM，請為您正在建立的 HSM 選擇一個可用區域 (AZ)。然後選取建立。
5. 當 CSR 準備好，您會看到可下載它的連結。



6. 選擇 Cluster CSR (叢集 CSR) 以下載並儲存 CSR。

取得 CSR (AWS CLI)

- 在命令提示字元，執行下列 describe-clusters 命令，其會擷取 CSR 並將它儲存到檔案。將 *<cluster ID>* 取代為先前建立的叢集 ID。

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

若要取得企業社會責任 (AWS CloudHSM API)

1. 傳送 DescribeClusters 要求。
2. 擷取並儲存來自回應的 CSR。

簽署 CSR

現在，您必須建立自我簽署的簽署憑證，並使用它來簽署您的叢集 CSR。您不需要執 AWS CLI 行此步驟，而且殼層不需要與您的 AWS 帳戶建立關聯。若要簽署 CSR，您必須執行以下操作：

1. 完成上一節 (請參閱 取得叢集 CSR)。
2. 建立私有金鑰。
3. 使用私有金鑰來建立簽署憑證。
4. 簽署您的叢集 CSR。

建立私有金鑰。

> **Note**
>
> 針對生產叢集，應使用信任的隨機來源以安全的方式建立您想建立的金鑰。我們建議您使用安全的離站和離線 HSM 或其他同等項目。安全地存放金鑰。金鑰會建立叢集的身分識別，以及您對叢集所包含的 HSM 的唯一控制權。
> 在開發和測試時，您可以使用任何方便的工具 (如 OpenSSL) 來建立和簽署叢集憑證。下列範例說明如何建立金鑰。在使用該金鑰建立自我簽署憑證 (請見以下) 後，您應該妥善存放它。若要登入 AWS CloudHSM 執行個體，憑證必須存在，但私密金鑰不存在。

使用下列命令來建立私有金鑰。初始化 AWS CloudHSM 叢集時，您必須使用 RSA 2048 憑證或 RSA 4096 憑證。

```
$ 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:
```

使用私有金鑰來建立自我簽署憑證。

您用來建立生產叢集私有金鑰的信任硬體，也應該提供軟體工具來產生使用該金鑰的自我簽署憑證。以下範例使用 OpenSSL 和您在上個步驟中建立的私有金鑰來建立簽署憑證。憑證有效期限是 10 年 (3652 天)。請閱讀畫面上的指示，並依照提示操作。

```
$ 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 []:
```

此命令會建立名為 customerCA.crt 的憑證檔案。將此憑證放在要連線至 AWS CloudHSM 叢集的每部主機上。如果您替檔案取了其他名稱，或將其存放在主機根目錄以外的路徑，請據此編輯您的用戶端組態檔案。使用您剛建立的憑證和私有金鑰來簽署後續步驟中的叢集憑證簽署請求 (CSR)。

簽署叢集 CSR

您用來建立生產叢集私有金鑰的信任硬體，也應該提供工具來簽署使用該金鑰的 CSR。以下範例會使用 OpenSSL 來簽署叢集的 CSR。範例會使用您的私有金鑰和以及在先前步驟中建立的自我簽署憑證。

```
$ 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:
```

此命令會建立名為 *<cluster ID>*_CustomerHsmCertificate.crt 的檔案。初始化叢集時，使用這個檔案做為已簽署的憑證。

初始化叢集

使用已簽署的 HSM 憑證和簽署憑證來初始化您的叢集。您可以使用AWS CloudHSM 主控台AWS CLI、或 AWS CloudHSM API。

初始化叢集 (主控台)

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 選取您要驗證的 HSM 叢集 ID 旁的選項按鈕。
3. 選取動作。從下拉式功能表中，選擇初始化。
4. 如果您未完成上一個步驟來建立 HSM，請為您正在建立的 HSM 選擇一個可用區域 (AZ)。然後選取建立。
5. 在 Download certificate signing request (下載憑證簽署請求) 頁面上，選擇 Next (下一步)。如果無法使用 Next (下一步)，先選擇其中一個 CSR 或憑證連結。然後選擇下一步。
6. 在 Sign certificate signing request (簽署憑證簽署請求) 頁面上，選擇 Next (下一步)。
7. 在 Upload the certificates (上傳憑證) 頁面上，執行下列動作：

    a. 選擇 Cluster certificate (叢集憑證) 旁的 Upload file (上傳檔案)。然後，尋找並選取您之前簽署的 HSM 憑證。如果您已完成先前區段的步驟，請選取 *<cluster ID>*_CustomerHsmCertificate.crt 檔案。

    b. 選擇 Issuing certificate (發行憑證) 旁的 Upload file (上傳檔案)。然後選擇您的簽署憑證。如果您已完成先前區段的步驟，請選取 customerCA.crt 檔案。

    c. 選擇 Upload and initialize (上傳並初始化)。

初始化叢集 (AWS CLI)

- 在命令提示中，執行 initialize-cluster 命令。提供下列資訊：
    - 您先前建立之叢集的 ID。

- 您先前簽署的 HSM 憑證。如果您已完成先前區段的步驟，它會儲存在名為 *<cluster ID>*_CustomerHsmCertificate.crt 的檔案中。
- 您的簽署憑證。如果您已完成先前區段的步驟，簽署憑證會儲存在名為 customerCA.crt 的檔案中。

```
$ 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."
}
```

若要初始化叢集 (AWS CloudHSM API)

- 使用下列項目傳送 InitializeCluster 請求：
  - 您先前建立之叢集的 ID。
  - 您先前簽署的 HSM 憑證。
  - 您的簽署憑證。

下載和設定 CloudHSM CLI

若要與 AWS CloudHSM 叢集中的 HSM 互動，您需要 CloudHSM CLI。

任務

- 安裝命 AWS CloudHSM 令行工具

安裝命 AWS CloudHSM 令行工具

Connect 線至用戶端執行個體並執行下列命令，以下載並安裝 AWS CloudHSM 命令列工具。如需詳細資訊，請參閱 啟動 Amazon EC2 用戶端執行個體。

使用以下命令來下載和安裝 CloudHSM CLI。

Amazon Linux 2

Amazon Linux 2 (x86_64 架構)：

```
$ 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
```

Amazon Linux 2 (ARM64 架構)：

```
$ 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

Amazon 架構

```
$ 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
```

在 ARM64 架構上的 Amazon

```
$ 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+)

CentOS 7 (x86_64 架構)：

```
$ 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+)

RHEL 7 (x86_64 架構)：

```
$ 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+)

RHEL 8 (x86_64 架構)：

```
$ 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+)

在 64 架構上的 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
```

關於 ARM64 架構的第 9 步：

```
$ 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

Ubuntu 20.04 LTS (x86_64 架構)：

```
$ 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

Ubuntu 22.04 LTS (x86_64 架構)：

```
$ 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
```

關於 ARM64 架構的 Ubuntu 22.04 研究所：

```
$ 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

對於 x86_64 架構上的視窗伺服器 2016，請 PowerShell 以系統管理員身分開啟，然後執行下列命令：

```
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

對於 x86_64 架構上的 Windows 伺服器 2019，請 PowerShell 以系統管理員身分開啟並執行下列命令：

```
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
```

使用下列命令來設定 CloudHSM CLI。

為用戶端 SDK 5 引導 Linux EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 地址。

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

為用戶端 SDK 5 引導 Windows EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 地址。

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

啟用叢集

當您啟動叢集時， AWS CloudHSM 叢集的狀態會從初始化變更為使用中。然後，您就可以管理硬體安全模組 (HSM) 的使用者和使用 HSM。

Important

在啟動叢集之前，您必須先將簽發的憑證複製到連線至叢集的每個 EC2 執行個體平台的預設位置 (您可以在初始化叢集時建立發行憑證)。

Linux

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

Windows

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

放置簽發的憑證後，請安裝 CloudHSM CLI，並在您的第一個 HSM 上執行 cluster activate 命令。您會注意到叢集中第一個 HSM 上的管理員帳戶具有未啟用的管理員角色。這是僅在叢集啟動之前暫時存在的角色。當您啟動叢集時，未啟用的管理員角色會變更為管理員。

啟用叢集

1. 連接到您先前啟動的用戶端執行個體。如需詳細資訊，請參閱 啟動 Amazon EC2 用戶端執行個體。您可以啟動 Linux 執行個體或 Windows Server。
2. 以互動式模式執行 CloudHSM CLI。

   Linux

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

   Windows

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

3. (選用) 使用 user list 命令來顯示現有的使用者。

   ```
   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. 使用 cluster activate 命令設定初始管理員密碼。

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

建議您將新密碼寫在密碼工作表。不要遺失此工作表。建議您列印一份密碼工作表，用來記錄重要的 HSM 密碼，並存放於安全之處。另外也建議您將一份這個工作表存放於安全的離站儲存區。

5. (選用) 使用 user list 命令，確認使用者的類型已變更為管理員/CO。

```
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. 使用 quit 命令來停止 CloudHSM CLI 工具。

```
aws-cloudhsm > quit
```

如需使用 CloudHSM CLI 或 CMU 的詳細資訊，請參閱了解 HSM 使用者和了解使用 CMU 的 HSM 使用者管理。

使用新憑證和私有金鑰重新設定 SSL (選用)

AWS CloudHSM 使用 SSL 憑證建立與 HSM 的連線。安裝用戶端時會包含預設金鑰和 SSL 憑證。不過，您可以建立和使用自己的金鑰和 SSL 憑證。請注意，您將需要在初始化叢集時所建立的自簽憑證 (*customerCA.crt*)。

在高層級，這個步驟分為兩個步驟：

1. 首先，您可以建立一個私有金鑰，然後用該金鑰建立憑證簽署要求 (CSR)。使用您在初始化叢集時建立的發行憑證來簽署 CSR。
2. 接下來，您可以使用設定工具將金鑰和憑證複製到適當的目錄。

建立一個金鑰和 CSR，然後簽署 CSR

用戶端 SDK 3 或用戶端 SDK 5 的步驟相同。

使用新憑證和私有金鑰重新設定 SSL

1. 使用下列 OpenSSL 命令來建立私有金鑰：

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

2. 使用以下 OpenSSL 命令來建立憑證簽署要求 (CSR)。將會詢問一連串有關憑證的問題。

```
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. 使用您初始化叢集時所建立的 *customerCA.crt* 憑證來簽署 CSR。

```
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
```

啟用自訂 SSL AWS CloudHSM

用戶端 SDK 3 或用戶端 SDK 5 的步驟不同。如需使用設定命令列工具的詳細資訊，請參閱 ???。

主題

- 用戶端 SDK 3 的自訂 SSL
- 用戶端 SDK 5 的自訂 SSL

用戶端 SDK 3 的自訂 SSL

使用用戶端 SDK 3 的設定工具來啟用自訂 SSL。如需有關用戶端 SDK 3 設定工具的詳細資訊，請參閱 ???。

使用自訂憑證和金鑰在 Linux 上使用用戶端 SDK 3 進行 TLS 用户端和服务器相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 使用設定工具來指定 `ssl-client.crt` 和 `ssl-client.key`。

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

3. 將 `customerCA.crt` 憑證新增至信任存放區。建立憑證主體名稱的雜湊。這會建立索引，以允許依該名稱來查詢憑證。

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

   建立目錄。

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

   建立檔案，其中包含憑證與雜湊名稱。

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

用戶端 SDK 5 的自訂 SSL

使用用戶端 SDK 5 任意設定工具來啟用自訂 SSL。如需有關用戶端 SDK 5 設定工具的詳細資訊，請參閱 ???。

PKCS #11 library

使用自訂憑證和金鑰在 Linux 上使用用戶端 SDK 5 進行 TLS 用户端和服务器相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 使用設定工具來指定 `ssl-client.crt` 和 `ssl-client.key`。

```
$ 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
```

使用自訂憑證和金鑰進行 TLS 用戶端伺服器與 Windows 用戶端 SDK 5 的相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 對於 PowerShell 解釋器，使用配置工具來指定`ssl-client.crt`和`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

使用自訂憑證和金鑰在 Linux 上使用用戶端 SDK 5 進行 TLS 用户端和服务器相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 使用設定工具來指定 ssl-client.crt 和 ssl-client.key。

```
$ 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

使用自訂憑證和金鑰在 Linux 上使用用戶端 SDK 5 進行 TLS 用户端和服务器相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 使用設定工具來指定 ssl-client.crt 和 ssl-client.key。

```
$ 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
```

使用自訂憑證和金鑰進行 TLS 用戶端伺服器與 Windows 用戶端 SDK 5 的相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 對於 PowerShell 解釋器，使用配置工具來指定ssl-client.crt和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

使用自訂憑證和金鑰在 Linux 上使用用戶端 SDK 5 進行 TLS 用户端和服务器相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 使用設定工具來指定 `ssl-client.crt` 和 `ssl-client.key`。

```
$ 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
```

使用自訂憑證和金鑰進行 TLS 用戶端伺服器與 Windows 用戶端 SDK 5 的相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 對於 PowerShell 解釋器，使用配置工具來指定`ssl-client.crt`和`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
```

建立應用程式

建置應用程式並使用 AWS CloudHSM.

如要開始在新叢集中建立和使用金鑰，您必須先使用 CloudHSM 管理公用程式 (CMU) 建立硬體安全模組 (HSM) 使用者。如需詳細資訊，請參閱了解 HSM 使用者管理任務、開始使用 AWS CloudHSM 命令列介面 (CLI)和如何管理 HSM 使用者。

> **Note**
>
> 如使用用戶端 SDK 3，請使用 CloudHSM 管理公用程式 (CMU) 而不是 CloudHSM CLI。

有了 HSM 使用者，您就可以登入 HSM，並使用下列任一選項來建立和使用金鑰：

- 使用金鑰管理公用程式，命令列工具
- 使用PKCS #11 程式庫建立 C 應用程式
- 使用 JCE 提供程序建立 Java 應用程式
- 使用直接從命令列使用 OpenSSL 動態引擎
- 使用 OpenSSL 動態引擎在 NGINX 和 Apache Web 伺服器上進行 TLS 卸載
- 使用 CNG 和 KSP 提供者 AWS CloudHSM 搭配 Microsoft 視窗伺服器憑證授權單位 (CA) 使用
- 使用 CNG 和 KSP 提供者 AWS CloudHSM 搭配 Microsoft 簽署工具使用
- 使用 CNG 和 KSP 提供程序在網際網路資訊伺服器 (IIS) Web 伺服器上進行 TLS 卸載

的最佳做法 AWS CloudHSM

執行本主題中的最佳作法以有效使用 AWS CloudHSM。

目錄

叢集管理

建立、存取和管理 AWS CloudHSM 叢集時，請遵循本節中的最佳作法。

擴展您的叢集以處理尖峰流量

有幾個因素會影響叢集可處理的最大輸送量，包括用戶端執行個體大小、叢集大小、網路拓撲，以及使用案例所需的密碼編譯作業。

作為起點，請參閱主題，以取得AWS CloudHSM 性能一般叢集大小和配置的效能預估。我們建議您使用預期的尖峰負載對叢集進行負載測試，以判斷目前的架構是否具有彈性且規模適當。

架構您的叢集以取得高可用性

為維護新增備援： AWS 可能會取代您的 HSM 進行排程維護或偵測到問題時。一般而言，您的叢集大小應該至少具有 +1 冗餘。例如，如果您的服務需要兩個 HSM 才能在尖峰時間運作，則理想的叢集大小將為三個。如果您遵循與可用性相關的最佳做法，這些 HSM 替換不應影響您的服務。不過，已取代 HSM 上進行中的作業可能會失敗，而且必須重試。

將 HSM 分散到許多可用區域：考慮您的服務在可用區域中斷期間如何運作。 AWS 建議您盡可能將 HSM 分散到盡可能多的可用區域。對於具有三個 HSM 的叢集，您應該將 HSM 分散到三個可用區域。視您的系統而定，您可能需要額外的備援。

至少有三個 HSM，以確保新產生金鑰的耐用性

對於需要新產生金鑰持久性的應用程式，建議至少有三個 HSM 分散在一個區域中的不同可用區域。

安全存取您的叢集

使用私有子網路限制對執行個體的存取：在 VPC 的私有子網路中啟動 HSM 和用戶端執行個體。這會限制從外部世界存取您的 HSM。

使用 VPC 端點存取 API： AWS CloudHSM 資料平面的設計可在不需要存取網際網路或 AWS API 的情況下運作。如果您的用戶端執行個體需要 AWS CloudHSM API 的存取權，您可以使用 VPC 端點存取 API，而不需要在用戶端執行個體上存取網際網路。如需詳細資訊，請參閱AWS CloudHSM 和 VPC 端點。

重新設定 SSL 以保護用戶端與伺服器之間的通訊安全： AWS CloudHSM 使用 TLS 建立與 HSM 的連線。初始化叢集之後，您可以取代用來建立外部 TLS 連線的預設 TLS 憑證和金鑰。如需詳細資訊，請參閱 使用 SSL/TLS 卸載來提高您的網絡服務器安全性 AWS CloudHSM。

根據您的需求擴展來降低成本

沒有使用前期成本 AWS CloudHSM。您必須為每個啟動的 HSM 支付小時費用，直到終止 HSM 為止。如果您的服務不需要連續使用 AWS CloudHSM，您可以在不需要 HSM 時將 HSM 縮減 (刪除) 為零來降低成本。當再次需要 HSM 時，您可以從備份還原 HSM。例如，如果您有一個工作負載要求您每月簽署一次程式碼 (特別是在月份的最後一天)，您可以在之前擴展叢集，在工作完成後刪除 HSM 以縮減叢集，然後還原叢集以在下個月底再次執行簽署作業。

AWS CloudHSM 自動定期備份叢集中的 HSM。稍後新增 HSM 時， AWS CloudHSM 會將最新的備份還原到新的 HSM，以便您可以從離開 HSM 的位置繼續使用。若要計算 AWS CloudHSM 架構成本，請參閱AWS CloudHSM 定價。

相關資源：

- 備份的一般概述
- Backup 保留政策
- 跨 AWS 區域複製備份

HSM 使用者管理

請遵循本節中的最佳作法，有效管理 AWS CloudHSM 叢集中的使用者。HSM 使用者與 IAM 使用者不同。具有以身分為基礎的政策且具有適當許可的 IAM 使用者和實體，可透過 AWS API 與資源互動來建立 HSM。HSM 建立後，您必須使用 HSM 使用者憑證來驗證 HSM 上的操作。如需 HSM 使用者的詳細指南，請參閱管理 HSM 使用者 AWS CloudHSM。

保護您的 HSM 使用者認證

由於 HSM 使用者是可以在 HSM 上存取和執行密碼編譯和管理作業的實體，因此必須安全地保護 HSM 使用者的認證。 AWS CloudHSM 無法存取您的 HSM 使用者憑證，如果您無法存取 HSM 使用者認證，將無法為您提供協助。

至少有兩名管理員以防止鎖定

若要避免叢集遭到鎖定，建議您至少有兩個管理員，以防遺失一個管理員密碼。如果發生這種情況，您可以使用其他管理員重置密碼。

> **Note**
>
> 用戶端 SDK 5 中的系統管理員是用戶端 SDK 3 中加密管理員 (CoS) 的代名詞。

啟用所有使用者管理作業的仲裁

仲裁可讓您設定必須先核准使用者管理作業的管理員數目，才能執行該作業。由於管理員擁有的權限，我們建議您為所有使用者管理作業啟用仲裁。如果您的其中一個管理員密碼遭到入侵，這可能會限制可能產生影響。如需詳細資訊，請參閱管理仲裁。

創建多個加密用戶，每個用戶都有限的權限

通過分離加密用戶的責任，沒有一個用戶可以完全控制整個系統。因此，我們建議您創建多個加密用戶並限制每個用戶的權限。通常情況下，這是通過給予不同的加密用戶明顯不同的責任和他們執行的操作來完成的（例如，有一個負責生成和與其他加密用戶共享密鑰的加密用戶，然後在您的應用程序中使用它們的加密用戶）來完成。

相關資源：

- 共用金鑰
- 取消共用金鑰

HSM 金鑰管理

在中管理金鑰時，請遵循本節中的最佳作法 AWS CloudHSM。

選擇正確的按鍵類型

使用工作階段金鑰時，每秒交易數 (TPS) 將限制為金鑰存在的一個 HSM。叢集中的額外 HSM 不會增加該金鑰要求的輸送量。如果您在相同的應用程式中使用 Token 金鑰，您的請求將在叢集中所有可用的 HSM 之間進行負載平衡。如需詳細資訊，請參閱 密鑰同步和耐久性設置 AWS CloudHSM。

管理金鑰儲存限制

HSM 對於一次可儲存在 HSM 上的權杖和工作階段金鑰數目上限有限制。如需金鑰儲存限制的資訊，請參閱AWS CloudHSM 配額。如果您的應用程式需要超過限制，您可以使用下列一或多個策略來有效管理金鑰：

使用受信任包裝將金鑰儲存在外部資料存放區中：使用受信任的金鑰環繞，您可以將所有包裝在外部資料存放區中的金鑰來克服金鑰儲存限制。當您需要使用此金鑰時，您可以將金鑰作為工作階段金鑰解除包裝到 HSM 中，使用金鑰進行所需作業，然後捨棄工作階段金鑰。原始金鑰資料會安全地儲存在您的資料存放區中，以供您在需要時使用。使用受信任的金鑰來達到最大程度的保護。

跨叢集散佈金鑰：克服金鑰儲存限制的另一個策略是將金鑰儲存在多個叢集中。在這種方法中，您可以維護每個叢集中儲存之金鑰的對應。使用此對應可使用所需金鑰將用戶端要求路由至叢集。如需如何從相同用戶端應用程式連線到多個叢集的詳細資訊，請參閱下列主題：

- 使用 JCE 提供者連線到多個叢集
- 使用 PKCS #11 連線到多個插槽

管理和保護金鑰包裝

密鑰可以通過屬性標記為可提取或不可提取。EXTRACTABLE根據預設，HSM 金鑰會標記為可擷取。

可擷取的金鑰是允許透過金鑰環繞從 HSM 匯出的金鑰。包裝的金鑰會加密，而且必須先使用相同的包裝金鑰解包，才能使用這些金鑰。在任何情況下都不得從 HSM 匯出不可擷取的金鑰。沒有辦法使不可提取的密鑰可提取。出於這個原因，考慮是否需要您的密鑰是否可提取，並相應地設置相應的密鑰屬性是非常重要的。

如果您需要在應用程式中進行金鑰包裝，您應該使用受信任的金鑰環繞來限制 HSM 使用者僅包裝/解除包裝已明確標示為受管理員信任的金鑰。如需詳細資訊，請參閱中有關受信任金鑰換行的主題管理金鑰 AWS CloudHSM。

相關資源

- 包裝和解包函數

- JCE 的密碼函數
- 支援的 Java 金鑰屬性
- CloudHSM CLI 的金鑰屬性

應用程式整合

請遵循本節中的最佳作法，以最佳化應用程式與 AWS CloudHSM 叢集整合的方式。

引導您的客戶端 SDK

您的用戶端 SDK 必須先啟動載入，才能連線到叢集。將 IP 位址啟動載入叢集時，建議盡可能使用`--cluster-id`參數。此方法會將叢集中的所有 HSM IP 位址填入您的組態，而不需要追蹤每個個別位址。如果 HSM 正在進行維護或在可用區域中斷期間，這麼做可為應用程式初始化增加額外的彈性。如需詳細資訊，請參閱引導用戶端 SDK。

驗證以執行操作

在中 AWS CloudHSM，您必須先對叢集進行驗證，才能執行大部分作業 (例如密碼編譯作業)。

使用 CloudHSM CLI 進行驗證：您可以使用 CloudHSM CLI 的單一命令模式或互動模式進行驗證。使用指登入令在互動模式下進行驗證。若要在單一指令模式中進行驗證，您必須設定環境變數`CLOUDHSM_ROLE`和`CLOUDHSM_PIN`。如需執行此操作的詳細資訊，請參閱單一命令模式。 AWS CloudHSM 建議您在應用程式未使用時，安全地儲存 HSM 認證。

使用 PKCS #11 進行驗證:在 PKCS #11 中，您可以在使用 C_ 開啟工作階段後使用 C_Login API 登入。OpenSession您只需要在每個插槽 (叢集) 中執行一個 C_Login。成功登入後，您可以使用 C_ 開啟其他工作階段，OpenSession 而不需要執行其他登入作業。如需驗證 PKCS #11 的範例，請參閱。PKCS #11 程式庫的程式碼範例

使用 JCE 進行身份驗證： AWS CloudHSM JCE 提供程序支持隱式和明確登錄。適合您的方法取決於您的使用案例。如果可能的話，我們建議您使用隱含登入，因為如果您的應用程式與叢集中斷連線且需要重新驗證，SDK 會自動處理驗證。使用隱含登入也可讓您在使用不允許您控制應用程式程式碼的整合時，為應用程式提供認證。如需登入方法的詳細資訊，請參閱提供憑證給 JCE 提供者。

使用 OpenSSL 進行驗證：使用 OpenSSL 動態引擎，您可以透過環境變數提供認證。 AWS CloudHSM 建議您在應用程式未使用時，安全地儲存 HSM 認證。如果可能的話，您應該將環境配置為有系統地擷取和設定這些環境變數，而無需手動輸入。如需使用 OpenSSL 進行驗證的詳細資訊，請參閱。安裝 OpenSSL 動態引擎

有效管理應用程式中的金鑰

使用索引鍵屬性來控制金鑰可執行的動作：產生金鑰時，請使用索引鍵屬性來定義一組權限，以允許或拒絕該金鑰的特定作業類型。我們建議使用完成任務所需的最少屬性來生成密鑰。例如，用於加密的 AES 金鑰也不應允許將金鑰包裝在 HSM 之外。如需詳細資訊，請參閱下列用戶端 SDK 的屬性頁面：

- PKCS #11 金鑰屬性
- JCE 金鑰屬性

盡可能快取金鑰物件以將延遲降至最低：金鑰尋找作業會查詢叢集中的每個 HSM。此作業非常昂貴，且不會隨叢集中的 HSM 計數進行調整。

- 使用 PKCS #11 時，您可以使用 API 找到金鑰。`C_FindObjects`
- 使用 JCE 時，您可以使用 KeyStore.

為了獲得最佳效能， AWS 建議您在應用程式啟動期間僅使用一次 key find 命令 (例如findKey和列出金鑰)，並快取應用程式記憶體中傳回的金鑰物件。如果您稍後需要此索引鍵物件，您應該從快取中擷取物件，而不是針對每個作業查詢此物件，這會增加顯著的效能負荷。

使用多執行緒

AWS CloudHSM 支持多線程應用程序，但有一些事情要記住與多線程應用程序。

使用 PKCS #11 時，您應該只初始化 PKCS #11 程式庫 (呼叫`C_Initialize`) 一次。每個線程應該被分配自己的會話（`C_OpenSession`）。不建議在多個執行緒中使用相同的工作階段。

使用 JCE， AWS CloudHSM 提供者應該只初始化一次。請勿跨執行緒共用 SPI 物件的執行個體。例如，密碼，簽名，摘要，Mac KeyFactory 或 KeyGenerator 對象應該只在自己的線程的上下文中使用。

處理節流錯誤

在下列情況下，您可能會遇到 HSM 節流錯誤：

- 您的叢集未正確調整以管理尖峰流量。
- 在維護事件期間，叢集的大小不會有 +1 備援。

- 可用區域中斷會導致叢集中可用 HSM 的數量減少。

如需有HSM 調節關如何最佳處理此案例的資訊，請參閱。

若要確保叢集的大小適當且不會受到限制， AWS 建議您在環境中以預期的尖峰流量進行負載測試。

整合叢集作業的重試

AWS 可能會因操作或維護原因更換您的 HSM。為了讓應用程式能夠回復這種情況， AWS 建議您在路由到叢集的所有作業上實作用戶端重試邏輯。因取代而導致失敗作業的後續重試預期會成功。

實作災難復原策略

為了回應某個事件，可能需要將流量從整個叢集或區域移開。以下幾節描述了執行此操作的多種策略。

使用 VPC 對等互連從其他帳戶或區域存取叢集：您可以利用 VPC 對等互連從其他帳戶或地區存取 AWS CloudHSM 叢集。如需如何設定此功能的詳細資訊，請參閱什麼是 VPC 對等互連？ 在 VPC 對等互連指南中。建立對等連線並適當設定安全群組後，您就可以使用與通常相同的方式與 HSM IP 位址進行通訊。

從相同應用程式 Connect 到多個叢集：用戶端 SDK 5 中的 JCE 提供者、PKCS #11 程式庫和 CloudHSM CLI 支援從同一應用程式連線到多個叢集。例如，您可以有兩個使用中叢集，每個叢集位於不同的區域，而且應用程式可以一次連線到兩者，並在兩者之間進行負載平衡，做為正常作業的一部分。如果您的應用程式未使用用戶端 SDK 5 (最新的 SDK)，則您無法從同一應用程式連線到多個叢集。或者，您可以讓另一個叢集保持正常運作，如果發生區域中斷，請將流量轉移到另一個叢集，以將停機時間降至最低。有關詳細信息，請參閱相應頁面：

- 使用 PKCS #11 連線到多個插槽
- 使用 JCE 提供者連線到多個叢集
- 使用 CloudHSM CLI 連線到多個叢集

從備份還原叢集：您可以從現有叢集的備份建立新叢集。如需詳細資訊，請參閱 管理 AWS CloudHSM 備份。

監控

本節說明您可以用來監視叢集和應用程式的多種機制。如需監視的其他詳細資訊，請參閱監控 AWS CloudHSM。

監控用戶端記錄

每個用戶端 SDK 都會寫入您可以監視的記錄檔。如需用戶端記錄的資訊，請參閱使用用戶端 SDK 日誌。

在設計為暫時性的平台上 (例如 Amazon ECS) 和 AWS Lambda從檔案收集用戶端日誌可能很困難。在這些情況下，最佳做法是將 Client SDK 記錄設定為將記錄檔寫入主控台。大多數服務會自動收集此輸出並將其發佈到 Amazon CloudWatch 日誌，以供您保留和查看。

如果您在 AWS CloudHSM Client SDK 之上使用任何第三方整合，您應該確保設定該軟體套件，以便將其輸出記錄到主控台。 AWS CloudHSM 用戶端 SDK 的輸出可能會由此封裝擷取，否則會寫入其自己的記錄檔。

如需有關用戶端 SDK 5 設定工具如何在應用程式中設定記錄選項的資訊，請參閱《》。

監控稽核記錄

AWS CloudHSM 將稽核日誌發佈到您的 Amazon CloudWatch 帳戶。稽核記錄來自 HSM，並追蹤特定作業以供稽核之用。

您可以使用稽核記錄來追蹤在 HSM 上叫用的任何管理命令。例如，您可以在發現未預期的管理作業正在執行時觸發警示。

如需詳細資訊，請參閱HSM 稽核記錄的運作方式。

監控 AWS CloudTrail

AWS CloudHSM 與 (提供中的使用者 AWS CloudTrail、角色或服務所採取的動作記錄) 的 AWS 服務整合 AWS CloudHSM。 AWS CloudTrail 擷取 AWS CloudHSM 作為事件的所有 API 呼叫。擷取的呼叫包括來自 AWS CloudHSM 主控台的呼叫和 AWS CloudHSM API 作業的程式碼呼叫。

您可 AWS CloudTrail 以使用稽核對 AWS CloudHSM 控制平面進行的任何 API 呼叫，以確保您的帳戶中不會發生任何不必要的活動。

如需詳細資訊，請參閱 使用 AWS CloudTrail 和 AWS CloudHSM。

監控 Amazon CloudWatch 指標

您可以使用 Amazon CloudWatch 指標即時監控 AWS CloudHSM 叢集。指標可依地區、叢集 ID 或 HSM ID 和叢集識別碼分組。

使用 Amazon CloudWatch 指標，您可以設定 Amazon CloudWatch 警示，以提醒您任何可能會影響服務的潛在問題。我們建議您設定警報以監控下列項目：

- 接近 HSM 上的金鑰限制
- 接近 HSM 上的 HSM 工作階段計數限制
- 接近 HSM 上的 HSM 使用者計數限制
- 用於識別同步問題的 HSM 使用者或金鑰計數差異
- 運作狀況不佳的 HSM 可擴充叢集，直到 AWS CloudHSM 可以解決問題

如需詳細資訊，請參閱使用 Amazon CloudWatch 日誌和 AWS CloudHSM 稽核日誌。

管理 AWS CloudHSM 叢集

您可以從AWS CloudHSM 主控台或其中一個 AWS SDK 或命令列工具管理 AWS CloudHSM 叢集。如需詳細資訊，請參閱下列主題。

若要建立叢集，請參閱開始使用。

叢集架構

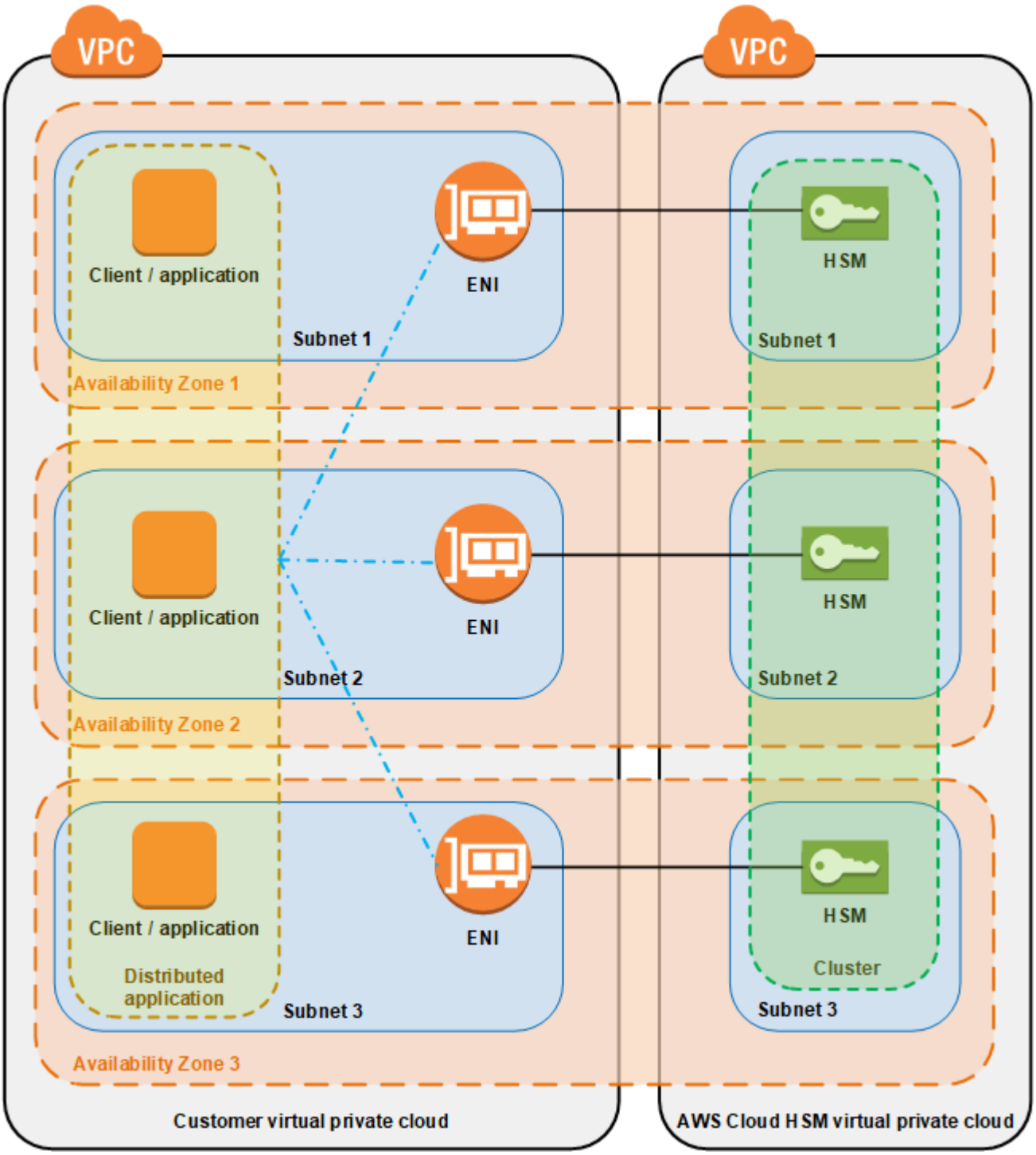
建立叢集時，您可以在 AWS 帳戶中指定 Amazon Virtual Private Cloud (VPC)，並在該 VPC 中指定一個或多個子網路。建議您在所選區域的每個可用區 AWS 域 (AZ) 中建立一個子網路。您可以在建立 VPC 時建立私有子網路。如需進一步了解，請參閱建立虛擬私有雲端 (VPC)。

每次建立 HSM 時，您都需要指定 HSM 的叢集和可用區域。只要將 HSM 放在不同的可用區域，萬一某個可用區域無法使用，就能發揮備援和高可用性。

建立 HSM 時，會在 AWS 帳戶中的指定子網路中 AWS CloudHSM 放置 elastic network interface (ENI)。彈性網路界面是用來與 HSM 互動的界面。HSM 位於所擁有 AWS 帳戶中的個別 VPC 中。AWS CloudHSM HSM 及其對應的網路界面位於相同的可用區域。

若要與叢集中的 HSM 互動，您需要 AWS CloudHSM 用戶端軟體。一般而言，您會將用戶端安裝在 Amazon EC2 執行個體 (稱為用戶端執行個體)，而這些執行個體與 HSM ENI 位於相同的 VPC，如下圖所示。在技術上不必如此；在任何相容的電腦上，只要可連接到 HSM ENI，都可以安裝用戶端。用戶端會透過 ENI 來與叢集的個別 HSM 通訊。

下圖表示具有三個 HSM 的 AWS CloudHSM 叢集，每個 HSM 都位於 VPC 中的不同可用區域中。



叢集同步

在 AWS CloudHSM 叢集中， AWS CloudHSM 使個別 HSM 上的金鑰保持同步。您不需要做任何動作來同步 HSM 上的金鑰。若要讓每個 HSM 上的使用者和原則保持同步，請先更新用 AWS CloudHSM 戶端組態檔案，然後再管理 HSM 使用者。如需詳細資訊，請參閱 讓 HSM 使用者保持同步。

將新的 HSM 新增至叢集時，會備份現有 HSM 上的所有金鑰、 AWS CloudHSM 使用者和原則。然後將該備份還原到新的 HSM。這樣就可讓兩個 HSM 保持同步。

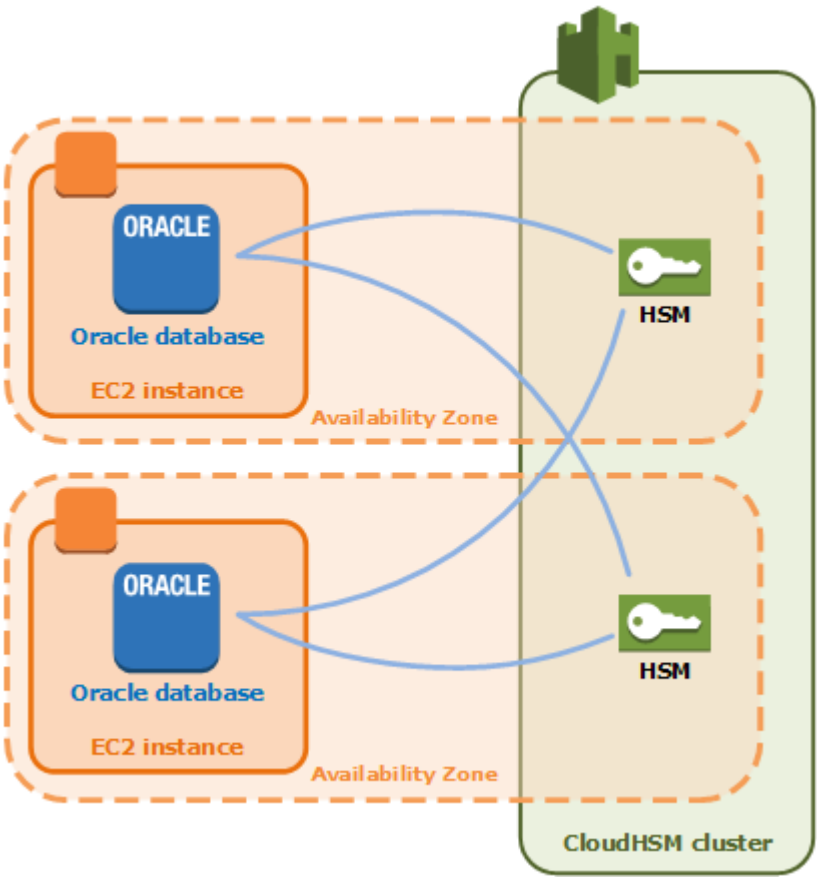
如果叢集中的 HSM 不同步化， AWS CloudHSM 會自動重新同步化它們。若要啟用此功能，請 AWS CloudHSM 使用設備使用者的認證。此使用者存在於由提供的所有 HSM 上， AWS CloudHSM 且權限有限。這個使用者可以取得 HSM 上的物件雜湊，並可以擷取和插入遮罩 (加密) 物件。 AWS 無法檢視或修改您的使用者或金鑰，也無法使用這些金鑰執行任何加密操作。

叢集高可用性和負載平衡

當您建立具有多個 HSM 的 AWS CloudHSM 叢集時，您會自動取得負載平衡。負載平衡表示 AWS CloudHSM 用戶端會根據每個 HSM 的額外處理能力，將密碼編譯操作分散給叢集中的所有 HSM 用戶端。

當您在不同的 AWS 可用區域中建立 HSM 時，您會自動取得高可用性。高可用性表示您可以獲得更高的可靠性，因為沒有任何一個 HSM 會發生單點失效。我們建議您在每個叢集中至少有兩個 HSM，每個 HSM 位於一個區域內的不同可用區域。 AWS

例如，下圖顯示 Oracle 資料庫應用程式分佈至兩個不同的可用區域。資料庫執行個體會將其主要金鑰儲存在每個可用區域中包含 HSM 的叢集中。 AWS CloudHSM 會自動將金鑰同步至兩個 HSM，以便立即存取和備援金鑰。



AWS CloudHSM 叢集模式和 HSM 類型

AWS CloudHSM 提供兩種叢集模式：FIPS 和非 FIP。 AWS CloudHSM 也提供兩種 HSM 類型：中型和 hsm2m。在決定適合您需求的叢集模式和 HSM 類型之前，請檢閱此頁面上的詳細資訊。

> **Note**
>
> 在 2024 年 6 月 10 日之前建立的所有叢集都處於 FIPS 模式，且 HSM 類型為 hsm1. 中型。

若要查看叢集的模式和 HSM 類型，請使用描述叢集命令。

叢集模式

AWS CloudHSM 提供兩種模式的叢集：FIPS 和非 FIP。在 FIPS 模式下，只能使用聯邦資訊處理標準 (FIPS) 核准的金鑰和演算法。非 FIPS 模式提供支援的所有金鑰和演算法 AWS CloudHSM，無論 FIPS 核准為何。

下表列出每個叢集模式之間的主要差異：

差異特徵	FIPS 模式	非 FIPS 模式
HSM 類型相容性	可搭配 hsm1. 中型。	適用於 HSM2 公尺。
Backup 相容性	只能用於在 FIPS 模式下備份還原叢集。	只能用於在非 FIP 模式下備份還原叢集。
按鍵選擇	支援 FIPS 核准[1]的 AWS CloudHSM 金鑰。	支援同時獲得 FIPS 核准和未通過 FIPS 核准的 AWS CloudHSM 金鑰。
演算法	支援 FIPS 核准[1]的 AWS CloudHSM 演算法。	支援已通過 FIPS 核准且未通過 FIPS 核准的 AWS CloudHSM 演算法。
認證	符合 FIPS 140-2、電容式引腳和 PCI-3DS 相容。	

[1] 有關詳細信息，請參見棄用通知。

在選擇叢集模式之前，請注意，叢集的模式 (FIPS 或非 FIP) 在建立之後無法變更，因此請務必根據需要選取正確的模式。

HSM 類型

除了叢集模式之外，還 AWS CloudHSM 提供兩種 HSM 類型：hsm1.medium 和 hsm 2m。每個 HSM 類型使用不同的硬體，而每個叢集只能包含一種類型的 HSM。下表列出了兩者之間的主要區別：

差異特徵	hsm1. 中	hsm2 米。中
叢集模式相容性	適用於 FIPS 模式下的叢集。	目前可用於非 FIP 模式下的叢集。
Backup 相容性	只能用於將還原備份到 hsm1. medium 叢集。	只能用來備份還原 hsm2 m. 媒體叢集。

差異特徵	hsm1. 中	hsm2 米。中
金鑰容量	每個叢集有 3,300 個。	總共 16,666 個金鑰，具有非對稱金鑰，每個叢集最多有 3,333 個金鑰。
用戶端開發套件	支援所有用戶端 SDK。	支援除 CNG 和 K SP 提供者以外的所有用戶端 SDK。
用戶端 SDK 版本	相容於 SDK 3.1.0 及更高版本。	與客戶端 SDK 版本 5.12.0 及更高版本兼容。
區域可用性	在提供 CloudHSM 服務的所有區域均可使用。	在有限數量的區域中提供，即將推出其他支援地區。若要查看此 HSM 類型可用的區域，請參閱定AWS CloudHSM 價計算器。
效能	若要查看每種 HSM 類型的效能，請參閱AWS CloudHSM 性能。	
認證	符合 FIPS 140-2、支援資料輸入資料輸入、PCI 引腳、SOC2 和 PCI-3DS 相容。	符合 PCI DSS 標準。

[1] 有關詳細信息，請參見棄用通知。

Connect 用戶端 SDK 連線至 AWS CloudHSM 叢集

如要使用用戶端 SDK 5 或用戶端 SDK 3 連接至叢集，您必須先執行兩項操作：

- 在 EC2 執行個體上擁有發行憑證
- 將用戶端 SDK 引導至叢集

在每個 EC2 執行個體上安裝發行憑證

您可以在初始化叢集時建立發行憑證。將發行憑證複製到連接至叢集每個 EC2 執行個體平台的預設位置。

Linux

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

Windows

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

指定憑證的位置。

如果是用戶端 SDK 5，您可使用設定工具指定簽發憑證的位置。

PKCS #11 library

將用戶端 SDK 5 的簽發憑證安裝於 Linux 上

- 使用設定工具指定簽發憑證的位置。

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

將用戶端 SDK 5 的簽發憑證安裝於 Windows 上

- 使用設定工具指定簽發憑證的位置。

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

OpenSSL Dynamic Engine

將用戶端 SDK 5 的簽發憑證安裝於 Linux 上

- 使用設定工具指定簽發憑證的位置。

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

JCE provider

將用戶端 SDK 5 的簽發憑證安裝於 Linux 上

- 使用設定工具指定簽發憑證的位置。

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

將用戶端 SDK 5 的簽發憑證安裝於 Windows 上

- 使用設定工具指定簽發憑證的位置。

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

CloudHSM CLI

將用戶端 SDK 5 的簽發憑證安裝於 Linux 上

- 使用設定工具指定簽發憑證的位置。

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

將用戶端 SDK 5 的簽發憑證安裝於 Windows 上

- 使用設定工具指定簽發憑證的位置。

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

如需詳細資訊，請參閱設定工具。

如需有關初始化叢集或建立和簽署憑證的詳細資訊，請參閱初始化叢集。

引導用戶端 SDK

用戶端 SDK 版本不同，啟動程序也不一樣，但您必須擁有叢集中其中一個硬體安全性模組 (HSM) 的 IP 地址。您可以使用任何連接至叢集的 HSM 的 IP 地址。用戶端 SDK 連接之後，會擷取其他任何 HSM 的 IP 地址，並執行負載平衡和用戶端金鑰同步處理操作。

取得叢集的 IP 地址

取得 HSM (主控台) 的 IP 位址

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 若要變更 AWS 區域，請使用頁面右上角的區域選擇器。
3. 若要開啟叢集詳細資訊頁面，請在叢集表格中選擇叢集 ID。
4. 若要取得 IP 地址，請在 HSM 索引標籤上，選擇 ENI IP 地址下列出的其中一個 IP 地址。

若要取得 HSM 的 IP 位址 ()AWS CLI

- 使用 AWS CLI中的 describe-clusters 命令取得 HSM 的 IP 地址。在命令輸出中，HSM 的 IP 地址是 `EniIp` 的值。

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
        { ... }
```

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

如需有關啟動程序的詳細資訊，請參閱設定工具。

如要啟用用戶端 SDK 5

PKCS #11 library

為用戶端 SDK 5 引導 Linux EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 位址。

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

為用戶端 SDK 5 引導 Windows EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 位址。

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

OpenSSL Dynamic Engine

為用戶端 SDK 5 引導 Linux EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 位址。

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

JCE provider

為用戶端 SDK 5 引導 Linux EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 位址。

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

為用戶端 SDK 5 引導 Windows EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 位址。

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

CloudHSM CLI

為用戶端 SDK 5 引導 Linux EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 地址。

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

為用戶端 SDK 5 引導 Windows EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 地址。

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

> **Note**
>
> 您可以使用 `--cluster-id` 參數來代替 `-a <HSM_IP_ADDRESSES>`。若要查看使用 `--cluster-id` 的需求，請參閱 用戶端 SDK 5 設定工具。

如要啟用用戶端 SDK 3

為用戶端 SDK 3 引導 Linux EC2 執行個體

- 用configure於指定叢集中 HSM 的 IP 位址。

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

為用戶端 SDK 3 引導 Windows EC2 執行個體

- 用configure於指定叢集中 HSM 的 IP 位址。

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

如需設定的詳細資訊，請參閱 ???。

在叢集中新增或移除 HSM AWS CloudHSM

若要擴展或縮減 AWS CloudHSM 叢集，請使用AWS CloudHSM 主控台或其中一個 AWS SDK 或命令列工具來新增或移除 HSM。我們建議您測試叢集的負載，以確定您應預期的尖峰負載，然後在叢集中再新增一個 HSM 以確保高可用性。

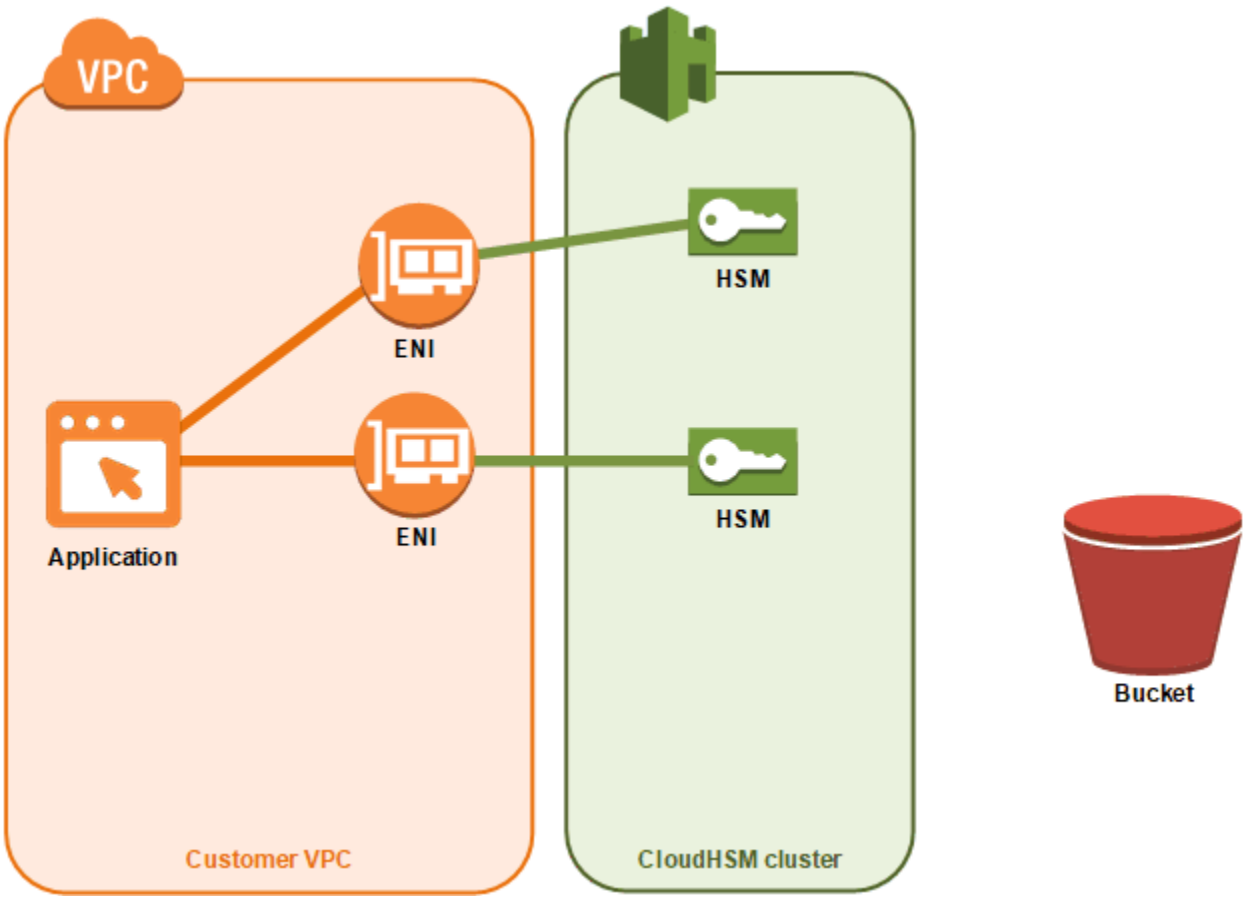
主題

- 新增 HSM

- 移除 HSM

新增 HSM

下圖說明將 HSM 新增到叢集時發生的事件。



1. 您將新的 HSM 新增到叢集。下列程序說明如何從 AWS CloudHSM 主控台、AWS Command Line Interface (AWS CLI) 和 AWS CloudHSM API 執行此作業。

   您只需要執行這個動作。剩餘的事件會自動發生。
2. AWS CloudHSM 建立叢集中現有 HSM 的備份副本。如需詳細資訊，請參閱 備份。
3. AWS CloudHSM 將備份還原至新的 HSM。這可確保此 HSM 與叢集的其他 HSM 同步。
4. 叢集中的現有 HSM 會通知用 AWS CloudHSM 戶端叢集中有新的 HSM。
5. 用戶端會對新的 HSM 建立連線。

新增 HSM (主控台)

1. 開啟主 AWS CloudHSM 控台，網址為 https://console.aws.amazon.com/cloudhsm/home。
2. 針對您要新增的 HSM 選擇叢集。
3. 在 HSM 標籤上，選擇 Create HSM (建立 HSM)。
4. 針對您要建立的 HSM 選擇可用區域 (AZ)。然後選擇 Create (建立)。

新增 HSM (AWS CLI)

- 在命令提示字元中，發出 create-hsm 命令，並指定您要建立之 HSM 的叢集 ID 和可用區域。如果您不知道慣用叢集的叢集 ID，請發出 describe-clusters 命令。指定可用區域，格式為 us-east-2a、us-east-2b 等。

```
$ 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"
    }
}
```

若要新增 HSM (AWS CloudHSM API)

- 傳送 CreateHsm 請求，並指定您要建立之 HSM 的叢集 ID 和可用區域。

移除 HSM

您可以使用AWS CloudHSM 主控台、或 AWS CloudHSM API 來移除 HSM。AWS CLI

移除 HSM (主控台)

1. 開啟主 AWS CloudHSM 控台，網址為 https://console.aws.amazon.com/cloudhsm/home。

2. 選擇叢集，其中包含您要移除的 HSM。
3. 在 HSM 標籤中，選擇您要移除的 HSM。然後選擇 Delete HSM (刪除 HSM)。
4. 確認您要刪除 HSM。然後選擇 Delete (刪除)。

移除 HSM (AWS CLI)

- 在命令提示字元中，發出 delete-hsm 命令。傳遞叢集 (包含您要刪除的 HSM) 的 ID 和以下其中一個 HSM 識別符：
  - HSM ID (`--hsm-id`)
  - HSM IP 地址 (`--eni-ip`)
  - HSM 的彈性網路界面 ID (`--eni-id`)

  如果您不知道這些識別符的值，請發出 describe-clusters 命令。

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

若要移除 HSM (AWS CloudHSM API)

- 傳送 DeleteHsm 請求，並指定您要刪除之 HSM 的叢集 ID 和識別符。

刪除 AWS CloudHSM 叢集

您必須先移除叢集中的所有 HSM 後，才能刪除叢集。如需詳細資訊，請參閱 移除 HSM。

移除所有 HSM 之後，您可以使用AWS CloudHSM 主控台、AWS Command Line Interface (AWS CLI) 或 AWS CloudHSM API 刪除叢集。

刪除叢集 (主控台)

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 選擇您要刪除的叢集。然後選擇刪除叢集。
3. 確認您要刪除叢集，然後選擇刪除。

刪除叢集 (AWS CLI)

- 在命令提示字元中，發出 delete-cluster 命令，並傳遞您要刪除的叢集 ID。如果您不知道叢集 ID，請發出 describe-clusters 命令。

```
$ 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"
    }
}
```

刪除AWS CloudHSM 叢集 (API)

- 傳送 DeleteCluster 請求，指定您要刪除的叢集 ID。

從備份建立 AWS CloudHSM 叢集

若要從備份還原 AWS CloudHSM 叢集，請遵循本主題中的步驟。您的叢集會包含備份中的相同使用者、金鑰資料、憑證、組態和政策。如需管理備份的詳細資訊，請參閱 管理備份。

從備份建立叢集 (主控台)

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 選擇建立叢集。
3. 在叢集組態區段中，執行下列操作：
   a. 對於 VPC，選擇您要建立叢集的 VPC。
   b. 對於 AZ，為您要新增至叢集的每個可用區域選擇私有子網路。
4. 在叢集來源區段中，執行下列操作：
   a. 選擇從當前備份還原。
   b. 選擇您要還原的備份。
5. 選擇下一步：檢閱。
6. 檢閱您的叢集組態，然後選擇建立叢集。
7. 指定服務應保留備份的時間長度。

   接受預設保留期 90 天，或輸入介於 7 到 379 天之間的新值。服務會自動刪除此叢集中超過您在此指定值的備份。您之後可以變更這個設定。如需詳細資訊，請參閱 設定備份保留原則。
8. 選擇下一步。
9. (選用) 輸入標籤索引鍵和選用標籤值。若要將多個標籤新增至叢集，請選擇新增標籤。
10. 選擇檢閱。
11. 檢閱您的叢集組態，然後選擇建立叢集。

> **Tip**
>
> 若要在此叢集中建立 HSM，其中包含與您還原之備份中相同的使用者、金鑰材料、憑證、組態和原則，請將 HSM 新增至叢集。

從備份建立叢集 (AWS CLI)

如要確定備份 ID，請執行 describe-backups 命令。

- 在命令提示字元中，發出 create-cluster 命令。指定 HSM 執行個體類型、您計劃在其中建立 HSM 子網路的子網路 ID，以及您要還原之備份的備份 ID。

```
$ 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"
    }
}
```

從備份 (AWS CloudHSM API) 建立叢集

請參閱下列主題，了解如何使用 API 從備份創建叢集。

- CreateCluster

管理 AWS CloudHSM 備份

AWS CloudHSM 至少每 24 小時定期備份叢集一次。每個備份包含以下資料的加密複本：

- 使用者 (CO、CU 和 AU)
- 金鑰資料與憑證
- 硬體安全模組 (HSM) 組態與原則

您無法指示服務進行備份，但您可以執行某些操作強制服務建立備份。這項服務會在您執行下列任一動作時，進行備份：

- 啟用叢集
- 將 HSM 新增至作用中叢集
- 從叢集移除 HSM

AWS CloudHSM 根據您在建立叢集時設定的備份保留原則刪除備份。如需管理備份保留原則的詳細資訊，請參閱 設定備份保留原則。

主題

使用備份

將 HSM 新增到之前包含一個或多個作用中 HSM 的叢集時，服務會將最近的備份還原到新的 HSM 上。使用備份來管理您不常使用的 HSM。當您刪除不需要使用的 HSM時，會觸發備份。之後，如果您需要使用 HSM，請在同一叢集中建立新的 HSM，此動作會還原先前您刪除 HSM 操作時建立的備份。

移除過期金鑰或非作用中使用者

您可能會想從您的環境移除不需要的密碼編譯資料，例如過期金鑰或非作用中使用者。這個程序需要兩個步驟：首先，從 HSM 中刪除這些材料。然後，刪除所有現有的備份。此步驟可確保您從備份初始

化新叢集時，不會還原備份中的已刪除資訊。如需詳細資訊，請參閱 the section called “刪除和還原備份”。

考量災難復原

您可以從備份中建立叢集。您可能需要執行此操作來設定叢集的復原點。指定一份包含復原點中所需的所有使用者、金鑰資料和憑證的備份，然後使用該備份建立新叢集。如需從備份中建立叢集的詳細資訊，請參閱 從備份建立叢集。

您也可以將叢集的備份複製到另一個區域中，然後可在該區域建立新的叢集，做為原始叢集的複本。您可能出於多種原因而希望這樣做，包括簡化災難復原程序。如需將備份複製到區域的詳細資訊，請參閱跨區域複製備份。

刪除和還原備份

刪除備份後，服務會將備份保留七天，在此期間您可以還原備份。七天後，您無法再還原備份。如需管理備份的詳細資訊，請參閱 管理備份。

刪除和還原備份 (控制台)

刪除備份原則 (主控台)

1. 開啟主 AWS CloudHSM 控台，網址為 https://console.aws.amazon.com/cloudhsm/home。
2. 如要變更 AWS 區域，請使用頁面右上角的 區域選擇器。
3. 在導覽窗格中，選擇備份。
4. 選擇要刪除的備份。
5. 如要刪除選取的備份，請選擇 Actions, Delete。

   刪除備份對話方塊隨即出現。
6. 選擇刪除。

   備份的狀態變更為PENDING_DELETE。您最多可以在請求刪除備份後的 7 天內，還原待刪除的備份。

如要備份原則 (主控台)

1. 開啟主 AWS CloudHSM 控台，網址為 https://console.aws.amazon.com/cloudhsm/home。

2. 如要變更 AWS 區域，請使用頁面右上角的 區域選擇器。
3. 在導覽窗格中，選擇備份。
4. 選擇 PENDING_DELETE 狀態下的備份進行還原。
5. 如要還原選取的備份，請選擇 Actions, Restore。

刪除和還原備份 (AWS CLI)

使用 AWS CLI中的 describe-backups 命令檢查備份狀態或尋找其 ID。

如要刪除備份 (AWS CLI)

- 在命令提示字元下，執行 delete-backup 命令，傳遞要刪除的備份 ID。

```
$ 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"
    }
}
```

如要還原備份 (AWS CLI)

- 若要還原備份，請發出 restore-backup 命令，傳遞處於 PENDING_DELETION 狀態的備份 ID。

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

如要列出備份 (AWS CLI)

- 如要查看 PENDING_DELETION 狀態下的所有備份清單，請執行 describe-backups 命令並指定 states=PENDING_DELETION 做為篩選條件。

```
$ 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": []
        }
}
```

刪除和還原備份 (AWS CloudHSM API)

請參閱下列主題，了解如何使用 API 刪除和還原備份。

- DeleteBackup
- RestoreBackup

設定 AWS CloudHSM 備份保留原則

叢集的預設備份保留原則為 90 天，但不包括 2020 年 11 月 18 日之前建立的叢集。您可以將此期間設置為 7 到 379 天之間的任何數字。 AWS CloudHSM 不會刪除叢集的上次備份。如需管理備份的詳細資訊，請參閱 管理備份。

了解備份保留原則

AWS CloudHSM 根據您在建立叢集時設定的備份保留原則來清除備份。備份保留原則適用於叢集。如果您將備份移至其他區域，該備份就不再與叢集產生關聯，也不再適用備份保留原則。您必須手動刪除任何與叢集沒有關聯的備份。 AWS CloudHSM 不會刪除叢集的上次備份。

AWS CloudTrail 會報告標記為刪除的備份。您可以還原服務清除的備份，跟還原手動刪除的備份一樣。如要防止競爭，您應該先變更叢集的備份保留原則，然後再還原由服務刪除的備份。如果您不想變更原則但又想保留選取的備份，您可以指定服務從叢集備保留原則中排除備份。

現有叢集排除情況

AWS CloudHSM 於 2020 年 11 月 18 日推出託管備份保留。對於 2020 年 11 月 18 日之前建立的叢集，其備份保留原則還要加上額外 90 天叢集的保留期限。例如，如果您在 2019 年 11 月 18 日建立叢集，服務會為您的叢集指派一年的備份保留原則，再加上 90 天 (一共 455 天)。

> **Note**
>
> 您可以聯絡支援人員 (https://aws.amazon.com/support)，選擇退出管理備份保留。

設定備份保留 (主控台)

如要設定備份保留 (主控台)

1. 開啟主 AWS CloudHSM 控台，網址為 https://console.aws.amazon.com/cloudhsm/home。
2. 若要變更 AWS 區域，請使用頁面右上角的區域選擇器。
3. 按一下載作用中狀態下叢集的叢集 ID，即可管理該叢集的備份保留原則。
4. 如要變更備份保留原則，請選擇 Actions, Change backup retention period。

   這時系統會顯示變更備份保留期限的對話方塊。
5. 在 Backup retention period (in days)，輸入介於 7 到 379 天之間的數字。
6. 選擇 Change backup retention period。

如要從備份保留原則 (主控台) 排除或包含一個備份

1. 開啟主 AWS CloudHSM 控台，網址為 https://console.aws.amazon.com/cloudhsm/home。
2. 若要檢視備份，請在導覽窗格中選擇 Backups。
3. 按一下處於「就緒」狀態的備份 ID 以排除或包含某個備份。
4. 在 Backup details 頁面上，採取下列其中一項動作。
   - 如要排除日期為 Expiration time 的備份，請選擇 Actions, Disable expiration。

- 如要包含未過期備份，請選擇 Actions, Use cluster retention policy。

設定備份保留原則 (AWS CLI)

使用 AWS CLI中的 describe-backups 命令檢查備份狀態或尋找其 ID。

如要設定備份保留 (AWS CLI)

- 在命令提示字元中，發出 modify-cluster 命令。指定叢集 ID 和備份保留原則。

```
$ 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"
   }
}
```

如要從備份保留原則 (AWS CLI) 排除或包含一個備份

- 在命令提示字元中，發出 modify-backup-attributes 命令。指定備份 ID 並設定永不過期旗標以保留備份。

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

如要在備份保留原則中包含一個備份 (AWS CLI)

- 在命令提示字元中，發出 modify-backup-attributes 命令。指定備份 ID 並設定 no-never-expires 旗標以將備份包含在備份保留原則中，這表示服務最後會刪除備份。

```
$ aws cloudhsmv2 modify-backup-attributes --backup-id <backup ID> \
                                          --no-never-expires
{
   "Backup": {
      "BackupId": "backup-ro5c4er4aac",
      "BackupState": "READY",
      "ClusterId": "cluster-dygnwhmscg5",
      "NeverExpires": false
   }
}
```

設定備份保留 (AWS CloudHSM API)

請參閱下列主題，了解如何使用 API 管理備份原則。

- ModifyCluster
- ModifyBackupAttributes

跨 AWS 區域複製備份

您可以基於多種原因，進行跨區域複製備份，包括跨區域復原能力、全域工作負載和災難復原。複製備份後，備份會以 CREATE_IN_PROGRESS 狀態顯示在目標區域。成功複製後，備份狀態會變成 READY。如果複製失敗，備份的狀態會變更為 DELETED。請檢查您輸入的參數是否有誤，並確保指定的來源備份處於 DELETED 狀態，然後再重新執行操作。如需備份或如何從備份建立叢集的資訊，請參閱 管理備份 或 從備份建立叢集。

注意下列事項：

- 若要複製叢集備份到目標區域，您的帳戶必須擁有適當的 IAM 政策許可。為將備份複製到其他區域，您的 IAM 政策必須允許訪問備份所在的來源區域。跨區域複製之後，您的 IAM 政策必須允許訪問目標區域，才能和複製的備份互動，其中包含使用 CreateCluster 操作。如需詳細資訊，請參閱 建立 IAM 管理員。
- 原始叢集和可從目標區域中備份建立的叢集並未連結。您必須分別管理每個叢集。如需詳細資訊，請參閱 管理 叢集。
- 無法在 AWS 限制區域和標準區域之間複製備份。備份可以在 AWS GovCloud (美國東部) 和 AWS GovCloud (美國西部) 區域之間複製。

將備份複製到不同的區域 (控制台)

將備份複製到不同的區域 (控制台)

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 如要變更 AWS 區域，請使用頁面右上角的 區域選擇器。
3. 在導覽窗格中，選擇備份。
4. 將備份複製到不同區域。
5. 如要複製所選備份，請選擇動作，複製到另一個區域。

   將會顯示 [將備份複製到其他區域] 對話方塊。
6. 在目標區域中，從選取一個區域選擇一個區域。
7. (選用) 輸入標籤索引鍵和選用標籤值。若要將多個標籤新增至叢集，請選擇新增標籤。
8. 選擇複製備份。

將備份複製到不同的區域 (AWS CLI)

如要確定備份 ID，請執行 describe-backups 命令。

將備份複製到不同的區域 (AWS CLI)

- 在命令提示中，執行 copy-backup-to-region 命令。指定目標區域以及來源備份的備份 ID。如果指定備份 ID，則會複製關聯的備份。

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

將備份複製到不同的區域 (AWS CloudHSM API)

請參閱下列主題，了解如何使用 API 將備份複製到不同區域。

- CopyBackupToRegion

使用共用備份

CloudHSM 與 AWS Resource Access Manager (AWS RAM) 整合以啟用資源共用功能。 AWS RAM 是一項服務，可讓您與其他人 AWS 帳戶 或透過共用某些 CloudHSM 資源。 AWS Organizations使用 AWS RAM，您可以透過建立資源共用來共用您擁有的資源。資源共享指定要共用的資源，以及共用它們的消費者。消費者可以包括：

- 特定於其組織 AWS 帳戶 內部或外部 AWS Organizations
- 其組織內部的組織單位 AWS Organizations
- 中的整個組織 AWS Organizations

若要取得有關的更多資訊 AWS RAM，請參閱AWS RAM 使用者指南。

本主題說明如何共用您擁有的資源，以及如何使用與您共用的資源。

目錄

- 取消共用共用備份
- 識別共用備份
- 共用備份的權限
- 計費和計量

共用備份的先決條件

- 要共享備份，您必須在 AWS 帳戶. 這表示必須在您的帳戶中配置或佈建資源。您無法共享已與您共享的備份。
- 若要共用備份，備份必須處於「就緒」狀態。
- 若要與中的組織或組織單位共用備份 AWS Organizations，您必須啟用與共用 AWS Organizations。如需詳細資訊，請參閱《AWS RAM 使用者指南》中的透過 AWS Organizations啟用共用。

共用備份

當您與其他人共用備份時 AWS 帳戶，您可以讓他們從包含儲存在備份中的金鑰和使用者的備份還原叢集。

若要共用備份，您必須將其新增至資源共用。資源共用是一 AWS RAM 種可讓您共用資源的資源 AWS 帳戶。資源共享指定要共用的資源，以及共用它們的消費者。使用 CloudHSM 主控台共用備份時，您可以將其新增至現有的資源共用。若要將備份新增至新的資源共用，您必須先使用AWS RAM 主控台建立資源共用。

如果您是組織中的一員， AWS Organizations 且已啟用組織內的共用功能，則組織中的取用者會自動獲得共用備份的存取權。否則，取用者會收到加入資源共用的邀請，並在接受邀請後授予共用備份的存取權。

您可以使用 AWS RAM 控制台或共享您擁有的備份 AWS CLI。

共用您使用 AWS RAM 主控台擁有的備份

請參閱《AWS RAM 使用者指南》中的建立資源共享。

共用您擁有的備份 (AWS RAM 指令)

使用 create-resource-share 命令。

若要共用您擁有的備份 (CloudHSM 命令)

> **Important**
>
> 雖然您可以使用 CloudHSM PutResourcePolicy 作業共用備份，但建議您改用 AWS Resource Access Manager (AWS RAM)。使用可為您建立原則、允許一次共用多個資源，以及增加共用資源的可探索性，因此可 AWS RAM 提供多項好處。如果您使用 PutResourcePolicy 並希望取用者能夠描述您與他們共用的備份，則必須使用 AWS RAM PromoteResourceShareCreatedFromPolicy API 作業將備份升級為標準 AWS RAM 資源共用。

使用 put-resource-policy 命令。

1. 建立名為的檔案，policy.json並將下列原則複製到其中。

```
{
  "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. policy.json使用備份 ARN 和標識符進行更新以與之共享。下列範例會針對 123456789012 所識別之 AWS 帳戶，授與根使用者的唯讀存取權。

```
{
  "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**
>
> 您只能 DescribeBackups 在帳戶層級授予權限。當您與其他客戶共用備份時，在該帳戶中具有 DescribeBackups 權限的任何主體都可以描述備份。

3. 執行 put-resource-policy 命令。

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

> **Note**
>
> 此時，取用者可以使用備份，但不會顯示在具有共用參數的 DescribeBackups 回應中。接下來的步驟說明如何升級 AWS RAM 資源共用，以便將備份包含在回應中。

4. 獲取資 AWS RAM 源共享 ARN。

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

這會傳回類似下列的回應：

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

```
}
```

從回應中，複製 *< resource-share-arn >* 值以在後續步驟中使用。

5. 執行來 AWS RAM promote-resource-share-created源政策命令。

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

6. 若要驗證資源共用是否已升級，您可以執行 AWS RAM get-resource-shares命令。

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

提升原則後，回應中`featureSet`列出的為`STANDARD`。這也意味著備份可以由策略中的新帳戶描述。

取消共用共用備份

當您取消共用資源時，取用者可能不再使用該資源來還原叢集。消費者仍然可以存取從共用備份還原的任何叢集。

若要取消共用您擁有的共用備份，您必須將其從資源共用中移除。您可以使用 AWS RAM 控制台或 AWS CLI.

取消共用您使用主控台擁有的 AWS RAM 共用備份

請參閱《AWS RAM 使用者指南》中的更新資源共享。

取消共用您擁有的共用備份 (AWS RAM 指令)

使用 disassociate-resource-share 命令。

若要取消共用您擁有的共用備份 (CloudHSM 命令)

使用 delete-resource-policy 命令。

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

識別共用備份

消費者可以使用 CloudHSM 主控台和識別與他們共用的備份。 AWS CLI

使用 CloudHSM 主控台識別與您共用的備份

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 若要變更 AWS 區域，請使用頁面右上角的「地區」選取器。
3. 在導覽窗格中，選擇備份。
4. 在表格中，選擇 [共用備份] 索引標籤。

若要識別與您共用的備份 AWS CLI

使用描述備份命令搭配`--shared`參數，可傳回與您共用的備份。

共用備份的權限

擁有者的許可

Backup 擁有者可以描述和管理共用備份，也可以使用它來還原叢集。

消費者的許可

Backup 用戶無法修改共用備份，但可以描述並使用它來還原叢集。

計費和計量

共用備份不會收取額外費用。

標記 AWS CloudHSM 資源

標籤是指派給 AWS 資源的標籤。您可以將標籤指派給 AWS CloudHSM 叢集。每個標籤皆包含由您定義的標籤索引鍵和標籤值。例如，標籤索引鍵可能是 Cost Center，而標籤值是 12345。每個叢集的標籤索引鍵必須是唯一的。

您可以將標籤用於各種用途。使用標籤來分類和追蹤 AWS 成本是常見的用途之一。您可以套用代表業務類別 (例如成本中心、應用程式名稱或擁有者) 的標籤，來整理多個服務中的成本。將標籤新增至資 AWS 源時， AWS 會產生成本分配報告，其中包含依標籤彙總的使用量和成本。您可以使用此報表來檢視專案或應用模組的 AWS CloudHSM 成本，而不是以單一明細項目的形式檢視所有 AWS CloudHSM 成本。

如需有關使用成本配置標籤的詳細資訊，請參閱《AWS Billing 使用者指南》中的使用成本分配標籤。

您可以使用 AWS CloudHSM 主控台或 AWS 開發套件或命令列工具的其中一個，來新增、更新、列出或移除標籤。

主題

新增或更新標籤

您可以從 AWS CloudHSM 主控台、AWS Command Line Interface (AWS CLI) 或 AWS CloudHSM API 新增或更新標籤。

新增或更新標籤 (主控台)

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 選擇您要新增標籤的叢集。
3. 選擇標籤。
4. 若要新增標籤，請執行以下操作：
    a. 選擇 Edit Tag (編輯標籤)，然後選擇 Add Tag (新增標籤)。
    b. 在 Key (索引鍵) 中，指定標籤的索引鍵。

c. (選用) 在 Value (值) 中，輸入標籤的值。

d. 選擇儲存。

5. 若要更新標籤，請執行以下操作：

a. 選擇 Edit Tag (編輯標籤)。

> **Note**
>
> 如果您更新現有標籤的標籤索引鍵，主控台會刪除現有的標籤，並建立新的標籤。

b. 輸入新的標籤值。

c. 選擇儲存。

新增或更新標籤 (AWS CLI)

1. 在命令提示字元，發出 tag-resource 命令來指定標籤以及您要加上標籤的叢集 ID。如果您不知道叢集 ID，請發出 describe-clusters 命令。

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

2. 若要更新標籤，請使用相同的命令，但指定現有的標籤索引鍵。當您為現有標籤指定新的標籤值，新值會覆寫原標籤。

若要新增或更新標籤 (AWS CloudHSM API)

- 傳送 TagResource 要求。指定標籤以及您要加上標籤的叢集 ID。

列出標籤

您可以從AWS CloudHSM 主控台、或 AWS CloudHSM API 列出叢集的標籤。AWS CLI

列出標籤 (主控台)

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 選擇您要列出其標籤的叢集。
3. 選擇標籤。

列出標籤 (AWS CLI)

- 在命令提示字元，發出 list-tags 命令來指定您要列出其標籤的叢集 ID。如果您不知道叢集 ID，請發出 describe-clusters 命令。

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

若要列出標籤 (AWS CloudHSM API)

- 傳送 ListTags 請求，以便指定您要列出標籤的叢集 ID。

移除標籤

您可以使用 AWS CloudHSM 主控台、AWS CLI 或 AWS CloudHSM API 來移除叢集的標籤。

移除標籤 (主控台)

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 選擇您要移除其標籤的叢集。
3. 選擇標籤。
4. 選擇 Edit Tag (編輯標籤)，然後針對您要移除的標籤，選擇 Remove tag (移除標籤)。
5. 選擇儲存。

移除標籤 (AWS CLI)

- 在命令提示字元，發出 untag-resource 命令來指定要移除之標籤的標籤索引鍵，以及您要移除其標籤的叢集 ID。當您使用移 AWS CLI 除標籤時，請僅指定標籤鍵，而不指定標籤值。

```
$ aws cloudhsmv2 untag-resource --resource-id <cluster ID> \
```

```
--tag-key-list "<tag key>"
```

若要移除標籤 (AWS CloudHSM API)

- 在 AWS CloudHSM API 中傳送UntagResource要求，指定叢集的 ID 以及您要移除的標籤。

管理 HSM 使用者和金鑰 AWS CloudHSM

您必須先在 AWS CloudHSM 叢集中的 HSM 上建立使用者和金鑰，才能使用叢集進行密碼處理。如需管理 AWS CloudHSM中的 HSM 使用者和金鑰的詳細資訊，請參閱下列主題。您也可以了解如何使用規定人數身分驗證 (也稱為 M of N 存取控制)。

主題

- 管理 HSM 使用者 AWS CloudHSM
- 管理金鑰 AWS CloudHSM
- 管理複製的叢集

管理 HSM 使用者 AWS CloudHSM

在中 AWS CloudHSM，您必須使用 CloudHSM CLI 或 CloudHSM 管理公用程式 (CMU) 命令列工具，在 HSM 上建立和管理使用者。CloudHSM CLI 可與最新的 SDK 版本系列搭配使用，而 CMU 則可與先前的 SDK 版本系列搭配使用。

主題

- 使用 CloudHSM CLI 管理 HSM 使用者
- 使用 CloudHSM 管理公用程式 (CMU) 管理 HSM 使用者

使用 CloudHSM CLI 管理 HSM 使用者

使用 CloudHSM CLI 命令列工具，利用最新的 SDK 在 HSM 上建立和管理使用者。

主題

- 了解 HSM 使用者
- HSM 使用者許可表
- 使用 CloudHSM CLI 管理使用者
- 使用 CloudHSM CLI 管理 MFA
- 使用 CloudHSM CLI 管理規定人數身分驗證 (M/N 個存取控制)

了解 HSM 使用者

您在 HSM 上執行的大多數操作，都需要 HSM 使用者的登入資料。HSM 會驗證每個 HSM 使用者，且每個 HSM 使用者都有一種類型，決定您可以該使用者身分在 HSM 上執行哪些操作。

> **Note**
>
> HSM 使用者與 IAM 使用者不同。擁有正確憑證的 IAM 使用者可利用 AWS API 與資源進行互動來建立 HSM。HSM 建立後，您必須使用 HSM 使用者憑證來驗證 HSM 上的操作。

使用者類型

- 未激活的管理員
- 管理員
- 加密使用者 (CU)
- 設備使用者 (AU)

未激活的管理員

在 CloudHSM CLI 中，未激活的管理員是暫時使用者，僅存在於尚未激活的 AWS CloudHSM 叢集中的第一個 HSM 上。如要啟動叢集，請執行 CloudHSM CLI 中的 cluster activate 命令。運行此命令後，系統將提示未激活的管理員更改密碼。更改密碼後，未激活的管理員將成為管理員。

管理員

在 CloudHSM CLI 中，管理員可以執行使用者管理操作。例如，管理員可以建立和刪除使用者，也可以變更使用者密碼。如需管理員的詳細資訊，請參閱 HSM 使用者許可表。

加密使用者 (CU)

加密使用者 (CU) 可以執行以下的金鑰管理和密碼編譯操作。

- 金鑰管理：建立、刪除、共用、匯入和匯出密碼編譯金鑰。
- 密碼編譯操作：使用密碼編譯金鑰進行加密、解密、簽署、驗證及其他操作。

如需更多資訊，請參閱HSM 使用者許可表。

設備使用者 (AU)

設備使用者 (AU) 可以在叢集的 HSM 上執行複製和同步作業。 AWS CloudHSM 使用 AU 來同步 AWS CloudHSM 叢集中的 HSM。AU 存在於由提供的所有 HSM 上 AWS CloudHSM，並具有有限的權限。如需更多資訊，請參閱HSM 使用者許可表。

AWS 無法對您的 HSM 執行任何作業。 AWS 無法檢視或修改您的使用者或金鑰，也無法使用這些金鑰執行任何密碼編譯作業。

HSM 使用者許可表

下表按 HSM 使用者類型或可執行該操作的工作階段依次列出了 HSM 操作。

	管理員	加密使用者 (CU)	設備使用者 (AU)	已驗證的工作階段
取得基本叢集資訊[1]	是	是	是	是
變更自己的密碼	是	是	是	不適用
變更任何使用者的密碼	是	否	否	否
新增、移除使用者	是	否	否	否

	管理員	加密使用者 (CU)	設備使用者 (AU)	已驗證的工作階段
取得同步狀態[2]	 是	 是	 是	 否
擷取、插入遮罩物件[3]	 是	 是	 是	 否
金鑰管理功能[4]	 否	 是	 否	 否
加密、解密	 否	 是	 否	 否
簽署、驗證	 否	 是	 否	 否
產生摘要和 HMAC	 否	 是	 否	 否

- [1] 基本叢集資訊包括叢集的 HSM 數目，以及每個 HSM 的 IP 地址、型號、序號、裝置 ID、韌體 ID 等。

- [2] 使用者可以取得一組摘要 (雜湊)，這組摘要會對應到 HSM 上的金鑰。應用程式可以比較這幾組摘要，以了解叢集中的 HSM 的同步狀態。
- [3] 遮罩物件就是在離開 HSM 之前加密的金鑰。這些金鑰無法在 HSM 外部解密。將金鑰插入 HSM，而且此 HSM 與先前從中擷取金鑰的 HSM 必須在同一個叢集，金鑰才會解密。應用程式可以擷取和插入遮罩物件，以同步叢集的 HSM。
- [4] 金鑰管理功能包括建立、刪除、包裝、取消包裝及修改金鑰的屬性。

使用 CloudHSM CLI 管理使用者

本主題提供有關使用 CloudHSM CLI 管理硬體安全模組 (HSM) 使用者的 step-by-step 指示。如需 CloudHSM CLI 或 HSM 使用者的詳細資訊，請參閱 CloudHSM CLI 和 使用 CloudHSM CLI。

章節

- 使用 CloudHSM CLI 了解 HSM 使用者管理
- 下載 CloudHSM CLI
- 如何使用 CloudHSM CLI 管理 HSM 使用者

使用 CloudHSM CLI 了解 HSM 使用者管理

如要管理 HSM 使用者，您必須使用管理員的使用者名稱和密碼登入 HSM。只有管理員可以管理使用者。HSM 包含名為 admin 的預設管理員。您可以在啟用叢集時設定 admin 管理員的密碼。

如要使用 CloudHSM CLI，您必須使用設定工具來更新本機組態。如需使用 CloudHSM CLI 執行設定工具的說明，請參閱 CloudHSM 命令列介面 (CLI) 入門。-a 參數需要在叢集中新增 HSM 的 IP 地址。如果您有多個 HSM，則可以使用任何 IP 地址。這可確保 CloudHSM CLI 可在整個叢集中傳播您所做的任何變更。請記住，CloudHSM CLI 會使用其本機檔案來追蹤叢集資訊。如果自上次從特定主機使用 CloudHSM CLI 後叢集已發生變更，則必須將這些變更新增至儲存在該主機上的本機組態檔案中。切勿在使用 CloudHSM CLI 時移除 HSM。

取得 HSM (主控台) 的 IP 位址

1. 開啟主 AWS CloudHSM 控台，網址為 https://console.aws.amazon.com/cloudhsm/home。
2. 若要變更 AWS 區域，請使用頁面右上角的區域選擇器。
3. 若要開啟叢集詳細資訊頁面，請在叢集表格中選擇叢集 ID。
4. 若要取得 IP 地址，請在 HSM 索引標籤上，選擇 ENI IP 地址下列出的其中一個 IP 地址。

取得 HSM 的 IP 位址 ()AWS CLI

- 使用 AWS CLI中的 describe-clusters 命令取得 HSM 的 IP 地址。在命令輸出中，HSM 的 IP 地址是 `EniIp` 的值。

```
$  aws cloudhsmv2 describe-clusters


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

下載 CloudHSM CLI

最新版本的 CloudHSM CLI 適用於用戶端 SDK 5 的 HSM 使用者管理任務。若要下載並安裝 CloudHSM CLI，請按照安裝和設定 CloudHSM CLI 中的說明進行操作。

如何使用 CloudHSM CLI 管理 HSM 使用者

本節介紹了使用 CloudHSM CLI 管理 HSM 使用者的基本命令。

> **Note**
>
> 備註：CloudHSM CLI 使用者命令列在 CloudHSM CLI 使用者命令參考中

主題

- 如要建立管理員
- 如要建立加密使用者
- 如要列出叢集中的所有 HSM 使用者

- 如要變更 HSM 使用者密碼
- 如要刪除 HSM 使用者

如要建立管理員

請依照以下步驟來建立管理員。

1. 使用下列命令啟動 CloudHSM CLI 互動式模式。

   Linux

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

   Windows

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

2. 使用 login 命令並以管理員身分登入叢集。

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

3. 系統會提示您輸入密碼。輸入密碼，然後輸出會顯示命令已成功執行。

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

4. 輸入下列命令建立管理員。

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

5. 輸入新使用者的密碼。
6. 重新輸入密碼以確認您輸入的密碼正確無誤。

如要建立加密使用者

請執行下列步驟建立使用者。

1. 使用下列命令啟動 CloudHSM CLI 互動式模式。

   Linux

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

   Windows

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

2. 使用 login 命令並以管理員身分登入叢集。

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

3. 系統會提示您輸入密碼。輸入密碼，然後輸出會顯示命令已成功執行。

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

4. 輸入以下命令來建立加密使用者：

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

5. 輸入新加密使用者的密碼。
6. 重新輸入密碼以確認您輸入的密碼正確無誤。

如要列出叢集中的所有 HSM 使用者

使用 user list 命令列出叢集上的所有使用者。您無需登入即可執行 user list。所有使用者類型均可列出使用者。

請依照下列步驟列出叢集中的所有使用者

1. 使用下列命令啟動 CloudHSM CLI 互動式模式。

   Linux

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

   Windows

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

2. 輸入下列命令列出叢集中的所有使用者：

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

如需 user list 的詳細資訊，請參閱使用者清單。

如要變更 HSM 使用者密碼

使用 user change-password 命令變更密碼。

使用者類型和密碼需區分大小寫，但使用者名稱不區分大小寫。

管理員、加密使用者 (CU) 和設備使用者 (AU) 只能變更自己的密碼。如要變更其他使用者的密碼，您必須以管理員身分登入。您無法變更目前已登入使用者的密碼。

如要變更您自己的密碼

1. 使用下列命令啟動 CloudHSM CLI 互動式模式。

   Linux

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

   Windows

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

2. 使用 login 命令並以要變更密碼的使用者身份登入。

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

3. 輸入使用者的密碼。

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

4. 輸入 user change-password 命令。

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

5. 輸入新密碼。
6. 再次輸入新密碼。

如要變更其他使用者的密碼

1. 使用下列命令啟動 CloudHSM CLI 互動式模式。

   Linux

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

   Windows

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

2. 使用 login 命令並以管理員身分登入。

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

3. 輸入管理員的密碼。

```
Enter password:
{
```

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

4. 輸入 user change-password 命令以及要變更密碼的使用者的使用者名稱。

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

5. 輸入新密碼。
6. 再次輸入新密碼。

如需 user change-password 的詳細資訊，請參閱 user change-password。

如要刪除 HSM 使用者

使用 user delete 刪除使用者。您必須以管理員身分登入才能刪除其他使用者。

> **Tip**
>
> 您無法刪除擁有金鑰的加密使用者 (CU)。

如要刪除使用者

1. 使用下列命令啟動 CloudHSM CLI 互動式模式。

   Linux

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

   Windows

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

2. 使用 login 命令並以管理員身分登入叢集。

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

3. 系統會提示您輸入密碼。輸入密碼，然後輸出會顯示命令已成功執行。

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

4. 使用 user delete 命令來刪除使用者。

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

如需 user delete 的詳細資訊，請參閱 deleteUser。

使用 CloudHSM CLI 管理 MFA

為提高安全性，您可設定多重要素驗證 (MFA)以協助保護叢集。如需詳細資訊，請參閱下列主題。

主題

- 了解 HSM 使用者的 MFA
- 使用適用於 HSM 使用者的 MFA

了解 HSM 使用者的 MFA

當您以啟用 MFA 的硬體服務模組 (HSM) 帳戶登入叢集時，您需向 CloudHSM CLI 提供您的密碼 (第一個要素，這是您已知的內容)，然後 CloudHSM CLI 會提供給您一個權杖並提示您簽署該權杖。

如要提供第二個要素 (您擁有的)，請使用您已經建立並與 HSM 使用者相關聯的金鑰對中的私有金鑰來簽署權杖。如要訪問叢集，請將已簽署的權杖提供給 CloudHSM CLI。

如需為使用者設定 MFA 的詳細資訊，請參閱 為 CloudHSM CLI 設定 MFA

規定人數驗證和 MFA

叢集使用同一金鑰進行規定人數驗證和 MFA。這意味著啟用了 MFA 的使用者可以有效地註冊了 MofN 或規定人數存儲控制。如要為同一 HSM 使用者成功使用 MFA 和規定人數驗證，請考慮下列幾點：

- 如果您今天要為使用者使用規定人數驗證，則應使用您為規定人數使用者建立的同一個金鑰對來為該使用者啟用 MFA。
- 如果您為非規定人數驗證使用者身分的非 MFA 使用者新增 MFA 要求，則您應將該使用者註冊為具有 MFA 驗證的規定人數 (MofN) 註冊使用者。
- 如果您要為兼具規定人數驗證使用者身分的 MFA 使用者移除 MFA 要求或變更其密碼，您也會同時移除規定人數使用者作為規定人數 (MofN) 使用者的註冊。
- 如果您要為兼具規定人數驗證使用者身分的 MFA 使用者移除 MFA 要求或變更其密碼，但仍想保留該使用者的規定人數驗證，則您須將該使用者註冊為規定人數 (MofN) 使用者。

如需規定人數驗證的詳細資訊，請參閱 管理規定人數 (M/N 個)。

使用適用於 HSM 使用者的 MFA

本主題介紹了使用 CloudHSM CLI 管理多重要素驗證 (MFA) 的相關資訊和說明。如需 CloudHSM CLI 的詳細資訊，請參閱 CloudHSM 命令列介面 (CLI)。

主題

- MFA 金鑰對要求
- 為 CloudHSM CLI 設定 MFA
- 建立啟用 MFA 的使用者
- 登入已啟用 MFA 的使用者
- 為已啟用 MFA 的使用者輪換金鑰
- 註冊 MFA 公有金鑰時，為管理員使用者的取消註冊 MFA 公有金鑰
- 權杖檔案參考

如需使用 HSM 使用者的詳細資訊，請參閱 CloudHSM 命令列介面 (CLI)

MFA 金鑰對要求

如要為 HSM 使用者啟用 MFA，您可建立新的金鑰對，或使用符合下列要求的現有金鑰。

- 金鑰類型：非對稱金鑰
- 金鑰用途：簽署和驗證
- 金鑰規格：RSA_2048

- 簽名算法包括：sha256WithRSAEncryption

> **Note**
>
> 如果您正在使用規定人數驗證或計劃使用法定驗證，請參閱 規定人數驗證和 MFA

您可以使用 CloudHSM CLI 和金鑰對來建立已啟用 MFA 的新管理員使用者。

為 CloudHSM CLI 設定 MFA

請依照這些步驟為 CloudHSM CLI 設定 MFA。

1. 如要使用字符簽署策略設定 MFA，您必須先產生 2048 位元的 RSA 私有錦綸和關聯的公有金鑰。

```
$ 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. 使用 CloudHSM CLI 登入您的使用者帳戶。

```
$ 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. 接下來，執行命令以變更 MFA 策略。您必須提供參數 `--token`。此參數指定將寫入未簽署權杖的檔案。

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

```
Enter password:
Confirm password:
```

4. 您現在有一個包含需要簽署但還未簽署權杖的檔案：unsigned-tokens.json。此檔案中的權杖數量取決於叢集中的 HSM 數目。每個權杖代表一個 HSM。這個檔案是 JSON 格式，包含需要簽署以證明您擁有私有金鑰的權杖。

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

5. 接下來，使用步驟 1 中建立的私有金鑰來簽署這些權杖。將簽署的權杖放回檔案中。首先，您必須擷取和解碼 base64 編碼的權杖。

```
$ 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. 現在，您擁有了可以使用步驟 1 中建立的 RSA 私有金鑰進行簽署的二進位權杖。

```
$ 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. 現在，您有了已簽署的二進位權杖。您必須使用 base64 對這些權杖進行編碼，然後將它們放回權杖檔案中。

```
$ 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. 最後，您可以將 base64 值複製並粘貼回權杖檔案中：

```
{
  "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. 現在，您的權杖文件已全部簽署，您可以繼續操作。輸入包含已簽署權杖的檔案名稱，然後按 Enter 鍵。最後，輸入您公有金鑰的路徑。

```
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"
  }
}
```

現在，您已經使用 MFA 設定了您的使用者。

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

```
},
```

建立啟用 MFA 的使用者

請依照下列步驟建立已啟用 MFA 的使用者。

1. 使用 CloudHSM CLI 以管理員身分登入 HSM。
2. 使用 user create 命令建立您選擇的使用者。然後依照 為 CloudHSM CLI 設定 MFA 的步驟為使用者設定 MFA。

登入已啟用 MFA 的使用者

請依照下列步驟登入已啟用 MFA 的使用者。

1. 使用 CloudHSM CLI 中的 login mfa-token-sign 命令，針對已啟用 MFA 的使用者啟動登入程序。

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

2. 輸入您的密碼。然後，系統將提示您輸入包含未簽署/已簽署權杖對的權杖檔案路徑，其中已簽署權杖是使用您私有金鑰產生的權杖。

```
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. 當提示輸入已簽署權杖文件路徑時，您可以在單獨的終端中檢查未簽署的權杖檔案。識別具有需要簽署但尚未權杖的檔案：unsigned-tokens.json。此檔案中的權杖數量取決於叢集中的 HSM 數目。每個權杖代表一個 HSM。這個檔案是 JSON 格式，包含需要簽署以證明您擁有私有金鑰的權杖。

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

```
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

4. 使用在步驟 2 中建立的私有金鑰簽署尚未簽署的權杖。首先，您必須擷取和解碼 base64 編碼的權杖。

```
$ 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. 您現在有二進制權杖。使用您先前在 MFA 設定步驟 1 中建立的 RSA 私有金鑰來簽署這些權杖。

```
$ 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. 現在，您有了已簽署的二進位權杖。使用 base64 對這些權杖進行編碼，然後將它們放回權杖檔案中。

```
$ 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. 最後，將 base64 值複製並粘貼回權杖檔案中：

```
{
  "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. 現在，您的權杖文件已全部簽署，您可以繼續操作。輸入包含已簽署權杖的檔案名稱，然後按 Enter 鍵。您現在應該成功登錄。

```
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>"
  }
}
```

為已啟用 MFA 的使用者輪換金鑰

請依照下列步驟登入已啟用 MFA 的使用者輪換金鑰。

<result>

您已使用私有金鑰簽署了產生的 JSON 格式的權杖文件檔案，並註冊了新的 MFA 公有金鑰。
</result>

1. 使用 CloudHSM CLI 以任何管理員或已啟用 MFA 的特定使用者身分登入 HSM (如需詳細資訊，請參閱已啟用 MFA的登入使用者)。
2. 接下來，執行命令以變更 MFA 策略。您必須提供參數 `--token`。此參數指定將寫入未簽署權杖的檔案。

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

3. 識別具有需要簽署但尚未權杖的檔案：`unsigned-tokens.json`。此檔案中的權杖數量取決於叢集中的 HSM 數目。每個權杖代表一個 HSM。這個檔案是 JSON 格式，包含需要簽署以證明您擁有私有金鑰的權杖。這將是新 RSA 公有/私有金鑰對中的新私有金鑰 (您想要用其輪換目前已註冊公有金鑰)。

```
$cat unsigned-tokens.json
{
  "version": "2.0",
  "tokens": [
```

```
    {
      "unsigned": "Vtf/9QOFY45v/E1osvpEMr59JsnP/hLDm4ItOO2vqL8=",
      "signed": ""
    },
    {
      "unsigned": "wVbC0/5IKwjyZK2NBpdFLyI7BiayZ24YcdUdlcxLwZ4=",
      "signed": ""
    },
    {
      "unsigned": "z6aW9RzErJBL5KqFG5h8lhTVt9oLbxppjod0Ebysydw=",
      "signed": ""
    }
  ]
}
```

4. 使用您先前在設置過程中建立的私有金鑰來簽署這些權杖。首先，必須擷取和解碼 base64 編碼的權杖。

```
$ 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. 您現在有二進制權杖。使用您先前在設定過程中建立的 RSA 私有金鑰來簽署這些權杖。

```
$ 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. 現在，您有了已簽署的二進位權杖。使用 base64 對這些權杖進行編碼，然後將它們放回權杖檔案中。

```
$ 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. 最後，將 base64 值複製並粘貼回權杖檔案中：

```
{
  "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. 現在，您的權杖文件已全部簽署，您可以繼續操作。輸入包含已簽署權杖的檔案名稱，然後按 Enter 鍵。最後，輸入您新公有金鑰的路徑。現在，您會看到以下用戶列表部分輸出的內容。

```
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"
  }
}
```

現在，您已經使用 MFA 設定了您的使用者。

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

註冊 MFA 公有金鑰時，為管理員使用者的取消註冊 MFA 公有金鑰

按照這些步驟在註冊 MFA 公有金鑰時，為管理員使用者取消註冊 MFA 公有金鑰。

1. 使用 CloudHSM CLI 以啟用 MFA 的管理員身分登入 HSM。
2. 使用 user change-mfa token-sign 命令為使用者移除 MFA。

```
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"
  }
}
```

權杖檔案參考

註冊 MFA 公有金鑰或嘗試使用 MFA 登入時產生的權杖檔案包含以下內容：

- 權杖：一組以 JSON 物件文字形式表示的 base64 編碼的未簽署/已簽署權杖對。
- 未簽署：一個 base64 編碼和 SHA256 雜湊的權杖。
- 已簽署：使用 RSA 2048 位元私有金鑰對未簽署權杖進行 base64 編碼的已簽署權杖 (簽章)。

```
{
  "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
    }
  ]
}
```

使用 CloudHSM CLI 管理規定人數身分驗證 (M/N 個存取控制)

AWS CloudHSM 叢集中的 HSM 支援仲裁驗證，也稱為 M in N 存取控制。有了規定人數身分驗證，HSM 上沒有任何單一使用者可以在 HSM 上執行由規定人數控制的操作。相對地，必須有最低數量的 HSM 使用者 (至少 2 個) 合作來執行這些操作。有了規定人數身分驗證，您可以透過要求來自一個以上 HSM 使用者的核准來新增一層額外的保護。

規定人數身分驗證可以控制以下操作：

- 由管理員管理的 HSM 使用者：會建立和刪除 HSM 使用者，以及變更不同 HSM 使用者的密碼。如需詳細資訊，請參閱 將規定人數身分驗證用於管理員。

下列主題提供 AWS CloudHSM中的規定人數身分驗證的詳細資訊。

主題

- 使用權杖簽署策略的規定人數身分驗證概觀
- 關於規定人數身分驗證的其他詳細資訊
- 支援規定人數身分驗證的服務名稱和類型
- 對管理員使用規定人數身分驗證：首次設定
- 將規定人數身分驗證用於管理員
- 變更管理員的規定人數最小值

使用權杖簽署策略的規定人數身分驗證概觀

下列步驟彙總規定人數身分驗證程序。如需特定步驟和工具，請參閱 將規定人數身分驗證用於管理員。

1. 每個 HSM 使用者會建立非對稱金鑰供簽署使用。使用者會在 HSM 外部執行此動作，以適當方式保護金鑰。
2. 每個 HSM 使用者會登入 HSM，並向 HSM 註冊使用者的簽署金鑰 (公有金鑰) 的公有部分。

3. HSM 使用者想要執行由規定人數控制的操作時，使用者會登入 HSM，並取得規定人數權杖。
4. HSM 使用者會將規定人數字符提供給一或多個其他 HSM 使用者，並要求其核准。
5. 其他 HSM 使用者透過使用自己的金鑰以密碼編譯方式簽署規定人數字符來進行核准。這是在 HSM 外部進行。
6. 當 HSM 使用者具有所需的核准數目時，相同的使用者會登入 HSM 並使用 --approval 引數執行規定人數控制的操作，並提供已簽署的規定人數權杖檔案，其中包含所有必要的核准 (簽章)。
7. HSM 會使用每個簽署者註冊的公有金鑰來驗證簽章。如果簽章有效，HSM 會核准權杖，並執行規定人數控制的操作。

關於規定人數身分驗證的其他詳細資訊

請注意關於在 AWS CloudHSM中使用規定人數身分驗證的下列額外資訊。

- HSM 使用者可以簽署自己的規定人數權杖—也就是說，要求的使用者可以針對規定人數身分驗證提供其中一個需要的核准。
- 您可以針對由規定人數控制的操作選擇最低數量的規定人數核准者。您可以選擇的最小數字是二 (2)，您可以選擇的最大數字是八 (8)。
- HSM 最多可存放最多 1024 個規定人數字符。當您嘗試建立新的字符時，如果 HSM 已經有 1024 個字符，HSM 會清除其中一個過期的字符。在預設情況下，字符會在建立後的十分鐘後過期。
- 如果已啟用 MFA，叢集會使用相同的金鑰進行規定人數身分驗證和多重要素驗證 (MFA)。如需使用規定人數身分驗證和 2FA 的詳細資訊，請參閱使用 CloudHSM CLI 管理 MFA。
- 每個 HSM 一次只能包含一個服務的權杖。

支援規定人數身分驗證的服務名稱和類型

管理員服務：規定人數身分驗證用於管理員特殊權限服務，例如建立使用者、刪除使用者、變更使用者密碼、設定規定人數值，以及停用規定人數和 MFA 功能。

每種服務類型都會進一步細分為合格的服務名稱，這包含一組特定的可執行法定人數支援的服務操作。

服務名稱	服務類型	服務操作
使用者	管理員	• 建立使用者 • 刪除使用者 • 變更使用者密碼

服務名稱	服務類型	服務操作
		• 變更使用者 MFA
規定人數	管理員	• 法定令牌符號 set-quorum-value

對管理員使用規定人數身分驗證：首次設定

下列主題說明設定硬體安全模組 (HSM) 必須完成的步驟，以讓管理員可以使用規定人數身分驗證。第一次針對管理員設定規定人數身分驗證時，您只需要執行一次下列步驟。完成這些步驟之後，請參閱將規定人數身分驗證用於管理員。

主題

- 必要條件
- 建立和註冊用於簽署的金鑰
- 設定 HSM 上的規定人數最小值

必要條件

若要了解此範例，您應該熟悉 CloudHSM CLI。在此範例中， AWS CloudHSM 叢集有兩個 HSM，每個 HSM 都有相同的管理員，如下列user list命令輸出所示。如需有關建立使用者的詳細資訊，請參閱使用 CloudHSM CLI。

```
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"
      }
    ]
  }
}
```

建立和註冊用於簽署的金鑰

若要使用規定人數身分驗證，每個管理員都必須完成下列所有步驟：

主題

- 建立 RSA 金鑰對
- 建立並簽署註冊權杖

- 用 HSM 註冊公有金鑰

建立 RSA 金鑰對

建立和保護金鑰對有許多不同的方式。下列範例示範使用 OpenSSL 的做法。

Example – 使用 OpenSSL 建立私有金鑰

以下範例示範如何使用 OpenSSL 來建立受到密碼短語保護的 2048 位元 RSA 金鑰。若要使用這個範例，請將 *<admin.key>* 以您要存放金鑰的檔案名稱來加以取代。

```
$ 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:
```

接下來，使用您剛建立的私有金鑰來產生公有金鑰。

Example – 使用 OpenSSL 建立一個公有金鑰

下列範例會示範如何使用 OpenSSL 根據您剛建立的私有金鑰建立公有金鑰。

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

建立並簽署註冊權杖

您可以建立一個權杖並使用在上一步中剛產生的私有金鑰簽署該權杖。

Example – 建立一個註冊權杖

1. 使用以下命令來啟動 CloudHSM CLI：

   Linux

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

Windows

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

2. 透過執行產生規定人數權杖簽署命令建立註冊權杖：

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

3. 產生規定人數權杖簽署命令在指定的檔案路徑產生註冊權杖。檢查權杖檔案：

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

權杖檔案由以下項目組成：

- approval_data：一個 base64 編碼的隨機資料權杖，其原始資料不超過 245 個位元組的最大值。
- unsigned：核准資料的 base64 編碼和 SHA256 雜湊權杖。
- signed：未簽署的權杖的 base64 編碼簽名權杖 (簽名)，使用先前使用 OpenSSL 產生的 RSA 2048 位元私有金鑰。

您可以使用私有金鑰簽署未簽署的權杖，以證明您可以訪問私有金鑰。您需要將註冊 Token 檔案完全填入簽章和公開金鑰，才能將管理員註冊為 AWS CloudHSM 叢集的法定使用者。

Example — 簽署未簽署的註冊權杖

1. 解碼 base64 編碼的未簽署權杖並將其放入二進位檔案中：

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

2. 使用 OpenSSL 和私有金鑰來簽署現在的二進位未簽署註冊權杖，並建立一個二進位簽署檔案：

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

3. 將二進位簽章編碼為 base64：

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

4. 將 base64 編碼的簽名複製並粘貼到權杖檔案中：

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

用 HSM 註冊公有金鑰

創建密鑰後，管理員必須在 AWS CloudHSM 集群中註冊公鑰。

向 HSM 註冊公有金鑰

1. 使用下列命令來啟動 CloudHSM CLI：

Linux

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

Windows

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

2. 使用 CloudHSM CLI，以管理員身分登入。

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

3. 使用 變更規定人數使用者 註冊字符簽署 命令來註冊公有金鑰。如需詳細資訊，請參閱下列範例或使用 help user change-quorum token-sign register 命令。

Example — 註冊 AWS CloudHSM 集群的公鑰

下列範例顯示如何在 CloudHSM CLI 中使用 user change-quorum token-sign register 命令，以向 HSM 註冊管理員的公有金鑰。若要使用此命令，管理員必須登入 HSM。以您自己的值取代這些值：

```
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：公有金鑰 PEM 檔案的檔案路徑
> 必要：是
> /path/tokenfile：帶有權杖由用戶私有金鑰簽名的檔案路徑
> 必要：是

在所有管理員註冊其公有金鑰之後，user list 命令的輸出會在規定人數欄位中顯示此資訊，說明使用中已啟用的規定人數策略，如下所示：

```
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"
      }
    ]
  }
}
```

設定 HSM 上的規定人數最小值

若要使用規定人數身分驗證，管理員必須登入 HSM，然後設定規定人數最小值。這是執行 HSM 使用者管理操作所需的管理員核准數下限。HSM 上的任何管理員 (包括尚未註冊用於簽署的金鑰的管理員) 可以設定規定人數最小值。您隨時可以變更規定人數最小值。如需更多資訊，請參閱 變更最小值。

設定 HSM 上的規定人數最小值

1. 使用下列命令來啟動 CloudHSM CLI：

   Linux

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

   Windows

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

2. 使用 CloudHSM CLI，以管理員身分登入。

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

3. 使用 法定令牌符號 set-quorum-value 命令來設定規定人數最小值。如需詳細資訊，請參閱下列範例或使用 help quorum token-sign set-quorum-value 命令。

Example – 設定 HSM 上的規定人數最小值

此範例使用值為 2 的規定人數最小值。您可以選擇二 (2) 到八 (8) 之間的任何值，最多可到 HSM 上的管理員總數。在此範例中，HSM 有四 (4) 個管理員，因此最大可能值為四 (4) 個。

若要使用以下範例命令，請將最終數字 (<2>) 以您偏好的規定人數最小值加以取代。

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

在上述範例中，服務識別您要設定的規定人數最小值的 HSM 服務。法定令牌符號 list-quorum-values 會列出服務中包含的 HSM 服務類型、名稱和說明。

管理員服務：規定人數身分驗證用於管理員特殊權限服務，例如建立使用者、刪除使用者、變更使用者密碼、設定規定人數值，以及停用規定人數和 MFA 功能。

每種服務類型都會進一步細分為合格的服務名稱，這包含一組特定的可執行法定人數支援的服務操作。

服務名稱	服務類型	服務操作
使用者	管理員	• 建立使用者 • 刪除使用者 • 變更使用者密碼 • 變更使用者 MFA
規定人數	管理員	• 法定令牌符號 set-quorum-value

使用 quorum token-sign list-quorum-values 命令取得服務的規定人數最小值。

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

上述 quorum token-sign list-quorum-values 命令的輸出顯示 HSM 使用者服務 (負責使用者管理操作) 的規定人數最小值現在是 2。完成這些步驟之後，請參閱 使用規定人數身分驗證 (M/N)。

將規定人數身分驗證用於管理員

HSM 上的管理員可以針對 AWS CloudHSM 叢集中的下列作業配置法定驗證：

- 建立使用者
- 刪除使用者

- 變更使用者密碼
- 變更使用者 MFA

設定 AWS CloudHSM 叢集進行法定驗證後，管理員無法自行執行 HSM 使用者管理作業。以下範例示範當管理員嘗試在 HSM 上建立新使用者時的輸出。命令失敗並顯示錯誤，指出需要進行規定人數身分驗證。

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

若要執行 HSM 使用者管理操作，管理員必須完成下列工作：

主題

- 取得規定人數字符
- 取得核准管理員的簽章
- 核准 AWS CloudHSM 叢集上的權杖並執行使用者管理作業

取得規定人數字符

首先，管理員必須使用 CloudHSM CLI 來請求規定人數權杖。

取得規定人數權杖

1. 使用下列命令來啟動 CloudHSM CLI。

   Linux

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

   Windows

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

2. 使用 login 命令並以管理員身分登入叢集。

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

3. 使用 quorum token-sign generate 命令來產生規定人數權杖。如需詳細資訊，請參閱下列範例或使用 help quorum token-sign generate 命令。

Example – 產生規定人數權杖

此範例會取得使用者名稱為 admin 的管理員的規定人數權杖並將權杖儲存至名為 admin.token 的檔案中。若要使用範例命令，請將這些值取代為您自己的值：

- *<admin>*：取得權杖的管理員名稱。此使用者必須是登入 HSM 的相同管理員並正在執行此命令。
- *<admin.token>*：用於存放規定人數權杖的檔案名稱。

在以下命令中，user 會識別您可以對其使用要生成權杖的服務名稱。在此情況下，此權杖可用於 HSM 使用者管理操作 (user 服務)。

```
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 命令會在指定的檔案路徑產生使用者服務規定人數權杖。可以檢查權杖文件：

```
$cat </path/admin.token>
{
  "version": "2.0",
```

```
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW/
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAABAAMAAAAUA
+b23gAAAAAAAAAA",
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": []
}
```

權杖檔案由以下項目組成：

- approval_data：由 HSM 產生的 base64 編碼原始資料權杖。
- token：approval_data 的 base64 編碼和 SHA-256 雜湊權杖
- signatures：base64 編碼的簽名權杖 (簽名) 陣列，其中核准者的每個簽名均採用 JSON 物件常值的形式：

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

  每個簽章均使用其對應的 RSA 2048 位元私有金鑰 (其公有金鑰已在 HSM 中註冊)，根據核准者的結果建立。

執行 quorum token-sign list 命令，可確認產生的使用者服務規定人數權杖在於 CloudHSM 叢集中是否存在：

```
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` 時間表示過期前需核准的所產生權杖的超時期 (以秒為單位)。

取得核准管理員的簽章

具有規定人數權杖的管理員必須取得其他管理員核准的權杖。為了提供核准，其他管理員會使用其簽署金鑰來以密碼編譯形式簽署權杖。他們會在 HSM 外部執行此操作。

簽署權杖的方式有很多。下列範例示範使用 OpenSSL 的做法。若要使用不同的簽署工具，請確認工具使用管理員的私有金鑰 (簽署金鑰) 來簽署權杖的 SHA-256 摘要。

Example 取得核准管理員的簽章

在這個範例中，具有權杖 (admin) 的管理員需要至少兩 (2) 個核准。以下範例命令示範兩 (2) 個管理員如何使用 OpenSSL 來以密碼編譯方式簽署權杖。

1. 解碼 base64 編碼的未簽署權杖並將其放入二進位檔案中：

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

2. 使用 OpenSSL 和核准者 (admin3) 各自的私有金鑰來簽署使用者服務的現在二進位規定人數未簽署的權杖，並建立二進位簽章檔案：

```
$openssl pkeyutl -sign \
-inkey admin3.key \
-pkeyopt digest:sha256 \
-keyform PEM \
-in admin.bin \
-out admin.sig.bin
```

3. 將二進位簽章編碼為 base64：

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

4. 最後，將 base64 編碼的簽名複製並粘貼到權杖檔案中，遵循先前為核准者簽名指定的 JSON 物件常值格式：

```
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW
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=="
    }
  ]
}
```

核准 AWS CloudHSM 叢集上的權杖並執行使用者管理作業

管理員擁有必要的核准/簽名後 (詳見上一節)，管理員可以將該權杖提供給 AWS CloudHSM 叢集，以及下列其中一項使用者管理操作：

- 建立
- 刪除
- 變更密碼
- user change-mfa

如需使用這些命令的詳細資訊，請參閱使用 CloudHSM CLI。

在交易期間，Token 將在 AWS CloudHSM 叢集內核准，並執行要求的使用者管理作業。使用者管理操作的成功取決於有效的已核准規定人數權杖和有效的使用者管理操作。

管理員僅可以使用該權杖來進行一個操作。當該操作成功，該字符即不再有效。若要執行其他 HSM 使用者管理操作，管理員必須重複上述程序。即：管理員必須產生新的規定人數權杖、向核准者取得新簽章，在 HSM 上核准新權杖並執行要求的使用者管理操作。

> **Note**
>
> 規定人數權杖僅在您目前的登入工作階段開啟時才有效。如果您登出 CloudHSM CLI 或網路中斷連線，則該權杖將不再有效。同樣地，授權的權杖只能在 CloudHSM CLI 中使用。其不能用於其他應用程式中進行身分驗證。

Example 以管理員身分建立新使用者

在下列範例命令中，登入的管理員會在 HSM 上建立新使用者。

```
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"
  }
}
```

然後，管理員輸入 user list 命令以確認新使用者的建立：

```
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"
      }
    ]
  }
}
```

如果管理員嘗試執行另一個 HSM 使用者管理操作，會因規定人數身分驗證錯誤而失敗，如以下範例所示。

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

如下所示，quorum token-sign list 命令顯示管理員沒有核准的權杖。若要執行另一個 HSM 使用者管理操作，管理員必須產生新的規定人數權杖、向核准者取得新簽章，並使用核准引數執行所需的使用者管理操作，以提供要在執行使用者管理操作期間核准和使用的規定人數權杖。

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

```
  }
}
```

變更管理員的規定人數最小值

當您為了讓管理員可以使用規定人數身分驗證而設定規定人數最小值之後，您可能想要變更規定人數最小值。只有當核准者數目等於或大於目前規定人數最小值時，HSM 才允許您變更規定人數最小值。例如，如果規定人數最小值是二 (2)，則至少兩 (2) 個管理員必須核准，才能變更規定人數最小值。

> **Note**
>
> 使用者服務的規定人數值必須永遠小於規定人數服務的規定人數值。如需關於服務名稱的資訊，例如規定人數服務和使用者服務，請參閱 支援規定人數身分驗證的服務名稱和類型。

若要取得規定人數核准來變更規定人數最小值，您需要規定人數權杖來用於使用 quorum token-sign set-quorum-value 命令的 quorum service。若要為使用 quorum token-sign set-quorum-value 命令的 quorum service 產生規定人數權杖，規定人數服務必須大於一 (1)。這表示您可能需要變更規定人數服務 的規定人數最小值才能變更使用者服務的規定人數最小值。

若要變更管理員的規定人數最小值

1. 使用下列命令啟動 CloudHSM CLI 互動式模式。

   Linux

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

   Windows

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

2. 使用 login 命令並以管理員身分登入叢集。

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

3. 使用 quorum token-sign list-quorum-values 命令取得所有服務名稱的規定人數最小值。如需詳細資訊，請參閱下列範例。

4. 如果規定人數服務的規定人數最小值低於使用者服務的值，請使用 quorum token-sign set-quorum-value 命令來變更規定人數服務的值。將規定人數服務的值變更為等於或大於使用者服務的值的一 (1)。如需詳細資訊，請參閱下列範例。
5. 產生規定人數權杖，且應該指定規定人數服務作為您可將權杖用於其中的服務。
6. 向其他管理員取得核准 (簽章)。
7. 核准 AWS CloudHSM 叢集上的權杖並執行使用者管理作業。 。
8. 使用 quorum token-sign set-quorum-value 命令變更使用者服務的規定人數最小值。

Example – 取得規定人數最小值並變更規定人數服務值

以下範例命令顯示使用者服務的規定人數最小值目前是二 (2)。

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

若要變更規定人數服務的規定人數最小值，請使用 quorum token-sign set-quorum-value 命令，並設定等於或大於使用者服務之值的值。以下範例將規定人數服務的規定人數最小值設為二 (2)，與為使用者服務設定的值相同。

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

以下命令顯示使用者服務和規定人數服務的規定人數最小值現在是二 (2)。

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

使用 CloudHSM 管理公用程式 (CMU) 管理 HSM 使用者

在中 AWS CloudHSM，您必須使用 CloudHSM CLI 或 CloudHSM 管理公用程式 (CMU) 命令列工具，在 HSM 上建立和管理使用者。CloudHSM CLI 可與最新的 SDK 版本系列搭配使用，而 CMU 則可與先前的 SDK 版本系列搭配使用。

主題

了解 HSM 使用者

您在 HSM 上執行的大多數操作，都需要 HSM 使用者的登入資料。HSM 會驗證每個 HSM 使用者，且每個 HSM 使用者都有一種類型，決定您可以該使用者身分在 HSM 上執行哪些操作。

> **Note**
>
> HSM 使用者與 IAM 使用者不同。擁有正確憑證的 IAM 使用者可利用 AWS API 與資源進行互動來建立 HSM。HSM 建立後，您必須使用 HSM 使用者憑證來驗證 HSM 上的操作。

使用者類型

- 準加密員 (PRECO)
- 加密管理員 (CO)
- 加密使用者 (CU)
- 設備使用者 (AU)

準加密員 (PRECO)

在雲端管理公用程式 (CMU) 和金鑰管理公用程式 (KMU) 中，PRECO 是暫時使用者，僅存在於 AWS CloudHSM 叢集中的第一個 HSM 上。新叢集中的第一個 HSM 包含一個 PRECO 使用者，這表示此叢集尚未啟用。如要啟用叢集，請執行 cloudhsm-cli 並執行 cluster activate 命令，然後登入 HSM 並變更 PRECO 的密碼。當您變更密碼時，此使用者會變成加密管理員 (CO)。

加密管理員 (CO)

在雲端管理公用程式 (CMU) 和金鑰管理公用程式 (KMU) 中，加密管理員 (CO) 可以執行使用者管理操作。例如，管理員可以建立和刪除使用者，也可以變更使用者密碼。如需有關加密使用者的詳細資訊，請參閱 HSM 使用者許可表。當您啟用新叢集時，使用者會由準加密員 (PRECO) 變成加密管理員 (CO)。-->

加密使用者 (CU)

加密使用者 (CU) 可以執行以下的金鑰管理和密碼編譯操作。

- 金鑰管理：建立、刪除、共用、匯入和匯出密碼編譯金鑰。
- 密碼編譯操作：使用密碼編譯金鑰進行加密、解密、簽署、驗證及其他操作。

如需更多資訊，請參閱HSM 使用者許可表。

設備使用者 (AU)

設備使用者 (AU) 可以在叢集的 HSM 上執行複製和同步作業。 AWS CloudHSM 使用 AU 來同步 AWS CloudHSM 叢集中的 HSM。AU 存在於由提供的所有 HSM 上 AWS CloudHSM，並具有有限的權限。如需更多資訊，請參閱HSM 使用者許可表。

AWS 無法對您的 HSM 執行任何作業。 AWS 無法檢視或修改您的使用者或金鑰，也無法使用這些金鑰執行任何密碼編譯作業。

HSM 使用者許可表

下表按 HSM 使用者類型或可執行該操作的工作階段依次列出了 HSM 操作。

	加密管理員 (CO)	加密使用者 (CU)	設備使用者 (AU)	已驗證的工作階段
取得基本叢集資訊[1]	是	是	是	是

	加密管理員 (CO)	加密使用者 (CU)	設備使用者 (AU)	已驗證的工作階段
變更自己的密碼	 是	 是	 是	不適用
變更任何使用者的密碼	 是	 否	 否	 否
新增、移除使用者	 是	 否	 否	 否
取得同步狀態[2]	 是	 是	 是	 否
擷取、插入遮罩物件[3]	 是	 是	 是	 否
金鑰管理功能[4]	 否	 是	 否	 否
加密、解密	 否	 是	 否	 否

	加密管理員 (CO)	加密使用者 (CU)	設備使用者 (AU)	已驗證的工作階段
簽署、驗證	 否	 是	 否	 否
產生摘要和 HMAC	 否	 是	 否	 否

- [1] 基本叢集資訊包括叢集的 HSM 數目，以及每個 HSM 的 IP 地址、型號、序號、裝置 ID、韌體 ID 等。
- [2] 使用者可以取得一組摘要 (雜湊)，這組摘要會對應到 HSM 上的金鑰。應用程式可以比較這幾組摘要，以了解叢集中的 HSM 的同步狀態。
- [3] 遮罩物件就是在離開 HSM 之前加密的金鑰。這些金鑰無法在 HSM 外部解密。將金鑰插入 HSM，而且此 HSM 與先前從中擷取金鑰的 HSM 必須在同一個叢集，金鑰才會解密。應用程式可以擷取和插入遮罩物件，以同步叢集的 HSM。
- [4] 金鑰管理功能包括建立、刪除、包裝、取消包裝及修改金鑰的屬性。

使用 CloudHSM 管理公用程式 (CMU) 來管理使用者。

本主題提供有關使用 CloudHSM 管理公用程式 (CMU) (用戶端 SDK 隨附的命令列工具) 管理硬體安全性模組 (HSM) 使用者的 step-by-step 指示。如需 CMU 或 HSM 使用者的詳細資訊，請參閱 CloudHSM 管理公用程式 和 了解 HSM 使用者。

章節

- 使用 CMU 了解 HSM 使用者管理
- 下載 CloudHSM 管理公用程式
- 如何使用 CMU 管理 HSM 使用者

使用 CMU 了解 HSM 使用者管理

如要管理 HSM 使用者，您須以加密管理員 (CO) 的使用者名稱和密碼登入 HSM。只有 CO 可以管理使用者。HSM 包含名為 admin 的預設 CO。您可以在啟用叢集時設定 admin 管理員的密碼。

如要使用 CMU，您必須使用設定工具來更新本機組態。CMU 會建立自己的與叢集的連接，且該連接不支援叢集。為追蹤叢集資訊，CMU 會維護本機組態檔案。這表示每次使用 CMU 時，您都應先透過執行帶有 `--cmu` 參數的設定命令列工具來更新組態檔案。如果您使用的是用戶端 SDK 3.2.1 或更早版本，則必須將 `--cmu` 換成其他參數。如需詳細資訊，請參閱 the section called “將 CMU 與用戶端 SDK 3.2.1 及更早版本搭配使用”。

`--cmu` 參數需要在叢集中新增 HSM 的 IP 地址。如果您有多個 HSM，則可以使用任何 IP 地址。這可確保 CMU 可在整個叢集中傳播您所做的任何變更。請記住，CMU 會使用其本機檔案來追蹤叢集資訊。如果自上次從特定主機使用 CMU 後叢集已發生變更，則必須將這些變更新增至儲存在該主機上的本機組態檔案中。切勿在使用 CMU 時新增或移除 HSM。

取得 HSM (主控台) 的 IP 位址

1. 開啟主 AWS CloudHSM 控台，網址為 https://console.aws.amazon.com/cloudhsm/home。
2. 若要變更 AWS 區域，請使用頁面右上角的區域選擇器。
3. 若要開啟叢集詳細資訊頁面，請在叢集表格中選擇叢集 ID。
4. 若要取得 IP 地址，請在 HSM 索引標籤上，選擇 ENI IP 地址下列出的其中一個 IP 地址。

取得 HSM 的 IP 位址 ()AWS CLI

- 使用 AWS CLI中的 describe-clusters 命令取得 HSM 的 IP 地址。在命令輸出中，HSM 的 IP 地址是 `EniIp` 的值。

```
$  aws cloudhsmv2 describe-clusters


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

```
                {
...
                    "EniIp": "10.0.1.6",
...
```

將 CMU 與用戶端 SDK 3.2.1 及更早版本搭配使用

透過用戶端 SDK 3.3.0， AWS CloudHSM 增加了對--cmu參數的支援，簡化了更新 CMU 組態檔的程序。如果您使用的是用戶端 SDK 3.2.1 或更早版本的 CMU，您必須繼續使用 -a 和 -m 參數來更新組態檔案。如需這些參數的詳細資訊，請參閱設定工具。

下載 CloudHSM 管理公用程式

無論您使用的是用戶端 SDK 5 還是用戶端 SDK 3，HSM 使用者管理任務均可使用最新版本的 CMU。

如要下載並安裝 CMU

- 下載並安裝 CMU。

  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. 下載 CloudHSM 管理公用程式。
2. 以視窗系統管理權限執行 CMU 安裝程式 (AWSCloudHSMManagementUtil-latest.msi)。

Windows Server 2012 R2

1. 下載 CloudHSM 管理公用程式。
2. 以視窗系統管理權限執行 CMU 安裝程式 (AWSCloudHSMManagementUtil-latest.msi)。

Windows Server 2016

1. 下載 CloudHSM 管理公用程式。
2. 以視窗系統管理權限執行 CMU 安裝程式 (AWSCloudHSMManagementUtil-latest.msi)。

如何使用 CMU 管理 HSM 使用者

本節介紹了使用 CMU 管理 HSM 使用者的基本命令。

如要建立 HSM 使用者

使用 createUser 在 HSM 上建立新使用者。您必須以 CO 的身份登入才能建立使用者。

如要建立新的 CO 使用者

1. 使用設定工具來更新 CMU 組態。

   Linux

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

   Windows

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

2. 啟動 CMU。

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. 以 CO 使用者身分登入 HSM。

```
aws-cloudhsm>loginHSM CO admin co12345
```

請確定連線 CMU 清單的數目與叢集中的 HSM 數目相符。如不符，請登出並重新啟動。

4. 使用 createUser 來建立名為 **example_officer** 的 CO 使用者，密碼為 **password1**。

```
aws-cloudhsm>createUser CO example_officer password1
```

CMU 會提示您建立使用者操作。

```
**********************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. 輸入 **y**。

如要建立新的 CU 使用者

1. 使用設定工具來更新 CMU 組態。

Linux

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

Windows

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

2. 啟動 CMU。

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. 以 CO 使用者身分登入 HSM。

```
aws-cloudhsm>loginHSM CO admin co12345
```

請確定連線 CMU 清單的數目與叢集中的 HSM 數目相符。如不符，請登出並重新啟動。

4. 使用 createUser 來建立名為 **example_user** 的 CU 使用者，密碼為 **password1**。

```
aws-cloudhsm>createUser CU example_user password1
```

CMU 會提示您建立使用者操作。

```
*************************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. 輸入 **y**。

如需 createUser 的詳細資訊，請參閱 createUser。

如要列出叢集中的所有 HSM 使用者

使用 listUsers 命令列出叢集上的所有使用者。您無需登入即可執行 listUsers 且所有使用者類型都可以列出使用者。

如要列出叢集中的所有使用者

1. 使用設定工具來更新 CMU 組態。

   Linux

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

   Windows

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

2. 啟動 CMU。

   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. 用 listUsers 來列出叢集上的所有使用者。

   ```
   aws-cloudhsm>listUsers
   ```

   CMU 列出了叢集上的所有使用者。

   ```
   
   ```

```
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
```

如需 listUsers 的詳細資訊，請參閱 listUsers。

如要變更 HSM 使用者密碼

使用 changePswd 變更密碼。

使用者類型和密碼需區分大小寫，但使用者名稱不區分大小寫。

管理員、加密使用者 (CU) 和設備使用者 (AU) 只能變更自己的密碼。如要變更其他使用者的密碼，您必須以 CO 身分登入。您無法變更目前已登入使用者的密碼。

如要變更您自己的密碼

1. 使用設定工具來更新 CMU 組態。

   Linux

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

   Windows

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

2. 啟動 CMU。

   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. 登入 HSM。

   ```
   aws-cloudhsm>loginHSM CO admin co12345
   ```

   請確定連線 CMU 清單的數目與叢集中的 HSM 數目相符。如不符，請登出並重新啟動。

4. 使用 changePswd 變更您自己的密碼。

   ```
   aws-cloudhsm>changePswd CO example_officer <new password>
   ```

   CMU 會提示您變更密碼的操作。

   ```
   
   ************************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. 輸入 **y**。

   CMU 會提示您變更密碼的操作。

```
Changing password for example_officer(CO) on 3 nodes
```

如要變更其他使用者的密碼

1. 使用設定工具來更新 CMU 組態。

   Linux

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

   Windows

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

2. 啟動 CMU。

   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. 以 CO 使用者身分登入 HSM。

```
aws-cloudhsm>loginHSM CO admin co12345
```

請確定連線 CMU 清單的數目與叢集中的 HSM 數目相符。如不符，請登出並重新啟動。

4. 使用 changePswd 變更其他使用者的密碼。

```
aws-cloudhsm>changePswd CU example_user <new password>
```

CMU 會提示您變更密碼的操作。

```
************************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. 輸入 **y**。

CMU 會提示您變更密碼的操作。

```
Changing password for example_user(CU) on 3 nodes
```

如需 changePswd 的詳細資訊，請參閱 changePswd。

如要刪除 HSM 使用者

使用 deleteUser 刪除使用者。您必須以 CO 身分登入才能刪除其他使用者。

> **Tip**
>
> 您無法刪除擁有金鑰的加密使用者 (CU)。

如要刪除使用者

1. 使用設定工具來更新 CMU 組態。

Linux

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

Windows

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

2. 啟動 CMU。

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. 以 CO 使用者身分登入 HSM。

```
aws-cloudhsm>loginHSM CO admin co12345
```

請確定連線 CMU 清單的數目與叢集中的 HSM 數目相符。如不符，請登出並重新啟動。

4. 使用 deleteUser 刪除使用者。

```
aws-cloudhsm>deleteUser CO example_officer
```

CMU 會刪除使用者。

```
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)
```

如需 deleteUser 的詳細資訊，請參閱 deleteUser。

使用 CloudHSM 管理公用程式 (CMU) 管理加密管理員的雙重要素身分驗證 (2FA)

為提高安全性，您可設定雙重要素驗證 (2FA) 以協助保護叢集。您只能為加密管理員 (CO) 啟用 2FA。

> **Note**
>
> 您無法為加密使用者 (CU) 或應用程式啟用 2FA。雙重要素驗證 (2FA) 僅適用於 CO 使用者。

主題

了解 HSM 使用者的 2FA

當您以啟用 2FA 的硬體服務模組 (HSM) 帳戶登入叢集時，您需向 cloudhsm_mgmt_util (CMU) 提供您的密碼 (第一個要素，這是您已知的內容)，然後 CMU 會提供給您一個權杖並提示您簽署該權杖。如要提供第二個要素 (您擁有的)，請使用您已經建立並與 HSM 使用者相關聯的金鑰對中的私有金鑰來簽署權杖。如要訪問叢集，請將已簽署的權杖提供給 CMU。

規定人數驗證和 2FA

叢集使用同一金鑰進行規定人數驗證和雙重要素驗證 (2FA)。這意味著啟用了 2FA 的使用者可以有效地註冊了 M-of-N-access-control (MofN)。如要為同一 HSM 使用者成功使用 2FA 和規定人數驗證，請考慮下列幾點：

- 如果您今天要為使用者使用規定人數驗證，則應使用您為規定人數使用者建立的同一個金鑰對來為該使用者啟用 2FA。
- 如果您為非規定人數驗證使用者身分的非 2FA 使用者新增 2FA 要求，則您應將該使用者註冊為具有 2FA 驗證的 MofN 使用者。
- 如果您要為兼具規定人數驗證使用者身分的 2FA 使用者移除 2FA 要求或變更其密碼，您也會同時移除規定人數使用者作為 MofN 使用者的註冊。
- 如果您要為兼具規定人數驗證使用者身分的 2FA 使用者移除 2FA 要求或變更其密碼，但仍想保留該使用者的規定人數驗證，則您須將該使用者註冊為 MofN 使用者。

如需規定人數驗證的詳細資訊，請參閱 使用 CMU 管理規定人數身分驗證。

使用適用於 HSM 使用者的 2FA

本節說明如何為 HSM 使用者使用 2FA，包括建立 2FA HSM 使用者、輪換金鑰，以及以啟用 2FA 使用者身分登入 HSM。如需使用 HSM 使用者的詳細資訊，請參閱 ???、???、???、??? 和 ???。

建立 2FA 使用者

如要為 HSM 使用者啟用 2FA，請使用符合下列要求的金鑰。

2FA 金鑰對要求

您可建立新的金鑰對，或使用符合下列要求的現有金鑰。

- 金鑰類型：非對稱金鑰
- 金鑰用途：簽署和驗證
- 金鑰規格：RSA_2048
- 簽署演算法包括：
  - `sha256WithRSAEncryption`

> **Note**
>
> 如果您正在使用規定人數驗證或計劃使用法定驗證，請參閱 the section called “規定人數驗證和 2FA”。

使用 CMU 和金鑰對來建立啟用 2FA 的新 CO 使用者。

如建立已啟用 2FA 的 CO 使用者

1. 如在一個終端中，請執行下列步驟：

   a. 訪問您的 HSM 並登入 CloudHSM 管理公用程式：

   ```
   /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

   b. 以 CO 身分登入，並使用下列命令建立含有 2FA 的新使用者 MFA：

   ```
   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. 將上述終端保留在此狀態。請勿按回車鍵或輸入任何檔案名稱。

2. 如在其他終端中，請執行下列步驟：

a. 訪問您的 HSM 並登入 CloudHSM 管理公用程式：

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
```

b. 使用下列指令產生公有/私有金鑰對：

```
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. 執行下列命令以安裝 Json 查詢功能，以便從 AuthData 檔案中擷取摘要：

```
sudo yum install jq
```

d. 如要擷取摘要值，請先在 AuthData 檔案中尋找下列資料：

```
{
  "Version":"1.0",
  "PublicKey":"",
  "Data":[
    {
      "HsmId": <"HSM ID">,
```

```
      "Digest": <"DIGEST">,
      "Signature":""
    }
  ]
}
```

> **Note**
>
> 已獲得的摘要是 base64 編碼的，但如要簽署該摘要，您需要先解碼該文件，然後簽署該文件。以下命令將解碼摘要並將解碼的內容存儲在「digest1.bin」中。
>
> ```
> cat authdata | jq '.Data[0].Digest' | cut -c2- | rev | cut -c2- | rev |
>  base64 -d > digest1.bin
> ```

e. 如要轉換公用金鑰的內容，請添加「\n」並刪除空格，如下所示：

```
-----BEGIN PUBLIC KEY-----\n<PUBLIC KEY>\n-----END PUBLIC KEY-----
```

> **Important**
>
> 以上命令顯示如何在 BEGIN PUBLIC KEY----- 後立即添加「\n」，如何刪除「\n」和公用金鑰第一個字元之間的空格，如何在 -----END PUBLIC KEY 之前添加「\n」，以及如何刪除「\n」和公用金鑰結尾的空格。

這是在 AuthData 文件中已接受的公用金鑰的 PEM 格式。

f. 將公有金鑰 PEM 格式的內容粘貼到 AuthData 文件的公用金鑰部分。

```
vi authdata
```

```
{
  "Version":"1.0",
  "PublicKey":"-----BEGIN PUBLIC KEY-----\n<"PUBLIC KEY">\n-----END PUBLIC
 KEY-----",
  "Data":[
{
  "HsmId":<"HSM ID">,
  "Digest":<"DIGEST">,
```

```
  "Signature":""
}
]
}
```

g. 請使用下列指命簽署權杖：

```
openssl pkeyutl -sign -in digest1.bin -inkey private_key.pem -pkeyopt
 digest:sha256 | base64

Output Expected:
<"THE SIGNATURE">
```

> **Note**
>
> 如以上命令所示，請改用 openssl pkeyutl 而不是 openssl dgst 來簽署。

h. 在「簽名」欄位的 AuthData File 中新增已簽署的摘要。

```
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. 返回到第一個終端，然後按 **Enter**：

```
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)
```

管理適用於 HSM 使用者的 2FA

使用變更密碼來變更 2FA 使用者的密碼，或啟用或停用 2FA，或輪換 2FA 金鑰。每次啟用 2FA 時，您都必須提供 2FA 登入的公用金鑰。

變更密碼的操作場景如下：

- 變更 2FA 使用者的密碼
- 變更 2FA 使用者的密碼
- 為非 2FA 使用者新增 2FA
- 移除 2FA 使用者的 2FA
- 輪換 2FA 使用者的金鑰

您也可以合併任務。例如，您可以移除使用者的 2FA 並同時變更密碼，或輪換 2FA 金鑰並變更使用者密碼。

如要為已啟用 2FA 的 CO 使用者變更密碼或輪換金鑰

1. 使用 CMU 以已啟用 2FA 的 CO 身分登入 HSM。
2. 用 changePswd 變更密碼或向已啟用 2FA 的 CO 使用者輪換金鑰。使用 `-2fa` 參數並在檔案系統中指定一個位置，以便系統寫入 `authdata` 檔案。此檔案包含叢集中每個 HSM 的摘要。

   ```
   aws-cloudhsm>changePswd CO example-user <new-password> -2fa /path/to/authdata
   ```

   CMU 會提示您使用私有金鑰簽署 `authdata` 檔案中的摘要，並傳回含有公有金鑰的簽章。

3. 使用私有金鑰簽署 authdata 檔案中的摘要、將簽章和公有金鑰新增至 JSON 格式的 authdata 檔案，然後將 authdata 檔案位置提供給 CMU。如需詳細資訊，請參閱 the section called “組態參考”。

> **Note**
>
> 叢集使用同一金鑰進行規定人數驗證和雙重要素驗證 (2FA)。如果您正在使用規定人數驗證或計劃使用法定驗證，請參閱 the section called “規定人數驗證和 2FA”。

如要為已啟用 2FA 的 CO 使用者停用 2FA

1. 使用 CMU 以已啟用 2FA 的 CO 身分登入 HSM。
2. 使用 changePswd 從已啟用 2FA 的 CO 使用者中移除 2FA。

```
aws-cloudhsm>changePswd CO example-user <new password>
```

CMU 會提示您確認變更密碼操作。

> **Note**
>
> 如果您要為兼具規定人數驗證使用者身分的 2FA 使用者移除 2FA 要求或變更其密碼，您也會同時移除規定人數使用者作為 MofN 使用者的註冊。如需規定人數使用者和 2FA 的詳細資訊，請參閱 the section called “規定人數驗證和 2FA”。

3. 輸入 **y**。

CMU 會確認變更密碼操作。

組態參考

以下範例顯示了 authdata 檔案中的 2FA 屬性 (包含 CMU 產生的請求和您的回應)。

```
{
    "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=="
        }
    ]
}
```

資料

頂層節點。包含叢集中每個 HSM 的次級節點。出現在所有 2FA 命令的請求和回應中。

摘要

這是您必須簽署後才能提供驗證的第二個要素。在所有 2FA 命令請求中產生 CMU。

HsmId

您 HSM 的識別碼。出現在所有 2FA 命令的請求和回應中。

PublicKey

您所產生金鑰對的公開金鑰部分會作為 PEM 格式字串插入。您要在 createUser 和 changePswd 的回應中輸入此內容。

Signature

Base 64 編碼的已簽署摘要。您要在所有 2FA 回應中輸入此內容。

版本

驗證資料的格式為 JSON 格式檔案。出現在所有 2FA 命令的請求和回應中。

使用 CloudHSM 管理公用程式 (CMU) 管理規定人數身分驗證 (M/N 個存取控制)

AWS CloudHSM 叢集中的 HSM 支援仲裁驗證，也稱為 M in N 存取控制。有了規定人數身分驗證，HSM 上沒有任何單一使用者可以在 HSM 上執行由規定人數控制的操作。相對地，必須有最低數量的 HSM 使用者 (至少 2 個) 合作來執行這些操作。有了規定人數身分驗證，您可以透過要求來自一個以上 HSM 使用者的核准來新增一層額外的保護。

規定人數身分驗證可以控制以下操作：

- 由加密管理員 (CO) 管理的 HSM 使用者：會建立和刪除 HSM 使用者，以及變更不同 HSM 使用者的密碼。如需詳細資訊，請參閱 對加密管理員使用規定人數身分驗證。

下列主題提供 AWS CloudHSM中的規定人數身分驗證的詳細資訊。

主題

- 規定人數身分驗證的概觀
- 關於規定人數身分驗證的其他詳細資訊
- 對加密管理員使用規定人數身分驗證：首次設定
- 對加密管理員使用規定人數身分驗證
- 變更加密管理員的規定人數最小值

規定人數身分驗證的概觀

下列步驟彙總規定人數身分驗證程序。如需特定步驟和工具，請參閱 對加密管理員使用規定人數身分驗證。

1. 每個 HSM 使用者會建立非對稱金鑰供簽署使用。使用者會在 HSM 外部執行此動作，以適當方式保護金鑰。
2. 每個 HSM 使用者會登入 HSM，並向 HSM 註冊使用者的簽署金鑰 (公有金鑰) 的公有部分。
3. HSM 使用者想要執行由規定人數控制的操作時，每個使用者會登入 HSM，並取得規定人數權杖。
4. HSM 使用者會將規定人數字符提供給一或多個其他 HSM 使用者，並要求其核准。
5. 其他 HSM 使用者透過使用自己的金鑰以密碼編譯方式簽署規定人數字符來進行核准。這是在 HSM 外部進行。
6. 當 HSM 使用者擁有需要的核准數量時，相同的使用者會登入 HSM，並向 HSM 提供規定人數權杖和核准 (簽章)。
7. HSM 會使用每個簽署者註冊的公有金鑰來驗證簽章。如果簽章有效，HSM 會核准字符。
8. HSM 使用者現在可以執行由規定人數控制的操作。

關於規定人數身分驗證的其他詳細資訊

請注意關於在 AWS CloudHSM中使用規定人數身分驗證的下列額外資訊。

- HSM 使用者可以簽署自己的規定人數權杖，即：要求的使用者可以針對規定人數身分驗證提供其中一個需要的核准。

- 您可以針對由規定人數控制的操作選擇最低數量的規定人數核准者。您可以選擇的最小數字是二 (2)，您可以選擇的最大數字是八 (8)。
- HSM 最多可存放最多 1024 個規定人數字符。當您嘗試建立新的字符時，如果 HSM 已經有 1024 個字符，HSM 會清除其中一個過期的字符。在預設情況下，字符會在建立後的十分鐘後過期。
- 叢集使用相同的金鑰進行規定人數身分驗證和雙重要素驗證 (2FA)。如需關於使用規定人數身分驗證和 2FA 的詳細資訊，請參閱規定人數身分驗證和 2FA。

對加密管理員使用規定人數身分驗證：首次設定

下列主題說明設定硬體安全模組 (HSM) 必須完成的步驟，以便加密管理員 (CO) 可以使用規定人數身分驗證。第一次針對 CO 設定規定人數身分驗證時，您只需要執行一次下列步驟。完成這些步驟之後，請參閱 對加密管理員使用規定人數身分驗證。

主題

- 必要條件
- 建立和註冊用於簽署的金鑰
- 設定 HSM 上的規定人數最小值

必要條件

若要了解此範例，您應熟悉 cloudhsm_mgmt_util (CMU) 命令列工具。在此範例中， AWS CloudHSM 叢集有兩個 HSM，每個 HSM 具有相同的 CO，如下列listUsers命令輸出所示。如需有關建立使用者的詳細資訊，請參閱管理 HSM 使用者。

```
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
```

建立和註冊用於簽署的金鑰

若要使用規定人數身分驗證，每個管理員都必須執行下列所有步驟：

主題

- 建立 RSA 金鑰對
- 建立並簽署註冊權杖
- 用 HSM 註冊公有金鑰

建立 RSA 金鑰對

建立和保護金鑰對有許多不同的方式。下列範例示範使用 OpenSSL 的做法。

Example – 使用 OpenSSL 建立私有金鑰

以下範例示範如何使用 OpenSSL 來建立受到密碼短語保護的 2048 位元 RSA 金鑰。若要使用這個範例，請將 *officer1.key* 以您要存放金鑰的檔案名稱來加以取代。

```
$ 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:
```

接下來，使用您剛建立的私有金鑰來產生公有金鑰。

Example – 使用 OpenSSL 建立一個公有金鑰

下列範例會示範如何使用 OpenSSL 根據您剛建立的私有金鑰建立公有金鑰。

```
$ openssl rsa -in officer1.key -outform PEM -pubout -out officer1.pub
Enter pass phrase for officer1.key:
writing RSA key
```

建立並簽署註冊權杖

您可以建立一個權杖並使用在上一步中剛產生的私有金鑰簽署該權杖。

Example – 建立權杖

註冊權杖只是一個最大不超過 245 位元組的任何隨機資料的檔案。您可以使用私有金鑰簽署權杖，以證明您可以訪問私有金鑰。下列命令使用 echo 將字串重新導向到一個檔案。

```
$ echo "token to be signed" > officer1.token
```

簽署權杖並將其儲存至簽章檔案。您需要已簽署的權杖、未簽署的權杖和公有金鑰，才能在 HSM 中將 CO 註冊為 MofN 使用者。

Example – 簽署權杖

使用 OpenSSL 和私有金鑰來簽署註冊權杖並建立簽章檔案。

```
$ openssl dgst -sha256 \
    -sign officer1.key \
    -out officer1.token.sig officer1.token
```

用 HSM 註冊公有金鑰

建立金鑰之後，CO 必須向 HSM 註冊金鑰的公有部分 (公有金鑰)。

向 HSM 註冊公有金鑰

1. 使用下列命令來啟動 cloudhsm_mgmt_util 命令列工具。

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

2. 使用 loginHSM 命令以 CO 身分登入 HSM。如需詳細資訊，請參閱 ???。
3. 使用 registerQuorumPubKey 命令來註冊公有金鑰。如需詳細資訊，請參閱下列範例或使用 help registerQuorumPubKey 命令。

Example – 向 HSM 註冊公有金鑰

下列範例顯示如何在 cloudhsm_mgmt_util 命令列工具中使用 registerQuorumPubKey 命令，以向 HSM 註冊 CO 的公有金鑰。若要使用此命令，CO 必須登入 HSM。以您自己的值取代這些值：

```
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>

包含未簽署註冊權杖的檔案路徑。可包含最大 245 個位元組的任何隨機資料。

必要：是

<officer1.token.sig>

包含註冊權杖的 SHA256_PKCS 機制簽名雜湊的檔案路徑。

必要：是

<officer1.pub>

包含非對稱 RSA-2048 金鑰對的公有金鑰的檔案路徑。使用私有金鑰簽署註冊權杖。

必要：是

在所有 CO 註冊其公有金鑰之後，來自 listUsers 命令的輸出會在 MofnPubKey 欄中顯示此資訊，如下列範例所示。

```
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
```

設定 HSM 上的規定人數最小值

若要對 CO 使用規定人數身分驗證，CO 必須登入 HSM，然後設定規定人數最小值，也稱為 m 值。這是執行 HSM 使用者管理操作所需的 CO 核准數下限。HSM 上的任何 CO (包括尚未註冊用於簽署的金鑰的 CO) 可以設定規定人數最小值。您隨時可以變更規定人數最小值。如需更多資訊，請參閱 變更最小值。

設定 HSM 上的規定人數最小值

1. 使用下列命令來啟動 cloudhsm_mgmt_util 命令列工具。

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

2. 使用 loginHSM 命令以 CO 身分登入 HSM。如需詳細資訊，請參閱 ???。
3. 使用 setMValue 命令來設定規定人數最小值。如需詳細資訊，請參閱下列範例或使用 help setMValue 命令。

Example – 設定 HSM 上的規定人數最小值

此範例使用值為 2 的規定人數最小值。您可以選擇二 (2) 到八 (8) 之間的任何值，最多可到 HSM 上的 CO 總數。在這個範例中，HSM 有 6 個 CO，因此最大的可能值為 6。

若要使用以下範例命令，請將最終數字 (*2*) 以您偏好的規定人數最小值加以取代。

```
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
```

在上述範例中，第一個數字 (3) 表示您要設定之規定人數最小值的 HSM 服務。

HSM 服務識別符及其名稱、說明和服務中包含的命令如下表所列。

服務識別符	服務名稱	服務描述	HSM 命令
3	USER_MGMT	HSM 使用者管理	• createUser • deleteUser • changePswd (僅在變更不同 HSM 使用者的密碼時適用)
4	MISC_CO	其他 CO 服務	• setMValue

若要取得服務的規定人數最小值，請使用 getMValue 命令，如下列範例所示。

```
aws-cloudhsm>getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

來自上述 getMValue 命令的輸出顯示 HSM 使用者管理操作 (服務 3) 的規定人數最小值現在是 2。

完成這些步驟之後，請參閱 對加密管理員使用規定人數身分驗證。

對加密管理員使用規定人數身分驗證

HSM 上的加密管理員 (CO) 可以針對 HSM 上的以下操作設定規定人數身分驗證：

- 建立 HSM 使用者
- 刪除 HSM 使用者
- 變更另一個 HSM 使用者的密碼

在將 HSM 設定為使用規定人數身分驗證之後，CO 無法自行執行 HSM 使用者管理操作。以下範例示範當 CO 嘗試在 HSM 上建立新使用者時的輸出。命令失敗，出現 RET_MXN_AUTH_FAILED 錯誤，這表示規定人數身分驗證失敗。

```
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
```

若要執行 HSM 使用者管理操作，CO 必須完成下列工作：

1. 取得規定人數字符。
2. 取得其他 CO 的核准 (簽章)。
3. 在 HSM 上核准字符。
4. 執行 HSM 使用者管理操作。

如果您尚未將 HSM 設定為使用 CO 的規定人數身分驗證，請立即設定。如需詳細資訊，請參閱 首次設定。

取得規定人數字符

首先，CO 必須使用 cloudhsm_mgmt_util 命令列工具，以請求規定人數權杖。

取得規定人數權杖

1. 使用下列命令來啟動 cloudhsm_mgmt_util 命令列工具。

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

2. 使用 loginHSM 命令以 CO 身分登入 HSM。如需詳細資訊，請參閱 ???。

3. 使用 getToken 命令來取得規定人數權杖。如需詳細資訊，請參閱下列範例或使用 help getToken 命令。

Example – 取得規定人數字符

此範例會取得使用者名稱 officer1 之 CO 的規定人數字符，並將字符儲存至名為 `officer1.token` 的檔案。若要使用範例命令，請將這些值取代為您自己的值：

- *`officer1`*：取得權杖的 CO 名稱。此使用者必須是登入到 HSM 的相同 CO 並正在執行此命令。
- *`officer1.token`*：用於存放規定人數權杖的檔案名稱。

在以下命令中，3 會識別您可以對其使用要取得字符的服務。在此情況下，此字符可用於 HSM 使用者管理操作 (服務 3)。如需詳細資訊，請參閱 設定 HSM 上的規定人數最小值。

```
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
```

取得核准 CO 的簽章

具有規定人數字符的 CO 必須取得其他 CO 核准的字符。為了提供核准，其他 CO 會使用其簽署金鑰來以密碼編譯形式簽署字符。他們會在 HSM 外部執行此操作。

簽署權杖的方式有很多。下列範例示範使用 OpenSSL 的做法。若要使用不同的簽署工具，請確定工具使用 CO 的私有金鑰 (簽署金鑰) 來簽署字符的 SHA-256 摘要。

Example – 取得核准 CO 的簽章

在這個範例中，具有字符 (officer1) 的 CO 需要至少兩個核准。以下範例命令示範兩個 CO 如何使用 OpenSSL 來以密碼編譯方式簽署字符。

在第一個命令中，officer1 會簽署自己的字符。若要使用下列範例命令，請將這些值取代為您自己的值：

- *officer1.key* 和 *officer2.key*：包含 CO 簽署金鑰的檔案名稱。
- *officer1.token.sig1* 和 *officer1.token.sig2*：用於存放簽章的檔案名稱。務必將每個簽章儲存在不同的檔案中。
- *officer1.token*：包含 CO 要簽署的權杖的檔案名稱。

```
$ openssl dgst -sha256 -sign officer1.key -out officer1.token.sig1 officer1.token
Enter pass phrase for officer1.key:
```

在以下命令中，officer2 會簽署相同的字符。

```
$ openssl dgst -sha256 -sign officer2.key -out officer1.token.sig2 officer1.token
Enter pass phrase for officer2.key:
```

在 HSM 上核准簽署的權杖

CO 在向其他 CO 取得最低數量的核准 (簽章) 後，必須在 HSM 上核准簽署的字符。

在 HSM 上核准簽署的字符

1. 建立字符核准檔案。如需詳細資訊，請參閱下列範例。
2. 使用下列命令來啟動 cloudhsm_mgmt_util 命令列工具。

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

3. 使用 loginHSM 命令以 CO 身分登入 HSM。如需詳細資訊，請參閱 ???。
4. 使用 approveToken 命令來核准已簽署的字符，以便傳遞字符核准檔案。如需詳細資訊，請參閱下列範例。

Example – 在 HSM 上建立權杖核准檔案和核准簽署的權杖

字符核准檔案是一個文字檔案，採用 HSM 要求的特定格式。該檔案包含有關該字符、其核准者和核准者簽章的資訊。以下範例示範的是範例字符核准檔案。

```
# 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;
```

建立字符核准檔案之後，CO 會使用 cloudhsm_mgmt_util 命令列工具來登入 HSM。接著 CO 會使用 approveToken 命令來核准字符，如下列範例所示。將 *approval.txt* 取代為字符核准檔案的名稱。

```
aws-cloudhsm>approveToken approval.txt
approveToken success on server 0(10.0.2.14)
approveToken success on server 1(10.0.1.4)
```

此命令成功時，HSM 即已核准規定人數字符。若要檢查字符的狀態，請使用 listTokens 命令，如下列範例所示。命令的輸出顯示字符擁有所需的核准數。

字符有效時間指出字符會維持在 HSM 上的保證時間。即使已超過字符有效時間 (零秒)，您仍可以繼續使用該字符。

```
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
```

針對使用者管理操作使用權杖

在 CO 具有所需核准數的字符之後，如先前區段中所示，CO 可以執行以下其中一個 HSM 使用者管理操作：

- 使用 createUser 命令建立 HSM 使用者
- 使用 deleteUser 命令來刪除 HSM 使用者
- 使用 changePswd 命令來變更不同 HSM 使用者的密碼

如需使用這些命令的詳細資訊，請參閱管理 HSM 使用者。

CO 僅可以使用該字符來進行一個操作。當該操作成功，該字符即不再有效。為了執行其他 HSM 使用者管理操作，CO 必須取得新的規定人數字符、向核准者取得新簽章，並在 HSM 上核准該新字符。

> **Note**
>
> MofN 權杖僅在您目前的登入工作階段開啟時才有效。如果您登出 cloudhsm_mgmt_util 或網路連線中斷連線，則該權杖會失效。同樣地，授權的權杖只能在 cloudhsm_mgmt_util 中使用，而不能用於其他應用程式中進行身分驗證。

在下列範例命令中，CO 會在 HSM 上建立新使用者。

```
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
```

先前命令成功之後，後續的 listUsers 命令會顯示新使用者。

```
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
```

如果 CO 嘗試執行其他 HSM 使用者管理操作，會因為規定人數身分驗證錯誤而失敗，如以下範例所示。

```
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
```

listTokens 命令顯示 CO 沒有已核准的憑證，如下列範例所示。為了執行其他 HSM 使用者管理操作，CO 必須取得新的規定人數字符、向核准者取得新的簽章，並在 HSM 上核准該新字符。

```
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
```

變更加密管理員的規定人數最小值

當您為了讓加密管理員 (CO) 可以使用規定人數身分驗證而設定規定人數最小值之後，您可能想要變更規定人數最小值。只有當核准者數目等於或大於目前規定人數最小值時，HSM 才允許您變更規定人數最小值。例如，如果規定人數最小值是 2，則至少兩個 CO 必須核准，才能變更規定人數最小值。

若要取得規定人數核准來變更規定人數最小值，您需要規定人數字符來用於 setMValue 命令 (服務 4)。若要取得 setMValue 命令的規定人數字符 (服務 4)，服務 4 的規定人數最小值必須大於 1。這表示您可能需要變更服務 4 的規定人數最小值，才能變更 CO 的規定人數最小值 (服務 3)。

HSM 服務識別符及其名稱、說明和服務中包含的命令如下表所列。

服務識別符	服務名稱	服務描述	HSM 命令
3	USER_MGMT	HSM 使用者管理	• createUser • deleteUser

服務識別符	服務名稱	服務描述	HSM 命令
			• changePswd (僅在變更不同 HSM 使用者的密碼時適用)
4	MISC_CO	其他 CO 服務	• setMValue

變更加密主管的規定人數最小值

1. 使用下列命令來啟動 cloudhsm_mgmt_util 命令列工具。

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```

2. 使用 loginHSM 命令以 CO 身分登入 HSM。如需詳細資訊，請參閱 ???。
3. 使用 getMValue 命令取得服務 3 的規定人數最小值。如需詳細資訊，請參閱下列範例。
4. 使用 getMValue 命令取得服務 4 的規定人數最小值。如需詳細資訊，請參閱下列範例。
5. 如果服務 4 的規定人數最小值低於服務 3 的值，請使用 setMValue 命令來變更服務 4 的值。將服務 4 的值變更為等於或大於服務 3 的值。如需詳細資訊，請參閱下列範例。
6. 取得規定人數權杖，且應該指定服務 4 做為您可將字符用於其中的服務。
7. 取得其他 CO 的核准 (簽章)。
8. 在 HSM 上核准字符。
9. 使用 setMValue 命令來變更服務 3 (由 CO 執行的使用者管理操作) 的規定人數最小值。

Example – 取得規定人數最小值並變更服務 4 的值

以下範例命令顯示服務 3 的規定人數最小值目前是 2。

```
aws-cloudhsm>getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

以下範例命令顯示服務 4 的規定人數最小值目前是 1。

```
aws-cloudhsm>getMValue 4
MValue of service 4[MISC_CO] on server 0 : [1]
MValue of service 4[MISC_CO] on server 1 : [1]
```

若要變更服務 4 的規定人數最小值，請使用 setMValue 命令，並設定等於或大於服務 3 之值的值。以下範例將服務 4 的規定人數最小值設為 2，與為服務 3 設定的值相同。

```
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
```

以下命令顯示服務 3 和服務 4 的規定人數最小值現在是 2。

```
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]
```

管理金鑰 AWS CloudHSM

在中 AWS CloudHSM，使用下列任何一項來管理叢集中 HSM 的金鑰：

- PKCS #11 程式庫
- JCE 提供者
- CNG 和 KSP 提供者
- CloudHSM CLI

在管理金鑰前，請以加密使用者 (CU) 的使用者名稱和密碼登入 HSM。只有 CU 才可建立金鑰。建立金鑰的 CU 擁有並管理該金鑰。

主題

- 密鑰同步和耐久性設置 AWS CloudHSM

- AES 密鑰包裝在 AWS CloudHSM
- 使用受信任的金鑰 AWS CloudHSM
- 使用 CloudHSM CLI 管理金鑰
- 使用 KMU 和 CMU 管理金鑰

密鑰同步和耐久性設置 AWS CloudHSM

本主題說明中的主要同步處理設定 AWS CloudHSM、客戶在叢集上使用金鑰時所面臨的常見問題，以及製作金鑰更耐用的策略。

主題

- 概念
- 了解金鑰同步
- 使用用戶端金鑰耐久性設定
- 同步複製的叢集之間的金鑰

概念

權杖金鑰

您在金鑰產生、匯入或解除換行作業期間建立的永久金鑰。 AWS CloudHSM 跨集群同步令牌密鑰。

工作階段金鑰

僅存在於叢集中的一個硬體安全性模組 (HSM) 上的暫時金鑰。 AWS CloudHSM 不會跨叢集同步處理工作階段金鑰。

用戶端金鑰同步

一種用戶端程序，可複製您在金鑰產生、匯入或解除包裝作業期間建立的權杖金鑰。您可以透過執行至少兩個 HSM 的叢集，使權杖金鑰更耐久。

伺服器端金鑰同步

定期將金鑰複製到叢集中的每個 HSM。不需要管理。

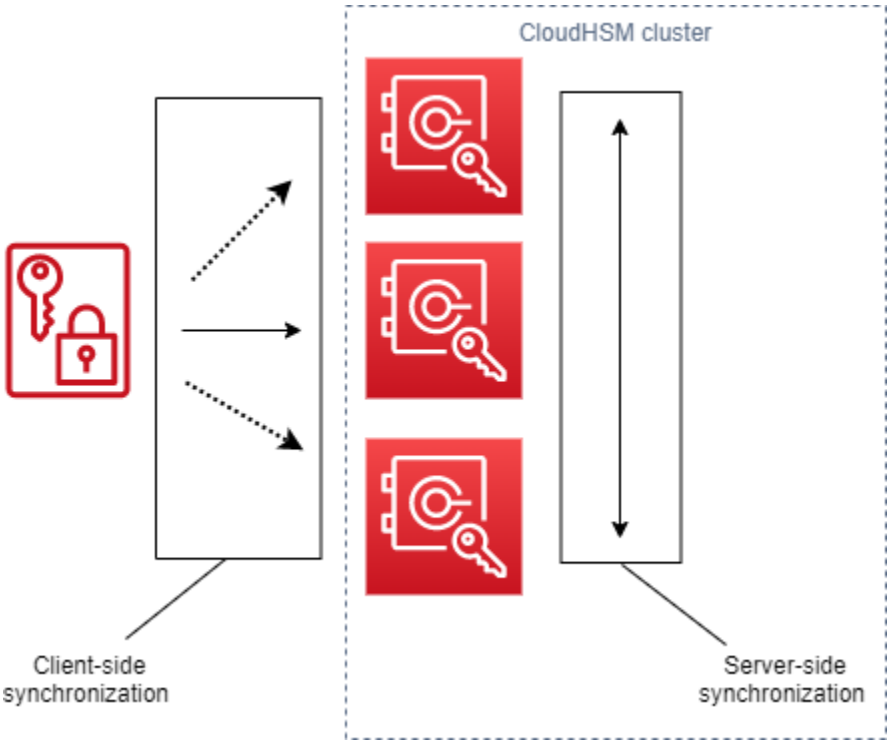
用戶端金鑰耐久性設定

您在用戶端上設定的會影響金鑰耐久性的設定。這些設定在用戶端 SDK 5 和用戶端 SDK 3 中的運作方式不同。

- 在用戶端 SDK 5 中，使用此設定來執行單一 HSM 叢集。
- 在用戶端 SDK 3 中，使用此設定可指定金鑰建立作業成功所需的 HSM 數目。

了解金鑰同步

AWS CloudHSM 使用金鑰同步處理來複製叢集中所有 HSM 的權杖金鑰。您可以在金鑰產生、匯入或解除包裝作業期間，將權杖金鑰建立為持久性金鑰。為了在叢集中分配這些金鑰，CloudHSM 提供用戶端和伺服器端金鑰同步處理。



金鑰同步處理 (伺服器端和用戶端) 的目標是在建立新金鑰之後，儘快在叢集中散發新金鑰。這很重要，因為您後續使用新金鑰所進行的呼叫可以路由傳送至叢集中任何可用的 HSM。如果您在沒有金鑰的情況下路由到 HSM 的呼叫，則呼叫會失敗。您可以指定應用程式重試在金鑰建立作業之後進行的後續呼叫，藉此減輕這些類型的失敗。同步處理所需的時間可能會有所不同，具體取決於叢集的工作負載和其他無形資產。使用 CloudWatch 指標來判斷應用程式在這種情況下應採用的時間。如需詳細資訊，請參閱CloudWatch 量度。

在雲端環境中，金鑰同步處理的挑戰在於金鑰的耐久性。您可以在單一 HSM 上建立金鑰，並且通常會立即開始使用這些金鑰。如果您建立金鑰的 HSM 在將金鑰複製到叢集中的其他 HSM 前失敗，您將遺失金鑰並存取金鑰所加密的任何項目。為了減輕此風險，我們提供用戶端同步處理。用戶端同步處理是用戶端程序，可複製您在金鑰產生、匯入或解除包裝作業期間建立的金鑰。在建立金鑰時複製金鑰會讓金鑰更耐久。當然，您無法使用單一 HSM 複製叢集中的金鑰。為了讓金鑰更耐久，我們也建議您將叢

集設定為至少使用兩個 HSM。透過用戶端同步處理和具有兩個 HSM 的叢集，您可以在雲端環境中應對金鑰耐久性的挑戰。

使用用戶端金鑰耐久性設定

金鑰同步處理大部分是自動程序，但您可以管理用戶端金鑰耐久性設定。用戶端金鑰耐久性設定在用戶端 SDK 5 和用戶端 SDK 3 中的運作方式不同。

- 在用戶端 SDK 5 中，我們介紹了金鑰可用性定額組的概念。此概念要求您至少使用兩個 HSM 執行叢集。您可以使用用戶端金鑰耐久性設定來選擇退出兩個 HSM 需求。如需定額組的詳細資訊，請參閱 the section called “用戶端 SDK 5 概念”。
- 在用戶端 SDK 3 中，您可以使用用戶端金鑰耐久性設定來指定必須成功建立金鑰的 HSM 數目，整體作業才會視為成功。

用戶端 SDK 5 用戶端金鑰耐久性設定

在用戶端 SDK 5 中，金鑰同步處理是完全自動的程序。透過金鑰可用性定額組，新建立的金鑰在您的應用程式使用金鑰前必須存在於叢集中的兩個 HSM 上。若要使用金鑰可用性定額組，您的叢集必須至少要有兩個 HSM。

如果您的叢集組態不符合金鑰耐久性需求，則建立或使用權杖金鑰的任何嘗試都會失敗，並在日誌中顯示下列錯誤訊息：

```
Key <key handle> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

您可以使用用戶端組態設定來選擇退出金鑰可用性定額組。例如，您可能想要選擇退出以單一 HSM 執行叢集。

用戶端 SDK 5 概念

金鑰可用性定額組

AWS CloudHSM 指定叢集中金鑰必須存在的 HSM 數目，您的應用程式才能使用金鑰。需要至少有兩個 HSM 的叢集。

管理用戶端金鑰耐久性設定

若要管理用戶端金鑰耐久性設定，您必須使用用戶端 SDK 5 的設定工具。

PKCS #11 library

在 Linux 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

在 Windows 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-
availability-check
```

OpenSSL Dynamic Engine

在 Linux 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

JCE provider

在 Linux 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

在 Windows 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-
availability-check
```

CloudHSM CLI

在 Linux 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

在 Windows 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-
availability-check
```

用戶端 SDK 3 用戶端金鑰耐久性設定

在用戶端 SDK 3 中，金鑰同步處理大部分是自動程序，但您可以使用用戶端金鑰耐久性設定，讓金鑰更耐久。您需要指定必須成功建立金鑰的 HSM 數目，整體作業才會視為成功。無論您選擇何種設定，用戶端同步處理始終都會盡最大努力嘗試將金鑰複製到叢集中的每個 HSM。您的設定會對您指定的 HSM 數量強制建立金鑰。如果您指定一個值，但系統無法將金鑰複製到該數目的 HSM，則系統會自動清除任何不需要的金鑰材料，您可以再試一次。

> **Important**
>
> 如果您未設定用戶端金鑰耐久性設定 (或使用預設值 1)，您的金鑰容易遺失。如果您目前的 HSM 應該在伺服器端服務將該金鑰複製到另一個 HSM 之前失敗，您就會遺失金鑰材料。

若要最大化金鑰持久性，請考慮為用戶端同步處理指定至少兩個 HSM。請記住，無論您指定了多少 HSM，叢集上的工作負載都保持不變。用戶端同步處理一律會盡最大努力嘗試將金鑰複製到叢集中的每個 HSM。

建議

- 最小值：每個叢集兩個 HSM
- 最大值：少於叢集中 HSM 總數的一個值

如果用戶端同步處理失敗，用戶端服務會清除任何可能已建立且現在不需要的多餘金鑰。這種清理是盡力而為的回應，可能並非不總是有效。如果清理失敗，您可能必須刪除不需要的金鑰材料。如需詳細資訊，請參閱金鑰同步處理失敗。

設定用戶端金鑰耐久性的組態檔案

若要指定用戶端金鑰耐久性設定，您必須編輯 `cloudhsm_client.cfg`。

編輯用戶端組態檔案

1. 打開 `cloudhsm_client.cfg`.

   Linux：

   ```
   /opt/cloudhsm/etc/cloudhsm_client.cfg
   ```

   Windows：

   ```
   C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
   ```

2. 在檔案的`client`節點中，新增`create_object_minimum_nodes`並指定 HSM 數目下限的值，AWS CloudHSM 必須成功建立金鑰才能成功建立金鑰，金鑰建立作業才能成功。

   ```
   "create_object_minimum_nodes" : 2
   ```

> **Note**
>
> key_mgmt_util (KMU) 命令列工具有用戶端金鑰耐久性的額外設定。如需更多資訊，請參閱the section called “KMU 與用戶端同步”

組態參考

以下是用戶端同步處理屬性，如 `cloudhsm_client.cfg` 的摘錄所示：

```
{
    "client": {
        "create_object_minimum_nodes" : 2,
        ...
    },

    ...
}
```

create_object_minimum_nodes

指定認為金鑰產生、金鑰匯入或金鑰解除包裝作業成功所需的 HSM 數目下限。如果設定，則預設值為 1。即：對於每個金鑰建立作業，用戶端服務都會嘗試在叢集中的每個 HSM 上建立金鑰，但若要傳回成功，只需要在叢集中的一個 HSM 上建立單一金鑰。

KMU 與用戶端同步

如果您使用 key_mgmt_util (KMU) 命令列工具建立金鑰，您可以使用選用的命令列參數 (`-min_srv`) 來限制需要複製金鑰的 HSM 數目。如果您在組態檔案中指定命令列參數和值，則 AWS CloudHSM 會使用這兩個值中較大的值。

如需詳細資訊，請參閱下列主題：

- 根德薩 KeyPair
- GenECC KeyPair
- 根納薩 KeyPair
- genSymKey
- importPrivateKey

- importPubKey
- imSymKey
- insertMaskedObject
- unWrapKey

同步複製的叢集之間的金鑰

用戶端和伺服器端同步處理僅適用於同步處理同一叢集內的金鑰。如果您將叢集備份複製到另一個區域，您可以使用 cloudhsm_mgmt_util (CMU) 的 syncKey 命令來同步叢集之間的金鑰。您可以使用複製的叢集進行跨區域備援，或簡化災難復原程序。如需詳細資訊，請參閱 syncKey。

AES 密鑰包裝在 AWS CloudHSM

本主題說明 AES 金鑰包裝在中的選項 AWS CloudHSM。AES 金鑰包裝使用 AES 金鑰 (包裝金鑰) 來包裝任何類型的另一個金鑰 (目標金鑰)。您可以使用金鑰包裝來保護儲存的金鑰，或透過不安全的網路傳輸金鑰。

主題

- 支援的演算法
- 使用 AES 密鑰包裝 AWS CloudHSM

支援的演算法

AWS CloudHSM 為 AES 密鑰包裝提供了三個選項，每個選項都基於在包裝之前如何填充目標密鑰。當您呼叫金鑰包裝時，會根據您使用的演算法自動完成填補。下表列出支援的演算法和相關詳細資訊，可協助您為應用程式選擇適當的包裝機制。

AES 金鑰包裝演算法	規格	支援的目標金鑰類型	填補方案	AWS CloudHSM 客戶可用性
AES 金鑰包裝，零填補	RFC 5649 和 SP 800 - 38F	全部	如果需要，在金鑰位元後增加零，以對齊區塊	SDK 3.1 及更新版本

AES 金鑰包裝演算法	規格	支援的目標金鑰類型	填補方案	AWS CloudHSM 客戶可用性
AES 金鑰包裝，無填補	RFC 3394 和 SP 800 - 38F	區塊對齊的金鑰，例如 AES 和 3DES	無	SDK 3.1 及更新版本
AES 金鑰包裝與 PKCS #5 填補	無	全部	根據 PKCS #5 填補方案新增至少 8 個位元組以對齊區塊	全部

若要了解如何在應用程式中使用上表的 AES 金鑰包裝演算法，請參閱 AWS CloudHSM中的使用 AES 金鑰包裝。

了解 AES 金鑰包裝中的初始化向量

在包裝之前，CloudHSM 會將初始化向量 (IV) 附加到目標金鑰，以確保資料完整性。每個金鑰包裝演算法對於允許什麼類型的 IV 都有特定的限制。要在中設置 IV AWS CloudHSM，您有兩個選擇：

- 隱含：將 IV 設定為 NULL，CloudHSM 會使用該演算法的預設值進行包裝和取消包裝操作 (建議)
- 明確：透過將預設 IV 值傳遞給金鑰包裝函數來設定 IV

> **Important**
>
> 您必須了解在您應用程式中使用什麼 IV。若要取消包裝金鑰，您必須提供您用來包裝金鑰的相同 IV。如果您使用隱含 IV 來包裝，則請使用隱含 IV 來取消包裝。對於隱含 IV，CloudHSM 將使用預設值來取消包裝。

下表說明包裝演算法指定之 IV 的允許值。

AES 金鑰包裝演算法	隱含 IV	明確 IV
AES 金鑰包裝，零填補	必要	不允許

AES 金鑰包裝演算法	隱含 IV	明確 IV
	預設值：(根據規格在內部計算 IV)	
AES 金鑰包裝，無填補	允許 (建議使用) 預設值：0xA6A6A6A6A6A6A6A6	允許 只接受此值：0xA6A6A6A6A6A6A6A6
AES 金鑰包裝與 PKCS #5 填補	允許 (建議使用) 預設值：0xA6A6A6A6A6A6A6A6	允許 只接受此值：0xA6A6A6A6A6A6A6A6

使用 AES 密鑰包裝 AWS CloudHSM

如下所示包裝和取消包裝金鑰：

- 在 PKCS #11 程式庫中，為 C_WrapKey 和 C_UnWrapKey 函數選取適當的機制，如下表所示。
- 在 JCE 提供者中，選取適當的演算法、模式和填補組合，實作加密方法 Cipher.WRAP_MODE 和 Cipher.UNWRAP_MODE，如下表所示。
- 在 CloudHSM CLI 中，從支援的密鑰展開演算法按鍵包裝和演算法清單中選擇適當的演算法，如下表所示。
- 在 key_mgmt_util (KMU) 中，使用 unWrapKey 和 wrapKey 命令搭配適當的 m 值，如下表所示。

AES 金鑰包裝演算法	PKCS #11 機制	Java 方法	CloudHSM 命令	金鑰管理公用程式 (KMU) 引數
AES 金鑰包裝，零填補	• CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD (供應商定義的機制)	AESWrap/ECB/ZeroPadding	aes-zero-pad	m = 6

AES 金鑰包裝演算法	PKCS #11 機制	Java 方法	CloudHSM 命令	金鑰管理公用程式 (KMU) 引數
AES 金鑰包裝，無填補	• CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD (供應商定義的機制)	AESWrap/ECB/NoPadding	aes-no-pad	m = 5
AES 金鑰包裝與 PKCS #5 填補	• CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD (供應商定義的機制)	AESWrap/ECB/PKCS5Padding	一個電腦墊	m = 4

使用受信任的金鑰 AWS CloudHSM

AWS CloudHSM 支援受信任的金鑰包裝，以保護資料金鑰免受內部威脅。本主題說明如何建立可信任金鑰來保護資料安全。

主題

- 了解可信任金鑰
- 可信任的金鑰屬性
- 如何使用可信任金鑰來包裝資料金鑰
- 如何使用可信任金鑰取包裝消資料金鑰

了解可信任金鑰

可信任金鑰是用來包裝其他金鑰的金鑰，管理員和加密管理員 (COs) 會使用屬性 CKA_TRUSTED 明確識別為可信任。此外，管理員和加密管理員 (CO) 會使用 CKA_UNWRAP_TEMPLATE 和相關屬性來指定可信任金鑰取消資料金鑰包裝後可執行的動作。由可信任金鑰取消資料金鑰包裝也必須包含這些屬性，取消包裝的操作才能成功，其有助於確保只允許已取消包裝的資料金鑰用於您想要的用途。

使用屬性 CKA_WRAP_WITH_TRUSTED 來識別您要使用可信任金鑰包裝的所有資料金鑰。這樣做可讓您限制資料金鑰，讓應用程式只能使用可信任金鑰來取消包裝。一旦您在資料金鑰上設定此屬性，屬

性就會變成唯讀，而且無法變更。設定這些屬性後，應用程式只能使用您信任的金鑰來取消資料金鑰包裝，並且取消包裝一律會產生具有會限制這些金鑰使用方式屬性的資料金鑰。

可信任的金鑰屬性

下列屬性可讓您將金鑰標示為可信任、指定資料金鑰只能使用可信任金鑰進行包裝和取消包裝，以及控制資料金鑰在取消包裝後可執行的操作：

- CKA_TRUSTED：將此屬性 (除 CKA_UNWRAP_TEMPLATE 外) 套用於將會包裝資料金鑰的金鑰，以表明管理員或加密管理員 (CO) 已完成必要的盡職調查並信任此金鑰。只有管理員或 CO 可以設定 CKA_TRUSTED。加密使用者 (CU) 擁有金鑰，但只有 CO 可以設置其 CKA_TRUSTED 屬性。
- CKA_WRAP_WITH_TRUSTED：將此屬性套用於可匯出的資料金鑰，以表明只能使用標記為 CKA_TRUSTED 的金鑰包裝此金鑰。一旦設定 CKA_WRAP_WITH_TRUSTED 為 true，屬性就會變成唯讀，而且您無法變更或移除屬性。
- CKA_UNWRAP_TEMPLATE：將此屬性套用於包裝金鑰 (除 CKA_TRUSTED 外)，以指定服務必須自動套用於服務取消包裝資料金鑰的屬性名稱和值。當應用程式提交金鑰以取消包裝時，還可提供自有的取消包裝範本。如果您指定取消包裝範本，且應用程式提供了自有的取消包裝範本，則 HSM 會使用這兩個範本將屬性名稱和值套用於金鑰。但是，如果用於包裝金鑰的 CKA_UNWRAP_TEMPLATE 中的值與應用程式在取消包裝請求期間提供的屬性衝突，取消包裝請求會失敗。

如需關於屬性的詳細資訊，請參閱下列主題：

- PKCS #11 金鑰屬性
- JCE 金鑰屬性
- CloudHSM CLI 金鑰屬性

如何使用可信任金鑰來包裝資料金鑰

若要使用可信任金鑰來包裝資料金鑰，您必須完成三個基本步驟：

1. 對於您計劃使用可信任金鑰包裝的資料金鑰，請將其 CKA_WRAP_WITH_TRUSTED 屬性設定為 true。
2. 對於您計劃用來包裝資料金鑰的可信任金鑰，請將其 CKA_TRUSTED 屬性設定為 true。
3. 使用可受信任金鑰來包裝資料金鑰。

步驟 1：將資料金鑰的 **CKA_WRAP_WITH_TRUSTED** 設定為 true

對於要包裝的資料金鑰，選擇以下選項之一以將金鑰的 CKA_WRAP_WITH_TRUSTED 屬性設定為 true。這樣做會限制資料金鑰，因此應用程式只能使用可信任金鑰來包裝。

選項 1：如果產生一個新的金鑰，將 **CKA_WRAP_WITH_TRUSTED** 設定為 true

使用 PKCS #11、JCE 或 CloudHSM CLI 產生金鑰。如需詳細資訊，請參閱下列範例。

PKCS #11

若要使用 PKCS #11 產生金鑰，您需要將金鑰的 CKA_WRAP_WITH_TRUSTED 屬性設定為 true。為此，請先將此屬性包含在金鑰的 CK_ATTRIBUTE template 中，然後將屬性設定為 true，如下列範例所示：

```
CK_BYTE_PTR label = "test_key";
CK_ATTRIBUTE template[] = {
        {CKA_WRAP_WITH_TRUSTED, &true_val,         sizeof(CK_BBOOL)},
        {CKA_LABEL,             label,             strlen(label)},
        ...
};
```

如需詳細資訊，請參閱使用 PKCS #11 產生金鑰的公開示範範例。

JCE

若要使用 JCE 產生金鑰，您需要將金鑰的 WRAP_WITH_TRUSTED 屬性設定為 true。為此，請先將此屬性包含在金鑰的 KeyAttributesMap 中，然後將屬性設定為 true，如下列範例所示：

```
final String label = "test_key";
final KeyAttributesMap keySpec = new KeyAttributesMap();
keySpec.put(KeyAttribute.WRAP_WITH_TRUSTED, true);
keySpec.put(KeyAttribute.LABEL, label);
...
```

如需詳細資訊，請參閱使用 JCE 產生金鑰的公開示範範例。

CloudHSM CLI

若要使用 CloudHSM CLI 產生金鑰，您需要將金鑰的 wrap-with-trusted 屬性設定為 true。為此，請將 wrap-with-trusted=true 包含在適合金鑰產生命令的引數：

- 對於對稱金鑰，請將 wrap-with-trusted 新增至 attributes 引數中。

- 對於公有金鑰，請將 `wrap-with-trusted` 新增至 `public-attributes` 引數中。
- 對於私有金鑰，請將 `wrap-with-trusted` 新增至 `private-attributes` 引數中。

如需關於產生金鑰對的詳細資訊，請參閱 關鍵 generate-asymmetric-pair 。

如需關於產生對稱金鑰的詳細資訊，請參閱 產生對稱金鑰 。

選項 2：如果使用現有金鑰，請使用 CloudHSM CLI 將其 **CKA_WRAP_WITH_TRUSTED** 設定為 true

若要將現有金鑰的 CKA_WRAP_WITH_TRUSTED 屬性設定為 true，請依照下列步驟執行：

1. 使用 登入 命令以加密使用者 (CU) 身分登入。
2. 使用 設定金鑰屬性 命令將金鑰的 `wrap-with-trusted` 屬性設定為 true。

```
aws-cloudhsm > key set-attribute --filter attr.label=test_key --name wrap-with-
trusted --value true
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

步驟 2：將可信任金鑰的 **CKA_TRUSTED** 設定為 true

若要讓金鑰成為可信任金鑰，其 CKA_TRUSTED 屬性必須設定為 true。您可以使用 CloudHSM CLI 或 CloudHSM 管理公用程式 (CMU) 來執行此操作。

- 如果使用 CloudHSM CLI 設定金鑰的 CKA_TRUSTED 屬性，請參閱 如何使用 CloudHSM CLI 將金鑰標示為可信任。
- 如果使用 CMU 來設定金鑰的 CKA_TRUSTED 屬性，請參閱 如何使用 CMU 將金鑰標示為可信任。

步驟 3。使用可受信任金鑰來包裝資料金鑰

若要使用您在步驟 2 中設定的可信任金鑰來包裝步驟 1 所述的資料金鑰，請參閱下列連結以取得程式碼範例。每個都示範如何包裝金鑰。

- AWS CloudHSM PKCS #11 範例

- AWS CloudHSM JCE 範例

如何使用可信任金鑰取包裝消資料金鑰

若要解開資料金鑰，您需要 CKA_UNWRAP 設定為 true 的可信任金鑰。若要成為金鑰，其還必須滿足下列條件：

- 金鑰的 CKA_TRUSTED 屬性必須設定為 true。
- 金鑰必須使用 CKA_UNWRAP_TEMPLATE 和相關屬性來指定資料金鑰在取消包裝後可以執行的動作。例如，如果您希望取消包裝的金鑰不可匯出，您可以將 CKA_EXPORTABLE = FALSE 設定為 CKA_UNWRAP_TEMPLATE 的一部分。

> **Note**
>
> CKA_UNWRAP_TEMPLATE 僅可與 PKCS #11 一同使用。

當應用程式提交要取消包裝的金鑰時，應用程式也可以提供自有的取消包裝範本。如果您指定取消包裝範本，且應用程式提供了自有的取消包裝範本，則 HSM 會使用這兩個範本將屬性名稱和值套用於金鑰。但是，如果在取消包裝請求期間，可信任金鑰中的值與應用程式提供的屬性 CKA_UNWRAP_TEMPLATE 衝突，則取消包裝請求會失敗。

若要查看關於使用可信任金鑰取消包裝金鑰的範例，請參閱此 PKCS #11 範例。

使用 CloudHSM CLI 管理金鑰

如果使用最新的 SDK 版本系列，請使用 CloudHSM CLI 管理叢集中的 AWS CloudHSM 金鑰。如需詳細資訊，請參閱下列主題。

- 使用受信任金鑰說明如何使用 CloudHSM CLI 建立受信任金鑰來保護資料安全。
- 產生金鑰包括建立金鑰的說明，包括對稱金鑰、RSA 金鑰和 EC 金鑰。
- 刪除金鑰會說明金鑰擁有者如何刪除金鑰。
- 共用和取消共用金鑰詳細說明金鑰擁有者如何共用和取消共用金鑰。
- 篩選金鑰提供如何使用篩選條件尋找金鑰的準則。

使用 CloudHSM CLI 產生金鑰

您必須先啟動 CloudHSM CLI 並以加密使用者 (CU) 身分登入，才能產生金鑰。若要在 HSM 中產生金鑰，請使用與您想產生之金鑰類型對應的命令。

主題

- 產生對稱金鑰
- 產生非對稱金鑰
- 相關主題

產生對稱金鑰

使用 產生對稱金鑰 中列出的命令產生對稱金鑰。若要查看所有可用的選項，請使用 help key generate-symmetric 命令。

產生 AES 金鑰

使用 key generate-symmetric aes 命令產生 AES 金鑰。若要查看所有可用的選項，請使用 help key generate-symmetric aes 命令。

Example

下列範例會產生 32 位元組 AES 金鑰。

```
aws-cloudhsm > key generate-symmetric aes \
    --label aes-example \
    --key-length-bytes 32
```

引數

<LABEL>

指使用者定義的 AES 金鑰標籤。

必要：是

<KEY-LENGTH-BYTES>

指金鑰長度 (以位元組為單位)。

有效值：

- 16、24 和 32

必要：是

<KEY_ATTRIBUTES>

指定一個空格分隔的金鑰屬性清單，對產生的 AES 金鑰進行設定，格式為 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例如，token=true)

如需支援的 AWS CloudHSM 索引鍵屬性清單，請參閱CloudHSM CLI 的金鑰屬性。

必要：否

<SESSION>

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

如要將工作階段金鑰更改為永久 (權杖) 金鑰，請使用金鑰設定屬性。

根據預設，產生的金鑰是持久性/權杖金鑰。使用 <SESSION> 改變這一點，確保用此參數產生的金鑰是工作階段/臨時金鑰

必要：否

產生一般私密金鑰

使用 key generate-symmetric generic-secret 命令產生一般私密金鑰。若要查看所有可用的選項，請使用 help key generate-symmetric generic-secret 命令。

Example

下列範例會產生 32 位元組的一般私密金鑰。

```
aws-cloudhsm > key generate-symmetric generic-secret \
    --label generic-secret-example \
    --key-length-bytes 32
```

引數

<LABEL>

為一般私密金鑰指定使用者定義的標籤。

必要：是

<KEY-LENGTH-BYTES>

指金鑰長度 (以位元組為單位)。

有效值：

- 1 到 800

必要：是

<KEY_ATTRIBUTES>

指一個空格分隔的金鑰屬性清單，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例如，token=true) 的形式為產生的一般私密金鑰設定

如需支援的 AWS CloudHSM 索引鍵屬性清單，請參閱CloudHSM CLI 的金鑰屬性。

必要：否

<SESSION>

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

如要將工作階段金鑰更改為永久 (權杖) 金鑰，請使用金鑰設定屬性。

根據預設，產生的金鑰是持久性/權杖金鑰。使用 <SESSION> 改變這一點，確保用此參數產生的金鑰是工作階段/臨時金鑰

必要：否

產生非對稱金鑰

使用 關鍵 generate-asymmetric-pair 中列出的命令產生非對稱金鑰對。

產生 RSA 金鑰

使用 key generate-asymmetric-pair rsa 命令來產生 RSA 金鑰對。若要查看所有可用的選項，請使用 help key generate-asymmetric-pair rsa 命令。

Example

以下範例會產生 RSA 2048 位元金鑰對。

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
    --public-exponent 65537 \
    --modulus-size-bits 2048 \
    --public-label rsa-public-example \
    --private-label rsa-private-example
```

引數

<PUBLIC_LABEL>

指使用者定義的公有金鑰標籤。

必要：是

<PRIVATE_LABEL>

指使用者定義的私有金鑰標籤。

必要：是

<MODULUS_SIZE_BITS>

指模數的長度 (以位元為單位)。最小值為 2048。

必要：是

<PUBLIC_EXPONENT>

指公有指數。值必須為大於或等於 65537 的奇數。

必要：是

<PUBLIC_KEY_ATTRIBUTES>

指以空格分隔的金鑰屬性清單，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例如，token=true) 的形式為產生的 RSA 公有金鑰設定。

如需支援的 AWS CloudHSM 索引鍵屬性清單，請參閱CloudHSM CLI 的金鑰屬性。

必要：否

<SESSION>

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

如要將工作階段金鑰更改為永久 (權杖) 金鑰，請使用金鑰設定屬性。

根據預設，產生的金鑰是持久性/權杖金鑰。使用 <SESSION> 改變這一點，確保用此參數產生的金鑰是工作階段/臨時金鑰

必要：否

產生 EC (橢圓曲線密碼編譯) 金鑰對

使用 key generate-asymmetric-pair ec 命令來產生金鑰對。若要查看所有可用的選項，包括支援的橢圓曲線清單，請使用 help key generate-asymmetric-pair ec 命令。

Example

下列範例會使用 Secp384r1 橢圓曲線產生 EC 金鑰對。

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp384r1 \
    --public-label ec-public-example \
    --private-label ec-private-example
```

引數

<PUBLIC_LABEL>

指使用者定義的公有金鑰標籤。用戶端 SDK 5.11 及之後允許的大小上限為 127 個字元。label用戶端 SDK 5.10 及之前版本的限制為 126 個字元。

必要：是

<PRIVATE_LABEL>

指使用者定義的私有金鑰標籤。用戶端 SDK 5.11 及之後允許的大小上限為 127 個字元。label用戶端 SDK 5.10 及之前版本的限制為 126 個字元。

必要：是

<CURVE>

指橢圓曲線的識別符。

有效值：

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1

必要：是

<PUBLIC_KEY_ATTRIBUTES>

指以空格分隔的金鑰屬性清單，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例如，token=true) 的形式為產生的 EC 公開金鑰設定。

如需支援的 AWS CloudHSM 索引鍵屬性清單，請參閱CloudHSM CLI 的金鑰屬性。

必要：否

<PRIVATE_KEY_ATTRIBUTES>

指以空格分隔的金鑰屬性清單，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例如，token=true) 的形式為產生的 EC 私有金鑰設定。

如需支援的 AWS CloudHSM 索引鍵屬性清單，請參閱CloudHSM CLI 的金鑰屬性。

必要：否

<SESSION>

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

如要將工作階段金鑰更改為永久 (權杖) 金鑰，請使用金鑰設定屬性。

根據預設，產生的金鑰是持久性 (權杖) 金鑰。在 <SESSION> 傳遞會變更此情況，確保使用此參數生成的金鑰是工作階段 (臨時) 金鑰。

必要：否

相關主題

- CloudHSM CLI 的金鑰屬性
- 關鍵 generate-asymmetric-pair
- 產生對稱金鑰

使用 CloudHSM CLI 刪除金鑰

使用本主題中的範例，透過 CloudHSM CLI 刪除金鑰。只有金鑰擁有者可以刪除金鑰。

主題

- 範例：刪除金鑰
- 相關主題

範例：刪除金鑰

1. 執行 key list 命令以識別您要刪除的金鑰：

```
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. 識別金鑰後，執行具有金鑰的唯一 `label` 屬性的 key delete 以刪除金鑰：

```
aws-cloudhsm > key delete --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
```

```
  }
}
```

3. 執行具有金鑰的唯一 `label` 屬性的 key list 命令並確認金鑰已被刪除。如下列範例所示，HSM 叢集中沒有含標籤 `my_key_to_delete` 的金鑰：

```
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
  }
}
```

相關主題

- CloudHSM CLI 的金鑰屬性
- 刪除金鑰

使用 CloudHSM CLI 共用和取消共用金鑰

使用本主題中的命令在 CloudHSM CLI 中共用和取消共用金鑰。在中 AWS CloudHSM，建立金鑰的加密使用者 (CU) 擁有該金鑰。擁有者可以使用 key share 和 key unshare 命令來與其他 CU 共用和取消共用金鑰。共用金鑰的使用者可以在密碼編譯操作中使用此金鑰，但無法匯出金鑰、刪除金鑰，或再與其他使用者共用。

共用金鑰前，請以擁有金鑰的加密使用者 (CU) 身分登入 HSM。

主題

- 範例：共用和停止共用金鑰
- 相關主題

範例：共用和停止共用金鑰

Example

以下示例顯示如何與加密使用者 (CU) `alice` 共用和取消共用金鑰。除 key share 和 key unshare 命令外，共用和取消共用命令還需要使用 CloudHSM CLI 金鑰篩選條件的特定金鑰以及要與之共用或取消共用金鑰的使用者的特定使用者名稱。

1. 先用篩選條件執行 key list 命令以傳回特定金鑰並查看已與誰共用金鑰。

```
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. 檢視 shared-users 輸出以識別金鑰目前與誰共用。
3. 如要與加密使用者 (CU) alice 共享此金鑰，請輸入以下命令：

```
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"
  }
}
```

請注意，除 key share 命令外，此命令還使用金鑰的唯一標籤以及與其共用金鑰的使用者名稱。

4. 執行 key list 命令以確認金鑰已與 alice 共用：

```
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. 若要與 alice 取消共用相同的金鑰，請執行下列 unshare 命令：

```
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"
  }
}
```

請注意，除 key unshare 命令外，此命令還使用金鑰的唯一標籤以及與其共用金鑰的使用者名稱。

6. 再次執行 key list 命令並確認是否未與加密使用者 alice 共用金鑰：

```
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
  }
}
```

相關主題

- CloudHSM CLI 的金鑰屬性
- 共用金鑰
- 取消共用金鑰
- 使用 CloudHSM CLI 篩選金鑰

使用 CloudHSM CLI 篩選金鑰

使用下列主要命令，將標準化金鑰篩選機制用於 CloudHSM CLI。

- key list
- key delete
- key share
- key unshare
- key set-attribute

若要使用 CloudHSM CLI 選取和/或篩選金鑰，主要命令會根據 CloudHSM CLI 的金鑰屬性 使用標準化篩選機制。您可以使用一或多個可識別單一或多個按鍵的 AWS CloudHSM 屬性，在按鍵指令中指定一個或一組按鍵。密鑰過濾機制僅對當前登錄用戶擁有和共享的密鑰以及 AWS CloudHSM 集群中的所有公共密鑰進行操作。

主題

要求

若要篩選金鑰，您必須以加密使用者 (CU) 的身分登入。

篩選以尋找單一金鑰

請注意，在下面的範例中，用作篩選條件的每個屬性都必須以 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 的形式寫入。例如，如果您想透過標籤屬性進行篩選，您會寫入 `attr.label=my_label`。

Example 使用單個屬性來尋找單一金鑰

此範例示範如何篩選為僅使用單一識別屬性的單一唯一金鑰。

```
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 使用多個屬性尋找單一金鑰

以下範例示範如何使用多個金鑰屬性尋找單一金鑰。

```
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 篩選以尋找一組金鑰

以下範例示範如何篩選以尋找一組私有 rsa 金鑰。

```
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
```

```
  }
}
```

篩選錯誤

某些金鑰作業一次只能在單一金鑰上執行。對於這些操作，CloudHSM CLI 會在篩選條件未充分細化且多個金鑰符合準則時報錯。一個此類範例與金鑰刪除如下所示。

Example 匹配太多金鑰時篩選錯誤

```
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."
}
```

相關主題

- CloudHSM CLI 的金鑰屬性

如何使用 CloudHSM CLI 將金鑰標示為可信任

本節中的內容提供如何使用 CloudHSM CLI 將金鑰標記為可信任的說明。

1. 使用 CloudHSM CLI login 命令，以加密使用者 (CU) 身分登入。
2. 使用 key list 命令來識別您要標記為可信任的金鑰的金鑰參照。下列範例會列出含有標籤 `key_to_be_trusted` 的金鑰。

```
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. 使用 登出 命令，以加密使用者 (CU) 的身分登出。
4. 使用 登入 命令，以管理員身分登入。
5. 使用 key set-attribute 命令搭配您在步驟 2 中識別的金鑰參照，將金鑰的信任值設定為 true：

```
aws-cloudhsm > key set-attribute --filter key-reference=<Key Reference> --name
 trusted --value true
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

使用 KMU 和 CMU 管理金鑰

如果使用最新的 SDK 版本系列，請使用 CloudHSM CLI 管理叢集中的 AWS CloudHSM 金鑰。

如果使用舊版 SDK 系列，您可以使用 key_mgmt_util 命令列工具來管理 AWS CloudHSM 叢集中 HSM 上的金鑰。您必須先啟動 AWS CloudHSM 用戶端、啟動 key_mgmt_util，然後登入 HSM，然後才能管理金鑰。如需詳細資訊，請參閱 key_mgmt_util 的入門指南。

- 使用可信任金鑰說明如何使用 PKCS #11 程式庫屬性和 CMU 來建立可信任金鑰以保護資料的安全。
- 產生金鑰具有產生金鑰的說明，包括對稱金鑰、RSA 金鑰和 EC 金鑰。
- 匯入金鑰提供金鑰擁有者如何匯入金鑰的詳細資訊。
- 匯出金鑰提供金鑰擁有者如何匯出金鑰的詳細資訊。
- 刪除金鑰提供金鑰擁有者如何刪除金鑰的詳細資訊。
- 共用和取消共用金鑰詳細說明金鑰擁有者如何共用和取消共用金鑰。

產生金鑰

若要在 HSM 中產生金鑰，請使用與您想產生之金鑰類型對應的命令。

主題

- 產生對稱金鑰
- 產生 RSA 金鑰對
- 產生 ECC (橢圓曲線密碼編譯) 金鑰對

產生對稱金鑰

使用指genSymKey令產生 AES 和其他類型的對稱金鑰。若要查看所有可用的選項，請使用 genSymKey -h 命令。

以下範例建立 256 位元 AES 金鑰。

```
Command: genSymKey -t 31 -s 32 -l aes256
CfM3GenerateSymmetricKey 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
```

產生 RSA 金鑰對

若要產生 RSA key pair，請使用 genR KeyPair SA 指令。若要查看所有可用的選項，請使用 genRSAKeyPair -h 命令。

以下範例會產生 RSA 2048 位元金鑰對。

```
Command: genRSAKeyPair -m 2048 -e 65537 -l rsa2048
CfM3GenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

CfM3GenerateKeyPair:    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
```

產生 ECC (橢圓曲線密碼編譯) 金鑰對

若要產生 ECC key pair，請使用 Gene KeyPair CC 指令。若要查看所有可用的選項，包括支援的橢圓曲線清單，請使用 genECCKeyPair -h 命令。

以下範例使用 NIST FIPS 出版物 186-4 中定義的 P-384 橢圓曲線來產生 ECC 金鑰對。

```
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
```

匯入金鑰

若要將私密金鑰 (即：對稱金鑰和非對稱私有金鑰) 匯入 HSM，您必須先在 HSM 上建立包裝金鑰。您可以直接匯入公有金鑰，而不需包裝金鑰。

主題

- 匯入私密金鑰
- 匯入公有金鑰

匯入私密金鑰

完成以下步驟，以匯入私密金鑰。匯入私密金鑰之前，請先儲存為檔案。將對稱金鑰儲存為原始位元組，將非對稱私有金鑰儲存為 PEM 格式。

此範例示範如何將檔案中的純文字私密金鑰匯入 HSM。若要將加密金鑰從檔案匯入 HSM，請使用指unWrapKey令。

匯入私密金鑰

1. 使用指genSymKey令建立環繞索引鍵。以下命令會建立僅對目前工作階段而言有效的 128 位元 AES 包裝金鑰。您可以使用工作階段金鑰或持久性金鑰做為包裝金鑰。

   ```
   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. 依據您要匯入的私密金鑰類型而定，使用以下其中一個命令。

   - 若要匯入對稱金鑰，請使用指imSymKey令。以下命令使用上個步驟中建立的包裝金鑰，從名為 aes256.key 的檔案匯入 AES 金鑰。若要查看所有可用的選項，請使用 imSymKey -h 命令。

     ```
     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
     ```

   - 若要匯入非對稱私密金鑰，請使用importPrivateKey指令。以下命令使用上個步驟中建立的包裝金鑰，從名為 rsa2048.key 的檔案匯入私有金鑰。若要查看所有可用的選項，請使用 importPrivateKey -h 命令。

     ```
     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
```

匯入公有金鑰

使用importPubKey指令匯入公開金鑰。若要查看所有可用的選項，請使用 importPubKey -h 命令。

以下範例會從名為 rsa2048.pub 的檔案匯入 RSA 公有金鑰。

```
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
```

匯出金鑰

若要從 HSM 匯出私密金鑰 (即：對稱金鑰和非對稱私有金鑰)，您必須先建立包裝金鑰。您可以直接匯出公有金鑰，而不需包裝金鑰。

只有金鑰擁有者可以匯出金鑰。共用金鑰的使用者可以在密碼編譯操作中使用此金鑰，但無法匯出此金鑰。執行這個範例時，請務必匯出您建立的金鑰。

> **Important**
>
> 此指exSymKey令會將私密金鑰的純文字 (未加密) 複本寫入檔案。匯出程序需要包裝金鑰，但檔案中的金鑰不是包裝金鑰。若要匯出金鑰的包裝 (加密) 複本，請使用 wrapKey 命令。

主題

- 匯出私密金鑰
- 匯出公有金鑰

匯出私密金鑰

完成以下步驟，以匯出私密金鑰。

匯出私密金鑰

1. 使用指genSymKey令建立環繞索引鍵。以下命令會建立僅對目前工作階段而言有效的 128 位元 AES 包裝金鑰。

```
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. 依據您要匯出的私密金鑰類型而定，使用以下其中一個命令。

   - 若要匯出對稱金鑰，請使用指exSymKey令。以下命令會將 AES 金鑰匯出到名為 aes256.key.exp 的檔案。若要查看所有可用的選項，請使用 exSymKey -h 命令。

```
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**
   >
   > 命令的輸出顯示「包裝對稱金鑰」已寫入輸出檔。不過，輸出檔案包含純文字 (而非包裝) 金鑰。若要將包裝 (加密) 金鑰匯出到檔案，請使用 wrapKey 命令。

   - 若要匯出私有金鑰，請使用 exportPrivateKey 命令。以下命令會將私有金鑰匯出到名為 rsa2048.key.exp 的檔案。若要查看所有可用的選項，請使用 exportPrivateKey -h 命令。

```
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
```

匯出公有金鑰

使用 exportPubKey 命令來匯出公有金鑰。若要查看所有可用的選項，請使用 exportPubKey -h 命令。

以下範例將 RSA 公有金鑰匯出到名為 `rsa2048.pub.exp` 的檔案。

```
Command: exportPubKey -k 524294 -out rsa2048.pub.exp
PEM formatted public key is written to rsa2048.pub.key

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

刪除金鑰

使用 deleteKey 命令來刪除金鑰，如下列範例所示。只有金鑰擁有者可以刪除金鑰。

```
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
```

共用和取消共用金鑰

在中 AWS CloudHSM，建立金鑰的 CU 擁有該金鑰。擁有者可以管理金鑰、匯出和刪除金鑰，以及可在密碼編譯操作中使用金鑰。擁有者還可以向其他 CU 使用者共用金鑰。共用金鑰的使用者可以在密碼編譯操作中使用此金鑰，但無法匯出或刪除此金鑰，或再與其他使用者共用。

您可以在建立金鑰時與其他 CU 使用者共用金鑰，例如使用genSymKey或 genRSA KeyPair 命令的-u參數。若要將現有的金鑰共用給另一個 HSM 使用者，請使用 cloudhsm_mgmt_util 命令列工具。這不同於本區段中大多數使用 key_mgmt_util 命令列工具的任務。

在您可以共用金鑰之前，您必須啟動 cloudhsm_mgmt_util、啟用 end-to-end加密，然後登入 HSM。若要共用金鑰，請以擁有金鑰的加密使用者 (CU) 身分登入 HSM。只有金鑰擁有者可以共用金鑰。

使用 shareKey 命令，並指定金鑰控點及一或多位使用者的 ID，以共用或取消共用金鑰。若要與多個使用者共用或取消共用，請指定以逗號分隔的使用者 ID 清單。若要共用金鑰，請使用 1 做為命令的最後一個參數，如下列範例所示。若要取消共用，請使用 0。

```
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)
```

下列顯示 shareKey 命令的語法。

```
aws-cloudhsm>shareKey <key handle> <user ID> <Boolean: 1 for share, 0 for unshare>
```

如何使用 CMU 將金鑰標示為可信任

本節中的內容提供如何使用 CMU 將金鑰標示為可信任的說明。

1. 使用 loginHSM 命令，以加密管理員 (CO) 身分登入。
2. 使用 setAttribute 命令，將 OBJ_ATTR_TRUSTED (值 134) 設定為 true (1)。

   ```
   setAttribute <Key Handle> 134 1
   ```

管理複製的叢集

如果該區域中的叢集最初是從另一個區域中的叢集備份建立的，請使用 CloudHSM 管理公用程式 (CMU) 同步處理遠端區域中的叢集。比如，您將叢集複製到另一個區域 (目的地)，然後您想要同步原始叢集 (來源) 的變更。在這種情況下，您可以使用 CMU 同步處理叢集。您可以建立新的 CMU 組態檔案，從新檔案中的兩個叢集指定硬體安全性模組 (HSM)，然後使用 CMU 連線至具有該檔案的叢集。

在複製的叢集間使用 CMU

1. 建立當前組態檔案的複本，並將複本的名稱更改為其他名稱。

   例如，使用下列檔案位置來尋找並建立目前組態檔案的複本，然後將複本名稱從 `cloudhsm_mgmt_config.cfg` 變更為 `syncConfig.cfg`。

   - Linux：`/opt/cloudhsm/etc/cloudhsm_mgmt_config.cfg`

- Windows：C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_config.cfg

2. 在重新命名的複本中，新增目的地 HSM (需要同步的外部區域中的 HSM) 的彈性網絡介面 (ENI) IP。建議您在來源 HSM 下方新增目的地 HSM。

```
{
    ...
    "servers": [
        {
            ...
            "hostname": "<ENI Source IP>",
            ...
        },
        {
            ...
            "hostname": "<ENI Destination IP>",
            ...
        }
    ]
}
```

如需如何取得 IP 地址的詳細資訊，請參閱 the section called “取得 HSM 的 IP 地址”。

3. 使用新的組態檔來初始化 CMU：

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. 檢查傳回的狀態訊息，確認 CMU 連線到所需的全部 HSM，並判斷傳回的 ENI IP 對應到哪個叢集。使用 syncUser 和 syncKey 手動同步處理使用者和金鑰。如需詳細資訊，請參閱 syncUser 和 syncKey。

取得 HSM 的 IP 地址

使用本節取得 HSM 的 IP 地址。

取得 HSM (主控台) 的 IP 位址

1. 請在以下位置開啟 AWS CloudHSM 主控台。 https://console.aws.amazon.com/cloudhsm/home
2. 若要變更 AWS 區域，請使用頁面右上角的區域選擇器。
3. 若要開啟叢集詳細資訊頁面，請在叢集表格中選擇叢集 ID。
4. 若要取得 IP 地址，請在 HSM 索引標籤上，選擇 ENI IP 地址下列出的其中一個 IP 地址。

取得 HSM 的 IP 位址 ()AWS CLI

- 使用 AWS CLI中的 describe-clusters 命令取得 HSM 的 IP 地址。在命令輸出中，HSM 的 IP 地址是 `EniIp` 的值。

```
$  aws cloudhsmv2 describe-clusters


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

相關主題

- syncUser
- syncKey
- 跨區域複製備份

AWS CloudHSM 命令行工具

本主題說明了可用於管理和使用 AWS CloudHSM的命令列工具。

主題

了解命令列工具

除了用於管理 AWS 資源的 AWS Command Line Interface (AWS CLI) 外，還 AWS CloudHSM 提供用於在 HSM 上建立和管理 HSM 使用者和金鑰的命令列工具。您 AWS CloudHSM 可以使用熟悉的 CLI 來管理叢集，並使用 CloudHSM 命令列工具來管理 HSM。

這些是各種命令列工具：

管理叢集和 HSM

模組中的 CloudSMv2 命令 AWS CLI和 HSM PowerShell 2 指令程式 AWSPowerShell

- 這些工具可取得、建立、刪除和標記 AWS CloudHSM 叢集和 HSM：
- 若要在 CLI 中使用 CloudSmv2 命令中的命令，您需要安裝和設定。 AWS CLI
- 模組中的 HSM2 PowerShell 指令程式可在 Windows AWSPowerShell 模組和跨平台 PowerShell 的核心 PowerShell模組中使用。

如何管理 HSM 使用者。

CloudHSM CLI

- 使用 CloudHSM CLI 建立使用者、刪除使用者、列出使用者、更改使用者密碼和更新使用者多重要素驗證 (MFA)。AWS CloudHSM 用戶端軟體中不包含此工具。如需有關安裝此工具的指引，請參閱安裝和設定 CloudHSM CLI。

協助程式工具

兩種工具可協助您使用 AWS CloudHSM 工具和軟體程式庫：

- 設定工具可更新您的 CloudHSM 用戶端組態檔案。這可 AWS CloudHSM 讓您同步叢集中的 HSM。

  AWS CloudHSM 提供了兩個主要版本，客戶端 SDK 5 是最新的。與用戶端 SDK 3 (上一個系列) 相比，用戶端 SDK 5 具有多種優勢。
- pkpspeed 可衡量 HSM 硬體的效能 (與軟體程式庫無關)。

適用於先前 SDK 的工具

使用金鑰管理工具 (KMU) 建立、刪除、匯入和匯出對稱金鑰和非對稱金鑰對：

- key_mgmt_util。此工具包含在 AWS CloudHSM 用戶端軟體之中。

使用 CloudHSM 管理工具 (CMU) 建立和刪除 HSM 使用者，包括執行使用者管理工作的規定人數身分驗證

- cloudhsm_mgmt_util。此工具包含在 AWS CloudHSM 用戶端軟體之中。

設定工具

AWS CloudHSM 自動同步叢集中所有硬體安全模組 (HSM) 之間的資料。configure 工具會更新同步機制所使用之組態檔案中的 HSM 資料。在您使用此命令列工具之前 (特別是在叢集中的 HSM 已變更時)，請使用 configure 重新整理 HSM 資料。

AWS CloudHSM 包括兩個主要的客戶端 SDK 版本：

- 用戶端 SDK 5：這是我們最新預設的用戶端 SDK。如需有關其優點和優勢的資訊，請參閱 用戶端 SDK 5 的優點 。
- 用戶端 SDK 3：這是我們較舊的用戶端 SDK。它包括一組完整的組件，用於基於平台和語言的應用程序兼容性和管理工具。

如需從用戶端 SDK 3 遷移至用戶端 SDK 5 的指示，請參閱從用戶端 SDK 3 遷移至用戶端 SDK 5。

主題

用戶端 SDK 5 設定工具

使用用戶端 SDK 5 設定工具來更新用戶端設定檔案。

用戶端 SDK 5 中的每個元件都包含一個設定工具，其中包含設定工具檔案名稱中的元件指示項。例如，用戶端 SDK 5 的 PKCS #11 程式庫包含一個名為 Linux `configure-pkcs11` 或 Windows `configure-pkcs11.exe` 的設定工具。

語法

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]
```

進階組態

如需用戶端 SDK 5 設定工具特定的進階組態清單，請參閱用戶端 SDK 5 設定工具的進階組態。

> **Important**
>
> 變更組態後，您需要重新啟動應用程式才能使變更生效。

範例

這些範例示範如何使用用戶端 SDK 5 的設定工具。

引導用戶端 SDK 5

Example

此範例使用 -a 參數來更新用戶端 SDK 5 的 HSM 資料。若要使用此 -a 參數，您必須擁有叢集中其中一個 HSM 的 IP 地址。

PKCS #11 library

為用戶端 SDK 5 引導 Linux EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 位址。

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

為用戶端 SDK 5 引導 Windows EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 位址。

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

OpenSSL Dynamic Engine

為用戶端 SDK 5 引導 Linux EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 位址。

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

JCE provider

為用戶端 SDK 5 引導 Linux EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 位址。

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

為用戶端 SDK 5 引導 Windows EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 位址。

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

CloudHSM CLI

為用戶端 SDK 5 引導 Linux EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 地址。

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

為用戶端 SDK 5 引導 Windows EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 地址。

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

> **Note**
>
> 您可以使用 `--cluster-id` 參數來代替 `-a <HSM_IP_ADDRESSES>`。若要查看使用 `--cluster-id` 的需求，請參閱 用戶端 SDK 5 設定工具。

如需 `-a` 參數的詳細資訊，請參閱 the section called “參數”。

指定用戶端 SDK 5 的叢集、區域和端點

Example

此範例會使用 `cluster-id` 參數，以 `DescribeClusters` 撥打呼叫的方式來啟動用戶端 SDK 5。

PKCS #11 library

使用 **`cluster-id`** 為用戶端 SDK 5 啟動 Linux EC2 執行個體

- 使用叢集識別碼`cluster-1234567`來指定叢集中 HSM 的 IP 位址。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id cluster-1234567
```

使用 **`cluster-id`** 為用戶端 SDK 5 啟動 Windows EC2 執行個體

- 使用叢集識別碼`cluster-1234567`來指定叢集中 HSM 的 IP 位址。

```
"C:\Program Files\Amazon\CloudHSM\configure-pkcs11.exe" --cluster-
id cluster-1234567
```

OpenSSL Dynamic Engine

使用 **cluster-id** 為用戶端 SDK 5 啟動 Linux EC2 執行個體

- 使用叢集識別碼`cluster-1234567`來指定叢集中 HSM 的 IP 位址。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id cluster-1234567
```

JCE provider

使用 **cluster-id** 為用戶端 SDK 5 啟動 Linux EC2 執行個體

- 使用叢集識別碼`cluster-1234567`來指定叢集中 HSM 的 IP 位址。

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id cluster-1234567
```

使用 **cluster-id** 為用戶端 SDK 5 啟動 Windows EC2 執行個體

- 使用叢集識別碼`cluster-1234567`來指定叢集中 HSM 的 IP 位址。

```
"C:\Program Files\Amazon\CloudHSM\configure-jce.exe" --cluster-
id cluster-1234567
```

CloudHSM CLI

使用 **cluster-id** 為用戶端 SDK 5 啟動 Linux EC2 執行個體

- 使用叢集識別碼`cluster-1234567`來指定叢集中 HSM 的 IP 位址。

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id cluster-1234567
```

使用 **cluster-id** 為用戶端 SDK 5 啟動 Windows EC2 執行個體

- 使用叢集識別碼cluster-1234567來指定叢集中 HSM 的 IP 位址。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --cluster-
id cluster-1234567
```

您可以將 --region 和 --endpoint 參數與 cluster-id 參數結合使用，以指定系統進行 DescribeClusters 呼叫的方式。例如，如果叢集區域與設定為 AWS CLI 預設的區域不同，則應使用該 --region 參數來使用該區域。此外，您還可以指定用於呼叫的 AWS CloudHSM API 端點，這對於各種網路設定來說可能是必要的，例如使用不使用預設 DNS 主機名稱的 VPC 介面端點。 AWS CloudHSM

PKCS #11 library

使用自訂端點和區域啟動 Linux EC2 執行個體

- 使用設定工具，透過自訂區域和端點來指定叢集中 HSM 的 IP 位址。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id cluster-1234567 --
region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

使用端點和區域啟動 Windows EC2 執行個體

- 使用設定工具，透過自訂區域和端點來指定叢集中 HSM 的 IP 位址。

```
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

使用自訂端點和區域啟動 Linux EC2 執行個體

- 使用設定工具，透過自訂區域和端點來指定叢集中 HSM 的 IP 位址。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

JCE provider

使用自訂端點和區域啟動 Linux EC2 執行個體

- 使用設定工具，透過自訂區域和端點來指定叢集中 HSM 的 IP 位址。

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

使用端點和區域啟動 Windows EC2 執行個體

- 使用設定工具，透過自訂區域和端點來指定叢集中 HSM 的 IP 位址。

```
"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

使用自訂端點和區域啟動 Linux EC2 執行個體

- 使用設定工具，透過自訂區域和端點來指定叢集中 HSM 的 IP 位址。

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id cluster-1234567 --region us-
east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

使用端點和區域啟動 Windows EC2 執行個體

- 使用設定工具，透過自訂區域和端點來指定叢集中 HSM 的 IP 位址。

```
"C:\Program Files\Amazon\CloudHSM\configure-cli.exe" --cluster-
id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

如需 `--cluster-id`、`--region` 和 `--endpoint` 參數的詳細資訊，請參閱 the section called “參數”。

更新 TLS 用戶端伺服器相互驗證的用戶端憑證和金鑰

Example

此範例顯示如何使用`server-client-cert-file`和`--server-client-key-file`參數，透過為下列項目指定自訂金鑰和 SSL 憑證來重新設定 SSL AWS CloudHSM

PKCS #11 library

使用自訂憑證和金鑰在 Linux 上使用用戶端 SDK 5 進行 TLS 用户端和服务器相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 使用設定工具來指定 `ssl-client.crt` 和 `ssl-client.key`。

```
$ 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
```

使用自訂憑證和金鑰進行 TLS 用戶端伺服器與 Windows 用戶端 SDK 5 的相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 對於 PowerShell 解釋器，使用配置工具來指定`ssl-client.crt`和`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

使用自訂憑證和金鑰在 Linux 上使用用戶端 SDK 5 進行 TLS 用户端和服务器相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 使用設定工具來指定 `ssl-client.crt` 和 `ssl-client.key`。

```
$ 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

使用自訂憑證和金鑰在 Linux 上使用用戶端 SDK 5 進行 TLS 用户端和服务器相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 使用設定工具來指定 `ssl-client.crt` 和 `ssl-client.key`。

```
$ 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
```

使用自訂憑證和金鑰進行 TLS 用戶端伺服器與 Windows 用戶端 SDK 5 的相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 對於 PowerShell 解釋器，使用配置工具來指定`ssl-client.crt`和`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

使用自訂憑證和金鑰在 Linux 上使用用戶端 SDK 5 進行 TLS 用户端和服务器相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 使用設定工具來指定 `ssl-client.crt` 和 `ssl-client.key`。

```
$ 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
```

使用自訂憑證和金鑰進行 TLS 用戶端伺服器與 Windows 用戶端 SDK 5 的相互驗證

1. 將您的金鑰和憑證複製到適當的目錄。

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

2. 對於 PowerShell 解釋器，使用配置工具來指定ssl-client.crt和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
```

如需 server-client-cert-file 和 --server-client-key-file 參數的詳細資訊，請參閱 the section called “參數”。

停用用戶端金鑰耐久性設定

Example

此範例使用 --disable-key-availability-check 參數來停用用戶端金鑰耐久性設定。如要使用單一 HSM 執行叢集，您必須停用用戶端金鑰耐久性設定。

PKCS #11 library

在 Linux 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

在 Windows 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-
availability-check
```

OpenSSL Dynamic Engine

在 Linux 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

JCE provider

在 Linux 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

在 Windows 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-
availability-check
```

CloudHSM CLI

在 Linux 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

在 Windows 上停用用戶端 SDK 5 的用戶端金鑰耐久性

- 使用設定工具來停用用戶端金鑰耐久性設定。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-availability-check
```

如需 `--disable-key-availability-check` 參數的詳細資訊，請參閱 the section called “參數”。

管理日誌選項

Example

用戶端 SDK 5 會使用 `log-file`、`log-level`、`log-rotation` 和 `log-type` 參數來管理記錄。

> **Note**
>
> 若要針對 AWS Fargate 或 AWS Lambda 等無伺服器環境設定您的開發套件，建議您將 AWS CloudHSM 日誌類型設定為。`term`用戶端記錄檔將輸出至針對該環境設定的 CloudWatch 記錄檔記錄群組，`stderr`並將其擷取到該環境中。

PKCS #11 library

預設日誌位置

- 如果您未指定檔案的位置，系統會將日誌寫入下列預設位置：

  Linux

```
/opt/cloudhsm/run/cloudhsm-pkcs11.log
```

  Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
```

設定日誌層級，並將其他日誌選項設定為預設值

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-level info
  ```

設定檔案日誌選項

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type file --log-file <file name
   with path> --log-rotation daily --log-level info
  ```

設定終端日誌選項

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type term --log-level info
  ```

OpenSSL Dynamic Engine

預設日誌位置

- 如果您未指定檔案的位置，系統會將日誌寫入下列預設位置：

  Linux

  ```
  stderr
  ```

設定日誌層級，並將其他日誌選項設定為預設值

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-level info
  ```

設定檔案日誌選項

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type <file name> --log-file file --
  log-rotation daily --log-level info
  ```
  ```

設定終端日誌選項

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type term --log-level info
  ```

JCE provider

預設日誌位置

- 如果您未指定檔案的位置，系統會將日誌寫入下列預設位置：

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-jce.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
  ```

設定日誌層級，並將其他日誌選項設定為預設值

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-level info
  ```

設定檔案日誌選項

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type file --log-file <file name> --
  log-rotation daily --log-level info
  ```

設定終端日誌選項

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type term --log-level info
  ```

CloudHSM CLI

預設日誌位置

- 如果您未指定檔案的位置，系統會將日誌寫入下列預設位置：

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-cli.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-cli.log
  ```

設定日誌層級，並將其他日誌選項設定為預設值

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-level info
  ```

設定檔案日誌選項

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type file --log-file <file name> --
  log-rotation daily --log-level info
  ```

設定終端日誌選項

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type term --log-level info
  ```

如需 `log-file`、`log-level`、`log-rotation` 和 `log-type` 參數的詳細資訊，請參閱 the section called “參數”。

放置用戶端 SDK 5 的簽發憑證

Example

此範例使用 `--hsm-ca-cert` 參數來更新用戶端 SDK 5 的簽發憑證位置。
  ```

PKCS #11 library

將用戶端 SDK 5 的簽發憑證安裝於 Linux 上

- 使用設定工具指定簽發憑證的位置。

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

將用戶端 SDK 5 的簽發憑證安裝於 Windows 上

- 使用設定工具指定簽發憑證的位置。

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

OpenSSL Dynamic Engine

將用戶端 SDK 5 的簽發憑證安裝於 Linux 上

- 使用設定工具指定簽發憑證的位置。

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

JCE provider

將用戶端 SDK 5 的簽發憑證安裝於 Linux 上

- 使用設定工具指定簽發憑證的位置。

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

將用戶端 SDK 5 的簽發憑證安裝於 Windows 上

- 使用設定工具指定簽發憑證的位置。

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

CloudHSM CLI

將用戶端 SDK 5 的簽發憑證安裝於 Linux 上

- 使用設定工具指定簽發憑證的位置。

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

將用戶端 SDK 5 的簽發憑證安裝於 Windows 上

- 使用設定工具指定簽發憑證的位置。

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

如需 `--hsm-ca-cert` 參數的詳細資訊，請參閱 the section called “參數”。

參數

-a ***<ENI IP address>***

將指定的 IP 地址新增至用戶端 SDK 5 設定檔案。輸入叢集中 HSM 的任何 ENI IP 位址。如需如何使用此選項的詳細資訊，請參閱引導用戶端 SDK 5。

必要：是

--hsm-ca-cert <customerCA certificate file path>

儲存用於將 EC2 用戶端執行個體連接到叢集的憑證授權機構 (CA) 憑證目錄的路徑。您可在初始化叢集時建立此檔案。按照預設，系統會在下列位置尋找此檔案：

Linux

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

Windows

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

如需有關初始化叢集或放置憑證的詳細資訊，請參閱 ??? 和 ???。

必要：否

--cluster-id ***<cluster ID>***

進行 `DescribeClusters` 呼叫以尋找叢集中與叢集 ID 關聯的所有 HSM 彈性網路介面 (ENI) IP 地址。系統會將 ENI IP 位址新增至組 AWS CloudHSM 態檔案。

> **Note**
>
> 如果您在無法存取公用網際網路的 VPC 中使用 EC2 執行個體的`--cluster-id`參數，則必須建立要連接的介面 VPC 端點。 AWS CloudHSM如需 VPC 端點的詳細資訊，請參閱 ???。

必要：否

--endpoint ***<endpoint>***

指定用於進行DescribeClusters呼叫的 AWS CloudHSM API 端點。您必須結合 `--cluster-id` 設定此選項。

必要：否

--region ***<region>***

指您叢集的區域。您必須結合 `--cluster-id` 設定此選項。

如果您未提供 `--region` 參數，系統會嘗試讀取 `AWS_DEFAULT_REGION` 或 `AWS_REGION` 環境變數來選擇區域。如果未設定這些變數，則除非您在 `AWS_CONFIG_FILE` 環境變數中指定了不同的檔案，否則系統會檢查 AWS config 檔案中 (通常是 `~/.aws/config`) 與您的設定檔相關聯的區域。如果未設定上述任何變數，系統會預設為 `us-east-1` 區域。

必要：否

--server-client-cert-file <client certificate file path>

用於 TLS 用戶端與伺服器相互驗證的用戶端憑證路徑。

只有在您不希望使用我們隨用戶端 SDK 5 提供的預設金鑰和 SSL/TLS 憑證時，才使用此選項。您必須結合 `--server-client-key-file` 設定此選項。

必要：否

--server-client-key-file <client key file path>

TLS 用戶端與伺服器相互驗證所使用的用戶端金鑰路徑。

只有在您不希望使用我們隨用戶端 SDK 5 提供的預設金鑰和 SSL/TLS 憑證時，才使用此選項。您必須結合 `--server-client-cert-file` 設定此選項。

必要：否

--log-level ***<error | warn | info | debug | trace>***

指定系統應寫入日誌檔的最低日誌層級。每個層級包括以前的層級，錯誤作為最低層級，追蹤為最高層級。這表示如果您指定錯誤，系統只會將錯誤寫入日誌。如果您指定追蹤，系統會將錯誤、警告、資訊 (info) 和偵錯訊息寫入日誌檔。如需詳細資訊，請參閱用戶端 SDK 5 記錄。

必要：否

--log-rotation ***<daily | weekly>***

指系統輪換日誌檔的頻率。如需詳細資訊，請參閱用戶端 SDK 5 記錄。

必要：否

--log-file *<file name with path>*

指系統將寫入日誌檔的位置。如需詳細資訊，請參閱用戶端 SDK 5 記錄。

必要：否

--log-type *<term | file>*

指系統是否將日誌寫入檔案或終端。如需詳細資訊，請參閱用戶端 SDK 5 記錄。

必要：否

-h | --help

顯示幫助。

必要：否

-v | --version

顯示版本。

必要：否

--disable-key-availability-check

停用金鑰可用性仲裁的旗標。使用此旗標表示 AWS CloudHSM 應停用金鑰可用性仲裁，而且您可以使用叢集中僅存在於一個 HSM 上的金鑰。如需有關使用此旗標設定金鑰可用性仲裁的詳細資訊，請參閱 ???。

必要：否

--enable-key-availability-check

啟動金鑰可用性仲裁的旗標。使用此旗標表示 AWS CloudHSM 應該使用金鑰可用性仲裁，而且在叢集中的兩個 HSM 上存在這些金鑰之前，不允許您使用金鑰。如需有關使用此旗標設定金鑰可用性仲裁的詳細資訊，請參閱 ???。

預設啟用。

必要：否

--初始disable-validate-key-at化

指定您可以跳過初始化呼叫來驗證後續呼叫之金鑰的權限，以此改善效能。請謹慎使用。

背景：PKCS #11 程式庫中的某些機制支援多部分操作，其中初始化呼叫會驗證您是否可以將金鑰用於後續呼叫。這一操作需要對 HSM 進行驗證呼叫，這會增加整體操作的延遲。此選項可讓您停用後續呼叫，並可能改善效能。

必要：否

--初始enable-validate-key-at化

指您應該使用初始化呼叫來驗證後續呼叫的金鑰權限。此為預設選項。使用 `disable-validate-key-at-init` 暫停這些初始化呼叫後，再使用 `enable-validate-key-at-init` 恢復這些初始化呼叫。

必要：否

相關主題

- DescribeClustersAPI 操作
- describe-clusters AWS CLI
- Get-HSM2Cluster PowerShell 指令程式
- 引導用戶端 SDK 5
- AWS CloudHSM VPC 端點
- 管理用戶端 SDK 5 金鑰持久性設定
- 用戶端 SDK 5 記錄

用戶端 SDK 5 設定工具的進階組態

用戶端 SDK 5 設定工具包含進階組能，這些組能不包含在大部分客戶使用的一般功能中。進階組態提供額外的功能。

- PKCS #11 的進階組態
  - 使用 PKCS #11 連線到多個插槽
  - PKCS #11 的重試命令
- 適用於 JCE 的進階組態
  - 使用 JCE 提供者連線到多個叢集
  - 重試適用於 JCE 的命令
  - 使用 JCE 擷取金鑰

- OpenSSL 的進階組態
  - 適用於 OpenSSL 的重試命令
- AWS CloudHSM 命令列介面 (CLI) 的進階組態
  - 使用 CloudHSM CLI 連線到多個叢集

用戶端 SDK 3 設定工具

使用用戶端 SDK 3 設定工具啟動用戶端常駐程式並設定 CloudHSM 管理公用程式。

語法

```
configure -h | --help
          -a <ENI IP address>
          -m [-i <daemon_id>]
          --ssl --pkey <private key file> --cert <certificate file>
          --cmu <ENI IP address>
```

範例

這些範例會示範如何使用 configure 工具。

Example ：更新用 AWS CloudHSM 戶端和金鑰的 HSM 資料

此範例使用的-a參數configure來更新用 AWS CloudHSM 戶端和 key_mgmt_util 的 HSM 資料。若要使用此 -a 參數，您必須擁有叢集中其中一個 HSM 的 IP 地址。使用主控台或 AWS CLI 取得 IP 地址。

取得 HSM (主控台) 的 IP 位址

1. 開啟主 AWS CloudHSM 控台，網址為 https://console.aws.amazon.com/cloudhsm/home。
2. 若要變更 AWS 區域，請使用頁面右上角的區域選擇器。
3. 若要開啟叢集詳細資訊頁面，請在叢集表格中選擇叢集 ID。
4. 若要取得 IP 地址，請在 HSM 索引標籤上，選擇 ENI IP 地址下列出的其中一個 IP 地址。

取得 HSM 的 IP 位址 ()AWS CLI

- 使用 AWS CLI中的 describe-clusters 命令取得 HSM 的 IP 地址。在命令輸出中，HSM 的 IP 地址是 `EniIp` 的值。

```
$  aws cloudhsmv2 describe-clusters


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

：更新 HSM 資料

1. 更新-a參數之前，請停止用 AWS CloudHSM 戶端。這可防止在 configure 編輯用戶端的組態檔案時可能發生的衝突。如果用戶端已停止，而此命令不會帶來任何影響，因此您可以在指令碼中使用它。

   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

- 用於 Windows 用戶端 1.1.2+：

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- 用於 Windows 用戶端 1.1.1 和更早版本：

  在您啟動用 AWS CloudHSM 戶端的命令視窗中使用 Ctrl + C。

2. 此步驟使用 configure 的 `-a` 參數，來將 `10.0.0.9` ENI IP 地址新增至組態檔案。

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. 接下來，重新啟動 AWS CloudHSM 用戶端。用戶端啟動時，會使用組態檔案中的 ENI IP 地址來查詢叢集。接著會將叢集中所有 HSM 的 ENI IP 地址新增至 `cluster.info` 檔案。

   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

- 用於 Windows 用戶端 1.1.2+：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 用於 Windows 用戶端 1.1.1 和更早版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

當命令完成時，AWS CloudHSM 用戶端和 key_mgmt_util 使用的 HSM 資料會完整且正確。

Example ：從用戶端 SDK 3.2.1 及更早版本更新 CMU 的 HSM 資料

此範例使用 -m configure 命令，將來自 cluster.info 檔案的更新 HSM 資料複製到 cloudhsm_mgmt_util 使用的 cloudhsm_mgmt_util.cfg 檔案。請搭配用戶端 SDK 3.2.1 及更早版本隨附的 CMU 使用此功能。

- 執行之前-m，請停止用 AWS CloudHSM 戶端、執行-a命令，然後重新啟動 AWS CloudHSM 用戶端，如前面的範例所示。這可確保從 cluster.info 檔案複製到 cloudhsm_mgmt_util.cfg 檔案的資料既完整且準確。

  Linux

  ```
  $ sudo /opt/cloudhsm/bin/configure -m
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\bin\ configure.exe -m
  ```

Example ：從用戶端 SDK 3.3.0 及更新版本更新 CMU 的 HSM 資料

此範例使用 configure 的 --cmu 參數來更新 CMU 的 HSM 資料。請搭配用戶端 SDK 3.3.0 及更新版本隨附的 CMU 使用此功能。如需有關使用 CMU 的詳細資訊，請參閱使用 CloudHSM 管理公用程式 (CMU) 管理使用者和使用用戶端 SDK 3.2.1 及更早版本隨附的 CMU。

- 使用此 --cmu 參數傳遞叢集中 HSM 的 IP 地址。

  Linux

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

  Windows

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

參數

-h | --help

顯示命令的語法。

必要：是

-a *<ENI IP address>*

將指定的 HSM 彈性網路界面 (ENI) IP 地址新增至 AWS CloudHSM 組態檔案。輸入叢集中任何一個 HSM 的 ENI IP 地址。選擇哪一個都可以。

若要取得叢集中 HSM 的 ENI IP 位址，請使用DescribeClusters作業、描述叢集命令或 AWS CLI 指令程式。Get-HSM2Cluster PowerShell

> **Note**
>
> 執行 `-a`configure命令之前，請先停止用 AWS CloudHSM 戶端。然後，當`-a`命令完成時，重新啟動用 AWS CloudHSM 戶端。如需詳細資訊，請參閱範例。

此參數會編輯以下組態檔案：

- `/opt/cloudhsm/etc/cloudhsm_client.cfg`：由用 AWS CloudHSM 戶端和密鑰使用。
- `/opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg`：由 cloudhsm_mgmt_util 使用。

用 AWS CloudHSM 戶端啟動時，會使用其組態檔案中的 ENI IP 位址來查詢叢集，並使用叢集中所有 HSM 的正確 ENI IP 位址更新`cluster.info`檔案 (`/opt/cloudhsm/daemon/1/cluster.info`)。

必要：是

-m

更新 CMU 所使用組態檔案中的 HSM ENI IP 地址。

> **Note**
>
> 此 `-m` 參數可與用戶端 SDK 3.2.1 及更早版本的 CMU 搭配使用。對於用戶端 SDK 3.3.0 及更新版本的 CMU，請參閱 `--cmu` 參數，該參數可簡化更新 CMU 的 HSM 資料的程序。

當您更新的-a參數configure並啟動 AWS CloudHSM 用戶端時，用戶端精靈會查詢叢集，並使用叢集中所有 HSM 的正確 HSM IP 位址更新cluster.info檔案。將 HSM IP 地址從 cluster.info 複製到 cloudhsm_mgmt_util 使用的 cloudhsm_mgmt_util.cfg 組態檔案，從而執行 -m configure 命令並完成更新。

執行-aconfigure命令之前，請務必執行命令並重新啟動 AWS CloudHSM 用戶端。-m這可確保從 cluster.info 複製到 cloudhsm_mgmt_util.cfg 檔案的資料既完整且準確。

必要：是

-i

指定替代的用戶端常駐程式。預設值代表 AWS CloudHSM 用戶端。

預設：1

必要：否

--ssl

以指定的私有金鑰和憑證取代叢集的 SSL 金鑰和憑證。使用此參數時，需要 --pkey 和 --cert 參數。

必要：否

--pkey

指定新的私有金鑰。輸入包含私有金鑰之檔案的路徑和名稱。

必要：如果指定的是 -ssl，則為是。否則不應使用此項目。

--cert

指定新的憑證。輸入包含憑證之檔案的路徑和名稱。此憑證應會鏈結至 customerCA.crt 憑證，即用於初始化叢集的自簽憑證。如需詳細資訊，請參閱初始化叢集。

必要：如果指定的是 -ssl，則為是。否則不應使用此項目。

--cmu ***<ENI IP address>***

將 -a 和 -m 參數合併為一個參數。將指定的 HSM elastic network interface (ENI) IP 位址新增至 AWS CloudHSM 組態檔，然後更新 CMU 組態檔案。輸入叢集中任何 HSM 的任何 IP 地址。如需用戶端 SDK 3.2.1 及更早版本，請參閱使用用戶端 SDK 3.2.1 及更早版本隨附的 CMU。

必要：是

相關主題

- 設定 key_mgmt_util

CloudHSM 命令列介面 (CLI)

CloudHSM CLI 可協助管理員管理使用者和加密使用者管理叢集中的金鑰。它包括可用於創建，刪除和列出用戶，更改用戶密碼，更新用戶多因素身份驗證（MFA）的工具。它還包括產生、刪除、匯入和匯出金鑰、取得和設定屬性、尋找金鑰以及執行密碼編譯作業的指令。

如需已定義的 CloudHSM CLI 使用者清單，請參閱 使用 CloudHSM CLI 管理 HSM 使用者 。如需 CloudHSM CLI 的已定義金鑰屬性清單，請參閱。CloudHSM CLI 的金鑰屬性如需如何使用 CloudHSM CLI 管理金鑰的相關資訊，請參閱。使用 CloudHSM CLI 管理金鑰

如需快速入門，請參閱 CloudHSM 命令列介面 (CLI) 入門。如需 CloudHSM CLI 命令的詳細資訊以及使用命令的範例，請參閱 CloudHSM CLI 命令的參考資料。

主題

- CloudHSM 命令列介面 (CLI) 支援的平台
- CloudHSM 命令列介面 (CLI) 入門
- 互動式和單一命令模式
- CloudHSM CLI 的金鑰屬性
- 從用戶端 SDK 3 CMU 和 KMU 遷移至用戶端開發套件 5 CloudHSM CLI
- CLI 的進階組態
- CloudHSM CLI 命令的參考資料

CloudHSM 命令列介面 (CLI) 支援的平台

Linux 支援

支援平台	X86_64 架構	ARM 架構
Amazon Linux 2	是	是

支援平台	X86_64 架構	ARM 架構
Amazon Linux 2023	是	是
CentOS 7 (7.8+)	是	否
紅帽企業版 7 (7.8+)	是	否
紅帽企業版 8 (8.3 以上)	是	否
紅帽企業版 9 (9.2+)	是	是
Ubuntu 20.04 LTS	是	否
Ubuntu 22.04 LTS	是	是

注意：軟件開發套件 5.4.2 是最後一個提供 CentOS 8 平台支援的發行版本。如需詳細資訊，請參閱 CentOS 網站。

Windows 支援

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

CloudHSM 命令列介面 (CLI) 入門

CloudHSM 命令列介面 (CLI) 可讓您管理叢集中的 AWS CloudHSM 使用者。使用本主題可開始進行基本的 HSM 使用者管理工作，例如建立使用者、列出使用者以及將 CloudHSM CLI 連線到叢集。

安裝 CloudHSM CLI

使用以下命令來下載和安裝 CloudHSM CLI。

Amazon Linux 2

Amazon Linux 2 (x86_64 架構)：

```
$ 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
```

Amazon Linux 2 (ARM64 架構) :

```
$ 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

Amazon 架構

```
$ 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
```

在 ARM64 架構上的 Amazon

```
$ 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+)

CentOS 7 (x86_64 架構) :

```
$ 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+)

RHEL 7 (x86_64 架構) :

```
$ 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+)

RHEL 8 (x86_64 架構)：

```
$ 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+)

在 64 架構上的 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
```

關於 ARM64 架構的第 9 步：

```
$ 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

Ubuntu 20.04 LTS (x86_64 架構)：

```
$ 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

Ubuntu 22.04 LTS (x86_64 架構)：

```
$ 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
```

關於 ARM64 架構的 Ubuntu 22.04 研究所：

```
$ 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

對於 x86_64 架構上的視窗伺服器 2016，請 PowerShell 以系統管理員身分開啟，然後執行下列命令：

```
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

對於 x86_64 架構上的 Windows 伺服器 2019，請 PowerShell 以系統管理員身分開啟並執行下列命令：

```
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
```

使用下列命令來設定 CloudHSM CLI。

為用戶端 SDK 5 引導 Linux EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 地址。

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

為用戶端 SDK 5 引導 Windows EC2 執行個體

- 使用設定工具指定叢集中 HSM 的 IP 地址。

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

使用 CloudHSM CLI

1. 使用下列命令來啟動 CloudHSM CLI。

   Linux

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

   Windows

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

2. 使用 login 命令登入叢集。所有使用者都可以使用此命令。

   以下範例中的命令將 admin 登入，這是預設的管理員賬戶。您已在啟用叢集時設定此使用者的密碼。

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

   系統會提示您輸入密碼。您輸入密碼，然後輸出顯示命令已成功。

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

3. 執行 user list 命令以列出叢集上的所有使用者。

```
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. 用於 user create 建立名為 **example_user** 的 CU 使用者。

   您可以建立 CU，因為在上一個步驟中，您以系統管理員使用者身分登入。只有管理員使用者可以執行使用者管理工作，例如建立和刪除使用者，以及變更其他使用者的密碼。

```
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. 用 user list 來列出叢集上的所有使用者。

```
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. 使用logout指令登出 AWS CloudHSM 叢集。

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

7. 使用 quit 命令停止 CLI。

```
aws-cloudhsm > quit
```

互動式和單一命令模式

在 CloudHSM CLI 中，您可以使用兩種不同的方式執行命令，分別是：單一命令模式和互動式模式。互動式模式專為使用者設計，而單一命令模式則是專為指令碼設計。

> **Note**
>
> 所有命令均在互動式模式和單一命令模式下工作。

互動式模式

使用以下命令來啟動 CloudHSM CLI 互動式模式。

Linux

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

Windows

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

在互動式模式下使用 CLI 時，您可使用 login 命令登入使用者帳戶。

若要列出所有 CloudHSM CLI 命令，請執行以下命令：

```
aws-cloudhsm > help
```

若要取得某一 CloudHSM CLI 命令的語法，請執行以下命令：

```
aws-cloudhsm >  help <command-name>
```

若要取得 HSM 上的使用者清單，請輸入 user list。

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

若要結束 CloudHSM CLI 工作階段，請執行以下命令：

```
aws-cloudhsm > quit
```

單一命令模式

如要在單一命令模式下執行 CloudHSM CLI，則需設定兩個環境變數以提供憑證：CLOUDHSM_PIN 和 CLOUDHSM_ROLE：

```
$ export CLOUDHSM_ROLE=admin
```

```
$ export CLOUDHSM_PIN=admin_username:admin_password
```

環境變數設定完成後，您可使用儲存在環境中的憑證來執行命令。

```
$ cloudhsm-cli user change-password --username alice --role crypto-user
Enter password:
Confirm password:
{
    "error_code": 0,
    "data": {
      "username": "alice",
      "role": "crypto-user"
    }
}
```

CloudHSM CLI 的金鑰屬性

本主題旨在說明如何使用 CloudHSM CLI 設定金鑰屬性。CloudHSM CLI 中的金鑰屬性可以定義金鑰的類型、金鑰的運作方式或金鑰的標記方式。某些屬性定義唯一的特性 (例如，金鑰的類型)。其他屬性可以設定為 true 或 false (變更它們會啟動或停用金鑰功能的一部分)。

如需展示如何使用金鑰屬性的範例，請參閱父命令 金鑰 下列出的命令。

支援的屬性

最佳實務是，只為您希望更具有限制性的屬性設定值。如果您未指定值，CloudHSM CLI 會使用下表中指定的預設值。

下表列出金鑰屬性、可能值、預設值及相關註記。值欄位中的空白儲存格表示屬性沒有獲派指定預設值。

CloudHSM CLI屬性	Value	可以使用密鑰集屬性進行修改	可在建立金鑰時設定
always-sensitive	如果 sensitive 一直被設置為 True 並且從未更改過，則該值是 True。	否	否
check-value	金鑰的金鑰檢查值。如需詳細資訊，請參閱其他詳細資訊。	否	否
class	可能的值：secret-key 、public-key 和 private-key 。	否	是
curve	用於產生 EC 金鑰對的橢圓曲線。 有效值：secp224r1 、secp256r1 、prime256v1 、secp384r1 、secp256k1 和 secp521r1	否	可使用 RSA 設定，而無法使用 EC 進行設定
decrypt	預設：False	是	是

CloudHSM CLI屬性	Value	可以使用密鑰集屬性進行修改	可在建立金鑰時設定
derive	預設：False	是	是
destroyable	預設：True	是	是
ec-point	如果是 EC 金鑰，則以十六進位格式對 ANSI X9.62 ECPoint 值「Q」進行 DER 編碼。 對於其他金鑰類型，這個屬性不存在。	否	否
encrypt	預設：False	是	是
extractable	預設：True	否	是
id	預設值：空	否	是
key-length-bytes	產生 AES 密鑰所需。 有效值：16、24 和 32 位元組。	否	否
key-type	可能的值：aes、rsa 和 ec。	否	是
label	預設值：空	是	是
local	預設值：True 適用於在 HSM 中產生的金鑰，False 適用於匯入至 HSM 的金鑰。	否	否
modifiable	預設：True	否	否

CloudHSM CLI屬性	Value	可以使用密鑰集屬性進行修改	可在建立金鑰時設定
modulus	用於產生 RSA 金鑰對的模數。對於其他金鑰類型，這個屬性不存在。	否	否
modulus-size-bits	產生 RSA 金鑰對時需要。 最小值為 2048。	否	可使用 RSA 設定，而無法使用 EC 進行設定
never-extractable	如果可擷取值從未設置為 False，則該值為 True。 如果可擷取值已被設置為 True，則該值為 False。	否	否
private	預設：True	否	是
public-exponent	產生 RSA 金鑰對時需要。 有效值：值必須為大於或等於 65537 的奇數。	否	可使用 RSA 設定，而無法使用 EC 進行設定
sensitive	預設： • 此值為 True，適用於 AES 金鑰以及 EC 和 RSA 私有金鑰。 • 此值為 False，適用於 EC 和 RSA 公有金鑰。	否	可使用私有金鑰設定，而不能使用公有金鑰設定。

CloudHSM CLI屬性	Value	可以使用密鑰集屬性進行修改	可在建立金鑰時設定
sign	預設： • 此值為 True，適用於 AES 金鑰。 • 此值為 False，適用於 RSA 和 EC 金鑰。	是	是
token	預設：False	否	是
trusted	預設：False	是	否
unwrap	預設：False	是	是
unwrap-template	這些值應使用已套用於以此包裝金鑰取消包裝之任何金鑰的屬性範本。	是	否
verify	預設： • 此值為 True，適用於 AES 金鑰。 • 此值為 False，適用於 RSA 和 EC 金鑰。	是	是
wrap	預設：False	是	是
wrap-template	這些值應使用屬性範本來匹配使用此包裝金鑰所包裝的金鑰。	是	否
wrap-with-trusted	預設：False	是	是

其他詳細資訊

檢查值

檢查值是 HSM 匯入或產生金鑰時所產生之金鑰的 3 位元組雜湊或總和檢查碼。您也可以在 HSM 之外計算檢查值，例如在匯出金鑰之後。然後，您可以比較檢查值以確認金鑰的身分和完整性。如需獲取金鑰的檢查值，請使用帶有詳細標誌的金鑰列表。

AWS CloudHSM 使用下列標準方法來產生檢查值：

- 對稱密鑰：使用金鑰加密零塊的結果的前 3 個位元組。
- 非對稱金鑰配對：公有金鑰 SHA-1 雜湊的前 3 個位元組。
- HMAC 金鑰：目前不支援 HMAC 金鑰的 KCV。

相關主題

- 金鑰
- CloudHSM CLI 命令的參考資料

從用戶端 SDK 3 CMU 和 KMU 遷移至用戶端開發套件 5 CloudHSM CLI

使用本主題可移轉使用用戶端 SDK 3 命令列工具、CloudHSM 管理公用程式 (CMU) 和金鑰管理公用程式 (KMU) 的工作流程，改為使用用戶端 SDK 5 命令列工具 CloudHSM CLI。

在中 AWS CloudHSM，客戶應用程式會使用用戶 AWS CloudHSM 端軟體開發套件 (SDK) 執行密碼編譯作業。客戶端 SDK 5 是繼續添加新功能和平台支持的主要 SDK。本主題提供從用戶端 SDK 3 遷移至命令列工具的用戶端 SDK 5 的特定詳細資料。

客戶端 SDK 3 包括兩個單獨的命令行工具：用於管理用戶的 CMU 和用於管理密鑰和使用密鑰執行操作的 KMU。客戶端 SDK 5 將 CMU 和 KMU（客戶端 SDK 3 提供的工具）的功能整合到一個單一的工具中，. CloudHSM 命令列介面 (CLI) 使用者管理作業可以在子指令使用者和規定人數下找到。您可以在金鑰子指令下找到金鑰管理作業，而且可以在 crypto 子指令下找到密碼編譯作業。CloudHSM CLI 命令的參考資料如需指令的完整清單，請參閱。

> **Note**
>
> 如果您在用戶端 SDK 3 中仰賴syncKey跨叢集同步處理syncUser功能，請繼續使用 CMU。用戶端開發套件 5 中的 CloudHSM CLI 目前不支援此功能。

如需有關移轉至用戶端 SDK 5 的指示，請參閱從用戶端 SDK 3 遷移至用戶端 SDK 5。如需移轉的優點，請參閱用戶端 SDK 5 的優點 。

CLI 的進階組態

命 AWS CloudHSM 令列介面 (CLI) 包括下列進階設定，這不是大多數客戶使用的一般組態的一部分。這些組態提供額外的功能。

- 連接至多個叢集

使用 CloudHSM CLI 連線到多個叢集

透過用戶端 SDK 5，您可以將 CloudHSM CLI 設定為允許從單一 CLI 執行個體連線到多個 CloudHSM 叢集。

使用本主題中的指示，使用 CloudHSM CLI 使用多叢集功能與多個叢集連線。

主題

- 多叢集先決條件
- 針對多叢集功能設定 CloudHSM CLI
- 配置-cli 添加集群
- 配置-cli 移除叢集
- 使用多個叢集

多叢集先決條件

- 您想要連線到的兩個或多個 AWS CloudHSM 叢集，以及其叢集憑證。
- 具有安全群組的 EC2 執行個體正確設定為連線到上述所有叢集。如需如何設定叢集和用戶端執行個體的詳細資訊，請參閱入門 AWS CloudHSM。
- 若要設定多叢集功能，您必須已下載並安裝 CloudHSM CLI。若您尚未完成此動作，請參閱 ??? 中的說明。
- 您將無法存取使用配置的叢集，`./configure-cli[.exe] -a`因為它不會與`cluster-id`. 您可以按照本指南中所`config-cli add-cluster`述重新配置它。

針對多叢集功能設定 CloudHSM CLI

若要針對多叢集功能設定 CloudHSM CLI，請依照下列步驟執行：

1. 識別您要連線到的叢集。
2. 使用設定 cli 子命令將這些叢集新增至 CloudHSM CLI 組態，如下所述。add-cluster
3. 重新啟動任何 CloudHSM CLI 程序，以使新組態生效。

配置-cli 添加集群

連線至多個叢集時，請使用configure-cli add-cluster指令將叢集新增至您的組態。

語法

```
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]
```

範例

使用 **cluster-id** 參數新增叢集

Example

搭配使用 configure-cli add-cluster 和 cluster-id 參數，將叢集 (ID為 cluster-1234567) 新增至您的組態。

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**
>
> 如果 `configure-cli add-cluster` 與 `cluster-id` 參數搭配使用不會導致新增叢集，請參閱下列範例，以取得此命令的更長版本，此命令也需要 `--region` 和 `--endpoint` 參數來識別要新增的叢集。例如，如果叢集區域與設定為 AWS CLI 預設值的區域不同，則應使用 `--region` 參數來使用正確的區域。此外，您還可以指定用於呼叫的 AWS CloudHSM API 端點，這對於各種網路設定來說可能是必要的，例如使用不使用預設 DNS 主機名稱的 VPC 介面端點。 AWS CloudHSM

使用 **`cluster-id`**、**`endpoint`** 和 **`region`** 參數新增叢集

Example

搭配使用 `configure-cli add-cluster` 以及 `cluster-id`、`endpoint` 和 `region` 參數將叢集 (ID為 `cluster-1234567`) 新增至您的組態。

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
```

如需 `--cluster-id`、`--region` 和 `--endpoint` 參數的詳細資訊，請參閱 the section called “參數”。

參數

--cluster-id ***<Cluster ID>***

進行 `DescribeClusters` 呼叫以尋找叢集中與叢集 ID 關聯的所有 HSM 彈性網路介面 (ENI) IP 地址。系統會將 ENI IP 位址新增至組 AWS CloudHSM 態檔案。

> **Note**
>
> 如果您在無法存取公用網際網路的 VPC 中使用 EC2 執行個體的`--cluster-id`參數，則必須建立要連接的介面 VPC 端點。 AWS CloudHSM如需 VPC 端點的詳細資訊，請參閱 ???。

必要：是

--endpoint ***<Endpoint>***

指定用於進行`DescribeClusters`呼叫的 AWS CloudHSM API 端點。您必須結合 `--cluster-id` 設定此選項。

必要：否

--hsm-ca-cert <HsmCA Certificate Filepath>

指定 HSM CA 憑證的檔案路徑。

必要：否

--region ***<Region>***

指您叢集的區域。您必須結合 `--cluster-id` 設定此選項。

如果您未提供 `--region` 參數，系統會嘗試讀取 `AWS_DEFAULT_REGION` 或 `AWS_REGION` 環境變數來選擇區域。如果未設定這些變數，則除非您在 `AWS_CONFIG_FILE` 環境變數中指定了不同的檔案，否則系統會檢查 AWS config 檔案中 (通常是 `~/.aws/config`) 與您的設定檔相關聯的區域。如果未設定上述任何變數，系統會預設為 `us-east-1` 區域。

必要：否

--server-client-cert-file <Client Certificate Filepath>

用於 TLS 用戶端與伺服器相互驗證的用戶端憑證路徑。

只有在您不希望使用我們隨用戶端 SDK 5 提供的預設金鑰和 SSL/TLS 憑證時，才使用此選項。您必須結合 `--server-client-key-file` 設定此選項。

必要：否

--server-client-key-file <Client Key Filepath>

TLS 用戶端與伺服器相互驗證所使用的用戶端金鑰路徑。

只有在您不希望使用我們隨用戶端 SDK 5 提供的預設金鑰和 SSL/TLS 憑證時，才使用此選項。您必須結合 `--server-client-cert-file` 設定此選項。

必要：否

配置-cli 移除叢集

使用 CloudHSM CLI 連線到多個叢集時，請使用指`configure-cli remove-cluster`令從組態中移除叢集。

語法

```
configure-cli remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

範例

使用 **cluster-id** 參數移除叢集

Example

搭配使用 `configure-cli remove-cluster` 和 `cluster-id` 參數，從您的組態中移除叢集 (ID 為 `cluster-1234567`)。

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
```

如需 `--cluster-id` 參數的詳細資訊，請參閱 the section called “參數”。

參數

--cluster-id ***<Cluster ID>***

要從配置中移除的叢集識別碼。

必要：是

使用多個叢集

使用 CloudHSM CLI 設定多個叢集之後，請使用命`cloudhsm-cli`令與它們互動。

範例

使用互動模式**`cluster-id`**時設定預設值

Example

搭配`cluster-id`參數使用，從您的組態設定預設叢集 (ID 為`cluster-1234567`)。???

Linux

```
$ cloudhsm-cli interactive --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\cloudhsm-cli.exe interactive --cluster-
id cluster-1234567
```

在執行單一命令**`cluster-id`**時設定

Example

使用`cluster-id`參數可設定要???從中取得的叢集 (ID 為`cluster-1234567`)。

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 命令的參考資料

CloudHSM CLI 可協助管理員管理其 AWS CloudHSM 叢集中的使用者。CloudHSM CLI 可以在兩種模式下執行：互動式模式和單一命令模式。如需快速入門，請參閱 CloudHSM 命令列介面 (CLI) 入門。

若要執行大多數 CloudHSM CLI 命令，您必須啟動 CloudHSM CLI 並登入 HSM。如果您新增或刪除 HSM，請更新 CloudHSM CLI 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

下列各主題說明 CloudHSM CLI 中的命令。

Command	描述	使用者類型
叢集啟動	啟動 CloudHSM 叢集並確認叢集是新的叢集。完成此操作後，方可執行其他操作。	未激活的管理員
叢集 hsm-info	列出叢集中的 HSM。	所有[1]，包括未經身分驗證的使用者。無需登入。
加密符號	使用 EC 私密金鑰和 ECDSA 簽署機制產生簽章。	加密使用者 (CU)
加密符號 RSA-pkcs	使用 RSA 私密金鑰和 RSA-PKCS 簽章機制產生簽章。	加密使用者
加密符號 rsa-pkcs-pss	使用 RSA 私密金鑰和 RSA-PCCS-PSS 簽署機制產生簽章。	加密使用者
加密貨幣驗證	確認檔案已透過指定的公開金鑰在 HSM 中簽署。驗證簽章是使用 ECDSA 簽署機制產生的。將已簽署的檔案與來源檔案進行比較，並根據指定的 ecdsa 公開金鑰和簽署機制，判斷兩者是否與密碼編譯相關。	加密使用者

Command	描述	使用者類型
加密貨幣驗證 RSA-pkcs	確認檔案已透過指定的公開金鑰在 HSM 中簽署。驗證簽章是使用 RSA-PKCS 簽署機制產生的。將已簽署的檔案與來源檔案進行比較，並根據指定的 rsa 公開金鑰和簽署機制，判斷兩者是否與密碼編譯相關。	加密使用者
加密驗證 rsa-pkcs-pss	確認檔案已透過指定的公開金鑰在 HSM 中簽署。驗證簽章是否使用 RSA-PKCS-PSS 簽署機制產生。將已簽署的檔案與來源檔案進行比較，並根據指定的 rsa 公開金鑰和簽署機制，判斷兩者是否與密碼編譯相關。	加密使用者
刪除金鑰	從 AWS CloudHSM 叢集中刪除金鑰。	加密使用者
產生金鑰的檔案	在 AWS CloudHSM 叢集中產生金鑰檔案。	加密使用者
關鍵 generate-asymmetric-pair RSA	在 AWS CloudHSM 叢集中產生非對稱 RSA 金鑰組。	加密使用者
關鍵 generate-asymmetric-pair ec	在叢集中產生非對稱的橢圓曲線 (EC) key pair 組。 AWS CloudHSM	加密使用者
產生對稱 AES 金鑰	在 AWS CloudHSM 叢集中產生對稱 AES 金鑰。	加密使用者
產生對稱通用私密金鑰	在 AWS CloudHSM 叢集中產生對稱的一般密鑰。	加密使用者

Command	描述	使用者類型
密鑰進口 PEM	將 PEM 格式金鑰匯入 HSM。您可以使用此命令匯入在 HSM 之外產生的公有金鑰。	加密使用者
列出金鑰	尋找 AWS CloudHSM 叢集中目前使用者的所有金鑰。	加密使用者
金鑰複製	將金鑰從來源叢集複製到複製的目的地叢集。	加密使用者
設定金鑰屬性	設定 AWS CloudHSM 叢集中金鑰的屬性。	加密使用者可以執行此命令，管理員可以設定受信任的屬性。
共用金鑰	與 AWS CloudHSM 叢集中的其他 CU 共用金鑰。	加密使用者
取消共用金鑰	與 AWS CloudHSM 叢集中的其他 CU 解除共用金鑰。	加密使用者
密鑰展開一個-gcm	使用 AES 包裝密鑰和 AES-GCM 解包機制將有效負載密鑰解開到集群中。	加密使用者
密鑰展開 aes-no-pad	使用 AES 包裝密鑰和 AES-NO-PAD 解包機制將有效負載密鑰解開到集群中。	加密使用者
密鑰打開一個-pkcs5 墊	使用 AES 包裝密鑰和 AES-PKCS5-PAD 解包機制解開有效載荷密鑰。	加密使用者
密鑰展開 aes-zero-pad	使用 AES 包裝密鑰和 AES-ZER-PAD 解包機制將有效負載密鑰解開到集群中。	加密使用者

Command	描述	使用者類型
密鑰展開 cloudhsm-aes-gcm	使用 AES 包裝密鑰和 CLOUDHSM-AES-GCM 解包機制將有效負載密鑰解開到集群中。	加密使用者
密鑰解開 RSA-艾斯	使用 RSA 私鑰和 RSA-AES 解包機制解開有效負載密鑰。	加密使用者
密鑰解開 RSA-工作	使用 RSA 私密金鑰和 RSA-OAEP 解包機制解除包裝承載金鑰。	加密使用者
密鑰解開 RSA-pkcs	使用 RSA 私密金鑰和 RSA-PKCS 解包機制解除包裝承載金鑰。	加密使用者
鑰匙包裝 AES-克厘米	使用 HSM 上的 AES 金鑰和 AES-GCM 包裝機制來封裝承載金鑰。	加密使用者
按鍵包裝 aes-no-pad	使用 HSM 上的 AES 金鑰和 AES-NO-PAD 環繞機制來封裝承載金鑰。	加密使用者
密鑰包裝一個-pkcs5 墊	使用 HSM 和 AES-PKCS5-PAD 包裝機制上的 AES 金鑰來封裝承載金鑰。	加密使用者
按鍵包裝 aes-zero-pad	使用 HSM 上的 AES 金鑰和 AES-ZER-PAD 環繞機制來封裝承載金鑰。	加密使用者
按鍵包裝 cloudhsm-aes-gcm	使用 HSM 和 CLOUDHSM-AES-GCM 包裝機制上的 AES 金鑰來封裝承載金鑰。	CUS

Command	描述	使用者類型
密鑰包裝 RSA-艾斯	使用 HSM 上的 RSA 公開金鑰和 RSA-AES 封裝機制來封裝承載金鑰。	加密使用者
密鑰包裝 RSA-	使用 HSM 上的 RSA 公開金鑰和 RSA-OAEP 包裝機制來封裝承載金鑰。	加密使用者

Command	描述	使用者類型

命key wrap rsa-pkcs令會使用 HSM 上的 RSA 公開金鑰和包裝機制來封RSA-PKCS裝承載金鑰。裝載金鑰的extractable 屬性必須設定為true。
 	
只有金鑰的擁有者 (即建立金鑰的加密使用者 (CU) 才能包裝金鑰。共用金鑰的使用者可以在密碼編譯作業中使用金鑰。
 	
若要使用此key wrap rsa-pkcs命令，您必須先在 AWS CloudHSM 叢集中具有 RSA 金鑰。您可以使用關鍵 generate-asymmetric-pair 指令和wrap屬性設定為來產生 RSA 金鑰配對。true
 	
使用者類型
 	
下列類型的使用者可以執行此命令。
 	
• 加密使用者 (CU)
 	
要求
 	
• 若要執行此命令，必須以 CU 的身分登入。
 	
語法
 	
`aws-cloudhsm > help key wrap rsa-pkcs`
 	
`Usage: key wrap rsa-pkcs [OPTIONS] --payload-filter [ <PAYLOAD_`
	使用 HSM 上的 RSA 公開金鑰和 RSA-PKCS 包裝機制來封裝承載金鑰。	加密使用者

Command	描述	使用者類型
登入	登入您的 AWS CloudHSM 叢集。	管理員、加密使用者 (CU) 和設備使用者 (AU)
登出	登出 AWS CloudHSM 叢集。	管理員、加密使用者和設備使用者 (AU)
刪除規定人數字符簽署	刪除規定人數授權服務的一個或多個權杖。	管理員
產生規定人數字符簽署	產生規定人數授權服務的權杖。	管理員
列出規定人數字符簽署	列出 CloudHSM 叢集中所有權杖簽署的規定人數權杖。	所有[1]，包括未經身分驗證的使用者。無需登入。
法定令牌符號 list-quorum-values	列出 CloudHSM 叢集中規定人數值集。	所有[1]，包括未經身分驗證的使用者。無需登入。
規定人數字符簽署 列出逾時	取得所有權杖類型的權杖逾時期間 (以秒為單位)。	管理員和加密使用者
法定令牌符號 set-quorum-value	為規定人數授權服務設定新的規定人數值。	管理員
規定人數字符簽署 設定逾時	設定所有權杖類型的權杖逾時期間 (以秒為單位)。	管理員
變更使用者 MFA	變更使用者的多重要素驗證 (MFA) 策略。	管理員、加密使用者
變更使用者密碼	變更 HSM 上使用者的密碼。任何使用者都可以變更自己的密碼。加密使用者可以變更任何人的密碼。	管理員、加密使用者
建立使用者	在 AWS CloudHSM 叢集中建立使用者。	管理員

Command	描述	使用者類型
刪除使用者	刪除 AWS CloudHSM 叢集中的使用者。	管理員
使用者清單	列出 AWS CloudHSM 叢集中的使用者。	所有[1]，包括未經身分驗證的使用者。無需登入。
變更規定人數使用者 註冊字符簽署	為使用者註冊規定人數字符簽署的規定人數策略。	管理員

註釋

- [1] 所有使用者都包含所有列出的角色和未登入的使用者。

叢集

cluster 是命令群組的父類別，當與父類別結合時，會建立使用者專用的命令。目前，用戶類別由以下命令組成：

- 叢集啟動
- 叢集 hsm-info

叢集啟動

使用 CloudHSM CLI 中的 cluster activate 命令來啟動新叢集。必須先執行此命令，叢集才可以用於執行密碼編譯操作。

使用者類型

下列類型的使用者可以執行此命令。

- 未激活的管理員

語法

此命令沒有任何參數。

```
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')
```

範例

此命令會為您的管理員使用者設定初始密碼來啟動叢集。

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

相關主題

- 建立使用者
- 刪除使用者
- 變更使用者密碼

叢集 hsm-info

使用 CloudHSM CLI 中的 cluster hsm-info 命令列出叢集中的 HSM。您無須登入 CloudHSM CLI 即可執行此命令。

> **Note**
>
> 如果您新增或刪除 HSM，請更新用 AWS CloudHSM 戶端和命令列工具使用的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列類型的使用者可以執行此命令。

- 所有使用者。您無須登入即可執行此命令。

語法

```
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
```

範例

此指令會列出 AWS CloudHSM 叢集中存在的 HSM。

```
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]"
      }
    ]
  }
}
```

輸出具有下列屬性：

- 廠商：HSM 的廠商名稱。
- 型號：HSM 的型號。

- 序號：HSM 的序號。其可能會因更換而改變。
- H ardware-version-major：主要硬體版本。
- H ardware-version-minor：次要硬體版本。
- F irmware-version-major：主要韌體版本。
- F irmware-version-minor：次要韌體版本。
- F irmware-build-number：韌體組建編號。
- Firmware-id：韌體 ID，其中包括主要和次要版本以及建置編號。
- FIPS 狀態：群集和其中的 HSM 的 FIPS 模式。如果在 FIPS 模式下，則輸出為「2 [具有單因素驗證的 FIPS 模式]」。如果在非 FIP 模式下，則輸出為「0 [具有單因素驗證的非 FIPS 模式]」。

相關主題

- 叢集啟動

加密

crypto是指令群組的父類別，當與父類別結合時，會建立密碼編譯作業專用的指令。目前，此類別由以下命令組成：

- 加密符號
  - 加密符號
  - 加密符號 RSA-pkcs
  - 加密符號 rsa-pkcs-pss
- 加密驗證
  - 加密貨幣驗證
  - 加密貨幣驗證 RSA-pkcs
  - 加密驗證 rsa-pkcs-pss

加密符號

crypto sign是指令群組的父類別，當與父類別結合時，會在 AWS CloudHSM 叢集中使用選擇的私密金鑰來產生簽章。 crypto sign具有以下子命令：

- 加密符號

- 加密符號 RSA-pkcs
- 加密符號 rsa-pkcs-pss

若要使用crypto sign，您的 HSM 中必須有私密金鑰。您可以使用以下命令生成私鑰：

- 關鍵 generate-asymmetric-pair ec
- 關鍵 generate-asymmetric-pair RSA

加密符號

該crypto sign ecdsa命令使用 EC 私鑰和 ECDSA 簽名機制生成簽名。

若要使用crypto sign ecdsa指令，您必須先在 AWS CloudHSM 叢集中擁有 EC 私密金鑰。您可以使用`sign`屬性設定為的關鍵 generate-asymmetric-pair ec命令產生 EC 私密金鑰`true`。

> **Note**
>
> 簽名可以在中 AWS CloudHSM 使用加密驗證子命令進行驗證。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如crypto sign ecdsa何使用 ECDSA 簽署機制和SHA256雜湊函數產生簽章。此命令使用 HSM 中的私密金鑰。

Example 範例：為基礎 64 編碼資料產生簽章

```
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 範例：產生資料檔案的簽章

```
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=="
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<DATA>

要簽署的 Base64 編碼資料。

必要：是 (除非透過資料路徑提供)

<DATA_PATH>

指定要簽署之資料的位置。

必要：是 (除非透過資料路徑提供)

<HASH_FUNCTION>

指定哈希函數。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必要：是

<KEY_FILTER>

鍵引用（例如鍵引用 = 0xabc）或以空格分隔的密鑰屬性列表，以屬性形式為屬性 = KEY_屬性 _ 名稱 = KEY_屬性 _ 值來選擇匹配的鍵。

如需支援的 CloudHSM CLI 金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：是

相關主題

- 加密符號
- 加密驗證

加密符號 RSA-pkcs

該crypto sign rsa-pkcs命令使用 RSA 私鑰和 RSA-PKCS 簽名機制生成簽名。

若要使用此crypto sign rsa-pkcs命令，您必須先在 AWS CloudHSM 叢集中擁有 RSA 私密金鑰。您可以使用`sign`屬性設定為`true`的關鍵 generate-asymmetric-pair RSA命令產生 RSA 私密金鑰。

> **Note**
>
> 簽名可以在中 AWS CloudHSM 使用加密驗證子命令進行驗證。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如crypto sign rsa-pkcs何使用 RSA-PKCS 簽署機制和雜湊函數產生簽章。SHA256此命令使用 HSM 中的私密金鑰。

Example 範例：為基礎 64 編碼資料產生簽章

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private --hash-function
 sha256 --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

Example 範例：產生資料檔案的簽章

```
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": "0x00000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<DATA>

要簽署的 Base64 編碼資料。

必要：是 (除非透過資料路徑提供)

<DATA_PATH>

指定要簽署之資料的位置。

必要：是 (除非透過資料提供)

<HASH_FUNCTION>

指定哈希函數。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必要：是

<KEY_FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以的形式選擇匹配的鍵。

如需支援的 CloudHSM CLI 金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：是

相關主題

- 加密符號
- 加密驗證

加密符號 rsa-pkcs-pss

該crypto sign rsa-pkcs-pss命令使用 RSA 私鑰和簽名機制生成RSA-PKCS-PSS簽名。

若要使用此crypto sign rsa-pkcs-pss命令，您必須先在 AWS CloudHSM 叢集中擁有 RSA 私密金鑰。您可以使用sign屬性設定為true的關鍵 generate-asymmetric-pair RSA命令產生 RSA 私密金鑰。

> **Note**
>
> 簽名可以在中 AWS CloudHSM 使用加密驗證子命令進行驗證。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如crypto sign rsa-pkcs-pss何使用RSA-PKCS-PSS簽署機制和SHA256雜湊函數產生簽章。此命令使用 HSM 中的私密金鑰。

Example 範例：為基礎 64 編碼資料產生簽章

```
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": "0x0000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg=="
  }
}
```

Example 範例：產生資料檔案的簽章

```
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=="
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<DATA>

要簽署的 Base64 編碼資料。

必要：是 (除非透過資料路徑提供)

<DATA_PATH>

指定要簽署之資料的位置。

必要：是 (除非透過資料提供)

<HASH_FUNCTION>

指定哈希函數。

有效值：

- sha1
- sha224
- sha256

- sha384
- sha512

必要：是

<KEY_FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以的形式選擇匹配的鍵。

如需支援的 CloudHSM CLI 金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：是

<MGF>

指定遮罩產生函數。

> **Note**
>
> 掩碼生成函數哈希函數必須匹配簽名機制哈希函數。

有效值：

- 毫克 1 沙 1
- 毫克 1-沙 224
- 毫克 1-沙 256
- 毫克 -1 沙 384
- 毫克 1-沙 512

必要：是

<SALT_LENGTH>

指定鹽的長度。

必要：是

相關主題

- 加密符號
- 加密驗證

相關主題

- 加密驗證

加密驗證

crypto verify是指令群組的父類別，當與父品類結合時，會確認檔案是否已使用指定的金鑰簽署。crypto verify具有以下子命令：

- 加密貨幣驗證
- 加密貨幣驗證 RSA-pkcs
- 加密驗證 rsa-pkcs-pss

該crypto verify命令將簽署的文件與源文件進行比較，並根據給定的公鑰和簽名機制分析它們是否與密碼編譯相關。

> **Note**
>
> 檔案可以 AWS CloudHSM 使用加密符號作業登入。

加密貨幣驗證

該crypto verify ecdsa命令用於完成以下操作：

- 確認檔案已透過指定的公開金鑰在 HSM 中簽署。
- 確認簽章是使用 ECDSA 簽署機制產生的。
- 將已簽署的檔案與來源檔案進行比較，並根據指定的 ecdsa 公開金鑰和簽署機制，判斷兩者是否與密碼編譯相關。

若要使用此crypto verify ecdsa命令，您必須先在 AWS CloudHSM 叢集中擁有 EC 公開金鑰。您可以使用`verify`屬性設定為的密鑰進口 PEM命令匯入 EC 公開金鑰`true`。

> **Note**
>
> 您可以使加密符號用子命令在 CloudHSM CLI 中產生簽章。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如何用crypto verify ecdsa來驗證使用 ECDSA 簽署機制和SHA256雜湊函數產生的簽章。此命令使用 HSM 中的公開金鑰。

Example 範例：使用 Base64 編碼資料驗證 Base64 編碼的簽章

```
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 範例：使用資料檔案驗證簽章檔

```
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 範例：證明錯誤的簽署關係

此指令會驗證位於的資料是否已使用 ECDSA 簽署機制，以產生位於中的簽章，ecdsa-public使用該標籤的公開金鑰簽署。/home/data /home/signature由於指定的引數不構成真正的簽署關係，因此命令會傳回錯誤訊息。

```
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"
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<DATA>

要簽署的 Base64 編碼資料。

必要：是 (除非透過資料路徑提供)

<DATA_PATH>

指定要簽署之資料的位置。

必要：是 (除非透過資料路徑提供)

<HASH_FUNCTION>

指定哈希函數。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必要：是

<KEY_FILTER>

鍵引用（例如，`key-reference=0xabc`）或空格分隔的鍵屬性列表，`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`以的形式選擇匹配的鍵。

如需支援的 CloudHSM CLI 金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：是

<SIGNATURE>

編碼簽名。

必要：是 (除非透過簽名路徑提供)

<SIGNATURE_PATH>

指定簽名的位置。

必要：是 (除非透過簽名路徑提供)

相關主題

- 加密符號
- 加密驗證

加密貨幣驗證 RSA-pkcs

該crypto verify rsa-pkcs命令用於完成以下操作：

- 確認檔案已透過指定的公開金鑰在 HSM 中簽署。
- 驗證簽名是使用RSA-PKCS簽名機制生成的。
- 將已簽署的檔案與來源檔案進行比較，並根據指定的 rsa 公開金鑰和簽章機制，判斷兩者是否與密碼編譯相關。

若要使用此crypto verify rsa-pkcs命令，您必須先在 AWS CloudHSM 叢集中擁有 RSA 公開金鑰。

> **Note**
>
> 您可以使用 CloudHSM CLI 搭配子命令來產生簽章。加密符號

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如何使用crypto verify rsa-pkcs來驗證使用 RSA-PKCS 簽章機制和雜湊函數產生的簽章。SHA256此命令使用 HSM 中的公開金鑰。

Example 範例：使用 Base64 編碼資料驗證 Base64 編碼的簽章

```
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 範例：使用資料檔案驗證簽名檔

```
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 範例：證明錯誤的簽署關係

此命令驗證無效數據是否通過rsa-public使用 RSAPKCS 簽名機制產生位於中的簽名的標籤的公鑰簽名簽名。/home/signature由於指定的引數不構成真正的簽署關係，因此命令會傳回錯誤訊息。

```
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"
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<DATA>

要簽署的 Base64 編碼資料。

必要：是 (除非透過資料路徑提供)

<DATA_PATH>

指定要簽署之資料的位置。

必要：是 (除非透過資料路徑提供)

<HASH_FUNCTION>

指定哈希函數。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必要：是

<KEY_FILTER>

鍵引用（例如，`key-reference=0xabc`）或空格分隔的鍵屬性列表，`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`以的形式選擇匹配的鍵。

如需支援的 CloudHSM CLI 金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：是

<SIGNATURE>

編碼簽名。

必要：是 (除非透過簽名路徑提供)

<SIGNATURE_PATH>

指定簽名的位置。

必要：是 (除非透過簽名路徑提供)

相關主題

- 加密符號
- 加密驗證

加密驗證 rsa-pkcs-pss

該crypto sign rsa-pkcs-pss命令用於完成以下操作。

- 確認檔案已透過指定的公開金鑰在 HSM 中簽署。
- 確認簽章是否使用 RSA-PKCS-PSS 簽署機制產生。
- 將已簽署的檔案與來源檔案進行比較，並根據指定的 rsa 公開金鑰和簽章機制，判斷兩者是否與密碼編譯相關。

若要使用此crypto verify rsa-pkcs-pss命令，您必須先在 AWS CloudHSM 叢集中擁有 RSA 公開金鑰。您可以使用密鑰導入 pem 命令導入 RSA 公鑰在此處添加解包鏈接），並將`verify`屬性設置為。`true`

> **Note**
>
> 您可以使用 CloudHSM CLI 搭配子命令來產生簽章。加密符號

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如何用crypto verify rsa-pkcs-pss來驗證使用 RSA-PKCS-PSS 簽署機制和雜湊函數產生的簽章。SHA256此命令使用 HSM 中的公開金鑰。

Example 範例：使用 Base64 編碼資料驗證 Base64 編碼的簽章

```
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 範例：使用資料檔案驗證簽章檔

```
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 範例：證明錯誤的簽署關係

此命令驗證無效數據是否通過rsa-public使用 RSAPKCSSS 簽名機制的標籤的公鑰簽名簽署，以產生位於中的簽名。/home/signature由於指定的引數不構成真正的簽署關係，因此命令會傳回錯誤訊息。

```
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"
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<DATA>

要簽署的 Base64 編碼資料。

必要：是 (除非透過資料路徑提供)

<DATA_PATH>

指定要簽署之資料的位置。

必要：是 (除非透過資料路徑提供)

<HASH_FUNCTION>

指定哈希函數。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必要：是

<KEY_FILTER>

鍵引用（例如，`key-reference=0xabc`）或空格分隔的鍵屬性列表，`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`以的形式選擇匹配的鍵。

如需支援的 CloudHSM CLI 金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：是

<MFG>

指定遮罩產生函數。

> **Note**
>
> 掩碼生成函數哈希函數必須與簽名機制哈希函數匹配。

有效值：

- 毫克 1
- 毫克 1-沙 224
- 毫克 1-沙 256
- 毫克 1 沙 384
- 毫克 1-沙 512

必要：是

<SIGNATURE>

編碼簽名。

必要：是 (除非透過簽名路徑提供)

<SIGNATURE_PATH>

指定簽名的位置。

必要：是 (除非透過簽名路徑提供)

相關主題

- 加密符號
- 加密驗證

金鑰

key 是命令群組的父類別，當與父類別結合時，會建立金鑰專用的命令。目前，此類別由以下命令組成：

- 刪除金鑰
- 產生金鑰的檔案
- 關鍵 generate-asymmetric-pair

  - 關鍵 generate-asymmetric-pair RSA
  - 關鍵 generate-asymmetric-pair ec
- 產生對稱金鑰
  - 產生對稱 AES 金鑰
  - 產生對稱通用私密金鑰
- 密鑰進口 PEM
- 列出金鑰
- 金鑰複製
- 設定金鑰屬性
- 共用金鑰
- 取消共用金鑰
- 密鑰展開
- 按鍵包裝

刪除金鑰

使用 CloudHSM CLI 中的key delete命令可從叢集中刪除金鑰。 AWS CloudHSM 您一次只能刪除一個金鑰。刪除金鑰對中的一個金鑰，對金鑰對中的另一個金鑰沒有影響。

只有建立金鑰並因此擁有金鑰的 CU 才能刪除金鑰。共用金鑰但卻未擁有金鑰的使用者可以在密碼編譯操作中使用金鑰，但不能刪除它。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

```
aws-cloudhsm > key delete --filter attr.label="ec-test-public-key"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以的形式選擇要刪除的匹配鍵。

如需支援的 CloudHSM CLI 金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性

必要：是

相關主題

- 列出金鑰

- 產生金鑰的檔案
- 取消共用金鑰
- CloudHSM CLI 的金鑰屬性
- 使用 CloudHSM CLI 篩選金鑰

產生金鑰的檔案

命key generate-file令會從 HSM 匯出非對稱金鑰。如果目標是私鑰，則對私鑰的引用將以偽造的 PEM 格式導出。如果目標是公鑰，則公鑰字節將以 PEM 格式導出。

假的 PEM 檔案不包含實際的私密金鑰材料，而是參考 HSM 中的私密金鑰，可用來建立從 Web 伺服器到的 SSL/TLS 卸載。 AWS CloudHSM如需詳細資訊，請參閱 SSL/TLS 卸載。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

若要執行此命令，必須以 CU 的身分登入。

語法

```
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')
```

範例

此範例顯示如key generate-file何使用在 AWS CloudHSM 叢集中產生金鑰檔案。

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"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以的形式選擇要刪除的匹配鍵。

如需支援的 CloudHSM CLI 金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性

必要：否

<ENCODING>

指金鑰檔案的編碼格式

必要：是

<PATH>

指要寫入金鑰檔案的檔案路徑

必要：是

相關主題

- CloudHSM CLI 的金鑰屬性
- 使用 CloudHSM CLI 篩選金鑰
- 關鍵 generate-asymmetric-pair
- 產生對稱金鑰

關鍵 generate-asymmetric-pair

key generate-asymmetric-pair 是命令群組的父類別，當與父類別結合時，會建立產生非對稱金鑰對的命令。目前，此類別由以下命令組成：

- 關鍵 generate-asymmetric-pair ec
- 關鍵 generate-asymmetric-pair RSA

關鍵 generate-asymmetric-pair ec

使用 CloudHSM CLI 中的key asymmetric-pair ec命令，在叢集中產生非對稱的橢圓曲線 (EC) key pair。 AWS CloudHSM

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

下列範例示範如何使用 key generate-asymmetric-pair ec 命令來建立 EC 金鑰對。

Example 範例：建立 EC 金鑰對

```
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 範例：建立具有選用屬性的 EC 金鑰對

```
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"
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<CURVE>

指橢圓曲線的識別符。

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1

必要：是

<PUBLIC_KEY_ATTRIBUTES>

指一個空格分隔的金鑰屬性清單，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例如，token=true) 的形式為產生的 EC 公有金鑰設定

如需支援的金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：否

<PUBLIC_LABEL>

指使用者定義的公有金鑰標籤。用戶端 SDK 5.11 及之後允許的大小上限為 127 個字元。label用戶端 SDK 5.10 及之前版本的限制為 126 個字元。

必要：是

<PRIVATE_KEY_ATTRIBUTES>

指一個空格分隔的金鑰屬性清單，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例如，token=true) 的形式為產生的 EC 私有金鑰設定

如需支援的金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：否

<PRIVATE_LABEL>

指使用者定義的私有金鑰標籤。用戶端 SDK 5.11 及之後允許的大小上限為 127 個字元。label用戶端 SDK 5.10 及之前版本的限制為 126 個字元。

必要：是

<SESSION>

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。

當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

根據預設，產生的金鑰是持久性 (權杖) 金鑰。在 <SESSION> 傳遞會變更此情況，確保使用此參數生成的金鑰是工作階段 (臨時) 金鑰。

必要：否

相關主題

- CloudHSM CLI 的金鑰屬性
- 使用 CloudHSM CLI 篩選金鑰

關鍵 generate-asymmetric-pair RSA

使用指key generate-asymmetric-pair rsa令會在 AWS CloudHSM 叢集中產生非對稱 RSA key pair。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

下列範例示範如何使用 `key generate-asymmetric-pair rsa` 來建立 RSA 金鑰對。

Example 範例：建立 RSA 金鑰對

```
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 範例：建立具有選用屬性的 RSA 金鑰對

```
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
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已[設定多個叢集。]

<MODULUS_SIZE_BITS>

指模數的長度 (以位元為單位)。最小值為 2048。

必要：是

<PRIVATE_KEY_ATTRIBUTES>

指一個空格分隔的金鑰屬性清單，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例如，token=true) 的形式為產生的 RSA 私有金鑰設定

如需支援的金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：否

<PRIVATE_LABEL>

指使用者定義的私有金鑰標籤。用戶端 SDK 5.11 及之後允許的大小上限為 127 個字元。label用戶端 SDK 5.10 及之前版本的限制為 126 個字元。

必要：是

<PUBLIC_EXPONENT>

指公有指數。值必須為大於或等於 65537 的奇數。

必要：是

<PUBLIC_KEY_ATTRIBUTES>

指一個空格分隔的金鑰屬性清單，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例如，token=true) 的形式為產生的 RSA 公有金鑰設定

如需支援的金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：否

<PUBLIC_LABEL>

指使用者定義的公有金鑰標籤。用戶端 SDK 5.11 及之後允許的大小上限為 127 個字元。label用戶端 SDK 5.10 及之前版本的限制為 126 個字元。

必要：是

<SESSION>

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。

當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

根據預設，產生的金鑰是持久性 (權杖) 金鑰。在 <SESSION> 傳遞會變更此情況，確保使用此參數生成的金鑰是工作階段 (臨時) 金鑰。

必要：否

相關主題

- CloudHSM CLI 的金鑰屬性
- 使用 CloudHSM CLI 篩選金鑰

產生對稱金鑰

key generate-symmetric 是命令群組的父類別，當與父類別結合時，會建立產生對稱金鑰的命令。目前，此類別由以下命令組成：

- 產生對稱 AES 金鑰
- 產生對稱通用私密金鑰

產生對稱 AES 金鑰

此指key generate-symmetric aes令會在 AWS CloudHSM 叢集中產生對稱 AES 金鑰。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

下列範例示範如何使用 key generate-symmetric aes 命令來建立 AES 金鑰。

Example 範例：建立 AES 金鑰

```
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 範例：建立具有選用屬性的 AES 金鑰

```
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
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<KEY_ATTRIBUTES>

指定一個空格分隔的金鑰屬性清單，對產生的 AES 金鑰進行設定，格式為 `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (例如，`token=true`)。

如需支援的金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：否

<KEY-LENGTH-BYTES>

指金鑰長度 (以位元組為單位)。

有效值：

- 16、24 和 32

必要：是

<LABEL>

指定使用者定義的 AES 金鑰標籤。用戶端 SDK 5.11 及之後允許的大小上限為 127 個字元。`label`用戶端 SDK 5.10 及之前版本的限制為 126 個字元。

必要：是

<SESSION>

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。

當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

根據預設，產生的金鑰是持久性 (權杖) 金鑰。在 <SESSION> 傳遞會變更此情況，確保使用此參數生成的金鑰是工作階段 (臨時) 金鑰。

必要：否

相關主題

- CloudHSM CLI 的金鑰屬性
- 使用 CloudHSM CLI 篩選金鑰

產生對稱通用私密金鑰

使用 key generate-asymmetric-pair 命令在 AWS CloudHSM 叢集中產生一組通用對稱私密金鑰。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

下列範例示範如何使用 key generate-symmetric generic-secret 命令來建立一組通用私密金鑰。

Example 範例：建立一組通用私密金鑰

```
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 範例：建立一組無選用屬性的通用私密金鑰

```
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
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<KEY_ATTRIBUTES>

指定一個空格分隔的金鑰屬性清單，對產生的 AES 金鑰進行設定，格式為 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例如，token=true)。

如需支援的金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：否

<KEY-LENGTH-BYTES>

指金鑰長度 (以位元組為單位)。

有效值：

- 1 到 800

必要：是

<LABEL>

指定使用者定義的通用私密金鑰標籤。用戶端 SDK 5.11 及之後允許的大小上限為 127 個字元。`label`用戶端 SDK 5.10 及之前版本的限制為 126 個字元。

必要：是

<SESSION>

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。

當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

根據預設，產生的金鑰是持久性 (權杖) 金鑰。在 <SESSION> 傳遞會變更此情況，確保使用此參數生成的金鑰是工作階段 (臨時) 金鑰。

必要：否

相關主題

- CloudHSM CLI 的金鑰屬性
- 使用 CloudHSM CLI 篩選金鑰

密鑰進口 PEM

中的`key import pem`命令會將 PEM 格式金鑰 AWS CloudHSM 匯入 HSM。您可以使用此命令匯入在 HSM 之外產生的公有金鑰。

> **Note**
>
> 使用指產生金鑰的檔案令從公開金鑰建立標準 PEM 檔案，或從私密金鑰建立參考 PEM 檔案。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例顯示如何使用key import pem命令從 PEM 格式的檔案匯入 RSA 公開金鑰。

Example 範例：匯入 RSA 公開金鑰

```
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 範例：匯入具有選用屬性的 RSA 公開金鑰

```
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"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<PATH>

指定金鑰檔所在的檔案路徑。

必要：是

<LABEL>

為匯入的金鑰指定使用者定義的標籤。label 的大小上限為 126 個字元。

必要：是

<KEY_TYPE_CLASS>

包裝鍵的密鑰類型和類別。

可能的值如下：

- EC-公共
- RSA-公共

必要：是

<IMPORT_KEY_ATTRIBUTES>

指定以空格分隔的索引鍵屬性清單，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE (例如，token=true) 形式為匯入的金鑰設定。如需支援的金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：否

相關主題

- 加密符號
- 加密驗證

列出金鑰

此命key list令會尋找 AWS CloudHSM 叢集中目前使用者的所有金鑰。輸出包含使用者擁有和共用的金鑰，以及 CloudHSM 叢集中的所有公有金鑰。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

語法

```
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
```

範例

下列範例顯示執行 key list 命令的不同方式。

Example 範例：尋找所有金鑰：預設

此命令列出了存在於 AWS CloudHSM 群集中的登錄用戶的密鑰。

> **Note**
>
> 根據預設，僅顯示目前登入使用者的 10 個金鑰，且只顯示 `key-reference` 和 `label` 作為輸出。使用適當的分頁選項來顯示更多或更少的金鑰作為輸出。

```
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 範例：尋找所有金鑰：詳細資訊

輸出包含使用者擁有和共用的金鑰，以及 HSM 中的所有公有金鑰。

> **Note**
>
> 注意：按照預設，僅顯示目前登入使用者的 10 個金鑰。使用適當的分頁選項來顯示更多或更少的金鑰作為輸出。

```
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 範例：分頁傳回

下列範例將顯示僅顯示兩個金鑰的金鑰分頁子集。然後，此範例會提供後續呼叫，以顯示接下來的兩個金鑰。

```
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"
  }
}
```

若要顯示接下來的兩個金鑰，可以進行後續呼叫：

```
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"
  }
}
```


如需更多有關 CloudHSM CLI 中金鑰篩選機制工作原理的演示範例，請參閱 使用 CloudHSM CLI 篩選金鑰。

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

按鍵參照 (例如，key-reference=0xabc) 或以空格分隔的索引鍵屬性清單，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以便選取要列出的相符鍵。

如需支援的 CloudHSM CLI 金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性

必要：否

<MAX_ITEMS>

要在命令輸出中傳回的總項目數。若可傳回的總項目數超過指定值，會在命令的輸出中提供 next-token。若要繼續分頁，請在後續 starting-token 命令引數中提供 next-token 值。

必要：否

<STARTING_TOKEN>

用以指定分頁開始位置的字符。這是來自先前已截斷回應的 next-token。

必要：否

<VERBOSE>

如已包含，則會列印每個相符金鑰的所有屬性和金鑰資訊。根據預設，每個相符金鑰僅顯示其金鑰參考和標籤屬性。

必要：否

相關主題

- 刪除金鑰
- 產生金鑰的檔案
- 取消共用金鑰
- CloudHSM CLI 的金鑰屬性

- 使用 CloudHSM CLI 篩選金鑰

金鑰複製

該key replicate命令會將金鑰從來源 AWS CloudHSM 叢集複製到目的地 AWS CloudHSM 叢集。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

> **Note**
>
> 加密用戶必須擁有密鑰才能使用此命令。

要求

- 來源叢集和目的地叢集必須是複製。這意味著一個是從另一個備份創建的，或者它們都是從通用備份創建的。如需詳細資訊，請參閱從備份建立叢集。
- 金鑰的擁有者必須存在於目的地叢集上。此外，如果金鑰與任何使用者共用，則這些使用者也必須存在於目的地叢集中。
- 若要執行此命令，您必須在來源叢集和目的地叢集上以 CU 的身分登入。
  - 在單一命令模式中，命令會使用 CLOUDHSM_PIN 和 CLOUDHSM_ROLE 環境變數，在來源叢集上進行驗證。如需詳細資訊，請參閱單一命令模式。若要提供目的地叢集的證明資料，您需要設定兩個額外的環境變數：

```
$ export DESTINATION_CLOUDHSM_ROLE=crypto-user
```

```
$ export DESTINATION_CLOUDHSM_PIN=username:password
```

  - 在互動模式下，使用者必須明確登入來源和目的地叢集。

語法

```
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
```

範例

Example 範例：複製金鑰

此命令會將來源叢集中的金鑰複製到複製的目的地叢集。

```
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"
  }
}
```

引數

<FILTER>

索引鍵參照 (例如key-reference=0xabc) 或以空格分隔的索引鍵屬性清單，以attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE便在來源叢集上選取相符的金鑰。

如需支援的 CloudHSM CLI 金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性

必要：是

<SOURCE_CLUSTER_ID>

來源叢集識別碼。

必要：是

<DESTINATION_CLUSTER_ID>

目的地叢集識別碼。

必要：是

相關主題

- 使用 CloudHSM CLI 連線到多個叢集

設定金鑰屬性

使用key set-attribute指令來設定 AWS CloudHSM 叢集中金鑰的屬性。只有建立金鑰並因此擁有金鑰的加密使用者才能變更金鑰的屬性。

如需可在 CloudHSM CLI 中使用的金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU) 可以執行此命令。
- 管理員可設定受信任的屬性。

要求

若要執行此命令，必須以 CU 的身分登入。若要設定受信任屬性，您必須以管理員使用者身分登入。

語法

```
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
```

範例：設定金鑰屬性

下列範例示範如何使用 key set-attribute 命令來設定標籤。

Example

1. 使用帶有標籤 my_key 的金鑰，如下所示：

```
aws-cloudhsm > key set-attribute --filter attr.label=my_key --name encrypt --value
 false
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
}
```

2. 使用 key list 命令確認 encrypt 屬性已變更：

```
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
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<KEY_ATTRIBUTES>

指金鑰屬性的名稱。

必要：是

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以的形式選擇要刪除的匹配鍵。

如需支援的 CloudHSM CLI 金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性

必要：否

<KEY_ATTRIBUTE_VALUE>

指金鑰屬性的名稱。

必要：是

<KEY_REFERENCE>

金鑰的十六進位或十進位表示法。(例如金鑰控制代碼)。

必要：否

相關主題

- 使用 CloudHSM CLI 篩選金鑰
- CloudHSM CLI 的金鑰屬性

共用金鑰

此命key share令會與 AWS CloudHSM 叢集中的其他 CUs 共用金鑰。

只有建立金鑰並因此擁有金鑰的加密使用者才能共用金鑰。共用金鑰的使用者可以在密碼編譯操作中使用此金鑰，但無法刪除、匯出、共用金鑰或取消共用金鑰。此外，這些使用者無法變更金鑰屬性。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

若要執行此命令，必須以 CU 的身分登入。

語法

```
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')
```

範例：與另一個加密使用者共用金鑰

下列範例顯示如何使用 key share 命令與加密使用者 `alice` 共用金鑰。

Example

1. 執行 key share 命令與 `alice` 共用金鑰。

```
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. 執行 key list 命令。

```
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. 以上列表中，驗證 alice 在列表 shared-users 中

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以的形式選擇要刪除的匹配鍵。

如需支援的金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：是

<USERNAME>

為使用者指定易記的名稱。長度上限為 31 個字元。允許的唯一特殊字元是底線 (_)。在此命令中，使用者名稱不區分大小寫，使用者名稱始終以小寫顯示。

必要：是

<ROLE>

指定指派給此使用者的角色。此為必要參數。若要取得使用者的角色，請使用使用者清單命令。如需 HSM 上使用者類型的詳細資訊，請參閱 了解 HSM 使用者。

必要：是

相關主題

- 使用 CloudHSM CLI 篩選金鑰
- CloudHSM CLI 的金鑰屬性

取消共用金鑰

此命key unshare令會取消與 AWS CloudHSM 叢集中其他 CU 的金鑰共用。

只有建立金鑰並因此擁有金鑰的加密使用者才能取消共用金鑰。共用金鑰的使用者可以在密碼編譯操作中使用此金鑰，但無法刪除、匯出、共用金鑰或取消共用金鑰。此外，這些使用者無法變更金鑰屬性。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

若要執行此命令，必須以 CU 的身分登入。

語法

```
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')
```

範例：取消與另一個加密使用者共用金鑰

下列範例顯示如何使用 key unshare 命令取消與加密使用者 `alice` 共用的金鑰。

Example

1. 執行 key list 命令並篩選出您想取消與 `alice` 共用的金鑰。

```
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. 確認 alice 在 shared-users 輸出中，並運行下列 key unshare 命令以取消與 alice 共用的金鑰。

```
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. 再次執行 key list 命令以確認已取消與 alice 共用該金鑰。

```
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
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以的形式選擇要刪除的匹配鍵。

如需支援的金鑰屬性清單，請參閱 CloudHSM CLI 的金鑰屬性。

必要：是

<USERNAME>

為使用者指定易記的名稱。長度上限為 31 個字元。允許的唯一特殊字元是底線 (_)。在此命令中，使用者名稱不區分大小寫，使用者名稱始終以小寫顯示。

必要：是

<ROLE>

指定指派給此使用者的角色。此為必要參數。若要取得使用者的角色，請使用使用者清單命令。如需 HSM 上使用者類型的詳細資訊，請參閱 了解 HSM 使用者。

必要：是

相關主題

- 使用 CloudHSM CLI 篩選金鑰
- CloudHSM CLI 的金鑰屬性

密鑰展開

CloudHSM CLI 中的key unwrap父命令會將加密 (包裝) 的對稱或非對稱私密金鑰從檔案匯入 HSM。此指令的設計目的是匯入由按鍵包裝命令包裝的加密金鑰，但也可以用來解除以其他工具包裝的金鑰。不過，在這些情況下，我們建議您使用 PKCS#11 或 JCE 軟體程式庫來取消包裝金鑰。

- 密鑰展開一個-gcm
- 密鑰展開 aes-no-pad
- 密鑰打開一個-pkcs5 墊
- 密鑰展開 aes-zero-pad
- 密鑰展開 cloudhsm-aes-gcm
- 密鑰解開 RSA-艾斯
- 密鑰解開 RSA-工作
- 密鑰解開 RSA-pkcs

密鑰展開一個-gcm

該key unwrap aes-gcm命令使用 AES 包裝密鑰和解包機制將有效負載密鑰AES-GCM解開到集群中。

未包裝的金鑰可以使用與產生的金鑰相同的方式使用 AWS CloudHSM。若要指出它們不是在本機產生，其`local`屬性會設定為`false`。

若要使用key unwrap aes-gcm命令，您必須在 AWS CloudHSM 叢集中有 AES 包裝金鑰，且其`unwrap`屬性必須設定為`true`。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如何使用 AES 金鑰並將unwrap屬性值設定為的key unwrap aes-gcm命令true。

Example 示例：從 Base64 編碼的包裝密鑰數據中解開有效負載密鑰

```
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 範例：解除包裝透過資料路徑提供的裝載金鑰

```
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
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇要解包的鍵的形式。

必要：是

<DATA_PATH>

包含包裝金鑰資料之二進位檔案的路徑。

必要：是 (除非透過 Base64 編碼資料提供)

<DATA>

以 Base64 編碼包裝的金鑰資料。

必要：是 (除非透過資料路徑提供)

<ATTRIBUTES>

以包裝金鑰的形式，以空格分隔的KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE索引鍵屬性清單。

必要：否

<AAD>

Aes GCM 其他驗證資料 (AAD) 值，以十六進位表示。

必要：否

<TAG_LENGTH_BITS>

Aes GCM 標籤長度 (以位為單位)。

必要：是

<KEY_TYPE_CLASS>

包裝鍵的鍵類型和類別 [可能的值：aesdes3,ec-private,generic-secret,,rsa-private]。

必要：是

<LABEL>

未包裝金鑰的標籤。

必要：是

<SESSION>

建立只存在於目前工作階段中的工作階段金鑰。工作階段結束後，金鑰無法復原。

必要：否

<IV>

用來包裝金鑰的初始值，以十六進位表示。

必要：否

相關主題

- 按鍵包裝

- 密鑰展開

密鑰展開 aes-no-pad

該key unwrap aes-no-pad命令使用 AES 包裝密鑰和解包機制將有效負載密鑰AES-NO-PAD解開到集群中。

未包裝的金鑰可以使用與產生的金鑰相同的方式使用 AWS CloudHSM。若要指出它們不是在本機產生，其local屬性會設定為false。

若要使用key unwrap aes-no-pad命令，您必須在 AWS CloudHSM 叢集中有 AES 包裝金鑰，且其unwrap屬性必須設定為true。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如何使用 AES 金鑰並將unwrap屬性值設定為的key unwrap aes-no-pad命令true。

Example 示例：從 Base64 編碼的包裝密鑰數據中解開有效負載密鑰

```
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 範例：解除包裝透過資料路徑提供的裝載金鑰

```
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
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇要解包的鍵的形式。

必要：是

<DATA_PATH>

包含包裝金鑰資料之二進位檔案的路徑。

必要：是 (除非透過 Base64 編碼資料提供)

<DATA>

以 Base64 編碼包裝的金鑰資料。

必要：是 (除非透過資料路徑提供)

<ATTRIBUTES>

以包裝金鑰的形式，以空格分隔的KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE索引鍵屬性清單。

必要：否

<KEY_TYPE_CLASS>

包裝鍵的鍵類型和類別 [可能的值：aesdes3,ec-private,generic-secret,,rsa-private]。

必要：是

<LABEL>

未包裝金鑰的標籤。

必要：是

<SESSION>

建立只存在於目前工作階段中的工作階段金鑰。工作階段結束後，金鑰無法復原。

必要：否

相關主題

- 按鍵包裝
- 密鑰展開

密鑰打開一個-pkcs5 墊

該key unwrap aes-pkcs5-pad命令使用 AES 包裝密鑰和解包機制AES-PKCS5-PAD解包裝有效負載密鑰。

未包裝的金鑰可以使用與產生的金鑰相同的方式使用 AWS CloudHSM。若要指出它們不是在本機產生，其`local`屬性會設定為`false`。

若要使用key unwrap aes-pkcs5-pad命令，您必須在 AWS CloudHSM 叢集中具有 AES 包裝金鑰，且其`unwrap`屬性必須設定為`true`。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如何使用 AES 金鑰並將unwrap屬性值設定為的key unwrap aes-pkcs5-pad命令true。

Example 示例：從 Base64 編碼的包裝密鑰數據中解開有效負載密鑰

```
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 範例：解除包裝透過資料路徑提供的有效負載金鑰

```
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
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇要解包的鍵的形式。

必要：是

<DATA_PATH>

包含包裝金鑰資料之二進位檔案的路徑。

必要：是 (除非透過 Base64 編碼資料提供)

<DATA>

以 Base64 編碼包裝的金鑰資料。

必要：是 (除非透過資料路徑提供)

<ATTRIBUTES>

以包裝金鑰的形式，以空格分隔的KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE索引鍵屬性清單。

必要：否

<KEY_TYPE_CLASS>

包裝鍵的鍵類型和類別 [可能的值：aesdes3,ec-private,generic-secret,,rsa-private]。

必要：是

<LABEL>

未包裝金鑰的標籤。

必要：是

<SESSION>

建立只存在於目前工作階段中的工作階段金鑰。工作階段結束後，金鑰無法復原。

必要：否

相關主題

- 按鍵包裝
- 密鑰展開

密鑰展開 aes-zero-pad

該key unwrap aes-zero-pad命令使用 AES 包裝密鑰和解包機制將有效負載密鑰AES-ZERO-PAD解開到集群中。

未包裝的金鑰可以使用與產生的金鑰相同的方式使用 AWS CloudHSM。若要指出它們不是在本機產生，其local屬性會設定為false。

若要使用key unwrap aes-no-pad命令，您必須在 AWS CloudHSM 叢集中有 AES 包裝金鑰，且其unwrap屬性必須設定為true。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如何使用 AES 金鑰並將unwrap屬性值設定為的key unwrap aes-zero-pad命令true。

Example 示例：從 Base64 編碼的包裝密鑰數據中解開有效負載密鑰

```
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 範例：解除包裝透過資料路徑提供的裝載金鑰

```
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
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇要解包的鍵的形式。

必要：是

<DATA_PATH>

包含包裝金鑰資料之二進位檔案的路徑。

必要：是 (除非透過 Base64 編碼資料提供)

<DATA>

以 Base64 編碼包裝的金鑰資料。

必要：是 (除非透過資料路徑提供)

<ATTRIBUTES>

以包裝金鑰的形式，以空格分隔的KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE索引鍵屬性清單。

必要：否

<KEY_TYPE_CLASS>

包裝鍵的鍵類型和類別 [可能的值：aesdes3,ec-private,generic-secret,,rsa-private]。

必要：是

<LABEL>

未包裝金鑰的標籤。

必要：是

<SESSION>

建立只存在於目前工作階段中的工作階段金鑰。工作階段結束後，金鑰無法復原。

必要：否

相關主題

- 按鍵包裝
- 密鑰展開

密鑰展開 cloudhsm-aes-gcm

該key unwrap cloudhsm-aes-gcm命令使用 AES 包裝密鑰和解包機制將有效負載密鑰CLOUDHSM-AES-GCM解開到集群中。

未包裝的金鑰可以使用與產生的金鑰相同的方式使用 AWS CloudHSM。若要指出它們不是在本機產生，其local屬性會設定為false。

若要使用該key unwrap cloudhsm-aes-gcm命令，您必須在 AWS CloudHSM 叢集中具有 AES 包裝金鑰，且其unwrap屬性必須設定為true。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如何使用 AES 金鑰並將unwrap屬性值設定為的key unwrap cloudhsm-aes-gcm命令true。

Example 示例：從 Base64 編碼的包裝密鑰數據中解開有效負載密鑰

```
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 範例：解除包裝透過資料路徑提供的有效負載金鑰

```
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
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇要解包的鍵的形式。

必要：是

<DATA_PATH>

包含包裝金鑰資料之二進位檔案的路徑。

必要：是 (除非透過 Base64 編碼資料提供)

<DATA>

以 Base64 編碼包裝的金鑰資料。

必要：是 (除非透過資料路徑提供)

<ATTRIBUTES>

以包裝金鑰的形式，以空格分隔的KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE索引鍵屬性清單。

必要：否

<AAD>

Aes GCM 其他驗證資料 (AAD) 值，以十六進位表示。

必要：否

<TAG_LENGTH_BITS>

Aes GCM 標籤長度 (以位為單位)。

必要：是

<KEY_TYPE_CLASS>

包裝鍵的鍵類型和類別 [可能的值：aesdes3,ec-private,generic-secret,,rsa-private]。

必要：是

<LABEL>

未包裝金鑰的標籤。

必要：是

<SESSION>

建立只存在於目前工作階段中的工作階段金鑰。工作階段結束後，金鑰無法復原。

必要：否

相關主題

- 按鍵包裝
- 密鑰展開

密鑰解開 RSA-艾斯

該key unwrap rsa-aes命令使用 RSA 私鑰和解包機制RSA-AES解包裝有效負載密鑰。

未包裝的金鑰可以使用與產生的金鑰相同的方式使用 AWS CloudHSM。若要指出它們不是在本機產生，其`local`屬性會設定為`false`。

若要使用key unwrap rsa-aes，您必須在 AWS CloudHSM 叢集中擁有 RSA 公開包裝金鑰的 RSA 私密金鑰，且其`unwrap`屬性必須設定為。`true`

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如何使用 RSA 私密金鑰且unwrap屬性值設定為true的key unwrap rsa-aes命令。

Example 示例：從 Base64 編碼的包裝密鑰數據中解開有效負載密鑰

```
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 範例：解除包裝透過資料路徑提供的有效負載金鑰

```
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
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇要解包的鍵的形式。

必要：是

<DATA_PATH>

包含包裝金鑰資料之二進位檔案的路徑。

必要：是 (除非透過 Base64 編碼資料提供)

<DATA>

以 Base64 編碼包裝的金鑰資料。

必要：是 (除非透過資料路徑提供)

<ATTRIBUTES>

以包裝金鑰的形式，以空格分隔的KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE索引鍵屬性清單。

必要：否

<KEY_TYPE_CLASS>

包裝鍵的鍵類型和類別 [可能的值：aesdes3,ec-private,generic-secret,,rsa-private]。

必要：是

<HASH_FUNCTION>

指定哈希函數。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必要：是

<MGF>

指定遮罩產生函數。

> **Note**
>
> 掩碼生成函數哈希函數必須與簽名機制哈希函數匹配。

有效值：

- 毫克 1 沙 1
- 毫克 1-沙 224
- 毫克 1-沙 256
- 毫克 -1 沙 384
- 毫克 1-沙 512

必要：是

<LABEL>

未包裝金鑰的標籤。

必要：是

<SESSION>

建立只存在於目前工作階段中的工作階段金鑰。工作階段結束後，金鑰無法復原。

必要：否

相關主題

- 按鍵包裝
- 密鑰展開

密鑰解開 RSA-工作

該key unwrap rsa-oaep命令使用 RSA 私鑰和解包機制RSA-OAEP解包裝有效負載密鑰。

未包裝的金鑰可以使用與產生的金鑰相同的方式使用 AWS CloudHSM。若要指出它們不是在本機產生，其`local`屬性會設定為`false`。

若要使用key unwrap rsa-oaep命令，您必須在 AWS CloudHSM 叢集中擁有 RSA 公開包裝金鑰的 RSA 私密金鑰，且其`unwrap`屬性必須設定為。`true`

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如何使用 RSA 私密金鑰且unwrap屬性值設定為true的key unwrap rsa-oaep命令。

Example 示例：從 Base64 編碼的包裝密鑰數據中解開有效負載密鑰

```
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 範例：解除包裝透過資料路徑提供的有效負載金鑰

```
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
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇要解包的鍵的形式。

必要：是

<DATA_PATH>

包含包裝金鑰資料之二進位檔案的路徑。

必要：是 (除非透過 Base64 編碼資料提供)

<DATA>

以 Base64 編碼包裝的金鑰資料。

必要：是 (除非透過資料路徑提供)

<ATTRIBUTES>

以包裝金鑰的形式，以空格分隔的KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE索引鍵屬性清單。

必要：否

<KEY_TYPE_CLASS>

包裝鍵的鍵類型和類別 [可能的值：aesdes3,ec-private,generic-secret,,rsa-private]。

必要：是

<HASH_FUNCTION>

指定哈希函數。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必要：是

<MGF>

指定遮罩產生函數。

> **Note**
>
> 掩碼生成函數哈希函數必須與簽名機制哈希函數匹配。

有效值：

- 毫克 1 沙 1
- 毫克 1-沙 224
- 毫克 1-沙 256
- 毫克 -1 沙 384
- 毫克 1-沙 512

必要：是

<LABEL>

未包裝金鑰的標籤。

必要：是

<SESSION>

建立只存在於目前工作階段中的工作階段金鑰。工作階段結束後，金鑰無法復原。

必要：否

相關主題

- 按鍵包裝
- 密鑰展開

密鑰解開 RSA-pkcs

該key unwrap rsa-pkcs命令使用 RSA 私鑰和解包機制RSA-PKCS解包裝有效負載密鑰。

未包裝的金鑰可以使用與產生的金鑰相同的方式使用 AWS CloudHSM。若要指出它們不是在本機產生，其`local`屬性會設定為`false`。

若要使用 key unwrap rsa-pkcs 命令，您必須在 AWS CloudHSM 叢集中擁有 RSA 公開包裝金鑰的 RSA 私密金鑰，且其`unwrap`屬性必須設定為。`true`

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這些範例說明如何使用 AES 金鑰並將unwrap屬性值設定為的key unwrap rsa-oaep命令true。

Example 示例：從 Base64 編碼的包裝密鑰數據中解開有效負載密鑰

```
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 範例：解除包裝透過資料路徑提供的有效負載金鑰

```
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
      }
    }
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇要解包的鍵的形式。

必要：是

<DATA_PATH>

包含包裝金鑰資料之二進位檔案的路徑。

必要：是 (除非透過 Base64 編碼資料提供)

<DATA>

以 Base64 編碼包裝的金鑰資料。

必要：是 (除非透過資料路徑提供)

<ATTRIBUTES>

以包裝金鑰的形式，以空格分隔的KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE索引鍵屬性清單。

必要：否

<KEY_TYPE_CLASS>

包裝鍵的鍵類型和類別 [可能的值：aesdes3,ec-private,generic-secret,,rsa-private]。

必要：是

<LABEL>

未包裝金鑰的標籤。

必要：是

<SESSION>

建立只存在於目前工作階段中的工作階段金鑰。工作階段結束後，金鑰無法復原。

必要：否

相關主題

- 按鍵包裝
- 密鑰展開

按鍵包裝

CloudHSM CLI 中的key wrap命令會將對稱或非對稱私密金鑰的加密副本從 HSM 匯出至檔案。執行時key wrap，您可以指定兩件事：要匯出的金鑰和輸出檔案。要匯出的金鑰是 HSM 上的金鑰，它會加密 (包裝) 您要匯出的金鑰。

此命key wrap令不會從 HSM 移除金鑰，也不會阻止您在密碼編譯作業中使用該金鑰。您可以多次匯出相同的金鑰。若要將加密金鑰重新匯入 HSM，請使用密鑰展開。只有金鑰的擁有者 (即建立金鑰的加密使用者 (CU) 才能包裝金鑰。與其共用金鑰的使用者只能在密碼編譯作業中使用金鑰。

該key wrap命令由以下子命令組成：

- 鑰匙包裝 AES-克厘米
- 按鍵包裝 aes-no-pad
- 密鑰包裝一個-pkcs5 墊
- 按鍵包裝 aes-zero-pad
- 按鍵包裝 cloudhsm-aes-gcm
- 密鑰包裝 RSA-艾斯
- 密鑰包裝 RSA-
- 密鑰包裝 RSA-pkcs

鑰匙包裝 AES-克厘米

命key wrap aes-gcm令會使用 HSM 和AES-GCM包裝機制上的 AES 金鑰來封裝承載金鑰。裝載金鑰的extractable屬性必須設定為true。

只有金鑰的擁有者 (即建立金鑰的加密使用者 (CU) 才能包裝金鑰。共用金鑰的使用者可以在密碼編譯作業中使用金鑰。

若要使用此key wrap aes-gcm命令，您必須先在 AWS CloudHSM 叢集中擁有 AES 金鑰。您可以產生 AES 金鑰，以使用產生對稱 AES 金鑰命令和wrap屬性設定為進行包裝true。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

這個範例說明如何使用 AES 金鑰來使用key wrap aes-gcm命令。

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"
```

```
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<PAYLOAD_FILTER>

索引鍵參照 (例如key-reference=0xabc) 或以空格分隔的索引鍵屬性清單，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選取有效負載金鑰的形式。

必要：是

<PATH>

二進位檔案的路徑，其中將儲存包裝的金鑰資料。

必要：否

<WRAPPING_FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇包裝鍵的形式。

必要：是

<AAD>

AES GCM 其他驗證資料 (AAD) 值，以十六進位表示。

必要：否

<TAG_LENGTH_BITS>

AES GCM 標籤長度 (以位元為單位)。

必要：是

相關主題

- 按鍵包裝
- 密鑰展開

按鍵包裝 aes-no-pad

命key wrap aes-no-pad令會使用 HSM 和AES-NO-PAD包裝機制上的 AES 金鑰來封裝承載金鑰。裝載金鑰的extractable屬性必須設定為true。

只有金鑰的擁有者 (即建立金鑰的加密使用者 (CU) 才能包裝金鑰。共用金鑰的使用者可以在密碼編譯作業中使用金鑰。

若要使用此key wrap aes-no-pad命令，您必須先在 AWS CloudHSM 叢集中擁有 AES 金鑰。您可以使用產生對稱 AES 金鑰命令和wrap屬性設置為生成用於包裝的 AES 密鑰true。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

此範例顯示如何使用 AES 金鑰 (將wrap屬性值設定為) 來使用key wrap aes-no-pad命令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"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<PAYLOAD_FILTER>

索引鍵參照 (例如key-reference=0xabc) 或以空格分隔的索引鍵屬性清單，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選取有效負載金鑰的形式。

必要：是

<PATH>

二進位檔案的路徑，其中將儲存包裝的金鑰資料。

必要：否

<WRAPPING_FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇包裝鍵的形式。

必要：是

相關主題

- 按鍵包裝
- 密鑰展開

密鑰包裝一個-pkcs5 墊

命key wrap aes-pkcs5-pad令會使用 HSM 和AES-PKCS5-PAD包裝機制上的 AES 金鑰來封裝承載金鑰。裝載金鑰的extractable屬性必須設定為true。

只有金鑰的擁有者 (即建立金鑰的加密使用者 (CU) 才能包裝金鑰。共用金鑰的使用者可以在密碼編譯作業中使用金鑰。

若要使用此key wrap aes-pkcs5-pad命令，您必須先在 AWS CloudHSM 叢集中擁有 AES 金鑰。您可以使用產生對稱 AES 金鑰命令和wrap屬性設置為生成用於包裝的 AES 密鑰true。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

此範例顯示如何使用 AES 金鑰 (將`wrap`屬性值設定為) 來使用key wrap aes-pkcs5-pad命令`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="
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<PAYLOAD_FILTER>

索引鍵參照 (例如`key-reference=0xabc`) 或以空格分隔的索引鍵屬性清單，`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`以選取有效負載金鑰的形式。

必要：是

<PATH>

儲存包裝金鑰資料之二進位檔案的路徑。

必要：否

<WRAPPING_FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇包裝鍵的形式。

必要：是

相關主題

- 按鍵包裝
- 密鑰展開

按鍵包裝 aes-zero-pad

命key wrap aes-zero-pad令會使用 HSM 和AES-ZERO-PAD包裝機制上的 AES 金鑰來封裝承載金鑰。裝載金鑰的extractable屬性必須設定為true。

只有金鑰的擁有者 (即建立金鑰的加密使用者 (CU) 才能包裝金鑰。共用金鑰的使用者可以在密碼編譯作業中使用金鑰。

若要使用此key wrap aes-zero-pad命令，您必須先在 AWS CloudHSM 叢集中擁有 AES 金鑰。您可以產生 AES 金鑰，以使用將wrap屬性設定為的產生對稱 AES 金鑰命令進行包裝true。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

此範例顯示如何使用 AES 金鑰 (將`wrap`屬性值設定為) 來使用key wrap aes-zero-pad 命令`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"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<PAYLOAD_FILTER>

索引鍵參照 (例如key-reference=0xabc) 或以空格分隔的索引鍵屬性清單，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選取有效負載金鑰的形式。

必要：是

<PATH>

二進位檔案的路徑，其中將儲存包裝的金鑰資料。

必要：否

<WRAPPING_FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇包裝鍵的形式。

必要：是

相關主題

- 按鍵包裝
- 密鑰展開

按鍵包裝 cloudhsm-aes-gcm

命key wrap cloudhsm-aes-gcm令會使用 HSM 和CLOUDHSM-AES-GCM包裝機制上的 AES 金鑰來封裝承載金鑰。裝載金鑰的extractable屬性必須設定為true。

只有金鑰的擁有者 (即建立金鑰的加密使用者 (CU) 才能包裝金鑰。共用金鑰的使用者可以在密碼編譯作業中使用金鑰。

若要使用此key wrap cloudhsm-aes-gcm命令，您必須先在 AWS CloudHSM 叢集中擁有 AES 金鑰。您可以產生 AES 金鑰，以使用產生對稱 AES 金鑰命令和wrap屬性設定為進行包裝true。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

此範例顯示如何使用 AES 金鑰使用key wrap cloudhsm-aes-gcm命令。

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"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<PAYLOAD_FILTER>

索引鍵參照 (例如key-reference=0xabc) 或以空格分隔的索引鍵屬性清單，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選取有效負載金鑰的形式。

必要：是

<PATH>

儲存包裝金鑰資料之二進位檔案的路徑。

必要：否

<WRAPPING_FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇包裝鍵的形式。

必要：是

<AAD>

AES GCM 其他驗證資料 (AAD) 值，以十六進位表示。

必要：否

<TAG_LENGTH_BITS>

AES GCM 標籤長度 (以位元為單位)。

必要：是

相關主題

- 按鍵包裝
- 密鑰展開

密鑰包裝 RSA-艾斯

命key wrap rsa-aes令會使用 HSM 上的 RSA 公開金鑰和 RSA-AES 包裝機制來封裝承載金鑰。裝載金鑰的`extractable`屬性必須設定為`true`。

只有金鑰的擁有者 (即建立金鑰的加密使用者 (CU) 才能包裝金鑰。共用金鑰的使用者可以在密碼編譯作業中使用金鑰。

若要使用此key wrap rsa-aes命令，您必須先在 AWS CloudHSM 叢集中具有 RSA 金鑰。您可以使用關鍵 generate-asymmetric-pair 指令和`wrap`屬性設定為來產生 RSA 金鑰配對。`true`

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

此範例顯示如何使用 RSA 公開金鑰 (將wrap屬性值設定為) 來true使用key wrap rsa-ae命令。

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=="
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<PAYLOAD_FILTER>

索引鍵參照 (例如key-reference=0xabc) 或以空格分隔的索引鍵屬性清單，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選取有效負載金鑰的形式。

必要：是

<PATH>

二進位檔案的路徑，其中將儲存包裝的金鑰資料。

必要：否

<WRAPPING_FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇包裝鍵的形式。

必要：是

<MGF>

指定遮罩產生函數。

> **Note**
>
> 掩碼生成函數哈希函數必須與簽名機制哈希函數匹配。

有效值

- 毫克 1
- 毫克 1-沙 224
- 毫克 1-沙 256

- 毫克 1 沙 384
- 毫克 1-沙 512

必要：是

相關主題

- 按鍵包裝
- 密鑰展開

密鑰包裝 RSA-

命key wrap rsa-oaep令會使用 HSM 上的 RSA 公開金鑰和包裝機制來封RSA-OAEP裝承載金鑰。裝載金鑰的`extractable`屬性必須設定為`true`。

只有金鑰的擁有者 (即建立金鑰的加密使用者 (CU) 才能包裝金鑰。共用金鑰的使用者可以在密碼編譯作業中使用金鑰。

若要使用此key wrap rsa-oaep命令，您必須先在 AWS CloudHSM 叢集中具有 RSA 金鑰。您可以使用關鍵 generate-asymmetric-pair 指令和`wrap`屬性設定為來產生 RSA 金鑰配對。`true`

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

此範例顯示如何使用 RSA 公開金鑰 (將`wrap`屬性值設定為) 來`true`使用key wrap rsa-oaep命令。

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=="
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<PAYLOAD_FILTER>

索引鍵參照 (例如key-reference=0xabc) 或以空格分隔的索引鍵屬性清單，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選取有效負載金鑰的形式。

必要：是

<PATH>

儲存包裝金鑰資料之二進位檔案的路徑。

必要：否

<WRAPPING_FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇包裝鍵的形式。

必要：是

<MGF>

指定遮罩產生函數。

> **Note**
>
> 掩碼生成函數哈希函數必須匹配簽名機制哈希函數。

有效值

- 毫克 1 沙 1
- 毫克 1-沙 224
- 毫克 1-沙 256

- 毫克 -1 沙 384
- 毫克 1-沙 512

必要：是

相關主題

- 按鍵包裝
- 密鑰展開

密鑰包裝 RSA-pkcs

命key wrap rsa-pkcs令會使用 HSM 上的 RSA 公開金鑰和包裝機制來封RSA-PKCS裝承載金鑰。裝載金鑰的`extractable`屬性必須設定為`true`。

只有金鑰的擁有者 (即建立金鑰的加密使用者 (CU) 才能包裝金鑰。共用金鑰的使用者可以在密碼編譯作業中使用金鑰。

若要使用此key wrap rsa-pkcs命令，您必須先在 AWS CloudHSM 叢集中具有 RSA 金鑰。您可以使用關鍵 generate-asymmetric-pair 指令和`wrap`屬性設定為來產生 RSA 金鑰配對。`true`

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

要求

- 若要執行此命令，必須以 CU 的身分登入。

語法

```
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
```

範例

此範例顯示如何使用 RSA 公開金鑰使用key wrap rsa-pkcs命令。

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=="
  }
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<PAYLOAD_FILTER>

索引鍵參照 (例如key-reference=0xabc) 或以空格分隔的索引鍵屬性清單，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選取有效負載金鑰的形式。

必要：是

<PATH>

儲存包裝金鑰資料之二進位檔案的路徑。

必要：否

<WRAPPING_FILTER>

鍵引用（例如，key-reference=0xabc）或空格分隔的鍵屬性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以選擇包裝鍵的形式。

必要：是

相關主題

- 按鍵包裝
- 密鑰展開

登入

您可以使用 CloudHSM CLI 中的 login 命令，來登入及登出叢集內的每個 HSM。

> **Note**
>
> 如果您錯誤登入超過 5 次，系統將鎖定您的帳戶。如要解除鎖定，管理員須使用 cloudhsm_cli 中的 user change-password 命令來重設您的密碼。

登入和登出疑難排解

如果您在叢集中有多個 HSM，系統將允許您再嘗試登入幾次，若均登入錯誤，系統才會鎖定您的賬戶。這是因為 CloudHSM 用戶端可平衡各個 HSM 之間的負載。因此，每次登入嘗試可能會在不同的 HSM 上開始。如果您正在測試此功能，建議您在只有一個作用中 HSM 的叢集上執行此操作。

如果您在 2018 年 2 月前建立叢集，則系統將在您 20 次錯誤登入後鎖定您的賬戶。

使用者類型

下列使用者可以執行這些命令。

- 未激活的管理員
- 管理員
- 加密使用者 (CU)

語法

```
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')
```

範例

Example

此命令可允許您使用以 admin1 的管理員使用者登入憑證，來登入叢集中的所有 HSM。

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<USERNAME>

為使用者指定易記的名稱。長度上限為 31 個字元。允許的唯一特殊字元是底線 (_)。在此命令中，使用者名稱不區分大小寫，使用者名稱始終以小寫顯示。

必要：是

<ROLE>

指定指派給此使用者的角色。此為必要參數。有效值為 admin、crypto-user。

若要取得使用者的角色，請使用 user list 命令。如需 HSM 上使用者類型的詳細資訊，請參閱了解 HSM 使用者。

<PASSWORD>

指定登入 HSM 使用者的密碼。

相關主題

- 開始使用 CloudHSM CLI
- 啟用叢集

登錄 mfa-token-sign

使用 AWS CloudHSM CloudHSM CLI 中的 login mfa-token-sign 命令來使用多重要素驗證登入。若要使用此命令，您必須先設定 MFA 的 CloudHSM CLI。

使用者類型

下列使用者可以執行這些命令。

- 管理員
- 加密使用者 (CU)

語法

```
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
```

範例

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"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<TOKEN>

將寫入未簽署權杖檔案的檔案路徑。

必要：是

相關主題

- 開始使用 CloudHSM CLI
- 啟用叢集
- 使用 CloudHSM CLI 管理 MFA

登出

您可以使用 CloudHSM CLI 中的 logout 命令，登出叢集內的每個 HSM。

使用者類型

下列使用者可以執行此命令。

- 管理員
- 加密使用者 (CU)

語法

```
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
```

範例

Example

此命令會讓您登出叢集中的所有 HSM。

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

相關主題

- 開始使用 CloudHSM CLI
- 啟用叢集

使用者

user 是命令群組的父類別，當與父類別結合時，會建立使用者專用的命令。目前，用戶類別由以下命令組成：

- 變更使用者 MFA
- 變更使用者密碼
- 建立使用者
- 刪除使用者
- 使用者清單

變更使用者 MFA

目前，此類別由以下子命令組成：

- 變更使用者 MFA 權杖簽署

變更使用者 MFA 權杖簽署

使用 CloudHSM CLI 中的 user change-mfa 命令來更新使用者帳戶的多重要素驗證 (MFA) 設定。任何使用者帳戶均可執行此命令。具有管理員角色的帳戶可以為其他使用者執行此命令。

使用者類型

下列使用者可以執行此命令。

- 管理員
- 加密使用者

語法

目前，只有一個多重要素策略供使用者使用：字符簽署。

```
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)
```

Token Sign 策略需要一個權杖檔案將未簽署的權杖寫入其中。

```
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')
```

範例

此命令會將叢集中每個 HSM 的未簽署權杖寫入 token 指定的檔案。當系統提示您時，請在檔案中簽署權杖。

Example ：寫入叢集中每個 HSM 的未簽署權杖

```
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"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<ROLE>

指賦予使用者帳戶的角色。此為必要參數。如需 HSM 上使用者類型的詳細資訊，請參閱了解 HSM 使用者。

有效值

- 管理員：管理員可以管理使用者，但無法管理金鑰。
- 加密使用者：加密使用者可以建立管理金鑰並在密碼編譯操作時使用該金鑰。

<USERNAME>

為使用者指定易記的名稱。長度上限為 31 個字元。允許的唯一特殊字元是底線 (_)。

建立使用者之後，您就無法再變更使用者的名稱。在 CloudHSM CLI 命令中，角色和密碼需區分大小寫，但使用者名稱不區分大小寫。

必要：是

<CHANGE_PASSWORD>

指正在註冊/取消註冊 MFA 使用者的純文字新密碼。

必要：是

<TOKEN>

將寫入未簽署權杖檔案的檔案路徑。

必要：是

<APPROVAL>

指要核准操作的已簽署規定人數權杖檔案的檔案路徑。只有在規定人數使用者服務規定人數值大於 1 時才需要執行此操作。

<DEREGISTER>

取消註冊 MFA 公有金鑰 (如有)。

<CHANGE-QUORUM>

變更規定人數公有金鑰以及 MFA 金鑰。

相關主題

- 了解 HSM 使用者的 2FA

變更使用者密碼

使用 CloudHSM CLI 中的user change-password命令來變更叢集中現有使用者的 AWS CloudHSM 密碼。若要為使用者啟用 MFA，請使用 `user change-mfa` 命令。

任何使用者都可以變更自己的密碼。此外，具有管理員角色的使用者可以變更叢集中其他使用者的密碼。您無需輸入目前的密碼即可進行變更。

> **Note**
>
> 您無法變更目前已登入叢集使用者的密碼。

使用者類型

下列使用者可以執行此命令。

- 管理員
- 加密使用者 (CU)

語法

> **Note**
>
> 若要為使用者啟用多重要素驗證 (MFA)，請使用 user change-mfa 命令。

```
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')
```

範例

以下範例說明如何使用 user change-password 重設叢集中目前使用者或其他任何使用者的密碼。

Example ：變更您的密碼

叢集中的任何使用者均可使用 user change-password 變更自己的密碼。

以下輸出顯示 Bob 目前正以加密使用者 (CU) 的身分登入。

```
aws-cloudhsm > user change-password --username bob --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "bob",
    "role": "crypto-user"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<APPROVAL>

指要核准操作的已簽署規定人數權杖檔案的檔案路徑。只有在規定人數使用者服務規定人數值大於 1 時才需要執行此操作。

<DEREGISTER-MFA>

取消註冊 MFA 公有金鑰 (如有)。

<DEREGISTER-QUORUM>

取消註冊規定人數公有鑰 (如有)。

<PASSWORD>

指使用者的純文字新密碼。

必要：是

<ROLE>

指賦予使用者帳戶的角色。此為必要參數。如需 HSM 上使用者類型的詳細資訊，請參閱了解 HSM 使用者。

有效值

- 管理員：管理員可以管理使用者，但無法管理金鑰。
- 加密使用者：加密使用者可以建立管理金鑰並在密碼編譯操作時使用該金鑰。

<USERNAME>

為使用者指定易記的名稱。長度上限為 31 個字元。允許的唯一特殊字元是底線 (_)。

建立使用者之後，您就無法再變更使用者的名稱。在 CloudHSM CLI 命令中，角色和密碼需區分大小寫，但使用者名稱不區分大小寫。

必要：是

相關主題

- 使用者清單
- 建立使用者
- 刪除使用者

變更使用者規定人數

user change-quorum 是命令群組的父類別，當與父類別結合時，會為使用者建立一個用於變更規定人數的命令。

user change-quorum 用於使用特定的規定人數策略註冊使用者規定人數驗證。如下所示：從 SDK 5.8.0 開始，使用者只能使用一種規定人數策略。

目前，此類別由以下類別和子命令組成：

- 字符簽署
  - 註冊

更改使用者規定人數權杖簽署

user change-quorum token-sign 是命令群組的父類別，當與該父類別結合時，會建立執行規定人數權杖簽署的命令。

目前，此類別由以下命令組成：

- 註冊

變更規定人數使用者 註冊字符簽署

使用 CloudHSM CLI 中的 user change-quorum token-sign register 命令為管理員使用者註冊規定人數字符簽署策略。

使用者類型

下列使用者可以執行此命令。

- 管理員

語法

```
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')
```

範例

Example

若要執行此命令，您須以您希望的使用者身分登入 register quorum token-sign。

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

此 user change-quorum token-sign register 命令會向 HSM 註冊您的公有金鑰。因此，該命令可讓您作為規定人數核准人執行必要的核准操作，這些操作需要使用者取得規定人數簽章，以此滿足必要的規定人數值閾值。

```
aws-cloudhsm > user change-quorum token-sign register \
    --public-key /home/mypemfile \
    --signed-token /home/mysignedtoken
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

您現在可以執行 user list 命令，並確認已為此使用者註冊規定人數字符簽署。

```
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"
      }
    ]
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<PUBLIC-KEY>

公有金鑰 PEM 檔案的檔案路徑。

必要：是

<SIGNED-TOKEN>

帶有由使用者私有金鑰簽署的權杖的文件路徑。

必要：是

相關主題

- 使用 CloudHSM CLI 管理規定人數驗證
- 對管理員使用規定人數身分驗證：首次設定
- 變更管理員的規定人數最小值
- 支援規定人數身分驗證的服務名稱和類型

建立使用者

CloudHSM CLI 中的user create AWS CloudHSM 命令會在您的叢集中建立一個使用者。只有具有管理員角色的使用者帳戶可以執行此命令。

使用者類型

下列類型的使用者可以執行此命令。

- 管理員

要求

若要執行此命令，必須以管理員使用者的身分登入。

語法

```
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')
```

範例

下列範例顯示如何使用 user create 在您的 HSM 中建立新使用者。

Example ：建立加密使用者

此範例會在您的 AWS CloudHSM 叢集中建立具有加密使用者角色的帳戶。

```
aws-cloudhsm > user create --username alice --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<USERNAME>

為使用者指定易記的名稱。長度上限為 31 個字元。允許的唯一特殊字元是底線 (_)。在此命令中，使用者名稱不區分大小寫，使用者名稱始終以小寫顯示。

必要：是

<ROLE>

指定指派給此使用者的角色。此為必要參數。有效值為 admin、crypto-user。

若要取得使用者的角色，請使用 user list 命令。如需 HSM 上使用者類型的詳細資訊，請參閱了解 HSM 使用者。

<PASSWORD>

指登入 HSM 使用者的密碼。

必要：是

<APPROVAL>

指要核准操作的已簽署規定人數權杖檔案的檔案路徑。只有在規定人數使用者服務規定人數值大於 1 時才需要執行此操作。

相關主題

- 使用者清單
- 刪除使用者
- 變更使用者密碼

刪除使用者

CloudHSM CLI 中的user delete AWS CloudHSM 命令會從您的叢集中刪除使用者。只有具有管理員角色的使用者帳戶可以執行此命令。您無法刪除目前已登入 HSM 的使用者。

使用者類型

下列類型的使用者可以執行此命令。

- 管理員

要求

- 您無法刪除擁有金鑰的使用者帳戶。
- 您的使用者帳戶必須具有管理員角色才能執行此命令。

語法

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
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
```

範例

```
aws-cloudhsm > user delete --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<USERNAME>

為使用者指定易記的名稱。長度上限為 31 個字元。允許的唯一特殊字元是底線 (_)。在此命令中，使用者名稱不區分大小寫，使用者名稱始終以小寫顯示。

必要：是

<ROLE>

指定指派給此使用者的角色。此為必要參數。有效值為 admin、crypto-user。

若要取得使用者的角色，請使用 user list 命令。如需 HSM 上使用者類型的詳細資訊，請參閱了解 HSM 使用者。

必要：是

<APPROVAL>

指要核准操作的已簽署規定人數權杖檔案的檔案路徑。只有在規定人數使用者服務規定人數值大於 1 時才需要執行此操作。

必要：是

相關主題

- 使用者清單
- 建立使用者
- 變更使用者密碼

使用者清單

CloudHSM CLI 中的 user list 命令會列出 CloudHSM 叢集中的使用者帳戶。您無須登入 CloudHSM CLI 即可執行此命令。

> **Note**
>
> 如果您新增或刪除 HSM，請更新用 AWS CloudHSM 戶端和命令列工具使用的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列類型的使用者可以執行此命令。

- 所有使用者。您無須登入即可執行此命令。

語法

```
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
```

範例

此命令會列出 CloudHSM 叢集中的使用者。

```
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"
      }
    ]
  }
}
```

輸出包含下列使用者屬性：

- 使用者名稱：顯示使用者定義的使用者易記名稱。使用者名稱一律以小寫顯示。
- 角色：決定使用者在 HSM 上可執行的操作。
- 已鎖定：表示此使用者帳戶是否已鎖定。
- MFA：表示此使用者帳戶支援的多重要素驗證機制。
- 叢集涵蓋範圍：表示此使用者帳戶在叢集範圍內的可用性。

相關主題

- key_mgmt_util 中的 listUsers
- 建立使用者
- 刪除使用者
- 變更使用者密碼

規定人數

quorum 是命令群組的父類別，當與 quorum 結合時，會建立規定人數身分驗證或 N 的 M 操作的專用命令。目前，此類別由 token-sign 子類別組成，該子類別由其自己的命令組成。按一下以下連結了解詳細資訊。

- 字符簽署

管理員服務：規定人數身分驗證用於管理員特殊權限服務，例如建立使用者、刪除使用者、變更使用者密碼、設定規定人數值，以及停用規定人數和 MFA 功能。

每種服務類型都會進一步細分為合格的服務名稱，這包含一組特定的可執行法定人數支援的服務操作。

服務名稱	服務類型	服務操作
使用者	管理員	• 建立使用者 • 刪除使用者 • 變更使用者密碼 • 變更使用者 MFA
規定人數	管理員	• 法定令牌符號 set-quorum-value

相關主題

- 對管理員使用規定人數身分驗證：首次設定
- 使用 CloudHSM CLI 管理規定人數身分驗證 (M/N 個存取控制)

簽署規定人數權杖

quorum token-sign 是命令群組的父類別，當與 quorum token-sign 結合時，會建立規定人數身分驗證或 N 的 M 操作的專用命令。

目前，此類別由以下命令組成：

- 刪除
- 產生
- 列出
- list-quorum-values
- 列出逾時
- set-quorum-value
- 設定逾時

刪除規定人數字符簽署

使用 CloudHSM CLI 中的 quorum token-sign delete 命令刪除規定人數授權服務的一個或多個權杖。

使用者類型

下列使用者可以執行此命令。

- 管理員

語法

```
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')
```

範例

下列範例顯示了如何使用 CloudHSM CLI 中的 quorum token-sign delete 命令刪除規定人數授權服務的一個或多個權杖。

Example ：刪除規定人數授權服務的一個或多個權杖

```
aws-cloudhsm > quorum token-sign delete --scope all
{
  "error_code": 0,
```

```
  "data": "Deletion of quorum token(s) successful"
}
```

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<SCOPE>

AWS CloudHSM 群集中令牌將被刪除的範圍。

有效值

- 使用者：僅用於刪除登入使用者擁有的權杖。
- 全部：用於刪除 AWS CloudHSM 群集中的所有令牌。

相關主題

- 使用者清單
- 建立使用者
- 刪除使用者

產生規定人數字符簽署

使用 CloudHSM CLI 中的 quorum token-sign generate 命令為規定人數授權服務產生一個權杖。

對於服務使用者和規定人數而言，HSM 叢集上，每位使用者每種服務取得作用中權杖有一定限制。

> **Note**
>
> 只有管理員可以產生一個服務權杖。

管理員服務：規定人數身分驗證用於管理員特殊權限服務，例如建立使用者、刪除使用者、變更使用者密碼、設定規定人數值，以及停用規定人數和 MFA 功能。

每種服務類型都會進一步細分為合格的服務名稱，這包含一組特定的可執行法定人數支援的服務操作。

服務名稱	服務類型	服務操作
使用者	管理員	• 建立使用者 • 刪除使用者 • 變更使用者密碼 • 變更使用者 MFA
規定人數	管理員	• 法定令牌符號 set-quorum-value

使用者類型

下列使用者可以執行此命令。

- 管理員
- 加密使用者 (CU)

語法

```
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
```

範例

此命令會將叢集中每個 HSM 的未簽署權杖寫入 token 指定的檔案。

Example ：寫入叢集中每個 HSM 的未簽署權杖

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

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<SERVICE>

指定要產生字符的規定人數授權服務。此為必要參數。

有效值

- 使用者：指使用者管理服務，用於執行規定人數授權的使用者管理操作。
- 規定人數：指規定人數管理服務，用於為任何規定人數授權服務設定規定人數授權的規定人數值。
- 登記：產生一個未簽署權杖，用於登記規定人數授權的公有金鑰。

必要：是

<TOKEN>

將寫入未簽署權杖檔案的檔案路徑。

必要：是

相關主題

- 支援規定人數身分驗證的服務名稱和類型

列出規定人數字符簽署

使用 CloudHSM CLI 中的quorum token-sign list命令列出叢集中存在的所有權杖符號仲裁權杖。 AWS CloudHSM

使用者類型

下列使用者可以執行此命令。

- 管理員
- 加密使用者 (CU)

語法

```
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
```

範例

此命令將列出集群中存在的所有令牌符號令牌。 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"
      }
    ]
  }
}
```

相關主題

- 產生規定人數字符簽署

法定令牌符號 list-quorum-values

使用 CloudHSM CLI 中的quorum token-sign list-quorum-values命令列出叢集中設定的 AWS CloudHSM 法定值。

使用者類型

下列使用者可以執行此命令。

- 所有使用者。您無須登入即可執行此命令。

語法

```
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
```

範例

此指令會列出 AWS CloudHSM 叢集中針對每個服務設定的法定值。

Example

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

相關主題

- 支援規定人數身分驗證的服務名稱和類型

規定人數字符簽署 列出逾時

使用 CloudHSM CLI 中的 quorum token-sign list-timeouts 命令來取得所有權杖類型的權杖逾時期間 (以秒為單位)。

使用者類型

下列使用者可以執行此命令。

- 所有使用者。您無須登入即可執行此命令。

語法

```
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
```

範例

Example

```
aws-cloudhsm > quorum token-sign list-timeouts
{
  "error_code": 0,
  "data": {
    "generated": 600,
    "approved": 600
  }
}
```

輸出包括以下內容：

- 已產生：待核准產生權杖的逾時期間 (以秒為單位)。
- 已核准：執行規定人數授權操作的已核准待使用權杖的逾時期間 (以秒為單位)。

相關主題

- 規定人數字符簽署 設定逾時

法定令牌符號 set-quorum-value

使用 CloudHSM CLI 中的 quorum token-sign set-quorum-value 命令為規定人數授權服務設定一個新的規定人數值。

使用者類型

下列使用者可以執行此命令。

- 管理員

語法

```
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')
```

範例

Example

下列範例中，此命令會將叢集中每個 HSM 的未簽署權杖寫入權杖指定的檔案。當系統提示您時，請在檔案中簽署權杖。

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

然後，您可以執行 list-quorum-values 命令來確認規定人數管理服務的規定人數值已設定完成。

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

引數

<CLUSTER_ID>

執行此作業的叢集識別碼。

必要：如果已設定多個叢集。

<APPROVAL>

HSM 上待核准的已簽署權杖檔案的檔案路徑。

<SERVICE>

指定要產生字符的規定人數授權服務。此為必要參數。如需有關服務類型和名稱的詳細資訊，請參閱支援規定人數身分驗證的服務名稱和類型。

有效值

- 使用者：使用者管理服務。用於執行規定人數授權的使用者管理操作的服務。
- 規定人數：規定人數管理服務。用於為任何規定人數授權服務而設定規定人數授權的規定人數值。
- 登記：產生一個未簽署權杖，用於登記規定人數授權的公有金鑰。

必要：是

<VALUE>

指要設定的規定人數值。最大規定人數值為八 (8)。

必要：是

相關主題

- 法定令牌符號 list-quorum-values
- 支援規定人數身分驗證的服務名稱和類型

規定人數字符簽署 設定逾時

使用 CloudHSM CLI 中的 quorum token-sign set-timeout 命令來設定每種權杖類型的權杖逾時期間 (以秒為單位)。

使用者類型

下列使用者可以執行此命令。

- 管理員

語法

```
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')
```

範例

下列範例示範如何使用 quorum token-sign set-timeout 命令來設定權杖逾時期間。

```
aws-cloudhsm > quorum token-sign set-timeout --generated 900
{
  "error_code": 0,
  "data": "Set token timeout successful"
}
```

相關主題

- 規定人數字符簽署 列出逾時

CloudHSM 管理公用程式 (CMU)

cloudhsm_mgmt_util 命令列工具可協助加密管理員在 HSM 中管理使用者。它包括可建立、刪除和列出使用者，以及變更使用者密碼的工具。

KMU 和 CMU 系統是用戶端 SDK 3 套件的一部分。用戶端 SDK 3 及其相關命令列工具 (金鑰管理公用程式和 CloudHSM 管理公用程式) 僅適用於 HSM 類型 hsm1.medium。

cloudhsm_mgmt_util 還包含允許加密使用者 (CU) 共用金鑰以及取得和設定金鑰屬性的命令。這些命令可以補充主索引鍵管理工具 key_mgmt_util 中的金鑰管理命令。

如需快速入門，請參閱 管理複製的叢集。如需 cloudhsm_mgmt_util 命令的詳細資訊以及使用命令的範例，請參閱 cloudhsm_mgmt_util command 參考。

主題

- 支援的 AWS CloudHSM 管理公用程式平台
- CloudHSM 管理公用程式 (CMU) 入門
- 安裝和設定 AWS CloudHSM 用戶端 (Linux)
- 安裝和設定 AWS CloudHSM 用戶端

- cloudhsm_mgmt_util command 參考

支援的 AWS CloudHSM 管理公用程式平台

Linux 支援

- 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

Windows 支援

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

CloudHSM 管理公用程式 (CMU) 入門

CloudHSM 管理公用程式 (CMU) 可讓您管理硬體安全模組 (HSM) 使用者。使用本主題可開始進行基本的 HSM 使用者管理工作，例如建立使用者、列出使用者以及將 CMU 連線到叢集。

1. 若要使用 CMU，您必須先使用設定工具，以 `--cmu` 參數和叢集中其中一個 HSM 的 IP 地址來更新本機 CMU 組態。每次使用 CMU 時，執行此操作，以確保您管理叢集中每個 HSM 上的 HSM 使用者。

Linux

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

Windows

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

2. 在互動式模式中輸入下列命令啟動 CLI。

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
```

輸出應該類似如下，視您有多少個 HSM 而定。

```
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.
```

cloudhsm_mgmt_util 執行時，系統提示會切換至 aws-cloudhsm>。

3. 使用 loginHSM 命令登入叢集。任何類型的使用者都可以使用此命令來登入叢集。

以下範例中的命令以 admin 登入，這是預設的加密管理員 (CO)。您已在啟用叢集時設定此使用者的密碼。您可以使用 -hpswd 參數來隱藏您的密碼。

```
aws-cloudhsm>loginHSM CO admin -hpswd
```

系統會提示您輸入密碼。您輸入密碼時，系統會隱藏密碼，並且輸出顯示命令已成功，並顯示您已連線至叢集上的所有 HSM。

```
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. 用 listUsers 來列出叢集上的所有使用者。

```
aws-cloudhsm>listUsers
```

CMU 列出了叢集上的所有使用者。

```


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. 使用 createUser 來建立名為 **example_user** 的 CU 使用者，密碼為 **password1**。

   您可以在應用程式中使用 CU 使用者來執行密碼編譯和金鑰管理操作。您可以建立 CU 使用者，因為在步驟 3 中您是以 CO 使用者身分登入的。只有 CO 使用者可以使用 CMU 執行使用者管理工作，例如建立和刪除使用者，以及變更其他使用者的密碼。

   ```
   aws-cloudhsm>createUser CU example_user password1
   ```

   CMU 會提示您建立使用者操作。

   ```
   
   ************************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. 若要建立 CU 使用者 **example_user**，請輸入 **y**。

7. 用 listUsers 來列出叢集上的所有使用者。

   ```
   aws-cloudhsm>listUsers
   ```

   CMU 會列出叢集上的所有使用者，包括您剛建立的新 CU 使用者。

   ```
   
   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. 使用 logoutHSM 命令來登出 HSM。

```
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. 使用 quit 命令來停止 cloudhsm_mgmt_util。

```
aws-cloudhsm>quit
```

```
disconnecting from servers, please wait...
```

安裝和設定 AWS CloudHSM 用戶端 (Linux)

若要與 AWS CloudHSM 叢集中的 HSM 互動，您需要適用於 Linux 的 AWS CloudHSM 用戶端軟體。應該將此軟體安裝到您先前建立的 Linux EC2 用戶端執行個體。如果是使用 Windows，您也可以安裝用戶端。如需詳細資訊，請參閱 安裝和設定 AWS CloudHSM 用戶端。

任務

- 安裝用 AWS CloudHSM 戶端和命令列工具
- 編輯用戶端組態

安裝用 AWS CloudHSM 戶端和命令列工具

Connect 線至用戶端執行個體並執行下列命令，以下載並安裝 AWS CloudHSM 用戶端和命令列工具。

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
```

編輯用戶端組態

您必須先編輯用 AWS CloudHSM 戶端組態，才能使用用戶端連線至叢集。

編輯用戶端組態

1. 如果在 cloudhsm_mgmt_util 上安裝用戶端 SDK 3，請完成下列步驟以確保叢集中的所有節點均已同步。

   a. 執行 configure -a ***<IP of one of the HSMs>***。

   b. 重新啟動用戶端服務。

   c. 執行 config -m。

2. 將您的發行憑證 (您使用此憑證來簽署叢集憑證) 複製到用戶端執行個體上的這個位置：`/opt/cloudhsm/etc/customerCA.crt`。您的用戶端執行個體上需要有執行個體根使用者許可，才能將憑證複製到此位置。

3. 使用下列 configure 命令來更新用 AWS CloudHSM 戶端和命令列工具的組態檔，並指定叢集中 HSM 的 IP 位址。若要取得 HSM 的 IP 位址，請在AWS CloudHSM 主控台中檢視叢集，或執行命describe-clusters AWS CLI 令。在命令輸出中，HSM 的 IP 地址是 `EniIp` 欄位的值。如果您有多個 HSM，請選擇任何 HSM 的 IP 地址；任何一個都可以。

   ```
   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. 前往 啟用叢集。

安裝和設定 AWS CloudHSM 用戶端

若要在 Windows 上使用 AWS CloudHSM 叢集中的 HSM，您需要適用於 Windows 的 AWS CloudHSM 用戶端軟體。應將此軟體安裝到您先前建立的 Windows Server 執行個體。

安裝 (或更新) 最新 Windows 用戶端和命令列工具

1. 連接至您的 Windows Server 執行個體。
2. 下載 AWSCloudHSMClient-latest.msi 安裝程式。
3. 如果在 cloudhsm_mgmt_util 上安裝用戶端 SDK 3，請完成下列步驟以確保叢集中的所有節點均已同步。
   a. 執行 configure -a ***<IP of one of the HSMs>***。
   b. 重新啟動用戶端服務。
   c. 執行 config -m。
4. 轉到您的下載位置並以管理權限運行安裝程序（AWSCloudHSMClient-latest.msi）。
5. 依照安裝程式指示操作，然後在安裝程式完成後選擇關閉。
6. 將您自己簽署的發行憑證 (用於簽署叢集憑證的憑證) 複製到 `C:\ProgramData\Amazon\CloudHSM` 資料夾。
7. 使用下列命令以更新組態檔案。如果您正在更新用戶端，務必在重新組態時停止和啟動用戶端。

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

8. 前往 啟用叢集。

備註：

- 如果您正在更新用戶端，系統將不會覆寫前一次安裝的現有組態檔案。
- 適用於 Windows 的 AWS CloudHSM 用戶端安裝程式會自動註冊密碼編譯 API：下一代 (CNG) 和金鑰儲存區提供者 (KSP)。若要解除安裝用戶端，請再次執行安裝程式並依照解除安裝指示操作。
- 如果您使用的 Linux，也可以安裝 Linux 用戶端。如需更多詳細資訊，請參閱 安裝和設定 AWS CloudHSM 用戶端 (Linux)。

cloudhsm_mgmt_util command 參考

cloudhsm_mgmt_util 命令列工具可協助加密管理員在 HSM 中管理使用者。此工具還包含允許加密使用者 (CU) 共用金鑰以及取得和設定金鑰屬性的命令。這些命令可以補充 key_mgmt_util 命令列工具中的主索引鍵管理命令。

如需快速入門，請參閱 管理複製的叢集。

您必須先啟動 cloudhsm_mgmt_util 並登入 HSM 後，才能執行任何 cloudhsm_mgmt_util 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

若要列出所有 cloudhsm_mgmt_util 命令，請執行下以下命令：

```
aws-cloudhsm> help
```

若要取得任一 cloudhsm_mgmt_util 命令的語法，請執行以下命令：

```
aws-cloudhsm> help <command-name>
```

> **Note**
>
> 根據文件使用語法。雖然內建的軟體說明可以提供其他選項，但您不應將這些選項視為支援選項，亦不應在生產程式碼中使用。

若要執行某個命令，請輸入命令名稱，或輸入該命令與 cloudhsm_mgmt_util 其他命令的名稱相區別的名稱。

例如，輸入 listUsers 或 listU 均可取得 HSM 上的使用者清單。

```
aws-cloudhsm> listUsers
```

若要結束 cloudhsm_mgmt_util 工作階段，請執行以下命令：

```
aws-cloudhsm> quit
```

如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

下列各主題說明 cloudhsm_mgmt_util 中的命令。

> **Note**
>
> key_mgmt_util 和 cloudhsm_mgmt_util 中的某些命令，名稱相同。但是，這些命令通常具有不同的語法，不同的輸出和略微不同的功能。

Command	描述	使用者類型
changePswd	變更 HSM 上使用者的密碼。任何使用者都可以變更自己的密碼。CO 可以變更任何人的密碼。	CO
createUser	在 HSM 上建立所有類型的使用者。	CO
deleteUser	從 HSM 刪除所有類型的使用者。	CO
findAllKeys	取得使用者擁有或共用的金鑰。也可取得每個 HSM 上所有金鑰的金鑰擁有權和共用資料的雜湊。	CO、AU
getAttribute	取得 AWS CloudHSM 金鑰的屬性值，並將其寫入檔案或標準輸出 (標準輸出)。	加密使用者
getCert	取得特定 HSM 的憑證並將其儲存為所需的憑證格式。	全部。
getHSMInfo	取得 HSM 執行所在硬體的相關資訊。	全部。無需登入。
getKeyInfo	取得金鑰的擁有者、共用使用者和仲裁身分驗證的狀態。	全部。無需登入。

Command	描述	使用者類型
info	取得 HSM 的資訊，包括 IP 地址、主機名稱、連接埠和目前的使用者。	全部。無需登入。
listUsers	取得每個 HSM 中的使用者、使用者類型和 ID，以及其他屬性。	全部。無需登入。
loginHSM 和 logoutHSM	登入及登出 HSM。	全部。
quit	結束 cloudhsm_mgmt_util。	全部。無需登入。
伺服器	由 HSM 上進入及結束伺服器模式。	全部。
registerQuorumPub鑰匙	將 HSM 使用者與非對稱 RSA-2048 金鑰對產生關聯。	CO
setAttribute	變更現有金鑰的標籤、加密、解密、包裝和取消包裝屬性的值。	加密使用者
shareKey	與其他使用者共用現有的金鑰。	加密使用者
syncKey	跨複製 AWS CloudHSM 的叢集同步金鑰。	CU、CO
syncUser	跨複製 AWS CloudHSM 的叢集同步使用者。	CO

changePswd

cloudhsm_mgmt_util 中的 changePswd 命令會變更叢集中 HSM 上現有使用者的密碼。

任何使用者都可以變更自己的密碼。此外，加密管理員 (CO 和 PCO) 可以變更其他加密管理員或加密使用者 (CU) 的密碼。您無需輸入目前的密碼即可進行變更。

> **Note**
>
> 您無法變更目前登入用 AWS CloudHSM 戶端或 key_mgmt_util 的使用者密碼。

changePswd 疑難排解

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列使用者可以執行此命令。

- 加密管理員 (CO)
- 加密使用者 (CU)

語法

依照語法圖表中指定的順序輸入引數。使用 `-hpswd` 參數來遮罩您的密碼。若要為 CO 使用者啟用雙重要素驗證 (2FA)，請使用 `-2fa` 參數並包含檔案路徑。如需詳細資訊，請參閱 the section called “引數”。

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

範例

以下範例說明如何使用 changePassword 重設 HSM 中目前使用者或任何其他使用者的密碼。

Example ：變更您的密碼

HSM 上的任何使用者均可使用 changePswd 變更自己的密碼。變更密碼之前，請使用 info 取得叢集中每個 HSM 的資訊，包括使用者名稱和登入使用者的使用者類型。

以下輸出顯示 Bob 目前正以加密使用者 (CU) 的身分登入。

```
        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)'
```

為了變更密碼，Bob 執行 changePswd 後接使用者類型、使用者名稱和新的密碼。

```
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 ：變更其他使用者的密碼

您必須是 CO 或 PCO 才能變更 HSM 上其他 CO 或 CU 的密碼。在變更其他使用者的密碼之前，請先使用 info 命令確認您的使用者類型為 CO 或 PCO。

以下輸出確認 Alice 目前已登入，且她是 CO。

```
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)'
```

Alice 想要重設其他使用者 (John) 的密碼。在變更密碼之前，她先使用 listUsers 命令來驗證 John 的使用者類型。

以下輸出列出 John 為 CO 使用者。

```
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
```

為了變更密碼，Alice 執行 changePswd 後接 John 的使用者類型、使用者名稱和新的密碼。

```
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
```

引數

依照語法圖表中指定的順序輸入引數。使用 -hpswd 參數來遮罩您的密碼。若要為 CO 使用者啟用雙重要素驗證 (2FA)，請使用 -2fa 參數並包含檔案路徑。如需使用 2FA 的詳細資訊，請參閱 使用 CMU 管理 2FA。

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user-type>

指您要變更密碼之使用者的目前類型。您不能使用 changePswd 來變更使用者類型。

有效值為 CO、CU、PCO、PRECO。

若要取得使用者類型，請使用 listUsers。如需 HSM 上使用者類型的詳細資訊，請參閱 了解 HSM 使用者。

必要：是

<user-name>

指使用者的易記名稱。此參數不區分大小寫。您不能使用 changePswd 來變更使用者名稱。

必要：是

<password | -hpswd >

為使用者指定新密碼。輸入長度 7 到 32 個字元的字串。此值區分大小寫。輸入密碼時，密碼會以純文字顯示。要隱藏您的密碼，請使用 -hpswd 參數代替密碼，然後按照提示進行操作。

必要：是

[-2fa </path/to/authdata>]

指為此 CO 使用者啟用 2FA。若要取得設定 2FA 所需的資料，請在 -2fa 參數後加入檔案系統帶有檔案名稱位置的路徑。如需使用 2FA 的詳細資訊，請參閱 使用 CMU 管理 2FA。

必要：否

相關主題

- info
- listUsers
- createUser
- deleteUser

createUser

cloudhsm_mgmt_util 中的 createUser 命令可在 HSM 上建立使用者。只有加密管理員 (CO 和 PRECO) 可以執行此命令。命令成功時，它會在叢集的所有 HSM 中建立使用者。

createUser 疑難排解

如果您的 HSM 組態不準確，可能不會在所有 HSM 上建立使用者。若要將使用者新增至未包含該使用者的任何 HSM，請僅在遺漏該使用者的 HSM 上使用 syncUser 或 createUser 命令。為了避免組態錯誤，請執行 configure 工具搭配 -m 選項。

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列類型的使用者可以執行此命令。

- 加密管理員 (CO、PRECO)

語法

依照語法圖表中指定的順序輸入引數。使用 -hpswd 參數來遮罩您的密碼。若要為 CO 使用者啟用雙重要素身分驗證 (2FA)，請使用 -2fa 參數並包含檔案路徑。如需詳細資訊，請參閱 the section called "引數"。

```
createUser <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

範例

下列範例顯示如何使用 createUser 在您的 HSM 中建立新使用者。

Example ：建立加密管理員

此範例會在叢集中的 HSM 上建立加密管理員 (CO)。第一個命令會使用 loginHSM 以加密管理員的身分登入 HSM。

```
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)
```

第二個命令會使用 createUser 命令來建立 alice (即 HSM 上新的加密管理員)。

警告訊息說明該命令會在叢集中的所有 HSM 上建立使用者。但如果此命令在任何 HSM 上失敗，使用者將不會存在於那些 HSM 上。若要繼續，請輸入 y。

輸出顯示已在叢集中的所有三個 HSM 上建立新使用者。

```
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
```

命令完成時，alice 會擁有與 HSM 上與 admin CO 使用者相同的許可，包括變更 HSM 上任何使用者的密碼。

最後一個命令會使用 listUsers 命令，來驗證在叢集的所有三個 HSM 上是否存在 alice。輸出也會顯示 alice 已獲指派使用者 ID 3。.您可以使用使用者 ID 來識別alice其他命令，例如findAllKeys。

```
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 ：建立加密使用者

此範例會在 HSM 上建立加密使用者 (CU) bob。加密使用者可以建立和管理金鑰，但不能管理使用者。

在您輸入 y 來回應警告訊息之後，輸出會顯示已在叢集中的全部三個 HSM 上建立 bob。新 CU 可以登入 HSM 來建立和管理金鑰。

此命令使用 `defaultPassword` 的密碼值。之後，bob 或任何 CO 可以使用 changePswd 命令來變更他的密碼。

```
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
```

引數

依照語法圖表中指定的順序輸入引數。使用 -hpswd 參數來遮罩您的密碼。若要建立啟用雙重要素驗證 (2FA) 的 CO 使用者，請使用 -2fa 參數並包含檔案路徑。如需 2FA 的詳細資訊，請參閱 使用 CMU 管理 2FA。

```
createUser <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user-type>

指使用者的類型。此為必要參數。

如需 HSM 上使用者類型的詳細資訊，請參閱 了解 HSM 使用者。

有效值：

- CO：加密管理員可以管理使用者，但不能管理金鑰。
- CU ：加密使用者可以建立管理金鑰並在密碼編譯操作時使用該金鑰。

當您在 HSM 啟用期間指派密碼時，PRECO 會轉換為 CO。

必要：是

<user-name>

為使用者指定易記的名稱。長度上限為 31 個字元。允許的唯一特殊字元是底線 (_)。

建立使用者之後，您就無法再變更使用者的名稱。在 cloudhsm_mgmt_util 命令中，使用者類型和密碼需區分大小寫，但使用者名稱不區分大小寫。

必要：是

<password | -hpswd >

為使用者指定密碼。輸入長度 7 到 32 個字元的字串。此值區分大小寫。輸入密碼時，密碼會以純文字顯示。要隱藏您的密碼，請使用 `-hpswd` 參數代替密碼，然後按照提示進行操作。

若要變更使用者密碼，請使用 changePswd。任何 HSM 使用者可以變更自己的密碼，但 CO 使用者可以變更 HSM 上任何使用者 (任何類型) 的密碼。

必要：是

[-2fa </path/to/authdata>]

指建立已啟用 2FA 的 CO 使用者。若要取得設定 2FA 驗證所需資料，請在 `-2fa` 參數後加入檔案系統帶有檔案名稱位置的路徑。如需設定並使用 2FA 的詳細資訊，請參閱 使用 CMU 管理 2FA。

必要：否

相關主題

- listUsers

- deleteUser
- syncUser
- changePswd

deleteUser

cloudhsm_mgmt_util 中的 deleteUser 命令會從硬體安全性模組 (HSM) 中刪除使用者。只有加密管理員 (CO) 可以執行此命令。您無法刪除目前已登入 HSM 的使用者。如需刪除使用者的詳細資訊，請參閱如何刪除 HSM 使用者。

> **Tip**
>
> 您無法刪除擁有金鑰的加密使用者 (CU)。

使用者類型

下列類型的使用者可以執行此命令。

- CO

語法

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
deleteUser <user-type> <user-name>
```

範例

此範例會從叢集的 HSM 刪除加密管理員 (CO)。第一個命令使用 listUsers 列出 HSM 上的所有使用者。

輸出顯示使用者 3 (alice) 是 HSM 上的 CO。

```
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
```

第二個命令使用 deleteUser 命令以從 HSM 刪除 `alice`。

輸出顯示命令在叢集中的三個 HSM 上均成功。

```
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)
```

最後一個命令使用 listUsers 命令，來驗證已從叢集上的所有三個 HSM 中刪除 `alice`。

```
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
```

引數

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
deleteUser <user-type> <user-name>
```

<user-type>

指使用者的類型。此為必要參數。

> **Tip**
>
> 您無法刪除擁有金鑰的加密使用者 (CU)。

有效值為 CO、CU。

若要取得使用者類型，請使用 listUsers。如需 HSM 上使用者類型的詳細資訊，請參閱 了解 HSM 使用者。

必要：是

<user-name>

為使用者指定易記的名稱。長度上限為 31 個字元。允許的唯一特殊字元是底線 (_)。

建立使用者之後，您就無法再變更使用者的名稱。在 cloudhsm_mgmt_util 命令中，使用者類型和密碼需區分大小寫，但使用者名稱不區分大小寫。

必要：是

相關主題

- listUsers
- createUser
- syncUser
- changePswd

findAllKeys

cloudhsm_mgmt_util 中的 findAllKeys 命令可取得指定的加密使用者 (CU) 所擁有或共用的金鑰。還會傳回每個 HSM 上的使用者資料的雜湊。有關使用者、金鑰擁有權和金鑰共用資料在叢集的所有 HSM 上是否都相同，此雜湊可讓您一目了然。在輸出中，由使用者擁有的金鑰會由（o）註釋，且共佣金鑰會由（s）註釋。

即使 HSM 上的所有 CU 可以使用任何公有金鑰，只有當指定的 CU 擁有金鑰時，findAllKeys 才會傳回公有金鑰。此行為不同於 key_mgmt_util 中的 findKey，此命令會傳回所有 CU 使用者的公有金鑰。

只有加密主管 (CO 和 PCO) 和設備使用者 (AU) 才可以執行此命令。加密使用者 (CU) 可執行以下命令：

- listUsers 尋找所有使用者
- 在 key_mgmt_util 中 findKey 可查找他們可以使用的金鑰

- getKeyInfo在 key_mgmt_util 中找到他們擁有或共享的特定密鑰的所有者和共享用戶

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列使用者可以執行此命令。

- 加密管理員 (CO、PCO)
- 設備使用者 (AU)

語法

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

範例

這些範例示範如何使用 `findAllKeys` 尋找每個 HSM 上使用者的所有金鑰，以及取得金鑰使用者資訊的雜湊。

Example ：尋找 CU 的金鑰

此範例使用 findAllKeys，在 HSM 中尋找使用者 4 所擁有和共用的金鑰。此命令使用 `0` 做為第二個引數的值，以隱藏雜湊值。由於省略選用的檔案名稱，此命令會寫入 stdout (標準輸出)。

輸出顯示使用者 4 可以使用 6 個金鑰：8、9、17、262162、19 和 31。輸出會使用 (s) 指出由使用者明確共用的金鑰。使用者擁有的金鑰由 (o) 指定，並包含使用者不共享的對稱和私有金鑰，以及可供所有加密使用者使用的公有金鑰。

```
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 ：驗證使用者資料已同步

此範例使用 findAllKeys，以驗證叢集中的所有 HSM 是否包含相同的使用者、金鑰擁有權和金鑰共用值。為了這樣做，此範例取得每個 HSM 上的金鑰使用者資料雜湊並比較雜湊值。

為了取得金鑰雜湊，此命令使用第二個引數中的 1 值。由於選用的檔案名稱已遭省略，此命令會將金鑰雜湊寫入 stdout。

此範例指定使用者 6，但對於 HSM 上擁有或共用任何金鑰的任何使用者，雜湊值都相同。如果指定的使用者未擁有或共用任何金鑰，例如 CO，則此命令不會傳回雜湊值。

輸出顯示叢集內的兩個 HSM 具有相同的金鑰雜湊。如果其中一個 HSM 有不同的使用者、不同的金鑰擁有者或不同的共用使用者，則雜湊值就不相等。

```
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)
```

此命令示範雜湊值代表 HSM 上所有金鑰的使用者資料。此命令針對使用者 3 使用 findAllKeys。與使用者 6 不同 (只擁有或共用 3 個金鑰)，使用者 3 擁有或共用 17 個金鑰，但金鑰雜湊值相同。

```
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)
```

引數

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

<user id>

取得指定之使用者擁有或共用的所有金鑰。輸入 HSM 上某個使用者的使用者 ID。若要尋找所有使用者的使用者 ID，請使用 listUsers。

所有使用者 ID 都有效，但 `findAllKeys` 只會傳回加密使用者 (CU) 的金鑰。

必要：是

<key hash>

納入 (1) 或排除 (0) 每個 HSM 中所有金鑰的使用者擁有權和共用資料的雜湊。

`user id` 引數代表擁有或共用金鑰的使用者時，將會填入金鑰雜湊。對於 HSM 上擁有或共用金鑰的所有使用者，即使他們擁有和共用不同的金鑰，金鑰雜湊值也都相同。不過，當 `user id` 代表未擁有或共用任何金鑰的使用者時 (例如 CO)，就不會填入雜湊值。

必要：是

<output file>

將輸出寫入指定的檔案。

必要：否

預設：Stdout

相關主題

- changePswd
- deleteUser
- listUsers
- syncUser
- key_mgmt_util 中的 findKey
- getKeyInfo在密鑰中

getAttribute

cloudhsm_mgmt_util 中的 getAttribute 命令可從叢集內的所有 HSM 取得金鑰的一個屬性值，並將其寫入 stdout (標準輸出) 或檔案。只有加密使用者 (CU) 可以執行此命令。

金鑰屬性是金鑰的屬性。其中包括的特性，例如金鑰類型、類別、標籤和 ID，以及代表您可對金鑰執行之動作的值，例如加密、解密、包裝、取消包裝和驗證。

您只可以對您擁有的金鑰和與您共用的金鑰使用 getAttribute。您可以執行此命令或 key_mgmt_util 中的 getAttribute 命令，將金鑰的一個或所有屬性值寫入檔案。

若要取得屬性的清單和代表它們的常數，請使用 listAttributes 命令。若要變更現有金鑰的屬性值，請使用 key_mgmt_util 中的 setAttribute 和 cloudhsm_mgmt_util 中的 setAttribute。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列使用者可以執行此命令。

- 加密使用者 (CU)

語法

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
getAttribute <key handle> <attribute id> [<filename>]
```

範例

此範例在 HSM 中取得金鑰之可擷取屬性的值。您可以使用如下命令，以判斷您是否可以從 HSM 匯出金鑰。

第一個命令使用 listAttributes，以尋找代表可擷取之屬性的常數。輸出顯示 OBJ_ATTR_EXTRACTABLE 的常數是 354。您也可以在金鑰屬性參考中找到這項資訊以及屬性和其值的描述。

```
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
```

第二個命令使用 getAttribute，在 HSM 中取得金鑰 (金鑰控點為 262170) 之可擷取屬性的值。為了指定可擷取屬性，此命令使用 354，這是代表屬性的常數。因為此命令未指定檔案名稱，所以 getAttribute 會將輸出寫入 stdout。

輸出顯示在所有 HSM 上，可擷取之屬性的值都是 1。這個值表示金鑰擁有者可以匯出金鑰。當值為 0 (0x0) 時，表示無法從 HSM 匯出金鑰。您在建立金鑰時會設定可擷取屬性的值，但無法變更。

```
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
```

引數

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
getAttribute <key handle> <attribute id> [<filename>]
```

<key-handle>

指定目標金鑰的金鑰控制代碼。每一個命令中只能指定一個金鑰。若要取得金鑰的金鑰控制代碼，請在 key_mgmt_util 中使用 findKey。

您必須擁有或共用指定的金鑰。要查找密鑰的用戶，請getKeyInfo在 key_mgmt_util 中使用。

必要：是

<attribute id>

識別屬性。輸入代表屬性的常數，或輸入 512 來代表所有屬性。例如，若要取得金鑰類型，請輸入 256，此常數代表 OBJ_ATTR_KEY_TYPE 屬性。

若要列出屬性及其常數，請使用 listAttributes。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

必要：是

<filename>

將輸出寫入指定的檔案。輸入檔案路徑。

如果指定的檔案已存在，getAttribute 會覆寫檔案且不會警告。

必要：否

預設：Stdout

相關主題

- key_mgmt_util 中的 getAttribute
- listAttributes
- cloudhsm_mgmt_util 中的 setAttribute
- key_mgmt_util 中的 setAttribute
- 金鑰屬性參考

getCert

使用 cloudhsm_mgmt_util 中的 getCert 命令可擷取叢集內特定 HSM 的憑證。執行此命令時，請指定欲擷取的憑證類型。為此，您應使用以下引數一節所述的其中一個對應的整數。若要了解該節所述各類憑證的角色，請參閱驗證 HSM 身分。

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列使用者可以執行此命令。

- 所有使用者。

必要條件

開始之前，您必須先由目標 HSM 上進入伺服器模式。如需詳細資訊，請參閱伺服器。

語法

在伺服器模式下使用 getCert 命令：

```
server> getCert <file-name> <certificate-type>
```

範例

首先，進入伺服器模式。此命令使用伺服器編號 0 由 HSM 上進入伺服器模式。

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

接著使用 getCert 命令。本範例使用 /tmp/P0.crt 做為儲存憑證的檔案名稱，使用 4 (客戶根憑證) 做為所需的憑證類型：

```
server0> getCert /tmp/P0.crt 4
getCert Success
```

引數

```
getCert <file-name> <certificate-type>
```

<file-name>

指定儲存憑證的檔案名稱。

必要：是

<certificate-type>

以整數指定欲擷取的憑證類型。各整數及其對應的憑證類型如下所示：

- 1：製造商根憑證
- 2：製造商硬體憑證
- 4：客戶根憑證
- 8：叢集憑證 (由客戶根憑證進行簽署)
- 16：叢集憑證 (鏈結到製造商根憑證)

必要：是

相關主題

- server

getHSMInfo

cloudhsm_mgmt_util 中的 getHSMInfo 命令可取得執行每個 HSM 之硬體的相關資訊，包括機型、序號、FIPS 狀態、記憶體、溫度，以及硬體和韌體的版本編號。此資訊還包括可讓 cloudhsm_mgmt_util 用來參考 HSM 的伺服器 ID。

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列類型的使用者可以執行此命令。

- 所有使用者。您無須登入即可執行此命令。

語法

此命令沒有任何參數。

```
getHSMInfo
```

範例

此範例使用 getHSMInfo，以取得叢集中 HSM 的相關資訊。

```
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

...
```

相關主題

- info

getKeyInfo

key_mgmt_util 中的 getKeyInfo 命令會傳回使用者的 HSM 使用者 ID，這些使用者可以使用該金鑰，包括擁有者和將金鑰與其共用的加密使用者 (CU)。對金鑰啟用規定人數身分驗證時，getKeyInfo 也會傳回必須核准使用該金鑰的加密操作的使用者數量。您只可以對您擁有的金鑰以及與您共用的金鑰執行 getKeyInfo。

當您對公有金鑰執行 getKeyInfo 時，即使 HSM 的所有使用者都可以使用該公有金鑰，getKeyInfo 只會傳回該金鑰擁有者。若要尋找 HSM 中使用者的 HSM 使用者 ID，請使用 listUsers。若要找出特定使用者的金鑰，請使用 key_mgmt_util 中的 findKey -u。加密軍官可以findAllKeys在雲中使用.

您擁有您建立的金鑰。您可以在建立金鑰時與其他使用者共用金鑰。之後，若要共用或取消共用現有的金鑰，請使用 cloudhsm_mgmt_util 中的 shareKey。

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

語法

```
getKeyInfo -k <key-handle> [<output file>]
```

範例

這些範例顯示如何使用 getKeyInfo，以取得金鑰使用者的相關資訊。

Example ：取得非對稱金鑰的使用者

此命令會取得可以對金鑰控制代碼 262162 使用 AES (非對稱) 金鑰的使用者。輸出顯示使用者 3 擁有金鑰，並且已將它與使用者 4 和使用者 6 共用。

只有使用者 3、4 和 6 可以對金鑰 262162 執行 getKeyInfo。

```
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 ：取得對稱金鑰對的使用者

這些命令使用 getKeyInfo，以取得可以在 ECC (對稱) 金鑰對中使用金鑰的使用者。公有金鑰具有金鑰控制代碼 262179。私有金鑰具有金鑰控制代碼 262177。

對私有金鑰 (262177) 執行 getKeyInfo 時，它會傳回金鑰擁有者 (3) 以及與其共用金鑰的加密使用者 (CU) 4。

```
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
```

對公有金鑰 (262179) 執行 getKeyInfo 時，只會傳回金鑰擁有者 (使用者 3)。

```
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
```

若要確認使用者 4 可以使用該公有金鑰 (和 HSM 上的所有公開金鑰)，請使用 key_mgmt_util 中 findKey 的 -u 參數。

輸出示範使用者 4 可以同時使用金鑰對中的公開 (262179) 與私密 (262177) 金鑰。使用者 4 也可以使用它們所建立或與其共用的所有其他公開金鑰和任何私密金鑰。

```
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 ：取得金鑰的規定人數身分驗證值 (m_value)

此範例示範如何取得金鑰的 m_value。m_value 為規定人數中的使用者數量，這些使用者必須核准使用該金鑰的任何密碼編譯操作與共用和取消共用金鑰的操作。

對金鑰啟用了仲裁身分驗證時，必須有仲裁的使用者核准使用該金鑰的任何密碼編譯操作。若要啟用規定人數驗證並設定規定人數數量，建立金鑰時請使用 -m_value 參數。

這個命令用genSymKey來建立與使用者 4 共用的 256 位元 AES 金鑰。它會使用 m_value 參數來啟用規定人數驗證，並將仲裁大小設定為兩個使用者。必須有足夠多的使用者，才能提供所需的核准。

輸出顯示此命令建立了金鑰 10。

```
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
```

此命令使用 cloudhsm_mgmt_util 中的 getKeyInfo 以取得金鑰 10 使用者的相關資訊。輸出顯示使用者 3 擁有該金鑰，並且將它與使用者 4 共用。它還示範兩個使用者的規定人數必須核准使用該金鑰的每個密碼編譯操作。

```
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
```

引數

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
getKeyInfo -k <key-handle> <output file>
```

<key-handle>

指定 HSM 中某個金鑰的金鑰控制代碼。輸入您擁有或共用的金鑰的金鑰控制代碼。此為必要參數。

必要：是

<output file>

將輸出寫入指定的檔案，而非 stdout。如果檔案存在，命令會覆寫檔案且不會有任何警告。

必要：否

預設：stdout

相關主題

- getKeyInfo在密鑰中
- key_mgmt_util 中的 findKey
- findAllKeys在雲中 _ 微米特
- listUsers
- shareKey

info

cloudhsm_mgmt_util 中的 info 命令可取得叢集內每個 HSM 的相關資訊，包括主機名稱、連接埠、IP 地址，以及在 HSM 上登入 cloudhsm_mgmt_util 之使用者的名稱和類型。

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列類型的使用者可以執行此命令。

- 所有使用者。您無須登入即可執行此命令。

語法

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
info server <server ID>
```

範例

此範例使用 info，以取得叢集中 HSM 的相關資訊。此命令使用 0 表示叢集中的第一個 HSM。輸出顯示 IP 地址、連接埠以及目前使用者的類型和名稱。

```
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)'
```

引數

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
info server <server ID>
```

<server id>

指定 HSM 的伺服器 ID。序號指派給 HSM，代表 HSM 新增到叢集的順序 (從 0 開始)。若要尋找 HSM 的伺服器 ID，請使用 getHSMInfo。

必要：是

相關主題

- getHSMInfo
- loginHSM 和 logoutHSM

listAttributes

在 cloudhsm_mgmt_util 中的listAttributes指令會列出索引鍵的屬性，以及代表這些索引 AWS CloudHSM 鍵的常數。您可以使用這些常數來識別 getAttribute 和 setAttribute 命令中的屬性。

如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

使用者類型

下列使用者可以執行此命令。

- 所有使用者。您無須登入即可執行此命令。

語法

```
listAttributes [-h]
```

範例

此命令可列出您在 key_mgmt_util 中可取得和變更的金鑰屬性，以及代表這些屬性的常數。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。若要代表所有屬性，請使用 512。

```
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
```

參數

-h

顯示命令的說明。

必要：是

相關主題

- getAttribute
- setAttribute
- 金鑰屬性參考

listUsers

cloudhsm_mgmt_util 中的 listUsers 命令可取得各 HSM 中的使用者，以及其使用者類型和其他屬性。所有類型的使用者可以執行此命令。您甚至不需要登入 cloudhsm_mgmt_util 即可執行此命令。

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列類型的使用者可以執行此命令。

- 所有使用者。您無須登入即可執行此命令。

語法

此命令沒有任何參數。

```
listUsers
```

範例

此命令列出叢集的每一個 HSM 上的使用者，並顯示其屬性。您可以在其他命令中 (例如 deleteUser、changePswd 和 findAllKeys) 使用 `User ID` 屬性來識別使用者。

```
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
```

輸出包含下列使用者屬性：

- 使用者 ID：在 key_mgmt_util 和 cloudhsm_mgmt_util 命令中識別使用者。
- 使用者類型：決定使用者在 HSM 上可執行的操作。
- 使用者名稱：顯示使用者定義的使用者易記名稱。
- MofnPubKey：指出使用者是否已註冊用於簽署仲裁驗證權杖的 key pair。
- LoginFailureCnt：指出使用者未成功登入的次數。
- 2FA：指出使用者已啟用多重要素驗證。

相關主題

- key_mgmt_util 中的 listUsers

- createUser
- deleteUser
- changePswd

loginHSM 和 logoutHSM

您可以使用 cloudhsm_mgmt_util 中的 loginHSM 和 logoutHSM 命令，登入及登出叢集內的每個 HSM。任何類型的任何使用者都可以使用這些命令。

> **Note**
>
> 如果您錯誤登入超過 5 次，系統將鎖定您的帳戶。若要解除鎖定帳戶，加密管理員 (CO) 必須在 cloudhsm_mgmt_util 中使用 changePswd 命令來重設您的密碼。

loginHSM 和 logoutHSM 疑難排解

您必須先啟動 cloudhsm_mgmt_util，然後再執行這些 cloudhsm_mgmt_util 命令。

如果您新增或刪除 HSM，請更新用 AWS CloudHSM 戶端和命令列工具使用的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

如果您在叢集中有多個 HSM，系統將允許您再嘗試登入幾次，若均登入錯誤，系統才會鎖定您的賬戶。這是因為 CloudHSM 用戶端可平衡各個 HSM 之間的負載。因此，每次登入嘗試可能會在不同的 HSM 上開始。如果您正在測試此功能，建議您在只有一個作用中 HSM 的叢集上執行此操作。

如果您在 2018 年 2 月前建立叢集，則系統將在您 20 次錯誤登入後鎖定您的賬戶。

使用者類型

下列使用者可以執行這些命令。

- 準加密員 (PRECO)
- 加密管理員 (CO)
- 加密使用者 (CU)

語法

依照語法圖表中指定的順序輸入引數。使用 -hpswd 參數來遮罩您的密碼。若要啟用雙重要素驗證 (2FA)，請使用 -2fa 參數並包含檔案路徑。如需詳細資訊，請參閱 the section called “引數”。

```
loginHSM <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

```
logoutHSM
```

範例

這些範例顯示如何使用 loginHSM 和 logoutHSM 登入和登出叢集中的所有 HSM。

Example ：登入叢集中的 HSM

此命令會使用名為 admin、密碼為 co12345 的 CO 使用者登入憑證資訊，來登入叢集中的所有 HSM。輸出會顯示您已成功執行此命令，且已連接到 HSM (在此案例中是 server 0 和 server 1)。

```
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 ：使用隱藏密碼登入

此命令與以上範例相同，只不過這次您指定系統應該隱藏密碼。

```
aws-cloudhsm>loginHSM CO admin -hpswd
```

系統會提示您輸入密碼。您輸入密碼時後，系統會隱藏密碼，並且輸出會顯示您已成功執行此命令，並顯示您已連線至叢集上的所有 HSM。

```
Enter password:

loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)

aws-cloudhsm>
```

Example ：登出 HSM

此命令會登出您目前登入的 HSM (在此案例中是 `server 0` 和 `server 1`)。輸出會顯示已成功執行此命令，且使用者已從 HSM 中斷開連線。

```
aws-cloudhsm>logoutHSM

logoutHSM success on server 0(10.0.2.9)
logoutHSM success on server 1(10.0.3.11)
```

引數

依照語法圖表中指定的順序輸入引數。使用 `-hpswd` 參數來遮罩您的密碼。若要啟用雙重要素驗證 (2FA)，請使用 `-2fa` 參數並包含檔案路徑。如需使用 2FA 的詳細資訊，請參閱 使用 CMU 管理 2FA。

```
loginHSM <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user type>

指登入 HSM 的使用者類型。如需詳細資訊，請參閱上面的使用者類型。

必要：是

<user name>

指登入 HSM 使用者的使用者名稱。

必要：是

<password | -hpswd >

指登入 HSM 使用者的密碼。要隱藏您的密碼，請使用 `-hpswd` 參數代替密碼，然後按照提示進行操作。

必要：是

[-2fa </path/to/authdata>]

指系統應使用第二個要素來驗證該啟用 2FA 的 CO 使用者。若要取得啟用 2FA 登入的所需資料，請在 `-2fa` 參數後加入檔案系統帶有檔案名稱位置的路徑。如需使用 2FA 的詳細資訊，請參閱 使用 CMU 管理 2FA。

必要：否

相關主題

- cloudhsm_mgmt_util 入門
- 啟用叢集

registerQuorumPub鑰匙

cloudhsm_mgmt_util 中的 registerQuorumPubKey 命令可將硬體安全性模組 (HSM) 使用者與非對稱式 RSA-2048 金鑰配對相關聯。您將 HSM 使用者與金鑰建立關聯後，這些使用者就可以使用私有金鑰來核准規定人數要求，且叢集可以使用已註冊的公有金鑰來確認簽章是否來自使用者。如需有關規定人數身分驗證的詳細資訊，請參閱管理規定人數身分驗證 (M/N 存取控制)。

> **Tip**
>
> 在 AWS CloudHSM 文件中，法定驗證有時稱為 M in N (MoFn)，這表示總數 N 個核准人中最少有 M 個核准人。

使用者類型

下列類型的使用者可以執行此命令。

- 加密管理員 (CO)

語法

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

範例

此範例說明如何使用 registerQuorumPubKey 將加密管理員 (CO) 註冊為規定人數身分驗證要求的核准者。若要執行此命令，您須擁有非對稱 RSA-2048 金鑰對、已簽署權杖和未簽署權杖。有關這些需求的詳細資訊，請參閱 the section called “引數”。

Example ：註冊 HSM 使用者以進行法定驗證

此範例會將名為 quorum_officer 的 CO 註冊為核准人，以進行規定人數身分驗證。

```
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)
```

最後一個命令使用 listUsers 命令來驗證 quorum_officer 是否註冊為 MofN 使用者。

```
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
```

引數

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-
token> <public-key>
```

<user-type>

指使用者的類型。此為必要參數。

如需 HSM 上使用者類型的詳細資訊，請參閱 了解 HSM 使用者。

有效值：

- CO：加密管理者可以管理使用者，但不能管理金鑰。

必要：是

<user-name>

為使用者指定易記的名稱。長度上限為 31 個字元。允許的唯一特殊字元是底線 (_)。

建立使用者之後，您就無法再變更使用者的名稱。在 cloudhsm_mgmt_util 命令中，使用者類型和密碼需區分大小寫，但使用者名稱不區分大小寫。

必要：是

<registration-token>

指定包含未簽署註冊權杖檔案的路徑。可包含最大 245 個位元組的任何隨機資料。如需有關建立未簽署註冊權杖的詳細資訊，請參閱建立並簽署註冊權杖。

必要：是

<signed-registration-token>

指包含註冊權杖的 SHA256_PKCS 機制簽署雜湊的文件的路徑。如需詳細資訊，請參閱建立並簽署註冊權杖。

必要：是

<public-key>

指包含非對稱 RSA-2048 金鑰對的公有金鑰檔案的路徑。使用私有金鑰簽署註冊權杖。如需詳細資訊，請參閱建立 RSA 金鑰對。

必要：是

> **Note**
>
> 叢集使用相同的金鑰進行規定人數身分驗證和雙重要素驗證 (2FA)。這表示您無法為使用 registerQuorumPubKey 啟用 2FA 的使用者輪換規定人數金鑰。若要輪換金鑰，您必須使用 changePswd。如需關於使用規定人數身分驗證和 2FA 的詳細資訊，請參閱規定人數身分驗證和 2FA。

相關主題

- 建立 RSA 金鑰對
- 建立並簽署註冊權杖
- 用 HSM 註冊公有金鑰
- 管理規定人數身分驗證 (M/N 存取控制)
- 規定人數驗證和 2FA
- listUsers

伺服器

一般而言，當您從 cloudhsm_mgmt_util 中發出命令時，該命令將會影響指定的叢集 (全域模式) 內的所有 HSM。不過，有時您可能需要向單一 HSM 發出命令。例如，若自動同步失敗，您可能需要同步某個 HSM 上的金鑰和使用者，以維持整個叢集的一致性。您可以使用 cloudhsm_mgmt_util 中的 server 命令進入伺服器模式，並直接與特定的 HSM 執行個體互動。

成功啟動後，`aws-cloudhsm>` 命令提示將取代為 `server>` 命令提示。

若要結束伺服器模式，請使用 `exit` 命令。成功退出後，您即會返回 cloudhsm_mgmt_util 命令提示。

您必須先啟動 cloudhsm_mgmt_util，然後再執行任何 cloudhsm_mgmt_util 命令。

使用者類型

下列使用者可以執行此命令。

- 所有使用者。

必要條件

若要進入伺服器模式，您必須事先得知目標 HSM 的伺服器編號。伺服器編號會列示於 cloudhsm_mgmt_util 在啟動時所產生的追蹤輸出中。伺服器編號將按照 HSM 顯示在組態檔案中的相同順序進行指派。本範例假設 `server 0` 是與所需 HSM 相對應的伺服器。

語法

啟動伺服器模式：

```
server <server-number>
```

結束伺服器模式：

```
server> exit
```

範例

此命令使用伺服器編號 0 由 HSM 上進入伺服器模式。

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

若要結束伺服器模式，請使用 exit 命令。

```
server0> exit
```

引數

```
server <server-number>
```

<server-number>

指定目標 HSM 的伺服器編號。

必要：是

exit 命令沒有任何引數。

相關主題

- syncKey
- createUser
- deleteUser

setAttribute

cloudhsm_mgmt_util 中的 setAttribute 命令可變更 HSM 中金鑰的標籤、加密、解密、包裝和取消包裝屬性的值。您也可以在 key_mgmt_util 中使用 setAttribute 命令，將工作階段金鑰轉換為持久性金鑰。您只能變更您所擁有金鑰的屬性。

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列使用者可以執行此命令。

- 加密使用者 (CU)

語法

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
setAttribute <key handle> <attribute id>
```

範例

此範例說明如何停用對稱金鑰的解密功能。您可使用這樣的命令來設定包裝金鑰，該金鑰應該能夠包裝和取消包裝其他金鑰，但不能加密或解密資料。

第一步是建立包裝金鑰。這個命令使用genSymKey在 key_mgmt_util 產生 256 位元的 AES 對稱金鑰。輸出顯示新金鑰的金鑰控制代碼是 14。

```
$  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
```

接著，我們想要確認目前解密屬性的值。若要取得解密屬性的屬性 ID，請使用 listAttributes。輸出顯示代表 OBJ_ATTR_DECRYPT 屬性的常數是 261。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

```
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
```

如要取得金鑰 14 目前解密屬性的值，下一個命令會在 cloudhsm_mgmt_util 中使用 getAttribute。

輸出顯示在叢集的兩個 HSM 上，解密屬性的值均為 true (1)。

```
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
```

此命令使用 setAttribute，將金鑰 14 的解密屬性 (屬性 261) 的值變更為 0。這將停用該金鑰的解密功能。

輸出顯示該命令在叢集中的兩個 HSM 上均成功執行。

```
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)
```

最後的命令會重複 getAttribute 命令。同樣地，執行該命令會取得金鑰 14 的解密屬性 (屬性 261)。

此時，輸出顯示在叢集中的兩個 HSM 上，解密屬性的值均為 false (0)。

```
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
```

引數

```
setAttribute <key handle> <attribute id>
```

<key-handle>

指定您擁有金鑰的金鑰控制代碼。每一個命令中只能指定一個金鑰。若要取得金鑰的金鑰控制代碼，請在 key_mgmt_util 中使用 findKey。若要尋找金鑰的使用者，請使用getKeyInfo。

必要：是

<attribute id>

指定常數，其代表您要變更的屬性。每一個命令中只能指定一個屬性。如要取得屬性及其整數值，請使用 listAttributes。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

有效值：

- 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。

必要：是

相關主題

- key_mgmt_util 中的 setAttribute
- getAttribute
- listAttributes
- 金鑰屬性參考

結束

cloudhsm_mgmt_util 中的 quit 命令可結束 cloudhsm_mgmt_util。任何類型的任何使用者均可使用此命令。

您必須先啟動 cloudhsm_mgmt_util，然後再執行任何 cloudhsm_mgmt_util 命令。

使用者類型

下列使用者可以執行此命令。

- 所有使用者。您無須登入即可執行此命令。

語法

```
quit
```

範例

這個命令會結束 cloudhsm_mgmt_util。成功結束後，您將返回慣常的命令列。此命令沒有任何輸出參數。

```
aws-cloudhsm> quit

disconnecting from servers, please wait...
```

相關主題

- cloudhsm_mgmt_util 入門

shareKey

cloudhsm_mgmt_util 中的 shareKey 命令可將您擁有的金鑰與其他加密使用者共用，也可以與其他加密使用者取消共用。只有金鑰擁有者可以共用和取消共用金鑰。您也可以在建立金鑰時共用金鑰。

共用金鑰的使用者可在加密操作中使用此金鑰，但無法刪除、匯出、共用、取消共用此金鑰，或變更其屬性。對金鑰啟用規定人數身分驗證時，任何會共用或取消共用此金鑰的操作，都必須經過規定人數核准。

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列類型的使用者可以執行此命令。

- 加密使用者 (CU)

語法

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

使用者類型：加密使用者 (CU)

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

範例

以下範例示範如何使用 shareKey，將您擁有的金鑰與其他加密使用者共用，也可以與其他加密使用者取消共用。

Example ：共用金鑰

此範例使用 shareKey，將目前使用者擁有的 ECC 私密金鑰與 HSM 上的另一位加密使用者共用。HSM 的所有使用者都能使用公開金鑰，因此您無法共用或取消共用公開金鑰。

第一個命令用getKeyInfo來取得金鑰的使用者資訊262177，也就是 HSM 上的 ECC 私密金鑰。

輸出顯示金鑰 262177 由使用者 3 擁有，但不共用。

```
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
```

此命令使用 shareKey 將金鑰 262177 與使用者 4 (HSM 上的另一位加密使用者) 共用。最後一個引數使用的值是 1，表示共用操作。

輸出顯示該操作在叢集的兩個 HSM 上均成功執行。

```
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)
```

為確認操作成功，範例會重複第一個 getKeyInfo 命令。

輸出顯示金鑰 262177 現在已共用給使用者 4。

```
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 ：取消共用金鑰

此範例取消共用對稱金鑰，即：從金鑰的共用使用者清單中移除加密使用者。

此命令使用 shareKey 從金鑰 6 的共用使用者清單中移除使用者 4。最後一個引數使用的值是 0，表示取消共用操作。

輸出顯示命令在兩個 HSM 上均成功執行。因此，使用者 4 無法再於加密操作中使用金鑰 6。

```
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)
```

引數

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

<key-handle>

指定您擁有金鑰的金鑰控制代碼。每一個命令中只能指定一個金鑰。若要取得金鑰的金鑰控制代碼，請在 key_mgmt_util 中使用 findKey。若要驗證您是否擁有金鑰，請使用getKeyInfo。

必要：是

<user id>

指定加密使用者 (CU) 的使用者 ID，表示您要將金鑰共用或取消共用給此使用者。如要尋找使用者的使用者 ID，請使用 listUsers。

必要：是

<share 1 or unshare 0>

如要將金鑰共用給指定的使用者，請輸入 1。如要取消共用金鑰，即：從金鑰的共用使用者清單中移除指定的使用者，請輸入 0。

必要：是

相關主題

- getKeyInfo

syncKey

您可使用 cloudhsm_mgmt_util 中的 syncKey 命令，在一個叢集或複製叢集間的 HSM 執行個體間同步金鑰。一般而言，您不需要使用此命令，這是因為叢集中的 HSM 執行個體會自動同步金鑰。但是您須手動同步複製叢集之間的金鑰。複製的叢集通常會在不同的 AWS 區域中建立，以簡化全域擴展和災難復原程序。

您不能用 syncKey 在任意叢集之間同步金鑰：其中一個叢集必須是從另一個叢集的備份建立而來。此外，兩個叢集必須擁有一致的 CO 和 CU 登入資料，操作才能成功。如需詳細資訊，請參閱 HSM 使用者。

若要使用syncKey，您必須先建立一個 AWS CloudHSM 組態檔，從來源叢集指定一個 HSM，另一個從目的地叢集指定一個 HSM。這可讓 cloudhsm_mgmt_util 連線到兩個 HSM 執行個體。使用此組態檔案啟動 cloudhsm_mgmt_util。然後，使用擁有您要同步金鑰的 CO 或 CU 登入資料登入。

使用者類型

下列類型的使用者可以執行此命令。

- 加密管理員 (CO)
- 加密使用者 (CU)

> **Note**
>
> CO 可以在任何金鑰上使用 syncKey，而 CU 只能在他們擁有的金鑰上使用此命令。如需詳細資訊，請參閱 the section called “了解 HSM 使用者”。

必要條件

開始之前，您須知道來源 HSM 上要與目標 HSM 同步之金鑰的 `key handle`。如要尋找 `key handle`，請使用 listUsers 命令列出具名使用者的所有識別符。然後，使用findAllKeys命令尋找屬於特定使用者的所有金鑰。

您也需了解指派給來源及目標 HSM 的 `server IDs`，這會顯示在啟動時由 cloudhsm_mgmt_util 所傳回的追蹤輸出中的具體內容。這些項目的指派順序就是 HSM 顯示在組態檔案中的順序。

遵循複製叢集間使用 CMU 的說明，並使用新的組態檔案初始化 cloudhsm_mgmt_util。然後，發出 server 命令，在來源 HSM 上進入伺服器模式。

語法

> **Note**
>
> 若要執行 syncKey，請先在包含要同步金鑰的 HSM 上進入伺服器模式。

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

使用者類型：加密使用者 (CU)

```
syncKey <key handle> <destination hsm>
```

範例

執行 server 命令，登入來源 HSM 並進入伺服器模式。在該範例中，我們假設 `server 0` 是來源 HSM。

```
aws-cloudhsm> server 0
```

現在執行 syncKey 命令。在這個範例中，我們假設金鑰 261251 要同步到 `server 1`。

```
aws-cloudhsm> syncKey 261251 1
syncKey success
```

引數

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
syncKey <key handle> <destination hsm>
```

<key handle>

指定要同步金鑰的金鑰控制代碼。每一個命令中只能指定一個金鑰。若要取得金鑰的金鑰控制代碼，請findAllKeys在登入 HSM 伺服器時使用。

必要：是

<destination hsm>

指定您要同步金鑰的伺服器數量。

必要：是

相關主題

- listUsers
- findAllKeys
- 描述叢集 AWS CLI
- server

syncUser

您可以在 cloudhsm_mgmt_util 中使用此syncUser命令，在叢集內或跨複製的叢集之間的 HSM 執行個體之間手動同步加密使用者 (CUs) 或加密主管 (COs)。 AWS CloudHSM 不會自動同步處理使用者。一般而言，您可以在全域模式中管理使用者，這樣，叢集中的所有 HSM 可以一起更新。如果 HSM 意外取消同步 (例如，由於密碼變更)，或者，如果您想要在複製叢集間輪換使用者憑證，您可能需要使用 syncUser。複製的叢集通常是在不同的 AWS 區域中建立，以簡化全域擴展和災難復原程序。

啟動 CMU 並登入 HSM 後，方可執行任何 CMU 命令。請確認您所登入的使用者帳戶類型能夠執行您要使用的命令。

如果您新增或刪除 HSM，請更新 CMU 的組態檔案。否則，您所進行的變更可能無法在叢集中的所有 HSM 上生效。

使用者類型

下列類型的使用者可以執行此命令。

- 加密管理員 (CO)

必要條件

開始之前，您必須知道來源 HSM 上要與目標 HSM 同步使用者的 `user ID`。如要尋找 `user ID`，請使用 listUsers 命令列出叢集中 HSM 上的所有使用者。

您也需了解指派給來源及目標 HSM 的 `server ID`，這會顯示在啟動時由 cloudhsm_mgmt_util 所傳回的追蹤輸出中的具體內容。這些項目的指派順序就是 HSM 顯示在組態檔案中的順序。

如果您要同步複製叢集間的 HSM，請遵循複製叢集間使用 CMU中的說明操作，並使用新的組態檔案初始化 cloudhsm_mgmt_util。

當您準備好執行 syncUser 時，請發出 server 命令，在來源 HSM 上進入伺服器模式。

語法

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
syncUser <user ID> <server ID>
```

範例

執行 server 命令，登入來源 HSM 並進入伺服器模式。在該範例中，我們假設 `server 0` 是來源 HSM。

```
aws-cloudhsm> server 0
```

現在執行 syncUser 命令。在這個範例中，我們假設使用者 6 是要同步的使用者，而 `server 1` 是目標 HSM。

```
server 0> syncUser 6 1
ExtractMaskedObject: 0x0 !
InsertMaskedObject: 0x0 !
syncUser success
```

引數

因為此命令未指明具體參數，所以您須依照語法圖表中指定的順序輸入引數。

```
syncUser <user ID> <server ID>
```

<user ID>

指定要同步的使用者 ID。每一個命令中只能指定一個使用者。若要取得使用者的 ID，請使用 listUsers。

必要：是

<server ID>

指定您要同步使用者的 HSM 伺服器編號。

必要：是

相關主題

- listUsers
- 描述叢集 AWS CLI
- server

金鑰管理公用程式 (KMU)

金鑰管理公用程式 (KMU) 是一組命令列工具，可協助加密使用者 (CU) 管理硬體安全模組 (HSM) 上的金鑰。KMU 包含多個命令，可產生、刪除、匯入和匯出金鑰、取得和設定屬性、尋找金鑰，以及執行密碼編譯操作。

KMU 和 CMU 系統是用戶端 SDK 3 套件的一部分。

如需快速入門，請參閱 key_mgmt_util 入門。如需命令的詳細資訊，請參閱 key_mgmt_util 命令參考。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

如果您使用的是 Linux，若要使用 key_mgmt_util，請連接到您的用戶端執行個體，然後請參閱安裝和設定 AWS CloudHSM 用戶端 (Linux)。如果您使用的是 Windows，請參閱 安裝和設定 AWS CloudHSM 用戶端。

主題

- key_mgmt_util 入門
- 安裝和設定 AWS CloudHSM 用戶端 (Linux)
- 安裝和設定 AWS CloudHSM 用戶端
- key_mgmt_util 命令參考

key_mgmt_util 入門

AWS CloudHSM 包含兩個AWS CloudHSM 用戶端軟體的命令列工具。cloudhsm_mgmt_util 工具包含管理 HSM 使用者的命令。key_mgmt_util 工具包含管理金鑰的命令。若要開始使用 key_mgmt_util 命令列工具，請參閱下列主題。

主題

- 設定 key_mgmt_util
- key_mgmt_util 的基本使用方式

如果您使用命令時遇到錯誤訊息或未預期的結果，請參閱疑難排 AWS CloudHSM 主題以取得協助。如需 key_mgmt_util 命令的詳細資訊，請參閱 key_mgmt_util 命令參考。

設定 key_mgmt_util

使用 key_mgmt_util 之前，請完成以下設定。

啟動用 AWS CloudHSM 戶端

在您使用 key_mgmt_util 之前，您必須啟動用戶端。 AWS CloudHSM 用戶端是與叢集中的 HSM 建立 end-to-end 加密通訊的精靈。key_mgmt_util 工具會使用用戶端連線來與叢集中的 HSM 通訊。如果沒有連線，key_mgmt_util 便無法運作。

啟動 AWS CloudHSM 用戶端

使用下面的命令來啟動 AWS CloudHSM 客戶端。

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

- 用於 Windows 用戶端 1.1.2+：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 用於 Windows 用戶端 1.1.1 和更早版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

啟動 key_mgmt_util

在您啟動用 AWS CloudHSM 戶端之後，請使用下列命令來啟動 key_mgmt_util。

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
```

key_mgmt_util 執行時，提示會切換至 `Command:`。

如果命令失敗 (例如傳回 `Daemon socket connection error` 訊息)，請嘗試更新您的組態檔。

key_mgmt_util 的基本使用方式

請參閱以下主題，以了解 key_mgmt_util 工具的基本使用方式。

主題

登入 HSM

使用 loginHSM 命令登入 HSM。以下命令會以名為 example_user 的加密使用者 (CU) 身分登入：輸出指出叢集中的所有三個 HSM 已成功登入。

```
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
```

下列顯示 loginHSM 命令的語法。

```
Command:  loginHSM -u <USER TYPE> -s <USERNAME> -p <PASSWORD>
```

從 HSM 登出

使用 logoutHSM 命令來登出 HSM。

```
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
```

停止 key_mgmt_util

使用 exit 命令來停止 key_mgmt_util。

```
Command:  exit
```

安裝和設定 AWS CloudHSM 用戶端 (Linux)

若要與 AWS CloudHSM 叢集中的 HSM 互動，您需要適用於 Linux 的 AWS CloudHSM 用戶端軟體。應該將此軟體安裝到您先前建立的 Linux EC2 用戶端執行個體。如果是使用 Windows，您也可以安裝用戶端。如需詳細資訊，請參閱 安裝和設定 AWS CloudHSM 用戶端。

任務

- 安裝用 AWS CloudHSM 戶端和命令列工具
- 編輯用戶端組態

安裝用 AWS CloudHSM 戶端和命令列工具

Connect 線至用戶端執行個體並執行下列命令，以下載並安裝 AWS CloudHSM 用戶端和命令列工具。

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
```

編輯用戶端組態

您必須先編輯用 AWS CloudHSM 戶端組態，才能使用用戶端連線至叢集。

編輯用戶端組態

1. 將您的發行憑證 (您使用此憑證來簽署叢集憑證) 複製到用戶端執行個體上的這個位置：/opt/cloudhsm/etc/customerCA.crt。您的用戶端執行個體上需要有執行個體根使用者許可，才能將憑證複製到此位置。
2. 使用下列 configure 命令來更新用 AWS CloudHSM 戶端和命令列工具的組態檔，並指定叢集中 HSM 的 IP 位址。若要取得 HSM 的 IP 位址，請在AWS CloudHSM 主控台中檢視叢集，或執行命describe-clusters AWS CLI 令。在命令輸出中，HSM 的 IP 地址是 EniIp 欄位的值。如果您有多個 HSM，請選擇任何 HSM 的 IP 地址；任何一個都可以。

   ```
   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. 前往 啟用叢集。

安裝和設定 AWS CloudHSM 用戶端

若要在 Windows 上使用 AWS CloudHSM 叢集中的 HSM，您需要適用於 Windows 的 AWS CloudHSM 用戶端軟體。應將此軟體安裝到您先前建立的 Windows Server 執行個體。

安裝 (或更新) 最新 Windows 用戶端和命令列工具

1. 連接至您的 Windows Server 執行個體。
2. 從下載頁面下載最新版本 (AWSCloudHSMClient-latest.msi)。

3. 轉到您的下載位置並以管理權限運行安裝程序（AWSCloudHSMClient-latest.msi）。
4. 依照安裝程式指示操作，然後在安裝程式完成後選擇關閉。
5. 將您自己簽署的發行憑證 (用於簽署叢集憑證的憑證) 複製到 `C:\ProgramData\Amazon\CloudHSM` 資料夾。
6. 使用下列命令以更新組態檔案。如果您正在更新用戶端，務必在重新組態時停止和啟動用戶端。

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

7. 前往 啟用叢集。

備註：

- 如果您正在更新用戶端，系統將不會覆寫前一次安裝的現有組態檔案。
- 適用於 Windows 的 AWS CloudHSM 用戶端安裝程式會自動註冊密碼編譯 API：下一代 (CNG) 和金鑰儲存區提供者 (KSP)。若要解除安裝用戶端，請再次執行安裝程式並依照解除安裝指示操作。
- 如果您使用的 Linux，也可以安裝 Linux 用戶端。如需更多詳細資訊，請參閱 安裝和設定 AWS CloudHSM 用戶端 (Linux)。

key_mgmt_util 命令參考

key_mgmt_util 命令列工具可協助您在叢集的 HSM 中管理金鑰，包括建立、刪除和尋找金鑰及其屬性。它包含多個命令，這個主題中詳細說明每個命令。

如需快速入門，請參閱 key_mgmt_util 入門。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。如需 cloudhsm_mgmt_util 命令列工具 (其中包含用來管理叢集中 HSM 和使用者的命令) 的詳細資訊，請參閱 CloudHSM 管理公用程式 (CMU)。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

如要列出所有 key_mgmt_util 命令，請輸入：

```
Command: help
```

若要取得特定 key_mgmt_util 命令的說明，請輸入：

```
Command: <command-name> -h
```

若要結束您的 key_mgmt_util 工作階段，請輸入：

```
Command: exit
```

下列各主題說明 key_mgmt_util 中的命令。

> **Note**
>
> key_mgmt_util 和 cloudhsm_mgmt_util 中的某些命令，名稱相同。但是，這些命令通常具有不同的語法，不同的輸出和略微不同的功能。

Command	描述
aesWrapUnwrap	加密和解密檔案中金鑰的內容。
deleteKey	從 HSM 刪除金鑰。
Error2String	取得與 key_mgmt_util 十六進位錯誤碼對應的錯誤。
exit	結束 key_mgmt_util。
exportPrivateKey	將私有金鑰的複本 從 HSM 匯出至磁碟上的檔案。
exportPubKey	從 HSM 匯出公有金鑰的複本至檔案。
exSymKey	將對稱金鑰的純文字複本從 HSM 匯出到檔案。
extractMaskedObject	從做為遮罩物件檔案的 HSM 擷取金鑰。
findKey	依金鑰屬性值搜尋金鑰。
findSingleKey	驗證叢集中所有 HSM 上存在金鑰。
根德薩 KeyPair	在 HSM 中產生數位簽署演算法 (DSA) 金鑰對。
GenECC KeyPair	在 HSM 中產生 Elliptic Curve Cryptography (ECC) 金鑰對。

Command	描述
根纳 KeyPair	在 HSM 中產生 RSA 非對稱金鑰對。
genSymKey	在 HSM 中產生對稱金鑰
getAttribute	取得 AWS CloudHSM 金鑰的屬性值並將值寫入檔案。
getCaviumPriv關鍵	建立私有金鑰的仿造 PEM 格式版本，並將它匯出至檔案。
getCert	擷取 HSM 的分割區憑證，並將它們儲存至檔案。
getKeyInfo	取得可以使用金鑰之使用者的 HSM 使用者 ID。 如果金鑰是受規定人數控制，它會取得規定人數中的使用者數目。
help	顯示在 key_mgmt_util 中可用命令的 help 資訊。
importPrivateKey	將私有金鑰匯入 HSM。
importPubKey	將公有金鑰匯入 HSM。
imSymKey	將對稱金鑰的純文字複本從檔案匯入 HSM。
insertMaskedObject	從磁碟上的檔案將遮罩物件插入由相關叢集包含到物件原始叢集的 HSM。相關叢集是以原始叢集備份產生的任何叢集。
???	判斷指定檔案是否包含真實的私有金鑰或仿造 PEM 金鑰。
listAttributes	列出索 AWS CloudHSM 引鍵的屬性以及代表它們的常數。

Command	描述
listUsers	取得 HSM 中的使用者、其使用者類型和 ID，以及其他屬性。
loginHSM 和 logoutHSM	登入和登出叢集中的 HSM。
setAttribute	將工作階段金鑰轉換為持久性金鑰。
sign	使用所選的私有金鑰產生檔案的簽章。
unWrapKey	將包裝 (加密) 的金鑰從檔案匯入到 HSM。
驗證	驗證指定的金鑰是否用於簽署指定的檔案。
wrapKey	將金鑰的加密複本從 HSM 匯出到檔案。

aesWrapUnwrap

aesWrapUnwrap 命令會加密或解密磁碟上檔案的內容。此命令旨在包裝和取消包裝加密金鑰，但您可以對包含小於 4 KB (4096 個位元組) 資料的任何檔案使用它。

aesWrapUnwrap 使用 AES 金鑰包裝。它會在 HSM 上使用 AES 金鑰做為包裝或取消包裝金鑰。然後，它會將結果寫入到磁碟上的另一個檔案。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
aesWrapUnwrap -h

aesWrapUnwrap -m <wrap-unwrap mode>
              -f <file-to-wrap-unwrap>
              -w <wrapping-key-handle>
              [-i <wrapping-IV>]
              [-out <output-file>]
```

範例

這些範例示範如何使用 aesWrapUnwrap 來加密和解密檔案中的加密金鑰。

Example ：包裝加密金鑰

此命令使用 aesWrapUnwrap 來包裝從 HSM 以純文字匯出的三重 DES 對稱金鑰到 3DES.key 檔案。您可以使用類似命令來包裝在檔案中儲存的任何金鑰。

此命令會使用 -m 參數搭配 1 值來指出包裝模式。它使用 -w 參數來指定 HSM 中的 AES 金鑰 (金鑰控制代碼 6) 做為包裝金鑰。它會將產生的包裝金鑰寫入 3DES.key.wrapped 檔案。

輸出顯示命令已成功，而操作使用了預設的 IV，這是慣用的操作。

```
 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 ：取消包裝加密金鑰

此範例示範如何使用 aesWrapUnwrap 來將檔案中已包裝 (加密) 的金鑰取消包裝 (解密)。在將金鑰匯入至 HSM 之前，您可能想要執行這類的操作。例如，如果您嘗試使用imSymKey命令匯入加密金鑰，則會傳回錯誤，因為加密的金鑰沒有該類型的純文字金鑰所需的格式。

此命令會取消包裝 3DES.key.wrapped 檔案中的金鑰，並將純文字寫入 3DES.key.unwrapped 檔案。此命令會使用 -m 參數搭配 0 值來指出取消包裝模式。它使用 -w 參數來指定 HSM 中的 AES 金鑰 (金鑰控制代碼 6) 做為包裝金鑰。它會將產生的包裝金鑰寫入 3DES.key.unwrapped 檔案。

```
 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
```

參數

-h

顯示命令的說明。

必要：是

-m

指定模式。若要包裝 (加密) 檔案內容，請輸入 1；若要取消包裝 (解密) 檔案內容，請輸入 0。

必要：是

-f

指定要包裝的檔案。輸入包含小於 4 KB (4096 個位元組) 資料的檔案。此操作旨在包裝和取消包裝加密金鑰。

必要：是

-w

指定包裝金鑰。輸入 HSM 上 AES 金鑰的金鑰控制代碼。此為必要參數。若要找出金鑰控制代碼，請使用 findKey 命令。

若要建立包裝金鑰，請使用genSymKey來產生 AES 金鑰 (類型 31)。

必要：是

-i

指定演算法的替代初始值 (IV)。除非您有需要替代方案的特殊條件，否則請使用預設值。

預設：0xA6A6A6A6A6A6A6A6。預設值是在 AES 金鑰包裝演算法規格中所定義。

必要：否

-out

指定包含已包裝或取消包裝金鑰之輸出檔的替代名稱。本機目錄中的預設值是 `wrapped_key` (針對包裝操作) 和 `unwrapped_key` (針對取消包裝操作)。

如果檔案存在，aesWrapUnwrap 會覆寫檔案且不會有任何警告。如果命令失敗，aesWrapUnwrap 會建立沒有內容的輸出檔。

預設值：針對包裝：`wrapped_key`。針對取消包裝：`unwrapped_key`。

必要：否

相關主題

- exSymKey
- imSymKey
- unWrapKey
- wrapKey

deleteKey

key_mgmt_util 中的 deleteKey 命令可從 HSM 刪除金鑰。您一次只能刪除一個金鑰。刪除金鑰對中的一個金鑰，對金鑰對中的另一個金鑰沒有影響。

只有金鑰擁有者可以刪除金鑰。共用金鑰的使用者可以在密碼編譯操作中使用它，但不能刪除它。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
deleteKey -h

deleteKey -k
```

範例

下列範例示範如何使用 deleteKey 來刪除 HSM 中的金鑰。

Example ：刪除金鑰

此命令會刪除金鑰控制代碼為 6 的金鑰。當命令成功時，deleteKey 會從叢集中每個 HSM 傳回成功訊息。

```
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 ：刪除金鑰 (失敗)

若因為沒有金鑰有指定的金鑰控制代碼而導致命令失敗，deleteKey 會傳回無效的物件控制代碼錯誤訊息。

```
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
```

若因為目前的使用者不是金鑰擁有者而導致命令失敗，命令會傳回存取遭拒錯誤。

```
Command:  deleteKey -k 262152

Cfm3DeleteKey returned: 0xc6 : HSM Error: Key Access is denied.
```

參數

-h

顯示命令的命令列說明。

必要：是

-k

指定要刪除之金鑰的金鑰控制代碼。若要尋找 HSM 中金鑰的金鑰控制代碼，請使用 findKey。

必要：是

相關主題

- findKey

Error2String

key_mgmt_util 中的 Error2String 協助程式命令可傳回與 key_mgmt_util 十六進位錯誤碼對應的錯誤。針對命令和指令碼進行疑難排解時，您可以使用此命令。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
Error2String -h

Error2String -r <response-code>
```

範例

這些範例示範如何使用 Error2String，以取得 key_mgmt_util 錯誤碼的錯誤字串。

Example ：取得錯誤描述

此命令取得 0xdb 錯誤代碼的錯誤描述。描述中說明嘗試登入 key_mgmt_util 失敗，因為使用者的使用者類型錯誤。只有加密使用者 (CU) 可以登入 key_mgmt_util。

```
        Command:  Error2String -r 0xdb

        Error Code db maps to HSM Error: Invalid User Type.
```

Example ：尋找錯誤碼

此範例顯示在 key_mgmt_util 錯誤中尋找錯誤碼的位置。錯誤代碼 0xc6 出現在此字串後面：Cfm3*command-name* returned: 。

在此範例中，getKeyInfo指出目前使用者 (使用者 4) 可以在密碼編譯作業中使用金鑰。不過，當使用者嘗試使用 deleteKey 刪除金鑰時，命令會傳回錯誤代碼 0xc6。

```
        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
```

如果您收到 0xc6 錯誤報告，您可以使用如下的 Error2String 命令來查詢錯誤。在此案例中，deleteKey 命令失敗，發生存取遭拒錯誤，因為金鑰與目前使用者共用，但擁有的使用者不同。只有金鑰擁有者有權刪除金鑰。

```
        Command:  Error2String -r 0xa8

        Error Code c6 maps to HSM Error: Key Access is denied
```

參數

-h

顯示命令的說明。

必要：是

-r

指定十六進位錯誤代碼。需要 0x 十六進位指標。

必要：是

exit

key_mgmt_util 中的 exit 命令可退出 key_mgmt_util。成功退出後，您即會返回您的標準命令列。

執行任何 key_mgmt_util 命令前，您必須先啟動 key_mgmt_util。

語法

```
exit
```

參數

此命令沒有任何參數。

相關主題

- 啟動 key_mgmt_util

exportPrivateKey

key_mgmt_util 中的 exportPrivateKey 命令會將 HSM 中的非對稱私有金鑰匯出至檔案。HSM 不允許以純文字形式直接匯出金鑰。該命令使用您指定的 AES 包裝密鑰來包裝私有金鑰，解密包裝的位元組，並以純文字形式將私有金鑰複製到檔案中。

exportPrivateKey 命令不會從 HSM 移除金鑰、變更其金鑰屬性，也不會阻止您在進一步的密碼編譯操作中使用該金鑰。您可以多次匯出相同的金鑰。

您只能匯出私有金鑰具有 OBJ_ATTR_EXTRACTABLE 屬性值 1。您須指定具有 OBJ_ATTR_WRAP 和 OBJ_ATTR_DECRYPT 屬性值 1 的 AES 包裝金鑰。若要尋找金鑰的屬性，請使用 getAttribute 命令。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
exportPrivateKey -h

exportPrivateKey -k <private-key-handle
                 -w <wrapping-key-handle>
                 -out <key-file>
                 [-m <wrapping-mechanism>]
                 [-wk <wrapping-key-file>]
```

範例

此範例顯示如何使用 exportPrivateKey 從 HSM 匯出私有金鑰。

Example ：匯出私有金鑰

此命命以包含控制代碼 16 的包裝金鑰，將包含控制代碼 15 的私有金鑰匯出至稱為 `exportKey.pem` 的 PEM 檔案。當命令成功時，exportPrivateKey 會傳回成功訊息。

```
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
```

參數

此命令會使用下列參數。

-h

顯示命令的命令列說明。

必要：是

-k

指定要匯出之私有金鑰的金鑰控制代碼。

必要：是

-w

指定包裝金鑰的金鑰控制代碼。此為必要參數。若要找出金鑰控制代碼，請使用 findKey 命令。

若要判斷金鑰是否可以做為包裝金鑰使用，請使用 getAttribute 取得 OBJ_ATTR_WRAP 屬性 (262) 的值。若要建立包裝金鑰，請使用 genSymKey 建立 AES 金鑰 (類型 31)。

如果您使用 -wk 參數來指定外部的取消包裝金鑰，則在匯出期間會使用 -w 包裝金鑰來進行包裝，但不會用來取消包裝。

必要：是

-out

指定將寫入匯出之私有金鑰的檔案名稱。

必要：是

-m

指定用來包裝匯出之私有金鑰的包裝機制。唯一的有效值是 4，這代表 NIST_AES_WRAP mechanism.。

預設：4 (NIST_AES_WRAP)

必要：否

-wk

指定用於將匯出之金鑰取消包裝的金鑰。輸入包含純文字 AES 金鑰的檔案路徑和名稱。

當您加入此參數時，exportPrivateKey 會使用在 -w 檔案中的金鑰來包裝要匯出的金鑰，並使用 -wk 參數指定的金鑰來取消包裝此金鑰。

預設：使用在 -w 參數中指定的包裝金鑰來進行包裝和取消包裝。

必要：否

相關主題

- importPrivateKey
- wrapKey
- unWrapKey
- genSymKey

exportPubKey

key_mgmt_util 中的 exportPubKey 命令會將 HSM 中的公有金鑰匯出至檔案。您可以使用此命令匯出在 HSM 上產生的公有金鑰。您也可以使用此命令匯出之前匯入 HSM 的公有金鑰，例如使用 importPubKey 命令匯入的公有金鑰。command.

exportPubKey 操作會複製金鑰材料至您指定的檔案。但不會從 HSM 移除金鑰、變更其金鑰屬性，或避免您將金鑰用於進一步的密碼編譯操作。您可以多次匯出相同的金鑰。

您僅可匯出具有 1 之 OBJ_ATTR_EXTRACTABLE 值的公有金鑰。若要尋找金鑰的屬性，請使用 getAttribute 命令。

執行任何 key_mgmt_util 命令之前，您必須啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
exportPubKey -h

exportPubKey -k <public-key-handle
             -out <key-file>
```

範例

此範例顯示如何使用 exportPubKey 從 HSM 匯出公有金鑰。

Example ：匯出公有金鑰

此命令將包含控制代碼 10 的公有金鑰匯出至稱為 public.pem 的檔案。當命令成功時，exportPubKey 會傳回成功訊息。

```
Command: exportPubKey -k 10 -out public.pem

PEM formatted public key is written to public.pem

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

參數

此命令會使用下列參數。

-h

顯示命令的命令列說明。

必要：是

-k

指定要匯出之公有金鑰的金鑰控制代碼。

必要：是

-out

指定將寫入匯出之公有金鑰的檔案名稱。

必要：是

相關主題

- importPubKey
- 產生金鑰

exSymKey

key_mgmt_util 工具中的 exSymKey 命令會從 HSM 匯出對稱金鑰的純文字複本，並將其儲存在磁碟上的檔案中。若要匯出金鑰的加密 (包裝) 複本，請使用 wrapKey 命令。若要匯入純文字金鑰，就像匯exSymKey出的金鑰一樣，請使用. imSymKey

在匯出程序時，exSymKey 會使用您指定要包裝 (加密) 的 AES 金鑰 (包裝金鑰)，然後將金鑰取消包裝 (解密)以便匯出。不過，匯出操作的結果是磁碟上的純文字 (取消包裝) 金鑰。

只有金鑰的擁有者 (也就是建立金鑰的 CU 使用者) 可以匯出金鑰。共用金鑰的使用者可以在密碼編譯操作中使用它，但不能將它匯出。

exSymKey 操作會將金鑰資料複製到您指定的檔案，但不會從 HSM 中移除該金鑰、變更其金鑰屬性、或阻止您在密碼編譯操作中使用金鑰。您可以多次匯出相同的金鑰。

exSymKey 只會匯出對稱金鑰。若要匯出公有金鑰，請使用 exportPubKey。若要匯出私有金鑰，請使用 exportPrivateKey。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
exSymKey -h
```

```
exSymKey -k <key-to-export>
         -w <wrapping-key>
         -out <key-file>
         [-m 4]
         [-wk <unwrapping-key-file> ]
```

範例

下列範例示範如何從 HSM 使用 exSymKey 來匯出您擁有的對稱金鑰。

Example ：匯出 3DES 對稱金鑰

此命令會將三重 DES (3DES) 對稱金鑰 (金鑰控制代碼 7) 匯出。它使用 HSM 中的現有 AES 金鑰 (金鑰控制代碼 6) 做為包裝金鑰。然後，將純文字的 3DES 金鑰寫入 3DES.key 檔案。

輸出顯示已成功將金鑰 7 (3DES 金鑰) 包裝和取消包裝，然後將其寫入 3DES.key 檔案。

> **Warning**
>
> 雖然輸出寫著已將 "Wrapped Symmetric Key"(包裝的對稱金鑰) 寫入輸出檔，但輸出檔中包含純文字 (取消包裝) 金鑰。

```
        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 ：匯出僅限工作階段的包裝金鑰

此範例示範如何使用只存在於工作階段中的金鑰做為包裝金鑰。因為要匯出的金鑰會被包裝、再立即取消包裝，然後以純文字傳送，所以不需要保留包裝金鑰。

這一系列的命令會從 HSM 匯出金鑰控制代碼為 8 的 AES 金鑰。它會使用特別為此目的建立的 AES 工作階段金鑰。

第一個命令使genSymKey用創建 256 位 AES 密鑰。它會使用 -sess 參數來建立只在目前工作階段中存在的金鑰。

輸出顯示 HSM 建立了金鑰 262168。

```
        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
```

接著，範例會驗證金鑰 8 (要匯出的金鑰) 是可擷取的對稱金鑰。它也會驗證包裝金鑰 (金鑰 262168) 是只存在於工作階段中的 AES 金鑰。您可以使用 findKey 命令，而本範例則是將兩個金鑰的屬性匯出到檔案，然後使用 grep 在檔案中尋找相關的屬性值。

這些命令使用 getAttribute，再加上 -a 的值為 512 (所有)，來取得 8 和 262168 金鑰的所有屬性。如需有關這些金鑰屬性的詳細資訊，請參閱the section called “金鑰屬性參考”。

```
getAttribute -o 8 -a 512 -out attributes/attr_8
getAttribute -o 262168 -a 512 -out attributes/attr_262168
```

這些命令使用 grep 來驗證要匯出金鑰 (金鑰 8) 的屬性，以及僅限工作階段的包裝金鑰 (金鑰 262168)。

```
    // 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
```

最後，我們使用工作階段金鑰 (金鑰 262168) 做為包裝金鑰，來使用 exSymKey 命令匯出金鑰 8。

當工作階段結束時，金鑰 262168 便不再存在。

```
        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 ：使用外部取消包裝金鑰

此範例示範如何使用外部取消包裝金鑰來從 HSM 匯出金鑰。

當您從 HSM 匯出金鑰時，會指定 HSM 上的 AES 金鑰做為包裝金鑰。依預設，此包裝金鑰會用於將要匯出的金鑰進行包裝和取消包裝。不過，您可以使用 -wk 參數來告訴 exSymKey 使用磁碟上檔案中的外部金鑰來取消包裝。當您這麼做時，由 -w 參數指定的金鑰會包裝目標金鑰，而 -wk 參數指定檔案中的金鑰會將金鑰取消包裝。

因為包裝金鑰必須是 AES 金鑰 (其為對稱)，在 HSM 中的包裝金鑰和磁碟上的取消包裝金鑰必須有相同的金鑰資料。為此，您必須在匯出操作前將包裝金鑰匯入 HSM，或從 HSM 匯出包裝金鑰。

此範例會在 HSM 外建立金鑰，並將其匯入 HSM。它使用內部的金鑰複本來包裝要匯出的對稱金鑰，然後複製檔案中的金鑰來將其取消包裝。

第一個命令會使用 OpenSSL 來產生 256 位元的 AES 金鑰。此命令會將金鑰儲存到 aes256-forImport.key 檔案。OpenSSL 命令不會傳回任何輸出，但您可以使用多種命令來確認操作是否成功。此範例中使用 wc (字數) 工具，該工具會確認檔案中是否包含 32 位元組的資料。

```
$  openssl rand -out keys/aes256-forImport.key 32

$ wc keys/aes256-forImport.key
 0  2 32 keys/aes256-forImport.key
```

此命令使用imSymKey命令將 AES 金鑰從aes256-forImport.key檔案匯入 HSM。命令完成後，HSM 中便有該金鑰，金鑰控制代碼為 262167，並位於 aes256-forImport.key 檔案中。

```
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
```

此命令在匯出操作時使用該金鑰。命令使用 exSymKey 來匯出金鑰 21，後者是 192 位元的 AES 金鑰。若要包裝金鑰，它使用金鑰 262167，這是之前匯入 HSM 的複本。為了要將金鑰取消包裝，它使用 aes256-forImport.key 檔案中的相同金鑰資料。命令完成時，會將金鑰 21 匯出到 aes192_h21.key 檔案。

```
        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"
```

參數

-h

顯示命令的說明。

必要：是

-k

指定要匯出之金鑰的金鑰控制代碼。此為必要參數。輸入您擁有之對稱金鑰的金鑰控制代碼。此為必要參數。若要找出金鑰控制代碼，請使用 findKey 命令。

若要確認金鑰是否匯出，請使用 getAttribute 命令取得 OBJ_ATTR_EXTRACTABLE 屬性的值 (以常數 354 表示)。此外，您只能匯出您擁有的金鑰。若要尋找金鑰的擁有者，請使用getKeyInfo指令。

必要：是

-w

指定包裝金鑰的金鑰控制代碼。此為必要參數。若要找出金鑰控制代碼，請使用 findKey 命令。

包裝金鑰 是 HSM 中的金鑰，用於將要匯出的金鑰進行加密 (包裝) 然後解密 (取消包裝)。只有 AES 金鑰可以做為包裝金鑰。

您可以使用任何 AES 金鑰 (任何大小) 做為包裝金鑰。因為包裝金鑰將目標金鑰包裝後又立即取消包裝，您可以使用僅限工作階段的 AES 金鑰做為包裝金鑰。若要判斷金鑰是否可做為包裝金鑰，使用 getAttribute 取得 OBJ_ATTR_WRAP 屬性之值 (以常數 262 表示)。若要建立包裝金鑰，請genSymKey使用建立 AES 金鑰 (類型 31)。

如果您使用 -wk 參數來指定外部的取消包裝金鑰，則在匯出期間會使用 -w 包裝金鑰來進行包裝，但不會用來取消包裝。

> **Note**
>
> Key 4 代表不支援的內部金鑰。建議您使用您所建立並做為包裝金鑰管理的 AES 金鑰。

必要：是

-out

指定輸出檔案的路徑和名稱。命令成功時，這個檔案中會包含匯出的純文字金鑰。如果檔案已存在，命令會覆寫檔案且不會有任何警告。

必要：是

-m

指定包裝機制。唯一的有效值是 4，這代表 NIST_AES_WRAP 機制。

必要：否

預設：4

-wk

使用指定檔案中的 AES 金鑰來將要匯出的金鑰取消包裝。輸入包含純文字 AES 金鑰的檔案路徑和名稱。

當您加入此參數時，exSymKey 會使用 -w 參數在 HSM 指定的金鑰來包裝要匯出的金鑰，並使用 -wk 檔案中的金鑰來取消包裝此金鑰。-w 和 -wk參數值必須解析為相同的純文字金鑰。

必要：否

預設：使用 HSM 上的包裝金鑰來取消包裝。

相關主題

- genSymKey
- imSymKey
- wrapKey

extractMaskedObject

key_mgmt_util 中的 extractMaskedObject 命令會從 HSM 擷取金鑰，並將金鑰做為遮罩物件，儲存至檔案。遮罩物件是「複製的」物件，僅可在使用 insertMaskedObject 命令將這些物件插回原始叢集後才可使用。您僅能將遮罩物件插入之前從中產生的相同叢集，或該叢集的複製品。這包括透過複製跨區域備份和使用該備份建立新叢集而產生之叢集的任何複製版本。

遮罩物件是卸載及同步金鑰的有效方式，包括無法擷取的金鑰 (即：具有 0 的 OBJ_ATTR_EXTRACTABLE 值的金鑰)。如此一來，您就可以在不同區域的相關叢集之間安全地同步金鑰，而不需要更新 AWS CloudHSM 設定檔案。

> **Important**
>
> 插入後，即將遮罩物件解密，並給予不同於原始金鑰之金鑰控制代碼的金鑰控制代碼。遮罩物件包括與原始金鑰關聯的所有中繼資料，包括屬性、所有權和分享資訊，以及仲裁設定。如您需要同步處理應用程式中叢集間的金鑰，請改用 cloudhsm_mgmt_util 中的 syncKey。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並登入 HSM。擁有金鑰的 CU 或任何 CO 都可使用 extractMaskedObject 命令。

語法

```
extractMaskedObject -h

extractMaskedObject -o <object-handle>
                    -out <object-file>
```

範例

此範例顯示如何使用 extractMaskedObject 命令從 HSM 擷取做為遮罩物件的金鑰。

Example ：擷取遮罩物件

此命令從包含控制代碼 524295 的金鑰，自 HSM 擷取遮罩物件，並另存為稱為 maskedObj 的檔案。當命令成功時，extractMaskedObject 會傳回成功訊息。

```
Command: extractMaskedObject -o 524295 -out maskedObj

Object was masked and written to file "maskedObj"

        Cfm3ExtractMaskedObject returned: 0x00 : HSM Return: SUCCESS
```

參數

此命令會使用下列參數。

-h

顯示命令的命令列說明。

必要：是

-o

指定金鑰的控制代碼，以做為遮罩物件擷取。

必要：是

-out

指定將儲存遮罩物件的檔案名稱。

必要：是

相關主題

- insertMaskedObject
- syncKey
- 跨區域複製備份
- 從先前的 Backup 建立 AWS CloudHSM 叢集

findKey

使用 key_mgmt_util 中的 findKey 命令，依金鑰屬性的值來搜尋金鑰。當一個金鑰符合您設定的所有條件時，findKey 會傳回金鑰控制代碼。若不指定任何參數，findKey 會傳回您在 HSM 中可使用之所有金鑰的金鑰控制代碼。若要尋找特定金鑰的屬性值，請使用 getAttribute。

就像所有 key_mgmt_util 命令一樣，findKey 也是針對特定使用者。只會傳回目前的使用者在密碼編譯操作中可以使用的金鑰。這包括目前使用者擁有的金鑰，以及已經與目前使用者共用的金鑰。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
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>]
```

範例

下列範例示範如何使用 findKey 以在 HSM 中尋找和識別金鑰。

Example ：尋找所有金鑰

此命令在 HSM 中找出目前使用者的所有金鑰。輸出包含使用者擁有和共用的金鑰，以及 HSM 中的所有公有金鑰。

若要取得具有特定金鑰控制代碼之金鑰的屬性，請使用 getAttribute。若要判斷目前的使用者是否擁有或共用特定金鑰，請使用getKeyInfo或findAllKeys在 cloudhsm_mgmt_util 中。

```
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 ：依類型、使用者和工作階段尋找金鑰

此命令會尋找目前使用者和使用者 3 可使用的持久性 AES 金鑰。(使用者 3 或許能夠使用目前使用者看不到的其他金鑰。)

```
Command: findKey -t 31 -sess 0 -u 3
```

Example ：依類別和標籤尋找金鑰

此命令針對目前使用者找出具有 2018-sept 標籤的所有公有金鑰。

```
Command: findKey -c 2 -l 2018-sept
```

Example ：依模數尋找 RSA 金鑰

此命令會針對目前的使用者，尋找以 m4.txt 檔案中的模數所建立之 RSA 金鑰 (類型 0)。

```
Command: findKey -t 0 -m m4.txt
```

參數

-h

顯示命令的說明。

必要：是

-t

尋找指定類型的金鑰。輸入代表金鑰類別的常數。例如，若要尋找 3DES 金鑰，請輸入 `-t 21`。

有效值：

- 0：RSA
- 1：DSA
- 3：EC
- 16：GENERIC_SECRET
- 18：RC4
- 21：三重 DES (3DES)
- 31：AES

必要：否

-c

尋找指定類別中的金鑰。輸入代表金鑰類別的常數。例如，若要尋找公有金鑰，請輸入 `-c 2`。

每個金鑰類型的有效值：

- 2：公有。此類別包含公有/私有金鑰對中的公有金鑰。
- 3：私有。此類別包含公有/私有金鑰對中的私有金鑰。
- 4：私密。此類別包含所有對稱金鑰。

必要：否

-l

尋找具有指定標籤的金鑰。輸入確切的標籤。`--l` 值中不可使用萬用字元或規則表達式。

必要：否

-id

尋找具有指定 ID 的金鑰。輸入確切的 ID 字串。`-id` 值中不可使用萬用字元或規則表達式。

必要：否

-sess

依工作階段狀態尋找金鑰。若要尋找只在目前工作階段中有效的金鑰，請輸入 1。若要尋找持久性金鑰，請輸入 0。

必要：否

-u

尋找指定的使用者和目前的使用者共用的金鑰。輸入以逗號分隔的 HSM 使用者 ID 清單，例如 -u 3 或 -u 4,7。若要尋找 HSM 上的使用者 ID，請使用 listUsers。

當您指定一個使用者 ID 時，findKey 會傳回該使用者的金鑰。當您指定多個使用者 ID 時，findKey 會傳回所有指定之使用者可以使用的金鑰。

由於 findKey 只會傳回目前使用者可以使用的金鑰，-u 結果一律與目前使用者的金鑰相同或為其中一部分。要獲取由任何用戶擁有或與任何用戶共享的所有密鑰，加密官員（COS）可以findAllKeys在 cloudhsm_mgmt_util 中使用。

必要：否

-m

尋找透過指定檔案中的 RSA 模數所建立的金鑰。輸入存放模數之檔案的路徑。

-m 指定要匹配包含 RSA 模數的二進位檔案 (選用)。

必要：否

-kcv

尋找具有指定之金鑰檢查值的金鑰。

金鑰檢查值 (KCV) 是 HSM 匯入或產生金鑰時所產生之金鑰的 3 位元組雜湊或總和檢查碼。您也可以在 HSM 之外計算 KCV，例如在匯出金鑰之後。然後，您可以比較 KCV 值以確認金鑰的身分和完整性。若要取得金鑰的 KCV，請使用 getAttribute。

AWS CloudHSM 使用下列標準方法來產生金鑰檢查值：

- 對稱密鑰：使用金鑰加密零塊的結果的前 3 個位元組。
- 非對稱金鑰配對：公有金鑰 SHA-1 雜湊的前 3 個位元組。
- HMAC 金鑰：目前不支援 HMAC 金鑰的 KCV。

必要：否

輸出

findKey 輸出會列出相符金鑰的總數及其金鑰控制代碼。

```
        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
```

相關主題

- findSingleKey
- getKeyInfo
- getAttribute
- findAllKeys在雲中 _ 微米特
- 金鑰屬性參考

findSingleKey

key_mgmt_util 工具中的 findSingleKey 命令可驗證叢集中所有 HSM 上存在金鑰。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
findSingleKey -h

findSingleKey -k <key-handle>
```

範例

Example

此命令可驗證叢集中所有三個 HSM 上存在金鑰 252136。

```
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
```

參數

-h

顯示命令的說明。

必要：是

-k

指定 HSM 中某個金鑰的金鑰控制代碼。此為必要參數。

若要找出金鑰控制代碼，請使用 findKey 命令。

必要：是

相關主題

- findKey
- getKeyInfo
- getAttribute

根德薩 KeyPair

key_mgmt_util 工具中的 genDSAKeyPair 命令可在 HSM 中產生數位簽章演算法 (DSA) 金鑰對。您必須指定模數長度；命令會產生模數值。您還可以指派 ID、將金鑰與其他 HSM 使用者共用、建立不可

擷取的金鑰，以及建立當工作階段結束時就過期的金鑰。命令成功時會傳回 HSM 指派給公有和私有金鑰的金鑰控制代碼。您可以使用金鑰控制代碼來向其他命令識別金鑰。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

> **Tip**
>
> 如要尋找您已建立之金鑰的屬性，例如類型、長度、標籤和 ID，請使用 getAttribute。若要尋找特定使用者的金鑰，請使用getKeyInfo。如要根據金鑰屬性值來尋找金鑰，請使用 findKey。

語法

```
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]
```

範例

下列範例示範如何使用 genDSAKeyPair 來建立 DSA 金鑰對。

Example ：建立 DSA 金鑰對

此命令建立標籤為 DSA 的 DSA 金鑰對。輸出顯示公有金鑰的金鑰控制代碼是 19，而私有金鑰的控制代碼是 21。

```
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 ：建立僅限工作階段的 DSA 金鑰對

此命令會建立只在目前工作階段中有效的 DSA 金鑰對。除了必要的 (非唯一) 標籤，此命令還會指派 DSA_temp_pair 的唯一 ID。您可以建立像這樣的金鑰對，以簽署和驗證僅限工作階段的字符。輸出顯示公有金鑰的金鑰控制代碼是 12，而私有金鑰的控制代碼是 14。

```
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
```

若要確認金鑰對只存在於工作階段中，請使用 -sessfindKey 的 參數，且值為 1 (true)。

```
  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 ：建立共用、不可擷取的 DSA 金鑰對

此命令建立 DSA 金鑰對。私有金鑰共用給其他三個使用者，且無法從 HSM 匯出。公有金鑰可供任何使用者使用，且一律可擷取。

```
        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 ：建立規定人數控制的金鑰對

此命令建立標籤為 DSA-mV2 的 DSA 金鑰對。此命令使用 -u 參數，將私有金鑰共用給使用者 4 和 6。此命令使用 -m_value 參數，針對任何使用私有金鑰的加密操作，要求至少兩次核准的仲裁。此命令還使用 -attest 參數，以驗證用來產生金鑰對的韌體完整性。

輸出顯示命令已產生金鑰控制代碼為 12 的公用金鑰，以及金鑰控制代碼 17 的私有金鑰，且已通過對叢集韌體的證實檢查。

```
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
```

此命令getKeyInfo在私鑰（密鑰句柄17）上使用。輸出確認金鑰由目前的使用者 (使用者 3) 擁有，且共用給使用者 4 和 6 (沒有其他任何人)。輸出也顯示規定人數身分驗證已啟用，且規定人數為 2。

```
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
```

參數

-h

顯示命令的說明。

必要：是

-m

指定模數的長度 (以位元為單位)。唯一有效的值為 2048。

必要：是

-l

指使用者定義的金鑰對標籤。輸入一個字串。相同標籤套用至金鑰對中的兩個金鑰。label 的大小上限為 127 個字元。

您可以使用任何可以幫助您識別金鑰的片語。標籤不具唯一性，因此您可使用標籤將金鑰進行分組和分類。

必要：是

-id

指使用者定義的金鑰對識別符。輸入叢集中唯一的字串。預設為空字串。您指定的 ID 會套用至金鑰對中的兩個金鑰。

預設：無 ID 值。

必要：否

-min_srv

指在 -timeout 參數值過期之前同步金鑰所需的 HSM 數量下限。如果未在規定時間內同步至指定數量的伺服器，金鑰就不會建立。

AWS CloudHSM 自動將每個金鑰同步到叢集中的每個 HSM。要加速流程，請將 min_srv 值設定為少於叢集中之 HSM 的數量，並設定低逾時值。然而，請注意有些請求可能不會產生金鑰。

預設：1

必要：否

-m_value

指須核准使用金鑰對中私有金鑰之密碼編譯操作的使用者數量。輸入 0 到 8 之間的值。

此參數會建立私有金鑰的規定人數驗證要求。預設值 0 會停用金鑰的規定人數驗證功能。啟用規定人數驗證時，指定數目的使用者必須簽署權杖，才能核准使用私用金鑰的密碼編譯操作，以及核准共用或取消共用私有金鑰的操作。

若要尋找金鑰m_value的，請使用getKeyInfo。

只有當命令中的 -u 參數與一定數量的使用者共用金鑰對以符合 m_value 要求時，此參數才有效。

預設：0

必要：否

-nex

使私有金鑰無法擷取。產生的私有金鑰無法從 HSM 匯出。公有金鑰始終可擷取。

預設：金鑰對中的公有金鑰和私有金鑰均可擷取。

必要：否

-sess

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。

當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

若要將工作階段金鑰更改為持久性 (權杖) 金鑰，請使用 setAttribute。

預設：此金鑰是持久性金鑰。

必要：否

-timeout

指命令等待金鑰同步到 min_srv 參數指定數目的 HSM 的時長 (以秒為單位)。

此參數只有在命令中同時使用 `min_srv` 參數時才有效。

預設：無逾時。該命令會無限期等待，並且僅在將金鑰同步到最小數目的伺服器時才返回。

必要：否

-u

與指定使用者共用金鑰對中的私有金鑰。此參數允許其他 HSM 加密使用者 (CU) 在密碼編譯操作中使用此金鑰。任何使用者均可在無需共用的情況下使用公有金鑰。

輸入以逗號分隔的 HSM 使用者 ID 清單，例如 `-u 5,6`。請勿包含現行使用者的 HSM 使用者 ID。若要尋找 HSM 上 CU 的 HSM 使用者 ID，請使用 listUsers。之後，如要共用或取消共用現有金鑰，請使用 cloudhsm_mgmt_util 中的 shareKey。

預設：只有目前使用者能使用匯入的私有金鑰。

必要：否

-attest

執行完整性檢查，以驗證執行叢集的韌體未被篡改。

預設：無認證檢查。

必要：否

相關主題

- 根納 KeyPair
- genSymKey
- GenECC KeyPair

GenECC KeyPair

key_mgmt_util 工具中的 `genECCKeyPair` 命令可在 HSM 中產生橢圓曲線密碼編譯 (ECC) 金鑰對。執行 `genECCKeyPair` 命令時，您必須指定橢圓曲線識別碼和金鑰對的標籤。您也可以將私有金鑰共用給其他 CU 使用者、建立不可擷取的金鑰、由規定人數控制的金鑰及工作階段結束時就過期的金鑰。命令成功時會傳回 HSM 指派給公有和私有 ECC 金鑰的金鑰控制代碼。您可以使用金鑰控制代碼來向其他命令識別金鑰。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

> **Tip**
>
> 如要尋找您已建立之金鑰的屬性，例如類型、長度、標籤和 ID，請使用 getAttribute。若要尋找特定使用者的金鑰，請使用getKeyInfo。如要根據金鑰屬性值來尋找金鑰，請使用 findKey。

語法

```
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]
```

範例

以下範例示範如何使用 genECCKeyPair 在 HSM 中建立 ECC 金鑰對。

Example ：建立和檢查 ECC 金鑰對

此命令使用 NID_secp384r1 橢圓曲線和 ecc14 標籤建立 ECC 金鑰對。輸出顯示私有金鑰的金鑰控制代碼是 262177，而公有金鑰的金鑰控制代碼是 262179。公有和私有金鑰都會套用此標籤。

```
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
```

產生金鑰後，您可以檢查其屬性。使用 getAttribute，將新 ECC 私有金鑰的所有屬性 (以常數 512 表示) 寫入 attr_262177 檔案。

```
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
```

然後使用 cat 命令來檢視 attr_262177 屬性檔案的內容。輸出顯示金鑰是橢圓曲線私有金鑰，可用於簽署，但不可用於加密、解密、包裝、取消包裝或驗證。金鑰為持久性且可匯出。

```
$  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 使用無效的 EEC 曲線

此命令嘗試使用 NID_X9_62_prime192v1 曲線建立 ECC 金鑰對。由於該橢圓曲線不適用於 FIPS 模式的 HSM，命令會失敗。訊息報告叢集內有一部伺服器無法使用，但這通常不表示叢集內的 HSM 有問題。

```
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
```

參數

-h

顯示命令的說明。

必要：是

-i

指定橢圓曲線的識別符。輸入識別符。

有效值：

- 2: NID_X9_62_prime256v1
- 14: NID_secp384r1

- 16: NID_secp256k1

必要：是

-l

指使用者定義的金鑰對標籤。輸入一個字串。相同標籤套用至金鑰對中的兩個金鑰。label 的大小上限為 127 個字元。

您可以使用任何可以幫助您識別金鑰的片語。標籤不具唯一性，因此您可使用標籤將金鑰進行分組和分類。

必要：是

-id

指使用者定義的金鑰對識別符。輸入叢集中唯一的字串。預設為空字串。您指定的 ID 會套用至金鑰對中的兩個金鑰。

預設：無 ID 值。

必要：否

-min_srv

指在 -timeout 參數值過期之前同步金鑰所需的 HSM 數量下限。如果未在規定時間內同步至指定數量的伺服器，金鑰就不會建立。

AWS CloudHSM 自動將每個金鑰同步到叢集中的每個 HSM。要加速流程，請將 min_srv 值設定為少於叢集中之 HSM 的數量，並設定低逾時值。然而，請注意有些請求可能不會產生金鑰。

預設：1

必要：否

-m_value

指須核准使用金鑰對中私有金鑰之密碼編譯操作的使用者數量。輸入 0 到 8 之間的值。

此參數會建立私有金鑰的規定人數驗證要求。預設值 0 會停用金鑰的規定人數驗證功能。啟用規定人數驗證時，指定數目的使用者必須簽署權杖，才能核准使用私用金鑰的密碼編譯操作，以及核准共用或取消共用私有金鑰的操作。

若要尋找金鑰m_value的，請使用getKeyInfo。

只有當命令中的 -u 參數與一定數量的使用者共用金鑰對以符合 m_value 要求時，此參數才有效。

預設：0

必要：否

-nex

使私有金鑰無法擷取。產生的私有金鑰無法從 HSM 匯出。公有金鑰始終可擷取。

預設：金鑰對中的公有金鑰和私有金鑰均可擷取。

必要：否

-sess

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。

當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

若要將工作階段金鑰更改為持久性 (權杖) 金鑰，請使用 setAttribute。

預設：此金鑰是持久性金鑰。

必要：否

-timeout

指命令等待金鑰同步到 `min_srv` 參數指定數目的 HSM 的時長 (以秒為單位)。

此參數只有在命令中同時使用 `min_srv` 參數時才有效。

預設：無逾時。該命令會無限期等待，並且僅在將金鑰同步到最小數目的伺服器時才返回。

必要：否

-u

與指定使用者共用金鑰對中的私有金鑰。此參數允許其他 HSM 加密使用者 (CU) 在密碼編譯操作中使用此金鑰。任何使用者均可在無需共用的情況下使用公有金鑰。

輸入以逗號分隔的 HSM 使用者 ID 清單，例如 `-u 5,6`。請勿包含現行使用者的 HSM 使用者 ID。若要尋找 HSM 上 CU 的 HSM 使用者 ID，請使用 listUsers。之後，如要共用或取消共用現有金鑰，請使用 cloudhsm_mgmt_util 中的 shareKey。

預設：只有目前使用者能使用匯入的私有金鑰。

必要：否

-attest

執行完整性檢查，以驗證執行叢集的韌體未被篡改。

預設：無認證檢查。

必要：否

相關主題

- genSymKey
- GrSA KeyPair
- 根德薩 KeyPair

根納 KeyPair

key_mgmt_util 工具中的 genRSAKeyPair 命令可產生 RSA 非對稱金鑰對。您需要指定金鑰類型、模數長度及公有指數。此命令產生指定長度的模數並建立金鑰對。您可以指派 ID、將金鑰與其他 HSM 使用者共用、建立不可擷取的金鑰及工作階段結束時就過期的金鑰。命令成功時會傳回 HSM 指派給金鑰的金鑰控制代碼。您可以使用金鑰控制代碼來向其他命令識別金鑰。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

> **Tip**
>
> 如要尋找您已建立之金鑰的屬性，例如類型、長度、標籤和 ID，請使用 getAttribute。若要尋找特定使用者的金鑰，請使用getKeyInfo。如要根據金鑰屬性值來尋找金鑰，請使用 findKey。

語法

```
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]
```

範例

下列範例示範如何使用 genRSAKeyPair 在 HSM 中建立非對稱金鑰對。

Example ：建立和檢查 RSA 金鑰對

此命令建立具有 2048 位元模數和指數為 65537 的 RSA 金鑰對。輸出顯示公有金鑰控制代碼是 2100177，私有金鑰控制代碼是 2100426。

```
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
```

下一個命令會使用 getAttribute，以取得剛建立之公有金鑰的屬性。此命令會將輸出寫入 attr_2100177 檔案。接著以 cat 命令來取得屬性檔案的內容。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

產生的十六進制值可確認這是類型為 RSA (OBJ_ATTR_CLASS 0x02) 的公有金鑰 (OBJ_ATTR_KEY_TYPE 0x00)。您可以使用此公有金鑰來加密 (OBJ_ATTR_ENCRYPT 0x01)，但不能解密 (OBJ_ATTR_DECRYPT 0x00)。結果還包括金鑰長度 (512，0x200)、模數、模數長度 (2048，0x800) 及公有指數 (65537，0x10001)。

```
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 ：產生共用 RSA 金鑰對

此命令會產生 RSA 金鑰對，並將私有金鑰向使用者 4 (HSM 上的另一個 CU) 共用。此命令會使用 m_value 參數來要求至少兩次核准，才能將金鑰對的私有金鑰用於密碼編譯操作。當您使用 m_value 參數時，您在命令中還必須使用 -u，且 m_value 不可超過使用者總數 (-u 的值個數 + 擁有者)。

```
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
```

參數

-h

顯示命令的說明。

必要：是

-m

指模數的長度 (以位元為單位)。最小值為 2048。

必要：是

-e

指公有指數。值必須為大於或等於 65537 的奇數。

必要：是

-l

指使用者定義的金鑰對標籤。輸入一個字串。相同標籤套用至金鑰對中的兩個金鑰。label 的大小上限為 127 個字元。

您可以使用任何可以幫助您識別金鑰的片語。標籤不具唯一性，因此您可使用標籤將金鑰進行分組和分類。

必要：是

-id

指使用者定義的金鑰對識別符。輸入叢集中唯一的字串。預設為空字串。您指定的 ID 會套用至金鑰對中的兩個金鑰。

預設：無 ID 值。

必要：否

-min_srv

指在 -timeout 參數值過期之前同步金鑰所需的 HSM 數量下限。如果未在規定時間內同步至指定數量的伺服器，金鑰就不會建立。

AWS CloudHSM 自動將每個金鑰同步到叢集中的每個 HSM。要加速流程，請將 min_srv 值設定為少於叢集中之 HSM 的數量，並設定低逾時值。然而，請注意有些請求可能不會產生金鑰。

預設：1

必要：否

-m_value

指須核准使用金鑰對中私有金鑰之密碼編譯操作的使用者數量。輸入 0 到 8 之間的值。

此參數會建立私有金鑰的規定人數驗證要求。預設值 0 會停用金鑰的規定人數驗證功能。啟用規定人數驗證時，指定數目的使用者必須簽署權杖，才能核准使用私用金鑰的密碼編譯操作，以及核准共用或取消共用私有金鑰的操作。

若要尋找金鑰m_value的，請使用getKeyInfo。

只有當命令中的 -u 參數與一定數量的使用者共用金鑰對以符合 m_value 要求時，此參數才有效。

預設：0

必要：否

-nex

使私有金鑰無法擷取。產生的私有金鑰無法從 HSM 匯出。公有金鑰始終可擷取。

預設：金鑰對中的公有金鑰和私有金鑰均可擷取。

必要：否

-sess

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。

當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

若要將工作階段金鑰更改為持久性 (權杖) 金鑰，請使用 setAttribute。

預設：此金鑰是持久性金鑰。

必要：否

-timeout

指命令等待金鑰同步到 min_srv 參數指定數目的 HSM 的時長 (以秒為單位)。

此參數只有在命令中同時使用 min_srv 參數時才有效。

預設：無逾時。該命令會無限期等待，並且僅在將金鑰同步到最小數目的伺服器時才返回。

必要：否

-u

與指定使用者共用金鑰對中的私有金鑰。此參數允許其他 HSM 加密使用者 (CU) 在密碼編譯操作中使用此金鑰。任何使用者均可在無需共用的情況下使用公有金鑰。

輸入以逗號分隔的 HSM 使用者 ID 清單，例如 -u 5,6。請勿包含現行使用者的 HSM 使用者 ID。若要尋找 HSM 上 CU 的 HSM 使用者 ID，請使用 listUsers。之後，如要共用或取消共用現有金鑰，請使用 cloudhsm_mgmt_util 中的 shareKey。

預設：只有目前使用者能使用匯入的私有金鑰。

必要：否

-attest

執行完整性檢查，以驗證執行叢集的韌體未被篡改。

預設：無認證檢查。

必要：否

相關主題

- genSymKey
- 根德薩 KeyPair
- GenECC KeyPair

genSymKey

key_mgmt_util 工具中的 genSymKey 命令會在您的 HSM 中產生對稱金鑰。您可以指定金鑰類型和大小、指派 ID 和標籤，以及與其他 HSM 使用者共用金鑰。您也可以建立不可擷取的金鑰，以及工作階段結束時就過期的金鑰。命令成功時會傳回 HSM 指派給金鑰的金鑰控制代碼。您可以使用金鑰控制代碼來向其他命令識別金鑰。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
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]
```

範例

下列範例示範如何使用 genSymKey 在 HSM 中建立對稱金鑰。

> **Tip**
>
> 如要使用您在這些範例中建立的金鑰進行 HMAC 操作，您必須在產生金鑰後將 OBJ_ATTR_SIGN 和 OBJ_ATTR_VERIFY 設為 TRUE。如要設定這些值，請使用 CloudHSM 管理公用程式 (CMU) 中的 setAttribute。如需詳細資訊，請參閱 setAttribute。

Example ：產生 AES 金鑰

此命令建立標籤為 aes256 的 256 位元 AES 金鑰。輸出顯示新金鑰的金鑰控制代碼是 6。

```
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 ：建立工作階段金鑰

此命令會建立不可擷取的 192 位元 AES 金鑰，此金鑰只在目前的工作階段中有效。您可以建立像這樣的金鑰來包裝 (然後立即拆封) 正在匯出的金鑰。

```
Command: genSymKey -t 31 -s 24 -l tmpAES -id wrap01 -nex -sess
```

Example ：快速返回

此命令會建立標籤為 IT_test_key 的一般 512 位元組金鑰。此命令不會等待金鑰同步至叢集內的所有 HSM。相反地，一旦在任何一個 HSM 上建立金鑰 (-min_srv 1) 或 1 秒之內 (-timeout 1) 就立即返回，以最快者為準。如果在逾時到期之前金鑰未同步至指定的 HSM 最少數量，金鑰就不會建立。您可以在指令碼中使用像這樣的命令來建立許多金鑰，例如下列範例中的 for 迴圈。

```
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 ：建立規定人數授權的一般金鑰

此命令會建立標籤為 generic-mV2 的 2048 位元一般私密金鑰。此命令使用 -u 參數來與另一個 CU (使用者 6) 共用金鑰。此命令使用 -m_value 參數，針對任何使用金鑰的密碼編譯操作，要求至少兩次核准的仲裁。此命令還使用 -attest 參數來驗證所產生金鑰的韌體完整性。

輸出顯示命令已產生金鑰控制代碼為 9 的金鑰，且已通過對叢集韌體的證實檢查。

```
                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 ：建立和檢查金鑰

此命令會建立標籤為 3DES_shared 和 ID 為 IT-02 的三重 DES 金鑰。金鑰可供目前使用者及使用者 4 和 5 使用。如果 ID 在叢集內不是唯一的，或目前使用者是使用者 4 或 5，命令會失敗。

輸出顯示新金鑰的金鑰控制代碼是 7。

```
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
```

若要驗證新的 3DES 金鑰是否為目前使用者所擁有，並與使用者 4 和 5 共用，請使用 getKeyInfo。此命令會使用指派給新金鑰的控制代碼 (Key Handle: 7)。

輸出確認金鑰為使用者 3 所擁有，且與使用者 4 和 5 共用。

```
Command:  getKeyInfo -k 7

        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

        Owned by user 3

        also, shared to following 2 user(s):

                4, 5
```

若要確認金鑰的其他屬性，請使用 getAttribute。第一個命令使用 getAttribute 來取得金鑰控制代碼 7 (-o 7) 的所有屬性 (-a 512)。然後將屬性寫入 attr_7 檔案。第二個命令使用 cat 以取得 attr_7 檔案的內容。

此命令確認金鑰 7 是 192 位元 (OBJ_ATTR_VALUE_LEN 0x00000018 或 24 位元組) 3DES (OBJ_ATTR_KEY_TYPE 0x15) 對稱金鑰 (OBJ_ATTR_CLASS 0x04)，標籤為 3DES_shared (OBJ_ATTR_LABEL 3DES_shared)，ID 為 IT_02 (OBJ_ATTR_ID IT-02)。金鑰是持久性 (OBJ_ATTR_TOKEN 0x01) 且可擷取 (OBJ_ATTR_EXTRACTABLE 0x01)，還可以用於加密、解密和包裝。

> **Tip**
>
> 如要尋找您已建立之金鑰的屬性，例如類型、長度、標籤和 ID，請使用 getAttribute。若要尋找特定使用者的金鑰，請使用getKeyInfo。如要根據金鑰屬性值來尋找金鑰，請使用 findKey。

如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

```
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**
>
> 如要使用您在這些範例中建立的金鑰進行 HMAC 操作，您必須在產生金鑰後將 OBJ_ATTR_SIGN 和 OBJ_ATTR_VERIFY 設為 TRUE。若要設定這些值，請使用 CMU 中的 setAttribute。如需詳細資訊，請參閱 setAttribute。

參數

-h

顯示命令的說明。

必要：是

-t

指對稱金鑰的類型。輸入代表金鑰類型的常數。例如，若要建立 AES 金鑰，請輸入 `-t 31`。

有效值：

- 16：GENERIC_SECRET。一般私密金鑰是位元組陣列，不符合任何特定標準，例如，AES 金鑰的需求。
- 18：RC4。RC4 金鑰不適用於 FIPS 模式的 HSM
- 21：三重 DES (3DES)。根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於處於非 FIP 模式的叢集，在 2023 之後仍然允許使用該叢集。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。
- 31：AES

必要：是

-s

指定金鑰大小 (位元組)。例如，若要建立 192 位元金鑰，請輸入 24。

每個金鑰類型的有效值：

- AES：16 (128 位元)、24 (192 位元)、32 (256 位元)
- 3DES：24 (192 位元)
- 一般密碼：<3584 (28672 位元)

必要：是

-l

指使用者定義的金鑰對標籤。輸入一個字串。

您可以使用任何可以幫助您識別金鑰的片語。標籤不具唯一性，因此您可使用標籤將金鑰進行分組和分類。

必要：是

-attest

執行完整性檢查，以驗證執行叢集的韌體未被篡改。

預設：無認證檢查。

必要：否

-id

指使用者定義的金鑰識別符。輸入叢集中唯一的字串。預設為空字串。

預設：無 ID 值。

必要：否

-min_srv

指在 `-timeout` 參數值過期之前同步金鑰所需的 HSM 數量下限。如果未在規定時間內同步至指定數量的伺服器，金鑰就不會建立。

AWS CloudHSM 自動將每個金鑰同步到叢集中的每個 HSM。要加速流程，請將 `min_srv` 值設定為少於叢集中之 HSM 的數量，並設定低逾時值。然而，請注意有些請求可能不會產生金鑰。

預設：1

必要：否

-m_value

指須核准使用金鑰之密碼編譯操作的使用者數量。輸入 `0` 到 `8` 之間的值。

此參數會建立私有金鑰的規定人數驗證要求。預設值 `0` 會停用金鑰的規定人數驗證功能。啟用規定人數驗證時，指定數目的使用者必須簽署權杖，才能核准使用金鑰的密碼編譯操作，以及核准共用或取消共用金鑰的操作。

若要尋找金鑰`m_value`的，請使用getKeyInfo。

只有當命令中的 `-u` 參數與足夠的使用者共享金鑰以符合 `m_value` 要求，此參數才有效。

預設：0

必要：否

-nex

使金鑰無法擷取。產生的金鑰無法從 HSM 匯出。

預設：此金鑰可擷取。

必要：否

-sess

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。

當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

若要將工作階段金鑰更改為持久性 (權杖) 金鑰，請使用 setAttribute。

預設：此金鑰是持久性金鑰。

必要：否

-timeout

指命令等待金鑰同步到 `min_srv` 參數指定數目的 HSM 的時長 (以秒為單位)。

此參數只有在命令中同時使用 `min_srv` 參數時才有效。

預設：無逾時。該命令會無限期等待，並且僅在將金鑰同步到最小數目的伺服器時才返回。

必要：否

-u

將金鑰共用給指定的使用者。此參數授與其他 HSM 密碼使用者 (CU) 在密碼編譯操作中使用此金鑰的許可。

輸入以逗號分隔的 HSM 使用者 ID 清單，例如 `-u 5,6`。請勿包含現行使用者的 HSM 使用者 ID。若要尋找 HSM 上 CU 的 HSM 使用者 ID，請使用 listUsers。之後，如要共用或取消共用現有金鑰，請使用 cloudhsm_mgmt_util 中的 shareKey。

預設：只有現行使用者能夠使用此金鑰。

必要：否

相關主題

- exSymKey
- 根納 KeyPair
- 根德薩 KeyPair
- GenECC KeyPair
- setAttribute

getAttribute

key_mgmt_util 中的getAttribute命令會將索引鍵的一個或所有屬性值寫入檔案。 AWS CloudHSM 如果您所指定金鑰類型的屬性不存在 (例如 AES 金鑰的模數)，則 getAttribute 會傳回錯誤。

金鑰屬性是金鑰的屬性。其中包括特性 (例如金鑰類型、類別、標籤和 ID)，以及代表您可對金鑰執行之動作的值 (例如加密、解密、包裝、取消包裝和驗證)。

您只可以對您擁有的金鑰和與您共用的金鑰使用 getAttribute。您可以執行此命令或 cloudhsm_mgmt_util 中的 getAttribute 命令，如此可從叢集中的所有 HSM 取得金鑰的一個屬性值，並將其寫入 stdout 或檔案。

若要取得屬性的清單和代表它們的常數，請使用 listAttributes 命令。若要變更現有金鑰的屬性值，請使用 key_mgmt_util 中的 setAttribute 和 cloudhsm_mgmt_util 中的 setAttribute。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
getAttribute -h

getAttribute -o <key handle>
             -a <attribute constant>
             -out <file>
```

範例

這些範例示範如何使用 getAttribute 來取得 HSM 中金鑰的屬性。

Example ：取得金鑰類型

此範例會取得金鑰的類型，例如 AES、3DES 或一般金鑰或 RSA 或橢圓曲線金鑰對。

第一個命令會執行 listAttributes，它會取得金鑰屬性和代表它們的常數。輸出顯示金鑰類型的常數是 256。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

```
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
```

第二個命令會執行 getAttribute。它會要求金鑰控制代碼 524296 的金鑰類型 (屬性 256)，並將它寫入 attribute.txt 檔案。

```
Command: getAttribute -o 524296 -a 256 -out attribute.txt
Attributes dumped into attribute.txt file
```

最終命令會取得金鑰檔案的內容。輸出顯示金鑰類型為 0x15 或 21，它是三重 DES (3DES) 金鑰。對於類別和類型值的定義，請參閱金鑰屬性參考。

```
$  cat attribute.txt
OBJ_ATTR_KEY_TYPE
0x00000015
```

Example ：取得金鑰的所有屬性

此命令會取得金鑰控制代碼 6 之金鑰的所有屬性，並將它們寫入 attr_6 檔案。它使用 512 的屬性值，其代表所有屬性。

```
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>
```

此命令示範範例屬性檔案的內容與所有屬性值。在值當中，其報告該金鑰是 256 位元 AES 金鑰並具有 ID test_01 和標籤 aes256。該金鑰可擷取且具持久性，亦即，不是僅限工作階段的金鑰。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

```
$  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
```

參數

-h

顯示命令的說明。

必要：是

-o

指定目標金鑰的金鑰控制代碼。每一個命令中只能指定一個金鑰。若要取得金鑰的金鑰控制代碼，請使用 findKey。

同時，您必須擁有該指定的金鑰或與您共用該金鑰。若要尋找金鑰的使用者，請使用getKeyInfo。

必要：是

-a

識別屬性。輸入代表屬性的常數，或輸入 512 來代表所有屬性。例如，若要取得金鑰類型，請輸入 256，這是代表 OBJ_ATTR_KEY_TYPE 屬性的常數。

若要列出屬性及其常數，請使用 listAttributes。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

必要：是

-out

將輸出寫入指定的檔案。輸入檔案路徑。您不能將輸出寫入 stdout。

如果指定的檔案已存在，getAttribute 會覆寫檔案且不會警告。

必要：是

相關主題

- cloudhsm_mgmt_util 中的 getAttribute
- listAttributes
- setAttribute
- findKey
- 金鑰屬性參考

getCaviumPriv關鍵

key_mgmt_util 中的getCaviumPriv金鑰命令會以偽造的 PEM 格式從 HSM 匯出私密金鑰。未包含實際私有金鑰材料，而是參照 HSM 中私有金鑰的仿造 PEM 檔案，之後可以用於從您的 Web 伺服器建立至 AWS CloudHSM的 SSL/TLS 卸載。如需詳細資訊，請參閱 Linux 上的 SSL/TLS 卸載。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
getCaviumPrivKey -h

getCaviumPrivKey -k <private-key-handle
                 -out <fake-PEM-file>
```

範例

此範例顯示如何使用 getCaviumPrivKey，以仿造 PEM 格式匯出私有金鑰。

Example ：匯出偽 PEM 檔案

此命令建立並匯出包含控制代碼 15 的仿造 PEM 版私有金鑰，並將它儲存至稱為 cavKey.pem 的檔案。當命令成功時，exportPrivateKey 會傳回成功訊息。

```
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
```

參數

此命令會使用下列參數。

-h

顯示命令的命令列說明。

必要：是

-k

指定要以仿造 PEM 格式匯出之私有金鑰的金鑰控制代碼。

必要：是

-out

指定將寫入仿造 PEM 金鑰的檔案名稱。

必要：是

相關主題

- importPrivateKey
- Linux 上的 SSL/TLS 卸載

getCert

key_mgmt_util 中的 getCert 命令會擷取 HSM 的分割區憑證，並將它們儲存至檔案。執行此命令時，請指定欲擷取的憑證類型。若要執行此操作，請使用參數一節所述的其中一個對應整數。若要了解該節所述各類憑證的角色，請參閱驗證 HSM 身分。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
getCert -h
```

```
getCert -f <file-name>
        -t <certificate-type>
```

範例

此範例說明如何使用 getCert 擷取叢集的客戶根憑證，並將它儲存為檔案。

Example ：擷取客戶根憑證

此命令會匯出客戶根憑證 (由整數 4 表示)，並將它儲存到名為 userRoot.crt 的檔案。當命令成功時，getCert 會傳回成功訊息。

```
Command: getCert -f userRoot.crt -s 4

Cfm3GetCert() returned 0 :HSM Return: SUCCESS
```

參數

此命令會使用下列參數。

-h

顯示命令的命令列說明。

必要：是

-f

指定要用來儲存擷取憑證的檔案名稱。

必要：是

-s

以整數指定欲擷取的分割區憑證類型。各整數及其對應的憑證類型如下所示：

- 1：製造商根憑證
- 2：製造商硬體憑證
- 4：客戶根憑證
- 8：叢集憑證 (由客戶根憑證進行簽署)
- 16：叢集憑證 (鏈結到製造商根憑證)

必要：是

相關主題

- 驗證 HSM 身分
- getCert (在 cloudhsm_mgmt_util 中)

getKeyInfo

key_mgmt_util 中的 getKeyInfo 命令會傳回使用者的 HSM 使用者 ID，這些使用者可以使用該金鑰，包括擁有者和將金鑰與其共用的加密使用者 (CU)。對金鑰啟用規定人數身分驗證時，getKeyInfo 也會傳回必須核准使用該金鑰的加密操作的使用者數量。您只可以對您擁有的金鑰以及與您共用的金鑰執行 getKeyInfo。

當您對公有金鑰執行 getKeyInfo 時，即使 HSM 的所有使用者都可以使用該公有金鑰，getKeyInfo 只會傳回該金鑰擁有者。若要尋找 HSM 中使用者的 HSM 使用者 ID，請使用 listUsers。若要找出特定使用者的金鑰，請使用 findKey -u。

您擁有您建立的金鑰。您可以在建立金鑰時與其他使用者共用金鑰。之後，若要共用或取消共用現有的金鑰，請使用 cloudhsm_mgmt_util 中的 shareKey。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
getKeyInfo -h

getKeyInfo -k <key-handle>
```

範例

這些範例顯示如何使用 getKeyInfo，以取得金鑰使用者的相關資訊。

Example ：取得對稱金鑰的使用者

此命令會取得可以對金鑰控制代碼 9 使用 AES (對稱) 金鑰的使用者。輸出顯示使用者 3 擁有該金鑰，並且已將它與使用者 4 共用。

```
Command:  getKeyInfo -k 9

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS
```

```
       Owned by user 3

       also, shared to following 1 user(s):

                4
```

Example ：取得非對稱金鑰對的使用者

這些命令使用 getKeyInfo，以取得可以在 RSA (非對稱) 金鑰對中使用金鑰的使用者。公有金鑰具有金鑰控制代碼 21。私有金鑰具有金鑰控制代碼 20。

對私有金鑰 (20) 執行 getKeyInfo 時，它會傳回金鑰擁有者 (3) 和與其共用金鑰的加密使用者 (CU) 4 和 5。

```
Command:  getKeyInfo -k 20

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 2 user(s):

                4
                5
```

對公有金鑰 (21) 執行 getKeyInfo 時，只會傳回金鑰擁有者 (3)。

```
Command:  getKeyInfo -k 21

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3
```

若要確認使用者 4 可以使用該公有金鑰 (和 HSM 上的所有公有金鑰)，請使用 -ufindKey 的 參數。

輸出示範使用者 4 可以同時使用金鑰對中的公開 (21) 與私密 (20) 金鑰。使用者 4 也可以使用它們所建立或與其共用的所有其他公開金鑰和任何私密金鑰。

```
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 ：取得金鑰的規定人數身分驗證值 (m_value)

此範例示範如何取得金鑰的 m_value，也就是在規定人數中的使用者數量，這些人必須核准使用該金鑰的任何密碼編譯操作。

對金鑰啟用了仲裁身分驗證時，必須有仲裁的使用者核准使用該金鑰的任何密碼編譯操作。若要啟用規定人數驗證並設定規定人數數量，建立金鑰時請使用 -m_value 參數。

這個命令使用 genRSA KeyPair 來建立與使用者 4 共用的 RSA key pair。它使用 m_value 參數來啟用金鑰中私有金鑰的規定人數身分驗證，並將規定人數大小設為兩個使用者。必須有足夠多的使用者，才能提供所需的核准。

輸出顯示命令建立了公有金鑰 27 和私有金鑰 28。

```
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
```

此命令使用 getKeyInfo 以取得私有金鑰使用者的相關資訊。輸出顯示使用者 3 擁有該金鑰，並且將它與使用者 4 共用。它還示範兩個使用者的規定人數必須核准使用該金鑰的每個密碼編譯操作。

```
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
```

參數

-h

顯示命令的命令列說明。

必要：是

-k

指定 HSM 中某個金鑰的金鑰控制代碼。輸入您擁有或共用的金鑰的金鑰控制代碼。此為必要參數。

若要找出金鑰控制代碼，請使用 findKey 命令。

必要：是

相關主題

- getKeyInfo在雲中 _ 微米特
- listUsers
- findKey
- findAllKeys在雲中 _ 微米特

help

help key_mgmt_util 中的 help 命令可顯示所有可用 key_mgmt_util 命令的有關資訊。

執行 help 前，您必須先啟動 key_mgmt_util。

語法

```
help
```

範例

此範例顯示 help 命令的輸出。

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
```

參數

此命令沒有任何參數。

相關主題

- loginHSM 和 logoutHSM

importPrivateKey

key_mgmt_util 中的 importPrivateKey 命令會將非對稱私有金鑰從檔案匯入 HSM。HSM 不允許以純文字形式直接匯入金鑰。此命令會使用您指定的 AES 包裝金鑰來加密私有金鑰，並在 HSM 內將該金鑰取消包裝。如果您嘗試將 AWS CloudHSM 金鑰與憑證產生關聯，請參閱本主題。

> **Note**
>
> 您無法使用對稱或私有金鑰匯入受密碼保護的 PEM 金鑰。

您須指定具有 OBJ_ATTR_UNWRAP 和 OBJ_ATTR_ENCRYPT 屬性值 1 的 AES 包裝金鑰。若要尋找金鑰的屬性，請使用 getAttribute 命令。

> **Note**
>
> 此命令不提供將匯入金鑰標記為不可匯出的選項。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
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]
```

範例

此範例顯示如何使用 importPrivateKey 將私有金鑰匯入 HSM。

Example ：匯入私有金鑰

此命令會從名為 rsa2048.key 且包含標籤 rsa2048-imported 的檔案，以及包含控制代碼 524299 的包裝金鑰，匯入私有金鑰。命令成功時，importPrivateKey 會傳回已匯入金鑰的金鑰控制代碼和成功訊息。

```
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
```

參數

此命令會使用下列參數。

-h

顯示命令的命令列說明。

必要：是

-l

指定使用者定義的私有金鑰標籤。

必要：是

-f

指定要匯入之金鑰的檔案名稱。

必要：是

-w

指定包裝金鑰的金鑰控制代碼。此為必要參數。若要找出金鑰控制代碼，請使用 findKey 命令。

若要判斷金鑰是否可以做為包裝金鑰使用，請使用 getAttribute 取得 OBJ_ATTR_WRAP 屬性 (262) 的值。若要建立包裝金鑰，請使用 genSymKey 建立 AES 金鑰 (類型 31)。

如果您使用 -wk 參數來指定外部的取消包裝金鑰，則在匯入期間會使用 -w 包裝金鑰來進行包裝，但不會用來取消包裝。

必要：是

-sess

將已匯入金鑰指定為工作階段金鑰。

預設：匯入的金鑰在叢集中保留為持久性 (符記) 金鑰。

必要：否

-id

指定要匯入之金鑰的 ID。

預設：無 ID 值。

必要：否

-m_value

指定必須核准使用匯入金鑰之密碼編譯操作的使用者數量。輸入從 **0** 到 **8** 的值。

只有當命令中的 -u 參數與足夠的使用者共享金鑰以符合 m_value 要求，此參數才有效。

預設：0

必要：否

-min_srv

指定在 -timeout 參數值過期之前同步匯入金鑰所需的 HSM 數量下限。如果未在規定時間內同步至指定數量的伺服器，金鑰就不會建立。

AWS CloudHSM 自動將每個金鑰同步到叢集中的每個 HSM。要加速流程，請將 min_srv 值設定為少於叢集中之 HSM 的數量，並設定低逾時值。然而，請注意有些請求可能不會產生金鑰。

預設：1

必要：否

-timout

指定當加入 min-serv 參數時要等待金鑰在 HSM 之間同步的秒數。如果未指定秒數，將持續輪詢下去。

預設：無限制

必要：否

-u

指定要分享匯入之私有金鑰的使用者清單。此參數授與其他 HSM 密碼使用者 (CU) 在密碼編譯操作中使用匯入金鑰的許可。

輸入 HSM 使用者 ID 清單並以逗號分隔，例如 -u 5,6。請勿包含現行使用者的 HSM 使用者 ID。若要找出 HSM 上 CU 的 HSM 使用者 ID，請使用 listUsers。

預設：只有現行使用者能夠使用匯入的金鑰。

必要：否

-wk

指定用於將匯入之金鑰包裝的金鑰。輸入包含純文字 AES 金鑰的檔案路徑和名稱。

當您加入此參數時，importPrivateKey 會使用 -wk 檔案中的金鑰來包裝匯入的金鑰。它也會使用 -w 參數指定的金鑰。

預設：使用在 -w 參數中指定的包裝金鑰來進行包裝和取消包裝。

必要：否

-attest

請對韌體回應進行證明檢查，以確保上面有叢集執行的韌體並未遭到盜用。

必要：否

相關主題

- wrapKey
- unWrapKey
- genSymKey

- exportPrivateKey

importPubKey

key_mgmt_util 中的 importPubKey 命令會將 PEM 格式的公有金鑰匯入 HSM。您可以使用此命令匯入在 HSM 之外產生的公有金鑰。您也可以使用指令匯入從 HSM 匯出的金鑰，例如exportPubKey指令匯出的金鑰。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
importPubKey -h

importPubKey -l <label>
             -f <key-file>
             [-sess]
             [-id <key-id>]
             [min_srv <minimum-number-of-servers>]
             [-timeout <number-of-seconds>]
```

範例

此範例顯示如何使用 importPubKey 將公有金鑰匯入 HSM。

Example ：匯入公有金鑰

此命令從名為 public.pem 且包含標籤 importedPublicKey 的檔案匯入公有金鑰。命令成功時，importPubKey 會傳回已匯入金鑰的金鑰控制代碼和成功訊息。

```
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
```

參數

此命令會使用下列參數。

-h

顯示命令的命令列說明。

必要：是

-l

指定使用者定義的公有金鑰標籤。

必要：是

-f

指定要匯入之金鑰的檔案名稱。

必要：是

-sess

將已匯入金鑰指定為工作階段金鑰。

預設：匯入的金鑰在叢集中保留為持久性 (符記) 金鑰。

必要：否

-id

指定要匯入之金鑰的 ID。

預設：無 ID 值。

必要：否

-min_srv

指定在 `-timeout` 參數值過期之前同步匯入金鑰所需的 HSM 數量下限。如果未在規定時間內同步至指定數量的伺服器，金鑰就不會建立。

AWS CloudHSM 自動將每個金鑰同步到叢集中的每個 HSM。要加速流程，請將 `min_srv` 值設定為少於叢集中之 HSM 的數量，並設定低逾時值。然而，請注意有些請求可能不會產生金鑰。

預設：1

必要：否

-timeout

指定當加入 min-serv 參數時要等待金鑰在 HSM 之間同步的秒數。如果未指定秒數，將持續輪詢下去。

預設：無限制

必要：否

相關主題

- exportPubKey
- 產生金鑰

imSymKey

key_mgmt_util 工具中的 imSymKey 命令可將檔案中對稱金鑰的純文字複本匯入 HSM。您可以使用它來匯入 HSM 以外任何方法所產生的金鑰，以及從 HSM 匯出的金鑰，例如exSymKey、命令寫入檔案的金鑰。

在匯入過程中，imSymKey 會使用您選取的 AES 金鑰 (「包裝金鑰」) 來「包裝」(加密) 然後「取消包裝」(解密) 要匯入的金鑰。不過，imSymKey 僅適用於包含純文字金鑰的檔案。若要匯出和匯入加密金鑰，請使用 wrapKey 和unWrapKey指令。

此外，imSymKey 命令只匯出對稱金鑰。若要匯入公有金鑰，請使用 importPubKey。要導入私鑰，請使用importPrivateKey或 wrapKey。

> **Note**
>
> 您無法使用對稱或私有金鑰匯入受密碼保護的 PEM 金鑰。

匯入的金鑰運作方式非常類似於 HSM 中產生的金鑰。不過，OBJ_ATTR_LOCAL 屬性的值為零，表示其不是在本機產生。當您匯入對稱金鑰時，您可以使用以下命令來共用金鑰。匯入金鑰之後，您可以在 shareKeycloudhsm_mgmt_util 中使用 命令來共用金鑰。

```
imSymKey -l aesShared -t 31 -f kms.key -w 3296 -u 5
```

在您匯入金鑰之後，請務必標記或刪除金鑰檔案。此命令不會阻止您將相同的金鑰資料匯入許多次。結果 (具有不同金鑰控制代碼和相同金鑰資料的多個金鑰) 會難以追蹤金鑰資料的使用情形，也就很難防止超過其加密限制。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
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>]
```

範例

以下範例示範如何使用 imSymKey 將對稱金鑰匯入 HSM。

Example ：匯入 AES 對稱金鑰

這個範例使用 imSymKey 將 AES 對稱金鑰匯入 HSM。

第一個命令使用 OpenSSL 產生隨機 256 位元 AES 對稱金鑰。此命令將金鑰儲存在 `aes256.key` 檔案中。

```
$  openssl rand -out aes256-forImport.key 32
```

第二個命令使用 imSymKey，將 `aes256.key` 檔案中的 AES 金鑰匯入 HSM。此命令使用金鑰 20 (HSM 中的 AES 金鑰) 作為包裝金鑰，並指定標籤為 `imported`。與 ID 不同，標籤在叢集內不需要是唯一的。`-t` (類型) 參數的值是 `31`，代表 AES。

輸出顯示檔案中的金鑰已包裝和取消包裝，然後匯入 HSM 中，並於其中指派金鑰控制代碼 262180。

```
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
```

下一個命令使用 getAttribute，以取得新匯入之金鑰的 OBJ_ATTR_LOCAL 屬性 (屬性 355)，並將此屬性寫入 attr_262180 檔案。

```
Command:  getAttribute -o 262180 -a 355 -out attributes/attr_262180
Attributes dumped into attributes/attr_262180_imported file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

當您檢查屬性檔案時，您可以看到 OBJ_ATTR_LOCAL 屬性值為零，這表示在金鑰資料不是在 HSM 中產生。

```
$  cat attributes/attr_262180_local
OBJ_ATTR_LOCAL
0x00000000
```

Example ：在叢集之間移動對稱金鑰

此範例顯示如何在叢集之間使用exSymKey和imSymKey移動純文字 AES 金鑰。您可以使用像這樣的程序來建立 AES 包裝，而此 AES 包裝同時存在於兩個叢集的 HSM 上。一旦共享包裝密鑰到位，您可以使用 wrapKey 和集群之間unWrapKey移動加密密鑰。

執行此操作的 CU 使用者必須有許可，而能夠登入兩個叢集上的 HSM。

第一個命令使exSymKey用將密鑰 14（32 位元 AES 金鑰）從叢集 1 匯出到aes.key檔案中。此命令使用金鑰 6 (叢集 1 的 HSM 上的 AES 金鑰) 作為包裝金鑰。

```
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"
```

接著，使用者在叢集 2 登入 key_mgmt_util，並執行 imSymKey 命令，將 aes.key 檔案中的金鑰匯入叢集 2 的 HSM。此命令使用金鑰 252152 (叢集 2 的 HSM 上的 AES 金鑰) 作為包裝金鑰。

由於包裝鍵exSymKey並imSymKey使用包裝並立即解開目標鍵，因此不同集群上的包裝鍵不需要相同。

輸出顯示金鑰已成功匯入叢集 2，且指派的金鑰控制代碼為 21。

```
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
```

若要證明叢集 1 的金鑰 14 和叢集 2 的金鑰 21 有相同的金鑰資料，請取得每個金鑰的金鑰檢查值 (KCV)。如果 KCV 值相同，即表示金鑰資料相同。

以下命令在叢集 1 使用 getAttribute，將金鑰 14 的 KCV 屬性 (屬性 371) 的值寫入 attr_14_kcv 檔案。然後，使用 cat 命令取得 attr_14_kcv 檔案的內容。

```
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
```

以下類似的命令在叢集 2 使用 getAttribute，將金鑰 21 的 KCV 屬性 (屬性 371) 的值寫入 attr_21_kcv 檔案。然後，使用 cat 命令取得 attr_21_kcv 檔案的內容。

```
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
```

輸出顯示兩個金鑰的 KCV 值相同，這證明金鑰資料相同。

由於相同的金鑰資料同時存在於兩個叢集的 HSM 中，您現在可以在叢集之間共用加密的金鑰，而完全不需要公有純文字金鑰。例如，您可以對包裝金鑰 14 使用 wrapKey 命令，從叢集 1 匯出加密的金鑰，然後對包裝金鑰 21 使用 unWrapKey，將加密的金鑰匯入叢集 2。

Example ：匯入工作階段金鑰

此命令使用 imSymKey 的 -sess 參數，以匯入只在目前工作階段中有效的 192 位元三重 DES 金鑰。

此命令使用 -f 參數來指定包含要匯入之金鑰的檔案、使用 -t 參數來指定金鑰類型，以及使用 -w 參數來指定包裝金鑰。此命令使用 -l 參數來指定金鑰的分類標籤，也使用 -id 參數來建立金鑰的易記 (且唯一的) 識別符。另外還使用 -attest 參數來驗證用於匯入金鑰的韌體。

輸出顯示金鑰已成功包裝和取消包裝、已匯入 HSM 中，且指派的金鑰控制代碼為 37。此外，也已通過證實檢查，表示韌體未被篡改。

```
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
```

接著，您可以使用 getAttribute 或 findKey 命令，以驗證新匯入之金鑰的屬性。以下命令使用 findKey，以驗證金鑰 37 具有命令所指定的類型、標籤和 ID，而且是工作階段金鑰。如輸出的第 5 行所示，findKey 報告唯一符合所有屬性的金鑰是金鑰 37。

```
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
```

參數

-attest

執行完整性檢查，以驗證執行叢集的韌體未被篡改。

預設：無認證檢查。

必要：否

-f

指定檔案，其中包含要匯入的金鑰。

這個檔案必須包含具有指定長度之 AES 或三重 DES 金鑰的純文字副本。RC4 和 DES 金鑰不適用於 FIPS 模式的 HSM。

- AES：16、24 或 32 位元組
- 三重 DES (3DES)：24 位元組

必要：是

-h

顯示命令的說明。

必要：是

-id

指使用者定義的金鑰識別符。輸入叢集中唯一的字串。預設為空字串。

預設：無 ID 值。

必要：否

-l

指使用者定義的金鑰對標籤。輸入一個字串。

您可以使用任何可以幫助您識別金鑰的片語。標籤不具唯一性，因此您可使用標籤將金鑰進行分組和分類。

必要：是

-min_srv

指在 `-timeout` 參數值過期之前同步金鑰所需的 HSM 數量下限。如果未在規定時間內同步至指定數量的伺服器，金鑰就不會建立。

AWS CloudHSM 自動將每個金鑰同步到叢集中的每個 HSM。要加速流程，請將 `min_srv` 值設定為少於叢集中之 HSM 的數量，並設定低逾時值。然而，請注意有些請求可能不會產生金鑰。

預設：1

必要：否

-sess

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。

當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

若要將工作階段金鑰更改為持久性 (權杖) 金鑰，請使用 setAttribute。

預設：此金鑰是持久性金鑰。

必要：否

-timeout

指命令等待金鑰同步到 `min_srv` 參數指定數目的 HSM 的時長 (以秒為單位)。

此參數只有在命令中同時使用 `min_srv` 參數時才有效。

預設：無逾時。該命令會無限期等待，並且僅在將金鑰同步到最小數目的伺服器時才返回。

必要：否

-t

指對稱金鑰的類型。輸入代表金鑰類型的常數。例如，若要建立 AES 金鑰，請輸入 `-t 31`。

有效值：

- 21：三重 DES (3DES)。
- 31：AES

必要：是

-u

將您匯入的金鑰共用給指定的使用者。此參數授與其他 HSM 密碼使用者 (CU) 在密碼編譯操作中使用此金鑰的許可。

輸入一個 ID，或以逗號分隔的 HSM 使用者 ID 清單，例如 `-u 5,6`。請勿包含現行使用者的 HSM 使用者 ID。若要尋找 ID，您可以在 cloudhsm_mgmt_util 命令列工具中使用 listUsers 命令，或在 key_mgmt_util 命令列工具中使用 listUsers 命令。

必要：否

-w

指定包裝金鑰的金鑰控制代碼。此為必要參數。若要找出金鑰控制代碼，請使用 findKey 命令。

包裝金鑰是 HSM 中的金鑰，用於匯入過程中加密 (包裝) 然後解密 (取消包裝) 金鑰。只有 AES 金鑰可以做為包裝金鑰。

您可以使用任何 AES 金鑰 (任何大小) 做為包裝金鑰。因為包裝金鑰將目標金鑰包裝後又立即取消包裝，您可以使用僅限工作階段的 AES 金鑰做為包裝金鑰。若要判斷金鑰是否可作為包

裝金鑰，請使用 getAttribute 以取得 OBJ_ATTR_WRAP 屬性 (262) 的值。若要建立包裝金鑰，請genSymKey使用建立 AES 金鑰 (類型 31)。

如果您使用 -wk 參數來指定外部包裝金鑰，則會使用 -w 包裝金鑰來取消包裝 (而不是包裝) 要匯入的金鑰。

> **Note**
>
> 金鑰 4 是不支援的內部金鑰。建議您使用您所建立並做為包裝金鑰管理的 AES 金鑰。

必要：是

-wk

使用指定檔案中的 AES 金鑰來包裝要匯入的金鑰。輸入包含純文字 AES 金鑰的檔案路徑和名稱。

當您加入此參數時，imSymKey 會使用 -wk 檔案中的金鑰來包裝要匯入的金鑰，並使用由 -w 參數在 HSM 中指定的金鑰來取消包裝此金鑰。-w 和 -wk參數值必須解析為相同的純文字金鑰。

預設：使用 HSM 上的包裝金鑰來取消包裝。

必要：否

相關主題

- genSymKey
- exSymKey
- wrapKey
- unWrapKey
- exportPrivateKey
- exportPubKey

insertMaskedObject

key_mgmt_util 中的 insertMaskedObject 命令會從檔案將遮罩物件插入指定的 HSM。遮罩物件是使用 extractMaskedObject 命令從 HSM 擷取的複製物件。它們僅可在插回原先的叢集後使用。您僅能將遮罩物件插入之前從中產生的相同叢集，或該叢集的複製品。這包括透過複製跨區域備份和使用該備份建立新叢集而產生之原始叢集的任何複製版本。

遮罩物件是卸載及同步金鑰的有效方式，包括無法擷取的金鑰 (即：具有 0 的 OBJ_ATTR_EXTRACTABLE 值的金鑰)。如此一來，您就可以在不同區域的相關叢集之間安全地同步金鑰，而不需要更新 AWS CloudHSM 設定檔案。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
insertMaskedObject -h

insertMaskedObject -f <filename>
                   [-min_srv <minimum-number-of-servers>]
                   [-timeout <number-of-seconds>]
```

範例

此範例顯示如何使用 insertMaskedObject 將遮罩物件檔案插入 HSM。

Example ：插入遮罩物件

此命令會從名為 maskedObj 的檔案將遮罩物件插入 HSM。命令成功時，insertMaskedObject 會傳回從遮罩物件解密之金鑰的金鑰控制代碼，以及成功訊息。

```
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
```

參數

此命令會使用下列參數。

-h

顯示命令的命令列說明。

必要：是

-f

指定要插入之遮罩物件的檔案名稱。

必要：是

-min_srv

指定在 -timeout 參數值過期之前同步所插入遮罩物件所需的伺服器數量下限。如果未在規定時間內同步至指定數量的伺服器，物件就不會插入。

預設：1

必要：否

-timeout

指定當加入 min-serv 參數時要等待金鑰在伺服器之間同步的秒數。如果未指定秒數，將持續輪詢下去。

預設：無限制

必要：否

相關主題

- extractMaskedObject
- syncKey
- 跨區域複製備份
- 從先前的 Backup 建立 AWS CloudHSM 叢集

IsValidKeyHandlefile

key_mgmt_util 中的IsValidKeyHandlefile命令用於確定密鑰文件是否包含真正的私鑰還是假的 RSA PEM 密鑰。仿造 PEM 檔案不含實際的私有金鑰材料，而是參照 HSM 中的私有金鑰。諸如此類的檔案可以用於從您的 Web 伺服器建立至 AWS CloudHSM的 SSL/TLS 卸載。如需詳細資訊，請參閱 Linux 上的 SSL/TLS 卸載。

> **Note**
>
> IsValidKeyHandlefile僅適用於 RSA 密鑰。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
IsValidKeyHandlefile -h

IsValidKeyHandlefile -f <rsa-private-key-file>
```

範例

這些範例顯示如何使用 IsValidKeyHandlefile 判斷指定的金鑰檔案是否包含真實的金鑰材料或仿造的 PEM 金鑰材料。

Example ：驗證真實的私有金鑰

此命令確認稱為 `privateKey.pem` 的檔案包含真實的金鑰材料。

```
Command: IsValidKeyHandlefile -f privateKey.pem

Input key file has real private key
```

Example ：驗證仿造的 PEM 金鑰

此命令確認稱為 `caviumKey.pem` 的檔案包含以金鑰控制代碼 15 製作的仿造 PEM 金鑰材料。

```
Command: IsValidKeyHandlefile -f caviumKey.pem

Input file has invalid key handle: 15
```

參數

此命令會使用下列參數。

`-h`

顯示命令的命令列說明。

必要：是

-f

指定要檢查有效金鑰材料的 RSA 私密金鑰檔案。

必要：是

相關主題

- getCaviumPriv鑰匙
- Linux 上的 SSL/TLS 卸載

listAttributes

key_mgmt_util 中的listAttributes命令會列出索引鍵的屬性以及代表這些索引 AWS CloudHSM 鍵的常數。您可以使用這些常數來識別 getAttribute 和 setAttribute 命令中的屬性。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

此命令沒有任何參數。

```
listAttributes
```

範例

此命令可列出您在 key_mgmt_util 中可取得和變更的金鑰屬性，以及代表這些屬性的常數。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

若要表示 key_mgmt_util 中 getAttribute 命令中所有屬性，請使用 512。

```
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
```

相關主題

- cloudhsm_mgmt_util 中的 listAttributes
- getAttribute
- setAttribute
- 金鑰屬性參考

listUsers

key_mgmt_util 中的 listUsers 命令可取得 HSM 中的使用者以及他們的使用者類型和其他屬性。

在 key_mgmt_util 中，listUsers 會傳回輸出，其代表叢集內的所有 HSM (即使不一致)。若要取得每個 HSM 中之使用者的相關資訊，請使用 cloudhsm_mgmt_util 中的 listUsers 命令。

key_mgmt_util 中的使用者命令是加密使用者 (CUs) 有權執行的唯讀命令。listUsers getKeyInfo剩餘的使用者管理命令是 cloudhsm_mgmt_util 的一部分。由具有使用者管理許可的加密主管 (CO) 執行。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
listUsers

listUsers -h
```

範例

此命令列出叢集內的 HSM 使用者及其屬性。您可以使用該User ID屬性來識別其他命令中的使用者，例如 findKey、getAttribute 和。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
```

輸出包含下列使用者屬性：

- 使用者 ID：在 key_mgmt_util 和 cloudhsm_mgmt_util 命令中識別使用者。
- 使用者類型：決定使用者在 HSM 上可執行的操作。
- 使用者名稱：顯示使用者定義的使用者易記名稱。
- MofnPubKey：指出使用者是否已註冊用於簽署仲裁驗證權杖的 key pair。
- LoginFailureCnt：指出使用者未成功登入的次數。
- 2FA：指出使用者已啟用多重要素驗證。

參數

-h

顯示命令的說明。

必要：是

相關主題

- cloudhsm_mgmt_util 中的 listUsers
- findKey
- getAttribute
- getKeyInfo

loginHSM 和 logoutHSM

key_mgmt_util 中的 loginHSM 和 logoutHSM 命令允許您登入及登出叢集內的 HSM。登入 HSM 後，您即可使用 key_mgmt_util 執行各式各樣的金鑰管理操作，包括公有和私有金鑰產生、同步處理和包裝。

執行任何 key_mgmt_util 命令前，您必須先啟動 key_mgmt_util。為使用 key_mgmt_util 管理金鑰，您必須以加密使用者 (CU) 的身分登入 HSM。

> **Note**
>
> 如果您錯誤登入超過 5 次，系統將鎖定您的帳戶。如果您在 2018 年 2 月前建立叢集，則系統將在您 20 次錯誤登入後鎖定您的賬戶。若要解除鎖定帳戶，加密管理員 (CO) 必須在 cloudhsm_mgmt_util 中使用 changePswd 命令來重設您的密碼。
> 如果您在叢集中有多個 HSM，系統將允許您再嘗試登入幾次，若均登入錯誤，系統才會鎖定您的賬戶。這是因為 CloudHSM 用戶端可平衡各個 HSM 之間的負載。因此，每次登入嘗試可能會在不同的 HSM 上開始。如果您正在測試此功能，建議您在只有一個作用中 HSM 的叢集上執行此操作。

語法

```
loginHSM -h

loginHSM -u <user type>
         { -p | -hpswd } <password>
         -s <username>
```

範例

此範例顯示如何使用 `loginHSM` 和 `logoutHSM` 命令登入和登出叢集內的 HSM。

Example ：登入 HSM

此命令以使用者名稱為 example_user 和密碼為 aws 的加密使用者 (CU) 身分，登入 HSM。輸出顯示您已登入叢集中的所有 HSM。

```
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 ：使用隱藏密碼登入

此命令與以上範例相同，只不過這次您指定系統應該隱藏密碼。

```
Command:  loginHSM -u CU -s example_user -hpswd
```

系統會提示您輸入密碼。您輸入密碼時後，系統會隱藏密碼，並且輸出會顯示您已成功執行此命令，並顯示您已連線至叢集上的所有 HSM。

```
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 ：登出 HSM。

此命令用於登出 HSM。輸出顯示您已登出叢集中的所有 HSM。

```
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
```

參數

-h

顯示此命令的說明。

-u

指定登入使用者的類型。為使用 key_mgmt_util，您必須以 CU 的身分登入。

必要：是

-s

指定登入使用者名稱。

必要：是

{ -p | -hpswd }

指帶 -p 的登入密碼。輸入密碼時，密碼會以純文字顯示。如要隱藏您的密碼，請使用 -hpswd 參數代替 -p，然後按照提示進行操作。

必要：是

相關主題

- exit

setAttribute

key_mgmt_util 中的 setAttribute 命令可將只在目前工作階段中有效的金鑰轉換為持久性金鑰。如果不刪除，持久性金鑰就會一直存在。在作法上，此命令將金鑰的字符屬性值 (OBJ_ATTR_TOKEN) 從 false (0) 變更為 true (1)。您只能變更您自己金鑰的屬性。

您也可以使用 cloudhsm_mgmt_util 中的 setAttribute 命令，以變更標籤、包裝、取消包裝、加密和解密屬性。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
setAttribute -h

setAttribute -o <object handle>
             -a 1
```

範例

此範例示範如何將工作階段金鑰轉換為持久性金鑰。

第一個命令使用的-sess參數genSymKey來建立僅在目前工作階段中有效的 192 位元 AES 金鑰。輸出顯示新工作階段金鑰的金鑰控制代碼是 262154。

```
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
```

此命令使用 findKey 來尋找目前工作階段中的工作階段金鑰。輸出確認金鑰 262154 是工作階段金鑰。

```
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
```

此命令使用 setAttribute，以將金鑰 262154 從工作階段金鑰轉換為持久性金鑰。在作法上，此命令將金鑰的權杖屬性值 (OBJ_ATTR_TOKEN) 從 0 (false) 變更為 1 (true)。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

此命令使用 -o 參數來指定金鑰控制代碼 (262154)，並使用 -a 參數來指定代表字符屬性的常數 (1)。執行此命令時會提示您輸入字符屬性的值。唯一的有效值是 1 (true)，此值代表持久性金鑰。

```
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
```

為了確認金鑰 262154 現在是持久性，此命令使用 findKey 來搜尋工作階段金鑰 (-sess 1) 和持久性金鑰 (-sess 0)。這一次，命令不會尋找任何工作階段金鑰，而是傳回在持久性金鑰清單中的 262154。

```
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
```

參數

-h

顯示命令的說明。

必要：是

-o

指定目標金鑰的金鑰控制代碼。每一個命令中只能指定一個金鑰。若要取得金鑰的金鑰控制代碼，請使用 findKey。

必要：是

-a

指定常數，其代表您要變更的屬性。唯一的有效值是 1，這代表字符屬性 OBJ_ATTR_TOKEN。

如要取得屬性及其整數值，請使用 listAttributes。

必要：是

相關主題

- cloudhsm_mgmt_util 中的 setAttribute
- getAttribute
- listAttributes
- 金鑰屬性參考

sign

key_mgmt_util 中的 sign 命令使用所選的私有金鑰產生檔案的簽章。

為使用 sign，您必須先在 HSM 中擁有私有金鑰。您可以使用 genSymKey、genRSAKeyPair 或 genECCKeyPair 命令產生私有金鑰。您也可以使用 importPrivateKey 命令匯入私有金鑰。如需關於這些金鑰的詳細資訊，請參閱產生金鑰。

sign 命令採用使用者指定的簽署機制 (以整數代表) 來簽署訊息檔案。如需可能的簽署機制清單，請參閱參數。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
sign -h

sign -f <file name>
     -k <private key handle>
     -m <signature mechanism>
     -out <signed file name>
```

範例

此範例顯示如何使用 sign 簽署檔案。

Example ：簽署檔案

此命令以包含控制代碼 266309 的私有金鑰簽署名為 messageFile 的檔案。此命令使用 SHA256_RSA_PKCS (1) 簽署機制，並將產生的簽署檔案另存為 signedFile。

```
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
```

參數

此命令會使用下列參數。

-f

待簽署檔案的名稱。

必要：是

-k

要用於簽署之私有金鑰的控制代碼。

必要：是

-m

代表用於簽署之簽署機制的整數。對應下列整數的可能機制：

簽署機制	對應整數
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
ECDSA_SHA256	17
ECDSA_SHA384	18
ECDSA_SHA512	19

必要：是

-out

將儲存簽署檔案的檔案名稱。

必要：是

相關主題

- 驗證
- importPrivateKey
- 根納 KeyPair
- GenECC KeyPair
- genSymKey
- 產生金鑰

unWrapKey

key_mgmt_util 工具中的 unWrapKey 命令可將檔案中包裝 (加密) 的對稱或私有金鑰匯入到 HSM。此命令旨在匯入由 key_mgmt_util 中 wrapKey 命令包裝的加密金鑰，同時也可對使用其他工具包裝的金鑰取消包裝。不過，在這些情況下，我們建議您使用 PKCS#11 或 JCE 軟體程式庫來取消包裝金鑰。

匯入的金鑰的運作方式就像產生的 AWS CloudHSM金鑰 不過，其 OBJ_ATTR_LOCAL 屬性值為零，表示其不是在本機產生。

匯入金鑰後，請務必標示或刪除金鑰檔案。此命令不會阻止您將相同的金鑰資料匯入許多次。結果 (具有不同金鑰控制代碼和相同金鑰資料的多個金鑰) 會難以追蹤金鑰資料的使用情形，也就很難防止超過其加密限制。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
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>]
```

範例

這些範例示範如何使用 unWrapKey 將檔案的包裝金鑰匯入 HSM。在第一個範例中，我們取消包裝由 wrapKey key_mgmt_util 命令包裝的金鑰，因此有標頭。在第二個範例中，我們取消包裝在 key_mgmt_util 外部包裝的金鑰，因此不會有標頭。

Example ：取消包裝金鑰 (含標頭)

此命令可將 3DES 對稱金鑰的包裝複本匯入 HSM。此金鑰是使用標籤為 6 的 AES 金鑰取消包裝，其加密方式與用來包裝 3DES 金鑰的金鑰相同。此輸出說明檔案中的金鑰已取消包裝並匯入，且匯入金鑰的控制代碼為 29。

```
        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 ：取消包裝金鑰 (無標頭)

此命令可將 3DES 對稱金鑰的包裝複本匯入 HSM。此金鑰是使用標籤為 6 的 AES 金鑰取消包裝，其加密方式與用來包裝 3DES 金鑰的金鑰相同。由於這個 3DES 金鑰並非使用 key_mgmt_util 包裝，因此指定了 noheader 參數，以及其必要的相關參數：金鑰標籤 (unwrapped3DES)、金鑰類別 (4) 和金鑰類型 (21)。此輸出說明檔案中的金鑰已取消包裝並匯入，且匯入金鑰的控制代碼為 8。

```
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
```

參數

-h

顯示命令的說明。

必要：是

-f

包含包裝金鑰的檔案路徑和名稱。

必要：是

-w

指定包裝金鑰。輸入 HSM 上 AES 金鑰或 RSA 金鑰的金鑰控制代碼。此為必要參數。若要找出金鑰控制代碼，請使用 findKey 命令。

要創建一個包裝密鑰，使genSymKey用生成一個 AES 密鑰（類型 31）或 genRSA 生KeyPair成一個 RSA key pair（類型 0）。如果您使用 RSA 金鑰對，請務必使用其中一個金鑰包裝該金鑰，並使用另一個金鑰對其取消包裝。如要驗證金鑰是否可做為包裝金鑰，請使用 getAttribute 取得 OBJ_ATTR_WRAP 的屬性值 (以常數 262 表示)。

必要：是

-sess

建立只在目前工作階段中存在的金鑰。工作階段結束後，金鑰無法復原。

當您僅短暫需要金鑰 (例如，加密後快速解密另一個金鑰的包裝金鑰) 時，請使用此參數。請勿使用工作階段金鑰來加密工作階段結束後可能需要解密的資料。

若要將工作階段金鑰更改為持久性 (權杖) 金鑰，請使用 setAttribute。

預設：此金鑰是持久性金鑰。

必要：否

-min_srv

指在 `-timeout` 參數值過期之前同步金鑰所需的 HSM 數量下限。如果未在規定時間內同步至指定數量的伺服器，金鑰就不會建立。

AWS CloudHSM 自動將每個金鑰同步到叢集中的每個 HSM。要加速流程，請將 `min_srv` 值設定為少於叢集中之 HSM 的數量，並設定低逾時值。然而，請注意有些請求可能不會產生金鑰。

預設：1

必要：否

-timeout

指命令等待金鑰同步到 `min_srv` 參數指定數目的 HSM 的時長 (以秒為單位)。

此參數只有在命令中同時使用 `min_srv` 參數時才有效。

預設：無逾時。該命令會無限期等待，並且僅在將金鑰同步到最小數目的伺服器時才返回。

必要：否

-attest

執行完整性檢查，以驗證執行叢集的韌體未被篡改。

預設：無認證檢查。

必要：否

-nex

使金鑰無法擷取。產生的金鑰無法從 HSM 匯出。

預設：此金鑰可擷取。

必要：否

-m

表示包裝機制的值。CloudHSM 支援以下機制：

Mechanism	Value
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP (如需資料大小上限，請參閱本節後文的備註)	8
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS (如需資料大小上限，請參閱本節後文的備註)。請參閱下列備註 1 查看即將進行的變更。	12

必要：是

> **Note**
>
> 使用RSA_OAEP包裝機制時，您可以換行的最大金鑰大小取決於 RSA 金鑰的模數和指定雜湊的長度，如下所示：最大金鑰大小 = modulusLengthIn 位元組-(2* 位元hashLengthIn組) -2。
> 使用 RSA_PKCS 環繞機制時，您可以換行的最大金鑰大小由 RSA 金鑰的模數決定，如下所示：最大金鑰大小 = (位元組 -11)。modulusLengthIn

-t

雜湊演算法	Value
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224 (對 RSA_AES 和 RSA_OAEP 機制有效)	6

必要：否

-noheader

如果您的金鑰是在 key_mgmt_util 外部包裝，則您必須指定此參數和所有其他相關參數。

必要：否

> **Note**
>
> 指定此參數時，您也必須指定下列 -noheader 參數：

- -l

  指定要新增到取消包裝金鑰的標籤。

  必要：是

- -kc

  指定要取消包裝的金鑰類別。以下是可接受的值：

  3 = 來自公私金鑰對的私有金鑰

  4 = 私密 (對稱) 金鑰

必要：是

- -kt

  指定要取消包裝的金鑰類型。以下是可接受的值：

  0 = RSA

  1 = DSA

  3 = ECC

  16 = GENERIC_SECRET

  21 = DES3

  31 = AES

  必要：是

您也可以選擇指定以下 -noheader 參數：

- -id

  要新增到取消包裝金鑰的 ID。

  必要：否

- -i

  要使用的取消包裝初始化向量 (IV)。

  必要：否

[1] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於處於非 FIP 模式的叢集，在 2023 之後仍然允許使用該叢集。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

相關主題

- wrapKey
- exSymKey
- imSymKey

驗證

key_mgmt_util 中的 verify 命令確認是否以指定金鑰簽署檔案。若要進行這些操作，verify 命令會依照來源檔案比較簽署的檔案，然後根據指定的公有金鑰和簽署機制，分析檔案是否在密碼編譯上相關。檔案可以 AWS CloudHSM 使用sign作業登入。

參數機制以列於參數區段中的整數代表。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
verify -h

verify -f <message-file>
       -s <signature-file>
       -k <public-key-handle>
       -m <signature-mechanism>
```

範例

這些範例顯示如何使用 verify 來檢查特定公有金鑰是否用於簽署指定的檔案。

Example ：驗證文件簽名

此命令會嘗試驗證是否使用 SHA256_RSA_PKCS 簽署機制產生 hardwareCertSigned 簽署檔案，以公有金鑰 262276 簽署名為 hardwarCert.crt 的檔案。由於指定的參數代表真實的簽署關係，因此命令會傳回成功訊息。

```
Command: verify -f hardwareCert.crt -s hardwareCertSigned -k 262276 -m 1

Signature verification successful

Cfm3Verify returned: 0x00 : HSM Return: SUCCESS
```

Example ：證明 False 簽署關係

此命令驗證是否使用 SHA256_RSA_PKCS 簽署機制產生 userCertSigned 簽署檔案，以公有金鑰 262276 簽署名為 hardwareCert.crt 的檔案。由於指定的參數並未組成真實的簽署關係，因此命令會傳回錯誤訊息。

```
Command: verify -f hardwarecert.crt -s usercertsigned -k 262276 -m 1
Cfm3Verify returned: 0x1b

CSP Error: ERR_BAD_PKCS_DATA
```

參數

此命令會使用下列參數。

-f

原始訊息檔案名稱。

必要：是

-s

簽署檔案的名稱。

必要：是

-k

被視為是用來簽署檔案的公有金鑰控制代碼。

必要：是

-m

整數代表用來簽署檔案之提出的簽署機制。對應下列整數的可能機制：

簽署機制	對應整數
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

簽署機制	對應整數
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
ECDSA_SHA256	17
ECDSA_SHA384	18
ECDSA_SHA512	19

必要：是

相關主題

- sign
- getCert
- 產生金鑰

wrapKey

key_mgmt_util 中的 wrapKey 命令會將對稱或私有金鑰的加密複本從 HSM 匯出至檔案。執行 wrapKey 時，您會指定要匯出的金鑰、HSM 上用來加密 (包裝) 您要匯出之金鑰的金鑰，以及輸出檔。

wrapKey 命令會將加密金鑰寫入您指定的檔案，但不會從 HSM 中移除該金鑰，或阻止您在密碼編譯操作中使用該金鑰。您可以多次匯出相同的金鑰。

只有金鑰的擁有者 (也就是建立金鑰的加密使用者 (CU)) 可以匯出金鑰。共用金鑰的使用者可以在密碼編譯操作中使用它，但不能將它匯出。

若要將加密金鑰重新匯入 HSM，請使用unWrapKey。若要從 HSM 匯出純文字金鑰，請視需要使用exSymKey或exportPrivateKey。該aesWrapUnwrap命令無法解密（解除包裝）wrapKey加密的密鑰。

執行任何 key_mgmt_util 命令之前，您必須先啟動 key_mgmt_util 並以加密使用者 (CU) 的身分登入 HSM。

語法

```
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>>]
```

範例

Example

此命令會將 192 位元的三重 DES (3DES) 對稱金鑰 (金鑰控制代碼 7) 匯出。它使用 HSM 中的 256 位元 AES 金鑰 (金鑰控制代碼 14) 來包裝金鑰 7。然後它會將加密的 3DES 金鑰寫入 3DES-encrypted.key 檔案。

輸出顯示金鑰 7 (3DES 金鑰) 已成功包裝並寫入指定的檔案。加密金鑰的長度為 307 個位元組。

```
        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
```

參數

-h

顯示命令的說明。

必要：是

-k

您要匯出之金鑰的金鑰控制代碼。輸入您擁有之對稱或私有金鑰的金鑰控制代碼。若要找出金鑰控制代碼，請使用 findKey 命令。

若要確認金鑰是否匯出，請使用 getAttribute 命令取得 OBJ_ATTR_EXTRACTABLE 屬性的值 (以常數 354 表示)。如需金鑰屬性的解譯說明，請參閱 金鑰屬性參考。

您只能匯出您擁有的金鑰。若要尋找金鑰的擁有者，請使用getKeyInfo指令。

必要：是

-w

指定包裝金鑰。輸入 HSM 上 AES 金鑰或 RSA 金鑰的金鑰控制代碼。此為必要參數。若要找出金鑰控制代碼，請使用 findKey 命令。

要創建一個包裝密鑰，使genSymKey用生成一個 AES 密鑰（類型 31）或 genRSA 生KeyPair成一個 RSA key pair（類型 0）。如果您使用 RSA 金鑰對，請務必使用其中一個金鑰包裝該金鑰，並使用另一個金鑰對其取消包裝。如要驗證金鑰是否可做為包裝金鑰，請使用 getAttribute 取得 OBJ_ATTR_WRAP 的屬性值 (以常數 262 表示)。

必要：是

-out

輸出檔案的路徑和名稱。命令成功時，這個檔案會包含匯出金鑰的加密複本。如果檔案已存在，命令會覆寫檔案且不會有任何警告。

必要：是

-m

表示包裝機制的值。CloudHSM 支援以下機制：

Mechanism	Value
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP (如需資料大小上限，請參閱本節後文的備註)	8
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS (如需資料大小上限，請參閱本節後文的備註)。請參閱下列備註 1 查看即將進行的變更。	12

必要：是

> **Note**
>
> 使用RSA_OAEP包裝機制時，您可以換行的最大金鑰大小取決於 RSA 金鑰的模數和指定雜湊的長度，如下所示：最大金鑰大小 = (位元組 -2* 位modulusLengthIn元組 hashLengthIn -2)。
> 使用 RSA_PKCS 環繞機制時，您可以換行的最大金鑰大小由 RSA 金鑰的模數決定，如下所示：最大金鑰大小 = (位元組 -11)。modulusLengthIn

-t

表示雜湊演算法的值。CloudHSM 支援以下演算法：

雜湊演算法	Value
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224 (對 RSA_AES 和 RSA_OAEP 機制有效)	6

必要：否

-aad

檔案名稱包含 AAD。

> **Note**
>
> 僅對 AES_GCM 和 CLOUDHSM_AES_GCM 機制有效。

必要：否

-noheader

忽略標頭，其指定 CloudHSM 特定的金鑰屬性。僅限您要使用 key_mgmt_util 以外的工具來取消包裝金鑰時，才使用此參數。

必要：否

-i

初始向量 (IV) (十六進位值)。

> **Note**
>
> 只在使用 CLOUDHSM_AES_KEY_WRAP 和 NIST_AES_WRAP 機制的 -noheader 參數傳遞時有效。

必要：否

-iv_file

您要寫入回應中包含之 IV 值的檔案。

> **Note**
>
> 只在使用 AES_GCM 機制的 -noheader 參數傳遞時有效。

必要：否

-tag_size

要與包裝的 Blob 一起儲存的標籤大小。

> **Note**
>
> 只在使用 AES_GCM 和 CLOUDHSM_AES_GCM 機制的 -noheader 參數傳遞時有效。標籤大小下限為 8。

必要：否

[1] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於處於非 FIP 模式的叢集，在 2023 之後仍然允許使用該叢集。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

相關主題

- exSymKey
- imSymKey
- unWrapKey

金鑰屬性參考

key_mgmt_util 命令使用常數來代表 HSM 中的金鑰屬性。這個主題可協助您識別屬性、尋找在命令中代表這些屬性的常數，以及了解屬性的值。

您在建立金鑰時會設定金鑰的屬性。如要變更權杖屬性 (指明金鑰是持久性金鑰還是工作階段金鑰)，請使用 key_mgmt_util 中的 setAttribute 命令。如要變更標籤、包裝、取消包裝、加密和解密屬性，請使用 cloudhsm_mgmt_util 中的 `setAttribute` 命令。

若要取得屬性及其常數的清單，請使用 listAttributes。若要取得金鑰的屬性值，請使用 getAttribute。

下表列出金鑰屬性、其常數及其有效值。

屬性	常數	值
OBJ_ATTR_ALL	512	代表所有屬性。
OBJ_ATTR_ALWAYS_SENSITIVE	357	0：False。 1：True。
OBJ_ATTR_CLASS	0	2：公有/私有金鑰對中的公有金鑰。 3：公有/私有金鑰對中的私有金鑰。 4：私密 (對稱) 金鑰。
OBJ_ATTR_DECRYPT	261	0：False。 1：True。金鑰可用來解密資料。
OBJ_ATTR_DERIVE	268	0：False。 1：True。此函數將衍生金鑰。
OBJ_ATTR_DESTROYABLE	370	0：False。 1：True。
OBJ_ATTR_ENCRYPT	260	0：False。 1：True。金鑰可用來加密資料。
OBJ_ATTR_EXTRACTABLE	354	0：False。

屬性	常數	值
		1：True。可以從 HSM 匯出金鑰。
OBJ_ATTR_ID	258	使用者定義的字串。在叢集內必須是唯一的。預設為空字串。
OBJ_ATTR_KCV	371	金鑰的金鑰檢查值。如需詳細資訊，請參閱其他詳細資訊。
OBJ_ATTR_KEY_TYPE	256	0：RSA。 1：DSA。 3：EC。 16：一般機密。 18：RC4。 21：三重 DES (3DES)。 31：AES。
OBJ_ATTR_LABEL	3	使用者定義的字串。在叢集內不必是唯一的。
OBJ_ATTR_LOCAL	355	0。False。金鑰已匯入到 HSM。 1：True。

屬性	常數	值
OBJ_ATTR_MODULUS	288	用於產生 RSA 金鑰對的模數。如果是 EC 金鑰，這個值則代表以十六進位格式表示 ANSI X9.62 ECPoint 值「Q」的 DER 編碼。 對於其他金鑰類型，這個屬性不存在。
OBJ_ATTR_MODULUS_BITS	289	用於產生 RSA 金鑰對的模數長度。如果是 EC 金鑰，則代表用於產生金鑰的橢圓曲線的 ID。 對於其他金鑰類型，這個屬性不存在。
OBJ_ATTR_NEVER_EXTRACTABLE	356	0：False。 1：True。金鑰無法從 HSM 匯出。
OBJ_ATTR_PUBLIC_EXPONENT	290	用於產生 RSA 金鑰對的公有指數。 對於其他金鑰類型，這個屬性不存在。
OBJ_ATTR_PRIVATE	2	0：False。 1：True。此屬性指出未經授權的使用者可以索引鍵屬性。由於 CloudHSM PKCS # 11 供應商目前不支援公有工作階段，所有金鑰 (包括公有金鑰的公有私有金鑰對) 有此屬性設定為 1。

屬性	常數	值
OBJ_ATTR_SENSITIVE	259	0：False。公有/私有金鑰對中的公有金鑰。 1：True。
OBJ_ATTR_SIGN	264	0：False。 1：True。金鑰可用於簽署 (私有金鑰)。
OBJ_ATTR_TOKEN	1	0：False。工作階段金鑰。 1：True。持久性金鑰。
OBJ_ATTR_TRUSTED	134	0：False。 1：True。
OBJ_ATTR_UNWRAP	263	0：False。 1：True。金鑰可用來解密金鑰。
OBJ_ATTR_UNWRAP_TEMPLATE	1073742354	這些值應使用已套用於以此包裝金鑰取消包裝之任何金鑰的屬性範本。
OBJ_ATTR_VALUE_LEN	353	金鑰長度 (以位元組為單位)。
OBJ_ATTR_VERIFY	266	0：False。 1：True。金鑰可用於驗證 (公有金鑰)。
OBJ_ATTR_WRAP	262	0：False。 1：True。金鑰可用來加密金鑰。

屬性	常數	值
OBJ_ATTR_WRAP_TEMPLATE	1073742353	這些值應使用屬性範本來匹配使用此包裝金鑰所包裝的金鑰。
OBJ_ATTR_WRAP_WITH_TRUSTED	528	0：False。 1：True。

其他詳細資訊

金鑰檢查值 (kcv)

金鑰檢查值 (KCV) 是 HSM 匯入或產生金鑰時所產生之金鑰的 3 位元組雜湊或總和檢查碼。您也可以在 HSM 之外計算 KCV，例如在匯出金鑰之後。然後，您可以比較 KCV 值以確認金鑰的身分和完整性。若要取得金鑰的 KCV，請使用 getAttribute。

AWS CloudHSM 使用下列標準方法來產生金鑰檢查值：

- 對稱密鑰：使用金鑰加密零塊的結果的前 3 個位元組。
- 非對稱金鑰配對：公有金鑰 SHA-1 雜湊的前 3 個位元組。
- HMAC 金鑰：目前不支援 HMAC 金鑰的 KCV。

AWS CloudHSM 用戶端開發套件

使用用戶端 SDK 將密碼編譯作業從平台或語言型應用程式卸載到硬體安全模組 (HSM)。

AWS CloudHSM 提供了兩個主要版本，客戶端 SDK 5 是最新的。與用戶端 SDK 3 (上一個系列) 相比，用戶端 SDK 5 具有多種優勢。如需詳細資訊，請參閱用戶端 SDK 5 的優點。如需關於平台支援的詳細資訊，請參閱 用戶端 SDK 5 支援的平台。

如需關於使用用戶端 SDK 3 的資訊，請參閱 先前的用戶端 SDK (用戶端 SDK 3)。

the section called “PKCS #11 程式庫”

PKCS #11 是在硬體安全模組 (HSMs) 上執行加密作業的標準。 AWS CloudHSM 提供與 PKCS #11 2.40 版相容的 PKCS #11 程式庫的實作方式。

the section called “OpenSSL 動態引擎”

AWS CloudHSM OpenSSL 動態引擎可讓您透過 OpenSSL API 將密碼編譯作業卸載到您的 CloudHSM 叢集。

the section called “JCE 提供者”

AWS CloudHSM JCE 提供者符合 Java 密碼編譯架構 (JCA)。提供者可讓您在 HSM 上執行密碼編譯作業。

the section called “KSP 和 CNG 提供者”

適 AWS CloudHSM 用於視窗的用戶端包括 CNG 和 KSP 提供者。目前，只有用戶端 SDK 3 支援 CNG 和 KSP 提供者。

用戶端 SDK 5 支援的平台

每個版本的 AWS CloudHSM 客戶端 SDK 的基本支持都不同。SDK 中元件的平台支援通常符合基礎支援，但並非總是如此。若要判斷特定元件的平台支援，請先確定您想要的平台出現在 SDK 的基礎區段中，然後在元件區段中檢查是否有任何排除項或任何其他相關資訊。

AWS CloudHSM 僅支援 64 位元作業系統。

平台支援會隨時間變更。舊版 CloudHSM 用戶端 SDK 可能不支援此處列出的所有作業系統。使用版本說明來判斷舊版 CloudHSM 用戶端 SDK 的作業系統支援。如需詳細資訊，請參閱 AWS CloudHSM 用戶端 SDK 的下載。

如需先前用戶端 SDK 支援的平台，請參閱 用戶端 SDK 3 支援的平台

用戶端 SDK 5 不需要用戶端常駐程式。

Linux 支援用戶端 SDK 5

支援平台	X86_64 架構	ARM 架構
Amazon Linux 2	是	是
Amazon Linux 2023	是	是
CentOS 7 (7.8+)	是	否
紅帽企業版 7 (7.8+)	是	否
紅帽企業版 8 (8.3 以上)	是	否
紅帽企業版 9 (9.2+)	是	是
Ubuntu 20.04 LTS	是	否
Ubuntu 22.04 LTS	是	是

注意：軟件開發套件 5.4.2 是最後一個提供 CentOS 8 平台支援的發行版本。如需詳細資訊，請參閱 CentOS 網站。

Windows 支援用戶端 SDK 5

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

無伺服器支援用戶端 SDK 5

- AWS Lambda
- Docker/ECS

用戶端 SDK 的 HSM 相容性 5

hsm1. 中	hsm2 米。中
與客戶端 SDK 版本 5.0.0 及更高版本兼容。	與客戶端 SDK 版本 5.12.0 及更高版本兼容。

元件支援

CloudHSM CLI

CloudHSM CLI 是一種命令列工具，可協助管理員管理叢集中的使用者。如需詳細資訊，請參閱 CloudHSM 命令列介面 (CLI)。

PKCS #11 程式庫

PKCS #11 程式庫是符合 Linux 和 Windows 用戶端 SDK 5 基礎支援的跨平台元件。如需詳細資訊，請參閱 the section called “Linux 支援用戶端 SDK 5” 及 the section called “Windows 支援用戶端 SDK 5”。

OpenSSL 動態引擎

該動態引擎是一個 Linux 唯一的組件，需要 OpenSSL 1.0.2，1.1.1 或 3.x。

JCE 提供者

JCE 提供者是一個 Java 開發套件，相容於所有支援平台上的 OpenJDK 8、OpenJDK 11、開發 JDK 17 和開發 JDK 21。

用戶端 SDK 5 的優點

與用戶端 SDK 3 相比，用戶端 SDK 5 更易於管理，提供卓越的設定性和更高的可靠性。用户端 SDK 5 還為用户端 SDK 3 提供了一些額外的關鍵優勢。

專為無伺服器架構所設計

用戶端 SDK 5 不需要用戶端常駐程式，因此您不再需要管理背景服務。其可以通過幾種重要方式幫助用戶：

- 簡化應用程式的啟動程序。開始使用 CloudHSM 需在執行應用程式之前先設定 SDK。
- 无需持續執行程序，可更輕鬆地與 Lambda 和彈性容器服務 (ECS) 等無伺服器元件的集成。

更好的第三方集成和更輕鬆的可攜性

用户端 SDK 5 嚴格遵循 JCE 規範，並在不同的 JCE 提供者和更好的第三方集成之間提供了更輕鬆的可移植性

改善使用者體驗和可設定性

用戶端 SDK 5 提高了日誌消息的可讀性，並提供了更清晰的異常和錯誤處理機制，让使用者更容易进行自助分類。SDK 5 還提供了各種組態，這些組態列在設定工具頁面中。

更廣泛的平台支援

用户端 SDK 5 為現代作業平台提供了更多支援。其包括對 ARM 技術的支援，以及對 JCE、PKCS #11 和 OpenSSL 的更多支援。如需詳細資訊，請參閱支援的平台。

附加功能和機制

用戶端 SDK 5 包含用戶端 SDK 3 中無法使用的其他功能和機制，而用戶端 SDK 5 未來會繼續新增更多機制。

從用戶端 SDK 3 遷移至用戶端 SDK 5

如需從用戶端 SDK 3 遷移至用戶端 SDK 5 的詳細說明，請參閱每個個別用戶端 SDK 的遷移指示：

- 將您的 PKCS #11 程式庫從用戶端 SDK 3 遷移到用戶端 SDK 5
- 將您的動態引擎從用戶端 SDK 3 遷移到用戶端 SDK 5
- 將您的 JCE 提供者從用戶端 SDK 3 遷移到用戶端 SDK 5
- 從用戶端 SDK 3 CMU 和 KMU 遷移至用戶端開發套件 5 CloudHSM CLI

對於 CloudHSM CLI 不支援的功能或使用案例，請聯絡支援。

> **Note**
>
> 視窗平台現在支援用戶端 SDK 5 PKCS #11 程式庫。它可以處理 CNG 和 KSP 提供商可以並且應該被視為替代品的大多數用例。KSP 目前僅適用於用戶端 SDK 3。

PKCS #11 程式庫

PKCS #11 是在硬體安全模組 (HSMs) 上執行加密作業的標準。 AWS CloudHSM 提供與 PKCS #11 2.40 版相容的 PKCS #11 程式庫的實作方式。

如需關於啟動載入的資訊，請參閱 連接至叢集。如需疑難排解，請參閱PKCS #11 程序庫的已知問題。

如需關於使用用戶端 SDK 3 的資訊，請參閱 先前的用戶端 SDK (用戶端 SDK 3)。

主題

- 為用戶端 SDK 5 安裝 PKCS #11 程式庫
- 對 PKCS #11 程式庫進行驗證
- PKCS #11 程式庫支援的金鑰類型
- PKCS #11 程式庫支援的機制
- PKCS #11 程式庫支援的 API 作業
- PKCS #11 程式庫支援的索引鍵屬性
- PKCS #11 程式庫的程式碼範例
- 將您的 PKCS #11 程式庫從用戶端 SDK 3 遷移到用戶端 SDK 5
- PKCS #11 的進階組態

為用戶端 SDK 5 安裝 PKCS #11 程式庫

本主題提供有關為 git 用戶端 SDK 5 版本系列安裝最新版本的 PKCS #11 程式庫的指示。如需關於用戶端 SDK 或 PKCS #11 程式庫的詳細資訊，請參閱使用用戶端 SDK 和 PKCS #11 程式庫。

安裝

使用用戶端 SDK 5 時，您無需安裝或執行用戶端常駐程式。

若要使用用戶端 SDK 5 執行單一 HSM 叢集，您必須先將 `disable_key_availability_check` 設定為 `True` 來管理用戶端金鑰持久性。如需詳細資訊，請參閱金鑰同步處理和用戶端 SDK 5 設定工具。

如需關於用戶端 SDK 5 中 PKCS #11 程式庫的詳細資訊，請參閱 PKCS #11 程式庫。

> **Note**
>
> 若要使用用戶端 SDK 5 執行單一 HSM 叢集，您必須先將 `disable_key_availability_check` 設定為 `True` 來管理用戶端金鑰持久性。如需詳細資訊，請參閱金鑰同步處理和用戶端 SDK 5 設定工具。

若要安裝和設定 PKCS #11 程式庫

1. 使用以下命令來下載和安裝 PKCS #11 程式庫。

   Amazon Linux 2

   安裝適用於 Amazon Linux 2 (X86_64 架構) 的 PKCS #11 程式庫：

   ```
   $ 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
   ```

   安裝適用於 Amazon Linux 2 (ARM64 架構) 的 PKCS #11 程式庫：

   ```
   $ 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

   在 X86_64 架構上安裝適用於 Amazon 2023 的 PKCS #11 程式庫：

   ```
   $ 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
   ```

   在 ARM64 架構上安裝適用於 Amazon 2023 的 PKCS #11 程式庫：

```
$ 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+)

安裝適用於 CentOS 7.8+ (X86_64 架構) 的 PKCS #11 程式庫：

```
$ 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+)

在 X86_64 架構上安裝適用於 RHEL 7 的 PKCS #11 程式庫：

```
$ 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+)

在 X86_64 架構上安裝適用於 RHEL 8 的 PKCS #11 程式庫：

```
$ 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+)

在 X86_64 架構上安裝適用於 RHEL 9 的 PKCS #11 程式庫：

```
$ 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
```

在 ARM64 架構上安裝適用於 RHEL 9 的 PKCS #11 程式庫：

```
$ 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

安裝適用於 Ubuntu 20.04 LTS (X86_64 架構) 的 PKCS #11 程式庫：

```
$ 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

安裝適用於 Ubuntu 22.04 LTS (X86_64 架構) 的 PKCS #11 程式庫：

```
$ 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
```

在 ARM64 架構上安裝用於 Ubuntu 22.04 LTS 的 PKCS #11 程式庫：

```
$ 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

安裝適用於 Windows Server 2016 (X86_64 架構) 的 PKCS #11 程式庫：

1. 下載適用於用戶端 SDK 5 的 PKCS #11 程式庫。
2. 以視窗系統管理權限執行 PKCS #11 程式庫安裝程式 (AWSCloudHSMPKCS11-latest.msi)。

Windows Server 2019

安裝適用於 Windows Server 2019 (X86_64 架構) 的 PKCS #11 程式庫：

1. 下載適用於用戶端 SDK 5 的 PKCS #11 程式庫。
2. 以視窗系統管理權限執行 PKCS #11 程式庫安裝程式 (AWSCloudHSMPKCS11-latest.msi)。

2. 使用設定工具指定發行憑證的位置。如需說明，請參閱指定憑證的位置。。
3. 若要連線到您的叢集，請參閱 引導用戶端 SDK。
4. 您可於下列位置找到 PKCS #11 程式庫檔案：

   - Linux 二進位檔案、組態指令碼和日誌文檔：

   ```
   /opt/cloudhsm
   ```

   Windows 二進位檔案：

   ```
   C:\ProgramFiles\Amazon\CloudHSM
   ```

   Windows組態指令碼和日誌檔案：

   ```
   C:\ProgramData\Amazon\CloudHSM
   ```

對 PKCS #11 程式庫進行驗證

當使用 PKCS #11 程式庫時，您的應用程式會以特定加密使用者 (CU) 在 HSM 中執行。您的應用程式只能檢視和管理 CU 所擁有和共用的金鑰。您可以使用 HSM 中現有的 CU，或為應用程式建立新的 CU。如需關於管理 CU 的資訊，請參閱使用 CloudHSM CLI 管理 HSM 使用者和使用 CloudHSM 管理公用程式 (CMU) 管理 HSM 使用者。

若要將 CU 指定給 PKCS #11 程式庫，請使用 PKCS #11 C_Login 函數的 pin 參數。對於 AWS CloudHSM，pin 參數具有以下格式：

```
<CU_user_name>:<password>
```

例如，以下命令會將 PKCS #11 程式庫 pin 設定給使用者名稱為 `CryptoUser` 和密碼為 `CUPassword123!` 的 CU。

```
CryptoUser:CUPassword123!
```

PKCS #11 程式庫支援的金鑰類型

PKCS #11 程式庫支援下列金鑰類型。

金鑰類型	描述
AES	產生 128 位元、192 位元和 256 位元的 AES 金鑰。
Triple DES (3DES, DESede)	產生 192 位元的三重 DES 金鑰。請參閱下列備註 1 查看即將進行的變更。
EC	使用 secp224r1 (P-224)、secp256r1 (P-256)、secp256k1 (Blockchain)、secp384r1 (P-384) 和 secp521r1 (P-521)曲線產生金鑰。
GENERIC_SECRET	產生 1 到 800 位元組的一般機密。
RSA	產生 2048 位元至 4096 位元的 RSA 金鑰，以 256 位元為單位遞增。

[1] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於處於非 FIP 模式的叢集，在 2023 之後仍然允許使用該叢集。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

PKCS #11 程式庫支援的機制

PKCS #11 程式庫與 PKCS #11 規格的 2.40 版相容。若要使用 PKCS #11 叫用加密功能，請利用指定機制呼叫函數。以下章節總結說明 AWS CloudHSM支援的函數和機制組合。

PKCS #11 程式庫支援下列演算法：

- 加密和解密：AES-CBC、AES-CTR、AES-ECB、AES-GCM、DES3-CBC、DES3-ECB、RSA-OAEP 和 RSA-PKCS
- 簽署和驗證：RSA、HMAC 和 ECDSA；無論是否使用雜湊皆不影響
- 雜湊/摘要：SHA1、SHA224、SHA256、SHA384 和 SHA512
- 金鑰包裝 ：AES 金鑰包裝 [1]、AES-GCM、RSA-AES 和 RSA-OAEP

主題

產生金鑰與金鑰對函數

PKCS #11 程式庫的 AWS CloudHSM 軟體程式庫可讓您使用下列機制來產生金鑰與金鑰配對功能。

- CKM_RSA_PKCS_KEY_PAIR_GEN
- CKM_RSA_X9_31_KEY_PAIR_GEN：此機制的功能與 CKM_RSA_PKCS_KEY_PAIR_GEN 機制完全相同，但在產生 p 和 q 時提供更強大的保證。
- CKM_EC_KEY_PAIR_GEN

- CKM_GENERIC_SECRET_KEY_GEN
- CKM_AES_KEY_GEN
- CKM_DES3_KEY_GEN：註腳 5 所列的即將進行的變更。

簽署和驗證函數

PKCS #11 程式庫的 AWS CloudHSM 軟體程式庫可讓您針對簽署和驗證功能使用下列機制。使用用戶端 SDK 5，資料會在軟體中進行本機雜湊處理。這意味著 SDK 可以對任何大小的資料進行雜湊處理。

使用用戶端 SDK 5 RSA 和 ECDSA 時，雜湊在本機完成，因此沒有資料限制。使用 HMAC 時，會有資料限制。如需詳細咨詢，請參閱註腳 2。

RSA

- CKM_RSA_X_509
- CKM_RSA_PKCS：僅限單一部分操作。
- CKM_RSA_PKCS_PSS：僅限單一部分操作。
- 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：僅限單一部分操作。
- 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

簽名恢復和驗證復原函數

用戶端 SDK 5 不支援簽名恢復和驗證復原函數。

Digest 函數

PKCS #11 程式庫的 AWS CloudHSM 軟體程式庫可讓您針對摘要函數使用下列機制。使用用戶端 SDK 5，資料會在軟體中進行本機雜湊處理。這意味著 SDK 可以對任何大小的資料進行雜湊處理。

- CKM_SHA_1
- CKM_SHA224
- CKM_SHA256
- CKM_SHA384
- CKM_SHA512

加密和解密函數

PKCS #11 程式庫的 AWS CloudHSM 軟體程式庫可讓您針對加密和解密功能使用下列機制。

- CKM_RSA_X_509
- CKM_RSA_PKCS：僅限單一部分操作。註腳 5 所列的即將進行的變更。
- CKM_RSA_PKCS_OAEP：僅限單一部分操作。
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_DES3_CBC：註腳 5 所列的即將進行的變更。
- CKM_DES3_ECB：註腳 5 所列的即將進行的變更。
- CKM_DES3_CBC_PAD：註腳 5 所列的即將進行的變更。
- CKM_AES_GCM [1, 2]
- CKM_CLOUDHSM_AES_GCM[3]

衍生金鑰函數

PKCS #11 程式庫的 AWS CloudHSM 軟體程式庫可讓您針對衍生函數使用下列機制。

- CKM_SP800_108_COUNTER_KDF

包裝和解包函數

PKCS #11 程式庫的 AWS CloudHSM 軟體程式庫可讓您使用下列機制來進行換行和解除換行功能。

如需有關 AES 金鑰包裝的其他資訊，請參閱AES 金鑰包裝。

- CKM_RSA_PKCS：僅限單一部分操作。註腳 5 所列的即將進行的變更。
- 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]

每個機制的最大資料大小

下表列出每個機制設定的資料大小上限：

資料集大小上限

Mechanism	資料大小上限 (位元組)
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

機制註釋

- [1] 當執行 AES-GCM 加密時，HSM 不會接受來自應用程式的初始化向量 (IV) 資料。請務必使用其產生的 IV。系統會將 HSM 所提供的 12 位元組 IV 寫入記憶體參考，該記憶體參考是由您提供的 CK_GCM_PARAMS 參數結構 pIV 元素所指向。為了確保使用者不會混淆，在初始化 AES-GCM 加密時，1.1.1 版和更新版本中的 PKCS # 11 開發套件會強制將該 pIV 指向歸零的緩衝區。
- [2] 當使用下列任一機制操作資料時，若資料緩衝區超過資料大小上限，該操作就會導致錯誤。對於這些機制，所有資料處理都必須在 HSM 內進行。如需有關每個機構的最大資料大小集合的資訊，請參閱每個機制的最大資料大小。
- [3] 廠商定義的機制。為了能使用 CloudHSM 廠商定義的機制，編譯期間 PKCS #11 應用程式必須加入 /opt/cloudhsm/include/pkcs11t.h。

CKM_CLOUDHSM_AES_GCM：這個專屬機制是標準 CKM_AES_GCM 程式設計更安全的替代方案。這將 HSM 生成的 IV 加入至加密文字的開頭，而不是將它寫回加密初始化期間提供的 CK_GCM_PARAMS 結構中。您可以搭配 C_Encrypt、C_WrapKey、C_Decrypt 和 C_UnwrapKey 函數搭配使用此機制。使用此機制時，CK_GCM_PARAMS 結構中的 pIV 變數必須設定為 NULL。與 C_Decrypt 和 C_UnwrapKey 搭配使用此機制時，IV 應至於取消包裝的加密文字之前。

CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD：AES 金鑰包裝與 PKCS #5 填補。

CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD：AES 金鑰包裝，零填補。

- [4] 支援下列 CK_MECHANISM_TYPE 和 CK_RSA_PKCS_MGF_TYPE，作為 CKM_RSA_PKCS_OAEP 的 CK_RSA_PKCS_OAEP_PARAMS：
  - CKM_SHA_1 使用 CKG_MGF1_SHA1
  - CKM_SHA224 使用 CKG_MGF1_SHA224
  - CKM_SHA256 使用 CKG_MGF1_SHA256
  - CKM_SHA384 使用 CKM_MGF1_SHA384
  - CKM_SHA512 使用 CKM_MGF1_SHA512
- [5] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於處於非 FIP 模式的叢集，在 2023 之後仍然允許使用該叢集。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

PKCS #11 程式庫支援的 API 作業

PKCS #11 程式庫支援下列 PKCS #11 API 操作。

- 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 程式庫支援的索引鍵屬性

金鑰物件可以是公有金鑰、私有金鑰或私密金鑰。系統會透過屬性來指定金鑰物件上允許的動作。金鑰物件建立時，即會一併建立屬性。在您使用 PKCS #11 程式庫時，我們會指派 PKCS #11 標準所指定的預設值。

AWS CloudHSM 不支援 PKCS #11 規格中列出的所有屬性。我們會符合所有支援屬性的規格，並在個別表格中列出這些屬性。

用來建立、修改或複製物件的加密函數 (如 C_CreateObject、C_GenerateKey、C_GenerateKeyPair、C_UnwrapKey 和 C_DeriveKey) 會採用屬性範本做為其中一個參數。如需關於在建立物件期間傳遞屬性範本的詳細資訊，請參閱透過 PKCS #11 程式庫產生金鑰 (以此為例)。

PKCS #11 程式庫屬性解譯表

PKCS #11 程式庫表包含金鑰類型不同的屬性清單。它指示一個給定的屬性是否支持與 AWS CloudHSM特定的密鑰類型使用特定的加密功能時。

圖例：

- ✓ 表示 CloudHSM 支援特定金鑰類型的屬性。
- ✖ 表示 CloudHSM 不支援特定金鑰類型的屬性。
- R 表示特定金鑰類型的屬性值設定為唯讀模式。
- S 表示屬性較為敏感，因此無法透過 GetAttributeValue 讀取。
- 預設值欄位中的空白儲存格表示屬性沒有獲派指定預設值。

GenerateKeyPair

屬性	金鑰類型				預設值
	EC 私有金鑰	EC 公有金鑰	RSA 私有金鑰	RSA 公有金鑰	
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

屬性	金鑰類型				預設值
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

屬性	金鑰類型				預設值
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	✖	✖	✖	✖	

屬性	金鑰類型				預設值
CKA_PUBLIC_EXPONENT	✖	✖	✖	✓[2]	
CKA_EC_PARAMS	✖	✓[2]	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	

GenerateKey

屬性	金鑰類型			預設值
	AES	DES3	一般機密	
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	False

屬性	金鑰類型			預設值
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

屬性	金鑰類型			預設值
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	✖	✖	✖	

屬性	金鑰類型			預設值
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

屬性	金鑰類型							預設值
	EC 私有金鑰	EC 公有金鑰	RSA 私有金鑰	RSA 公有金鑰	AES	DES3	一般機密	
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

屬性	金鑰類型							預設值
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

屬性	金鑰類型							預設值
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]	✖	✖	✖	✖	

屬性	金鑰類型							預設值
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

屬性	金鑰類型						預設值
		EC 私有金鑰	RSA 私有金鑰	AES	DES3	一般機密	
CKA_CLASS		✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE		✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL		✓	✓	✓	✓	✓	
CKA_ID		✓	✓	✓	✓	✓	

屬性	金鑰類型					預設值
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	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✓	✓	✖	False

屬性	金鑰類型					預設值
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	✖	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	✖	✖	

屬性	金鑰類型					預設值
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

屬性	金鑰類型			預設值
	AES	DES3	一般機密	
CKA_CLASS	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	

屬性	金鑰類型			預設值
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	✖	✖	✖	
CKA_WRAP	✓	✓	✖	False

屬性	金鑰類型			預設值
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	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	

屬性	金鑰類型			預設值
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

屬性	金鑰類型						
	EC 私有金鑰	EC 公有金鑰	RSA 私有金鑰	RSA 公有金鑰	AES	DES3	一般機密
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓

屬性	金鑰類型						
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	✖	✖	✓	✖	✖	✖	✖
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓

屬性	金鑰類型						
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	✓	✖	✓	✖	✓	✓	✓
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	R	R

屬性	金鑰類型						
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	✓	✓	✖	✖	✖	✖	✖
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
CKA_VALUE	S	✖	✖	✖	✓	✓	✓

屬性	金鑰類型						
CKA_VALUE_LEN	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	✖

屬性註釋

- [1] 此屬性受韌體部分支援，且需明確設定為僅限預設值。
- [2] 必要屬性。

修改屬性

有些物件屬性可在物件建立後進行修改，但有些不行。若要修改屬性，請使用來自 cloudhsm_mgmt_util 的 setAttribute 命令。您也可以使用來自 cloudhsm_mgmt_util 的 listAttribute 命令，來衍生屬性和代表這些屬性的常數清單。

下列清單會顯示物件建立後可修改的屬性：

- CKA_LABEL
- CKA_TOKEN

> **Note**
>
> 只有在將工作階段金鑰變更為符記金鑰時，才允許進行修改。使用 key_mgmt_util 中的 setAttribute 命令來變更屬性值。

- CKA_ENCRYPT
- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL

- CKA_SENSITIVE
- CKA_DERIVE

> **Note**
>
> 這個屬性支援金鑰衍生。所有公有金鑰的屬性須為 False，不能設定為 True。如果是私密金鑰和 EC 私有金鑰，則該屬性可設定為 True 或 False。

- CKA_TRUSTED

> **Note**
>
> 唯有加密管理員 (CO) 可將這個屬性設定成 True 或 False。

- CKA_WRAP_WITH_TRUSTED

> **Note**
>
> 將此屬性套用於可匯出的資料金鑰，以表明只能使用標記為 CKA_TRUSTED 的金鑰包裝此金鑰。一旦設定 CKA_WRAP_WITH_TRUSTED 為 true，屬性就會變成唯讀，而且您無法變更或移除屬性。

解譯錯誤代碼

若在範本中指定特定金鑰不支援的屬性，就會導致錯誤。下表包含違反規格時所產生的錯誤代碼：

錯誤代碼	Description
CKR_TEMPLATE_INCONSISTENT	當您在屬性範本中指定的屬性符合 PKCS #11 規格，卻不受 CloudHSM 支援時，就會收到此錯誤。
CKR_ATTRIBUTE_TYPE_INVALID	當您擷取的屬性值符合 PKCS #11 規格，卻不受 CloudHSM 支援時，就會收到此錯誤。
CKR_ATTRIBUTE_INCOMPLETE	當您沒有在屬性範本中指定必要屬性時，就會收到此錯誤。

錯誤代碼	Description
CKR_ATTRIBUTE_READ_ONLY	當您在屬性範本中指定唯讀屬性時，就會收到此錯誤。

PKCS #11 程式庫的程式碼範例

上的程式碼範例說 GitHub 明如何使用 PKCS #11 程式庫完成基本工作。

必要條件

執行範例之前，請執行以下步驟來設定您的環境：

- 安裝並設定適用於用戶端 SDK 5 的 PKCS #11 程式庫。
- 設定密碼編譯使用者 (CU)。您的應用程式會使用此 HSM 帳戶在 HSM 上執行程式碼範例。

程式碼範例

PKCS #11 AWS CloudHSM 軟體程式庫的程式碼範例可在上取得。GitHub此儲存庫包括如何使用 PKCS #11 執行一般作業的範例，包括加密、解密、簽署和驗證。

- 產生金鑰 (AES、RSA、EC)
- 列出金鑰屬性
- 使用 AES GCM 加密和解密資料
- 使用 AES_CTR 加密和解密資料
- 使用 3DES 加密和解密資料
- 使用 RSA 簽署和驗證資料
- 使用 HMAC KDF 衍生金鑰
- 使用以 PKCS #5 填補的 AES 包裝和取消包裝金鑰
- 使用不填補的 AES 包裝和取消包裝金鑰
- 使用零填補的 AES 包裝和取消包裝金鑰
- 使用 AES-GCM 包裝和取消包裝金鑰
- 使用 RSA 包裝和取消包裝金鑰

將您的 PKCS #11 程式庫從用戶端 SDK 3 遷移到用戶端 SDK 5

您可以使用本主題，將 PKCS #11 程式庫從用戶端 SDK 3 移轉至用戶端 SDK 5。如需移轉的優點，請參閱用戶端 SDK 5 的優點 。

在中 AWS CloudHSM，客戶應用程式會使用用戶 AWS CloudHSM 端軟體開發套件 (SDK) 執行密碼編譯作業。客戶端 SDK 5 是繼續添加新功能和平台支持的主要 SDK。

若要檢閱所有提供者的移轉指示，請參閱從用戶端 SDK 3 遷移至用戶端 SDK 5。

通過解決突破性更改做好

檢閱這些重大變更，並相應地在開發環境中更新您的應用程式。

換行機制已變更

用戶端 SDK 3 機制	等效的用戶端 SDK 5 機制
CKM_AES_KEY_WRAP	CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
CKM_AES_KEY_WRAP_PAD	CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD
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

在用戶端 SDK 3 中，您可以使用 ECDH 並指定 KDF。用戶端 SDK 5 目前無法使用此功能。如果您的應用程序需要此功能，請聯繫支持。

關鍵控制點現在是工作階段特定的

若要將金鑰控制代碼成功用於用戶端 SDK 5，您必須在每次執行應用程式時取得金鑰控制代碼。如果您的現有應用程式會在不同的工作階段中使用相同的金鑰控制代碼，則必須在每次執行應用程式時修改

程式碼，以取得金鑰控制代碼。如需擷取金鑰控點的相關資訊，請參閱此 AWS CloudHSM PKCS #11 範例。此變更符合 PKCS #11 2.40 規格。

遷移至用戶端 SDK 5

請遵循本節中的指示，從用戶端 SDK 3 遷移至用戶端 SDK 5。

> **Note**
>
> 用戶端軟體開發套件 5 目前不支援 Amazon Linux、Ubuntu 18.04、CentOS 6、CentOS 8 及 RHEL 6。如果您目前正在搭配用戶端 SDK 3 使用其中一個平台，則在移轉至用戶端 SDK 5 時，必須選擇不同的平台。

1. 解除安裝用戶端 SDK 3 的 PKCS #11 程式庫。

   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. 解除安裝用戶端 SDK 3 的用戶端常駐程式。

   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**
>
> 自訂組態需要再次啟用。

3. 依照中的步驟，安裝用戶端 SDK PKCS #11 程式庫。為用戶端 SDK 5 安裝 PKCS #11 程式庫
4. 客戶端 SDK 5 引入了新的配置文件格式和命令行引導工具。若要啟動您的用戶端 SDK 5 PKCS #11 程式庫，請遵循使用者指南中列出的指示。引導用戶端 SDK
5. 在您的開發環境中，測試您的應用程式。在最終移轉之前，更新現有程式碼以解決您的重大變更。

相關主題

- 的最佳做法 AWS CloudHSM

PKCS #11 的進階組態

AWS CloudHSM PKCS #11 提供者包含下列進階組態，這不是大多數客戶使用的一般組態的一部分。這些組態提供額外的功能。

- 使用 PKCS #11 連線到多個插槽
- PKCS #11 的重試組態

使用 PKCS #11 連線到多個插槽

用戶端 SDK 5 PKCS #11 程式庫中的單一插槽代表 AWS CloudHSM中叢集的單一連線。使用用戶端 SDK 5，您可以將 PKCS11 程式庫設定為允許多個插槽，從單一 PKCS #11 應用程式將使用者連線到多個 CloudHSM 叢集。

使用本主題中的說明，讓您的應用程式使用多插槽功能來連線到多個叢集。

主題

多插槽先決條件

- 您想要連線到的兩個或多個 AWS CloudHSM 叢集，以及其叢集憑證。
- 具有安全群組的 EC2 執行個體正確設定為連線到上述所有叢集。如需如何設定叢集和用戶端執行個體的詳細資訊，請參閱入門 AWS CloudHSM。
- 若要設定多插槽功能，您必須已下載並安裝 PKCS #11 程式庫。若您尚未完成此動作，請參閱 ??? 中的說明。

針對多插槽功能設定 PKCS #11 程式庫

若要針對多插槽功能設定 PKCS #11 程式庫，請依照下列步驟執行：

1. 使用多插槽功能識別要連線到的叢集。
2. 依照 ??? 中的說明，將這些叢集新增至 PKCS #11 組態
3. PKCS #11 應用程式下次執行時，其將具有多插槽功能。

configure-pkcs11 新增叢集

使用 PKCS #11 連線到多個插槽時，請使用 configure-pkcs11 add-cluster 命令將叢集新增至您的組態。

語法

```
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]
```

範例

使用 **cluster-id** 參數新增叢集

Example

搭配使用 configure-pkcs11 add-cluster 和 `cluster-id` 參數，將叢集 (ID為 `cluster-1234567`) 新增至您的組態。

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**
>
> 如果 configure-pkcs11 add-cluster 與 `cluster-id` 參數搭配使用不會導致新增叢集，請參閱下列範例，以取得此命令的更長版本，此命令也需要 `--region` 和 `--endpoint` 參數來識別要新增的叢集。例如，如果叢集區域與設定為 AWS CLI 預設值的區域不同，則應使用 `--region` 參數來使用正確的區域。此外，您還可以指定用於呼叫的 AWS CloudHSM API 端點，這對於各種網路設定來說可能是必要的，例如使用不使用預設 DNS 主機名稱的 VPC 介面端點。 AWS CloudHSM

使用 **cluster-id**、**endpoint** 和 **region** 參數新增叢集

Example

搭配使用 configure-pkcs11 add-cluster 以及 `cluster-id`、`endpoint` 和 `region` 參數將叢集 (ID為 `cluster-1234567`) 新增至您的組態。

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
```

如需 `--cluster-id`、`--region` 和 `--endpoint` 參數的詳細資訊，請參閱 the section called “參數”。

參數

--cluster-id ***<Cluster ID>***

進行 `DescribeClusters` 呼叫以尋找叢集中與叢集 ID 關聯的所有 HSM 彈性網路介面 (ENI) IP 地址。系統會將 ENI IP 位址新增至組 AWS CloudHSM 態檔案。

> **Note**
>
> 如果您在無法存取公用網際網路的 VPC 中使用 EC2 執行個體的`--cluster-id`參數，則必須建立要連接的介面 VPC 端點。 AWS CloudHSM如需 VPC 端點的詳細資訊，請參閱 ???。

必要：是

--endpoint ***<Endpoint>***

指定用於進行DescribeClusters呼叫的 AWS CloudHSM API 端點。您必須結合 `--cluster-id` 設定此選項。

必要：否

--hsm-ca-cert <HsmCA Certificate Filepath>

指定 HSM CA 憑證的檔案路徑。

必要：否

--region ***<Region>***

指您叢集的區域。您必須結合 `--cluster-id` 設定此選項。

如果您未提供 `--region` 參數，系統會嘗試讀取 `AWS_DEFAULT_REGION` 或 `AWS_REGION` 環境變數來選擇區域。如果未設定這些變數，則除非您在 `AWS_CONFIG_FILE` 環境變數中指定了不同的檔案，否則系統會檢查 AWS config 檔案中 (通常是 `~/.aws/config`) 與您的設定檔相關聯的區域。如果未設定上述任何變數，系統會預設為 `us-east-1` 區域。

必要：否

--server-client-cert-file <Client Certificate Filepath>

用於 TLS 用戶端與伺服器相互驗證的用戶端憑證路徑。

只有在您不希望使用我們隨用戶端 SDK 5 提供的預設金鑰和 SSL/TLS 憑證時，才使用此選項。您必須結合 `--server-client-key-file` 設定此選項。

必要：否

--server-client-key-file <Client Key Filepath>

TLS 用戶端與伺服器相互驗證所使用的用戶端金鑰路徑。

只有在您不希望使用我們隨用戶端 SDK 5 提供的預設金鑰和 SSL/TLS 憑證時，才使用此選項。您必須結合 `--server-client-cert-file` 設定此選項。

必要：否

configure-pkcs11 移除叢集

使用 PKCS #11 連線到多個插槽時，請使用 configure-pkcs11 remove-cluster 命令從可用的 PKCS #11 插槽中移除叢集。

語法

```
configure-pkcs11 remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

範例

使用 **cluster-id** 參數移除叢集

Example

搭配使用 configure-pkcs11 remove-cluster 和 `cluster-id` 參數，從您的組態中移除叢集 (ID 為 `cluster-1234567`)。

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
```

如需 `--cluster-id` 參數的詳細資訊，請參閱 the section called “參數”。

參數

--cluster-id ***<Cluster ID>***

需從組態中移除的叢集的 ID

必要：是

PKCS #11 的重試命令

用戶端 SDK 5.8.0 及更新版本具有內建的自動重試策略，該策略將從用戶端重試 HSM 限流操作。當 HSM 因過於忙於執行先前操作而無法接受更多要求而限制操作時，用戶端 SDK 會嘗試重試限流操作 (最多 3 次)，同時以指數形式回退。此自動重試策略可以設定為兩種模式中的其中一種：關閉模式和標準模式。

- 關閉：用戶端 SDK 將不會針對 HSM 的任何限流操作執行任何重試政策。
- 標準：這是用戶端 SDK 5.8.0 及更新版本的預設模式。在此模式下，用戶端 SDK 會以指數回退形式自動重試限流操作。

如需詳細資訊，請參閱 HSM 調節。

將重試命令設定為關閉模式

Linux

將 Linux 上用戶端 SDK 5 重試命令設定為 off

- 您可以使用下列命令將重試組態設定為 off 模式：

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --default-retry-mode off
```

Windows

將 Windows 上用戶端 SDK 5 重試命令設定為 off

- 您可以使用下列命令將重試組態設定為 off 模式：

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-pkcs11.exe --default-retry-
mode off
```

OpenSSL 動態引擎

AWS CloudHSM OpenSSL 動態引擎可讓您透過 OpenSSL API 將密碼編譯作業卸載到您的 CloudHSM 叢集。

AWS CloudHSM 提供了一個 OpenSSL 動態引擎，您可以在中Linux 上的 SSL/TLS 卸載閱讀。如需 AWS CloudHSM 搭配 OpenSSL 使用的範例，請參閱此 AWS 安全部落格。如需關於 SDK 平台支援的詳細資訊，請參閱 the section called “支援平台”。如需疑難排解，請參閱OpenSSL 動態引擎的已知問題。

如需關於使用用戶端 SDK 3 的資訊，請參閱 先前的用戶端 SDK (用戶端 SDK 3)。

如需詳細資訊，請參閱下列主題。

主題

- 安裝 OpenSSL 動態引擎
- 動態 OpenSSL 擎金鑰類型
- OpenSSL 引擎機制
- 將您的動態引擎從用戶端 SDK 3 遷移到用戶端 SDK 5
- OpenSSL 的進階組態

安裝 OpenSSL 動態引擎

> **Note**
>
> 若要使用用戶端 SDK 5 執行單一 HSM 叢集，您必須先將 `disable_key_availability_check` 設定為 `True` 來管理用戶端金鑰持久性。如需詳細資訊，請參閱金鑰同步處理和用戶端 SDK 5 設定工具。

安裝和設定 OpenSSL 動態引擎

1. 使用以下命令來下載和安裝 OpenSSL 引擎。

   Amazon Linux 2

   在 x86_64 架構上安裝適用於 Amazon 的動態引擎：

   ```
   $ 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
   ```

在 ARM64 架構 OpenSSL 安裝 Amazon Linux 2 的動態引擎：

```
$ 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

在 64 架構上安裝適用於 Amazon 的動態引擎：

```
$ 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
```

在 ARM64 架構上安裝適用於 Amazon 2023 的動態引擎：

```
$ 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+)

在 x86_64 架構上安裝用於 CentOS 7 的動態引擎：

```
$ 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+)

在 64 架構 OpenSSL 安裝適用於 RHEL 7 的動態引擎：

```
$ 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+)

在 64 架構 OpenSSL 安裝適用於 RHEL 8 的動態引擎：

```
$ 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+)

在 64 架構 OpenSSL 安裝適用於 RHEL 9 的動態引擎：

```
$ 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
```

在 ARM64 架構上安裝適用於 RHEL 9 的動態引擎：

```
$ 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

在 64 架構上安裝用於 Ubuntu 的動態引擎：

```
$ 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

在 64 架構上安裝適用於 Ubuntu 的動態引擎：

```
$ 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
```

在 ARM64 架構上安裝動態引擎

```
$ 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
```

您已在 /opt/cloudhsm/lib/libcloudhsm_openssl_engine.so 中安裝動態引擎的共用程式庫。

2. 引導用戶端 SDK 5。如需關於啟動程序的詳細資訊，請參閱 引導用戶端 SDK。
3. 使用加密使用者 (CU) 的憑證設定環境變數。如需關於建立 CU 的資訊，請參閱 使用 CMU 管理使用者。

```
$ export CLOUDHSM_PIN=<HSM user name>:<password>
```

> **Note**
>
> 用戶端 SDK 5 引入了用於存儲 CU 憑證的 CLOUDHSM_PIN 環境變量。在用戶端 SDK 3 中，您可以將 CU 憑證儲存在 n3fips_password 環境變數中。用戶端 SDK 5 支援這兩個環境變數，但建議使用 CLOUDHSM_PIN。

4. 將您的 OpenSSL 動態引擎連接到叢集。如需詳細資訊，請參閱連接至叢集。
5. 引導用戶端 SDK 5。如需詳細資訊，請參閱 the section called “引導用戶端 SDK”。

驗證用戶端 SDK 5 OpenSSL 的動態引擎

使用下面的命令來驗證您的 OpenSSL 動態引擎的安裝。

```
$ openssl engine -t cloudhsm
```

下列輸出會驗證組態：

```
(cloudhsm) CloudHSM OpenSSL Engine
     [ available ]
```

動態 OpenSSL 擎金鑰類型

AWS CloudHSM OpenSSL 動態引擎支援下列金鑰類型。

金鑰類型	描述
EC	對 P-256、P-384 和 secp256k1 金鑰類型進行 ECDSA 簽署/驗證。若要產生可與 OpenSSL 引擎互通的 EC 金鑰，請參閱 產生金鑰的檔案。
RSA	產生適用於 2048、3072 和 4096 位元金鑰的 RSA 金鑰。驗證程序已卸載到 OpenSSL 軟件中。

OpenSSL 引擎機制

瞭解如何使用 AWS CloudHSM OpenSSL 動態引擎機制。

簽署和驗證函數

AWS CloudHSM OpenSSL 動態引擎可讓您針對「簽署」和「驗證」功能使用下列機制。

使用用戶端 SDK 5，資料會在軟體中進行本機雜湊處理。這意味著可以散列的數據的大小沒有限制。

RSA 簽章類型

- SHA1withRSA
- SHA224withRSA

- SHA256withRSA
- SHA384withRSA
- SHA512withRSA

ECDSA 簽章類型

- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

將您的動態引擎從用戶端 SDK 3 遷移到用戶端 SDK 5

您可以使用本主題，將您的 OpenSSL 動態引擎從用戶端 SDK 3 遷移至用戶端 SDK 5。如需移轉的優點，請參閱用戶端 SDK 5 的優點 。

在中 AWS CloudHSM，客戶應用程式會使用用戶 AWS CloudHSM 端軟體開發套件 (SDK) 執行密碼編譯作業。客戶端 SDK 5 是繼續添加新功能和平台支持的主要 SDK。

> **Note**
>
> 使用 OpenSSL 動態引擎的用戶端 SDK 5 目前不支援隨機數產生。

若要檢閱所有提供者的移轉指示，請參閱從用戶端 SDK 3 遷移至用戶端 SDK 5。

遷移至用戶端 SDK 5

請遵循本節中的指示，從用戶端 SDK 3 遷移至用戶端 SDK 5。

> **Note**
>
> 用戶端軟體開發套件 5 目前不支援 Amazon Linux、Ubuntu 18.04、CentOS 6、CentOS 8 及 RHEL 6。如果您目前正在搭配用戶端 SDK 3 使用其中一個平台，則在移轉至用戶端 SDK 5 時，必須選擇不同的平台。

1. 解除安裝用戶端 SDK 3 的 OpenSSL 動態引擎。

   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. 解除安裝用戶端 SDK 3 的用戶端常駐程式。

   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**
>
> 自訂組態需要再次啟用。

3. 依照中安裝 OpenSSL 動態引擎的步驟，安裝用戶端 SDK OpenSSL 動態引擎。
4. 客戶端 SDK 5 引入了新的配置文件格式和命令行引導工具。若要啟動您的用戶端 SDK 5 OpenSSL 動態引擎，請遵循下引導用戶端 SDK方的使用者指南中列出的指示。
5. 在您的開發環境中，測試您的應用程式。在最終移轉之前，更新現有程式碼以解決您的重大變更。

相關主題

- 的最佳做法 AWS CloudHSM

OpenSSL 的進階組態

AWS CloudHSM OpenSSL 提供者包含下列進階設定，這不是大多數客戶使用的一般組態的一部分。這些組態提供額外的功能。

- 適用於 OpenSSL 的重試命令

適用於 OpenSSL 的重試命令

用戶端 SDK 5.8.0 及更新版本具有內建的自動重試策略，該策略將從用戶端重試 HSM 限流操作。當 HSM 因過於忙於執行先前操作而無法接受更多要求而限制操作時，用戶端 SDK 會嘗試重試限流操作 (最多 3 次)，同時以指數形式回退。此自動重試策略可以設定為兩種模式中的其中一種：關閉模式和標準模式。

- 關閉：用戶端 SDK 將不會針對 HSM 的任何限流操作執行任何重試政策。
- 標準：這是用戶端 SDK 5.8.0 及更新版本的預設模式。在此模式下，用戶端 SDK 會以指數回退形式自動重試限流操作。

如需詳細資訊，請參閱 HSM 調節。

將重試命令設定為關閉模式

Linux

將 Linux 上用戶端 SDK 5 重試命令設定為 off

- 您可以使用下列命令將重試命令設定為 off 模式：

```
$ sudo /opt/cloudhsm/bin/configure-dyn --default-retry-mode off
```

Windows

將 Windows 上用戶端 SDK 5 重試命令設定為 off

- 您可以使用下列命令將重試命令設定為 off 模式：

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-dyn.exe --default-retry-mode
 off
```

JCE 提供者

AWS CloudHSM JCE 提供者是根據 Java 密碼編譯延伸 (JCE) 提供者架構建置的提供者實作。JCE 可讓您使用 Java 開發套件 (JDK) 執行密碼編譯操作。在本指南中， AWS CloudHSM JCE 提供者有時稱為 JCE 提供者。使用 JCE 提供者和 JDK 將密碼編譯操作卸載至 HSM。如需疑難排解，請參閱JCE 開發套件的已知問題。

如需關於使用用戶端 SDK 3 的資訊，請參閱 先前的用戶端 SDK (用戶端 SDK 3)。

主題

- 安裝並使用用戶端 SDK 5 的 AWS CloudHSM JCE 提供者
- JCE 提供者支援的金鑰類型
- 支援的 JCE 提供者機制
- 支援的 Java 金鑰屬性
- Java AWS CloudHSM 軟體程式庫的程式碼範例
- AWS CloudHSM JCE 提供者爪哇
- 如 AWS CloudHSM KeyStore 何使用 Java 類別

- 將您的 JCE 提供者從用戶端 SDK 3 遷移到用戶端 SDK 5
- 適用於 JCE 的進階組態

安裝並使用用戶端 SDK 5 的 AWS CloudHSM JCE 提供者

JCE 提供者與 OpenJDK 8、OpenJDK 11、OpenJDK 17 和 OpenJDK 21 相容。您可以從 OpenJDK 網站下載 OpenJDK 8 和 OpenJDK11。

> **Note**
>
> 若要使用用戶端 SDK 5 執行單一 HSM 叢集，您必須先將 `disable_key_availability_check` 設定為 `True` 來管理用戶端金鑰持久性。如需詳細資訊，請參閱金鑰同步處理和用戶端 SDK 5 設定工具。

主題

- 安裝 JCE 提供者
- 提供憑證給 JCE 提供者
- JCE 提供者中的金鑰管理基礎知識

安裝 JCE 提供者

1. 使用下列命令下載和安裝 JCE 提供者。

   Amazon Linux 2

   在 x86_64 架構上安裝 Amazon Linux 2 的 JCE 提供商：

   ```
   $ 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
   ```

   在 ARM64 架構上安裝 Amazon Linux 2 的 JCE 提供商：

```
$ 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

在 x86_64 架構上安裝適用於 Amazon 2023 的 JCE 供應商：

```
$ 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
```

在 ARM64 架構上安裝 Amazon 2023 的 JCE 供應商：

```
$ 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+)

在 x86_64 架構上安裝 CentOS 7 的 JCE 供應商：

```
$ 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+)

在 x86_64 架構上安裝 RHEL 7 的 JCE 提供者：

```
$ 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+)

在 x86_64 架構上安裝 RHEL 8 的 JCE 提供者：

```
$ 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+)

在 x86_64 架構上安裝 RHEL 9 (9.2 以上) 的 JCE 提供者：

```
$ 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
```

在 ARM64 架構上安裝 RHEL 9 (9.2 以上) 的 JCE 提供者：

```
$ 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

在 x86_64 架構上安裝 Ubuntu 20.04 LTS 的 JCE 提供程序：

```
$ 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

在 64 架構上安裝 Ubuntu 22.04 LTS 的 JCE 提供者：

```
$ 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
```

在 ARM64 架構上安裝 Ubuntu 22.04 LTS 的 JCE 提供程序：

```
$ 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

在 x86_64 架構上安裝適用於 Windows 伺服器 2016 的 JCE 提供者，以系統管理員身分開啟 PowerShell ，然後執行下列命令：

```
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

在 x86_64 架構上安裝適用於 Windows 服務器 2019 的 JCE 提供程序，以管理員身份打開並運行 PowerShell 以下命令：

```
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. 引導用戶端 SDK 5。如需有關啟動程序的詳細資訊，請參閱 引導用戶端 SDK。
3. 找出下列 JCE 提供者檔案：

   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

提供憑證給 JCE 提供者

HSM 需要先驗證您的 Java 應用程式，應用程式才能使用 HSM。HSM 會使用明確登入或隱含登入方法，進行驗證。

明確登入：此方法可讓您直接在應用程式中提供 AWS CloudHSM 憑證。它使用 AuthProvider 中的方法，讓您可以在 pin 模式中傳遞 CU 使用者名稱和密碼。如需詳細資訊，請參閱登入 HSM 程式碼範例。

隱含登入：此方法可讓您在新的屬性檔案、系統屬性或環境變數中設定 AWS CloudHSM 憑證。

- 系統屬性：執行應用程式時，您可以透過系統屬性設定憑證。以下範例示範兩種不同的作法：

  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>");
```

- 環境變數：將憑證設定為環境變數。

  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>"
  ```

如果應用程式未提供登入資料，或者，如果您在 HSM 驗證工作階段之前嘗試操作，就可能無法使用登入資料。在這些情況下，適用於 Java 的 CloudHSM 軟體程式庫會按照以下順序，搜尋登入資料：

1. 系統屬性
2. 環境變數

JCE 提供者中的金鑰管理基礎知識

JCE 提供者的金鑰管理基本概念包括匯入金鑰、匯出金鑰、透過控制代碼載入金鑰或刪除金鑰。如需管理金鑰的詳細資訊，請參閱管理金鑰的程式碼範例。

您也可以在 程式碼範例 中尋找更多 JCE 提供者程式碼範例。

JCE 提供者支援的金鑰類型

Java AWS CloudHSM 軟體程式庫可讓您產生下列金鑰類型。

金鑰類型	描述
AES	產生 128 位元、192 位元和 256 位元的 AES 金鑰。

金鑰類型	描述
Triple DES (3DES, DESede)	產生 192 位元三重 DES 金鑰[1]的變更。請參閱註腳以瞭解即將到來
EC	產生 EC 金鑰對：NIST curves secp224r1 (P-224)、secp256r1 (P-256)、secp256k1 (Blockchain)、secp384r1 (P-384) 和 secp521r1 (P-521)。
GENERIC_SECRET	產生 1 到 800 位元組的一般機密。
HMAC	支援 SHA1、SHA224、SHA256、SHA384 和 SHA512 雜湊。
RSA	產生 2048 位元至 4096 位元的 RSA 金鑰，以 256 位元為單位遞增。

[1] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於非 FIP 模式下的叢集，在 2023 之後仍然允許。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

支援的 JCE 提供者機制

本主題提供有關用戶端 SDK 5 的 JCE 提供者支援機制的相關資訊。如需支援之 Java 密碼編譯架構 (JCA) 介面和引擎類別的相關資訊 AWS CloudHSM，請參閱下列主題。

主題

- 產生金鑰與金鑰對函數
- 密碼函數
- 簽署和驗證函數
- Digest 函數
- 雜湊訊息驗證碼 (HMAC) 函數
- 加密式訊息驗證程式碼 (CMAC) 函數
- 使用金鑰 Factory 將金鑰轉換為金鑰規格
- 機制註釋

產生金鑰與金鑰對函數

Java AWS CloudHSM 軟體程式庫可讓您使用下列作業來產生金鑰與 key pair 功能。

- RSA
- EC
- AES
- DESede (Triple DES)[請參閱備註 1]
- GenericSecret

密碼函數

Java 的 AWS CloudHSM 軟體程式庫支援下列演算法、模式和填補組合。

演算法	Mode	填補	備註
AES	CBC	AES/CBC/NoPadding AES/CBC/PKCS5Padding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。 實作 Cipher.UNWRAP_MODE for AES/CBC NoPadding
AES	ECB	AES/ECB/PKCS5Padding AES/ECB/NoPadding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。
AES	CTR	AES/CTR/NoPadding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。

演算法	Mode	填補	備註
AES	GCM	AES/GCM/NoPadding	實作 Cipher.WRAP_MODE 、Cipher.UNWRAP_MODE 、Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。 執行 AES-GCM 加密時，HSM 會忽略請求中的初始化向量 (IV)，並使用自己產生的 IV。操作完成後，您必須呼叫 Cipher.getIV() 以取得 IV。
AESWrap	ECB	AESWrap/ECB/NoPadding AESWrap/ECB/PKCS5Padding AESWrap/ECB/ZeroPadding	實作 Cipher.WRAP_MODE 和 Cipher.UNWRAP_MODE 。
DESede (三重 DES)	CBC	DESede/CBC/PKCS5Padding DESede/CBC/NoPadding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。請參閱下列備註 1 查看即將進行的變更。

演算法	Mode	填補	備註
DESede (三重 DES)	ECB	DESede/ECB/NoPadding DESede/ECB/PKCS5Padding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。請參閱下列備註 1 查看即將進行的變更。

演算法	Mode	填補	備註
RSA	ECB	RSA/ECB/PKCS1Padding 請參閱備註 1 RSA/ECB/OAEPPadding RSA/ECB/OAEPWithSHA-1ANDMGF1Padding RSA/ECB/OAEPWithSHA-224ANDMGF1Padding RSA/ECB/OAEPWithSHA-256ANDMGF1Padding RSA/ECB/OAEPWithSHA-384ANDMGF1Padding RSA/ECB/OAEPWithSHA-512ANDMGF1Padding	實作 Cipher.WRAP_MODE 、Cipher.UNWRAP_MODE 、Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。
RSA	ECB	RSA/ECB/NoPadding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。

演算法	Mode	填補	備註
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	實作 Cipher.WRAP_MODE 和 Cipher.UNWRAP_MODE 。

簽署和驗證函數

Java AWS CloudHSM 軟體程式庫支援下列類型的簽章和驗證。使用用戶端 SDK 5 和具有雜湊功能的簽章演算法，資料會先在軟體中進行本機雜湊處理，然後再傳送至 HSM 以進行簽章/驗證。這意味著 SDK 可以對任何大小的資料進行雜湊處理。

RSA 簽章類型

- NONEwithRSA
- RSASSA-PSS
- SHA1withRSA
- SHA1withRSA/PSS
- SHA1withRSAandMGF1
- SHA224withRSA
- SHA224withRSAandMGF1
- SHA224withRSA/PSS
- SHA256withRSA
- SHA256withRSAandMGF1
- SHA256withRSA/PSS
- SHA384withRSA
- SHA384withRSAandMGF1
- SHA384withRSA/PSS
- SHA512withRSA
- SHA512withRSAandMGF1
- SHA512withRSA/PSS

ECDSA 簽章類型

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

Digest 函數

Java 的 AWS CloudHSM 軟體程式庫支援下列訊息摘要。使用用戶端 SDK 5，資料會在軟體中進行本機雜湊處理。這意味著 SDK 可以對任何大小的資料進行雜湊處理。

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

雜湊訊息驗證碼 (HMAC) 函數

適用於 Java 的 AWS CloudHSM 軟體程式庫支援下列 HMAC 演算法。

- HmacSHA1（最大數據大小（以字節為單位：16288）
- HmacSHA224(資料大小上限 (位元組)：16256)
- HmacSHA256（最大數據大小（以字節為單位：16288）
- HmacSHA384（最大數據大小（以字節為單位：16224）
- HmacSHA512（最大數據大小（以字節為單位：16224）

加密式訊息驗證程式碼 (CMAC) 函數

CMAC (加密式訊息驗證程式碼) 使用區塊密碼和金鑰建立訊驗證代碼 (MAC)。與 HMAC 不同之處在於，它們使用 MAC 區塊對稱金鑰方法而非雜湊方法。

適用於 Java 的 AWS CloudHSM 軟體程式庫支援下列 CMAC 演算法。

- AESCMAC

使用金鑰 Factory 將金鑰轉換為金鑰規格

您可以使用金鑰工廠將金鑰轉換為金鑰規格。 AWS CloudHSM 有兩種類型的 JCE 關鍵工廠：

SecretKeyFactory：用於匯入或導出對稱金鑰。使用時 SecretKeyFactory，您可以傳遞支援的金鑰或支援的金鑰 KeySpec 來匯入或衍生對稱金鑰。 AWS CloudHSM以下是支援的規格 KeyFactory：

- FOR SecretKeyFactory 的generateSecret方法支持以下KeySpec類：
  - KeyAttributesMap可用於將具有其他屬性的金鑰位元組匯入為 CloudHSM 金鑰。可從此處查看範例。

- SecretKeySpec可用來將對稱金鑰規格匯入為 CloudHSM 金鑰。
- AesCmacKdfParameterSpec可用於使用另一個 CloudHSM AES 金鑰衍生對稱金鑰。

> **Note**
>
> SecretKeyFactory的`translateKey`方法採用任何實現密鑰接口的密鑰。

KeyFactory：用於匯入非對稱金鑰。使用時 KeyFactory，您可以傳遞支援的金鑰或支援 KeySpec 將非對稱金鑰匯入 AWS CloudHSM。如需詳細資訊，請參閱下列資源：

- 對於 KeyFactory的`generatePublic`方法，支持以下KeySpec類：
- 適用於 RSA 和 EC KeyAttributesMap KeyTypes 的 CloudHSM，包括：
  - 適用於 RSA 和 KeyAttributesMap EC 公用的 CloudHSM。 KeyTypes可從此處查看範例。
  - EncodedKeySpec適用於 RSA 和 EC 公鑰的 X509
  - RSA 公開金PublicKeySpec鑰的 RSA
  - EC 公開金鑰PublicKeySpec的 EC
- 對於 KeyFactory的`generatePrivate`方法，支持以下KeySpec類：
- 適用於 RSA 和 EC KeyAttributesMap KeyTypes 的 CloudHSM，包括：
  - 適用於 RSA 和 KeyAttributesMap EC 公用的 CloudHSM。 KeyTypes可從此處查看範例。
  - PKCS8 EncodedKeySpec 適用於 EC 和 RSA 私密金鑰
  - RSA 私密金PrivateCrtKeySpec鑰的 RSA
  - EC PrivateKeySpec 用於 EC 私鑰

For KeyFactory 的`translateKey`方法，它接受任何實現密鑰接口的密鑰。

機制註釋

[1] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於非 FIP 模式下的叢集，在 2023 之後仍然允許。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

支援的 Java 金鑰屬性

本主題介紹了客戶端 SDK 5 支持的 Java 關鍵屬性的信息。本主題說明如何使用 JCE 提供程序的專屬擴充功能來設定金鑰屬性。請在以下操作期間，使用此擴充功能來設定支援的金鑰屬性及其值：

- 金鑰產生
- 金鑰匯入

如需如何使用金鑰屬性的範例，請參閱 the section called “程式碼範例”。

主題

- 了解屬性
- 支援的屬性
- 設定金鑰的屬性

了解屬性

使用金鑰屬性來指定金鑰物件上允許的動作，包括公有金鑰、私有金鑰或秘密金鑰。在建立金鑰物件操作期間定義金鑰屬性和值。

Java Cryptography Extension (JCE) 未指定應該如何設定金鑰屬性的值，因此預設情況下會允許大部分動作。相反地，PKCS#11 標準定義了一組具有更嚴格預設值的完整屬性。從 JCE 提供者 3.1 開始， AWS CloudHSM 提供專屬的擴充功能，可讓您為常用屬性設定更嚴格的值。

支援的屬性

您可以設定下表中屬性的值。最佳實務是，只為您希望更具有限制性的屬性設定值。如果未指定值，AWS CloudHSM 會使用下表中指定的預設值。預設值欄位中的空白儲存格表示屬性沒有獲派指定預設值。

屬性	預設值			備註
	對稱金鑰	金鑰對中的公有金鑰	金鑰對中的私有金鑰	
DECRYPT	TRUE		TRUE	True 表示您可以使用金鑰來解密任何緩衝區。對於其 WRAP 設定為 true 的金鑰，通常將此設定為 FALSE。

屬性	預設值			備註
	對稱金鑰	金鑰對中的公有金鑰	金鑰對中的私有金鑰	
DERIVE				可以使用金鑰來衍生其他金鑰。
ENCRYPT	TRUE	TRUE		True 表示您可以使用金鑰來加密任何緩衝區。
EXTRACTABLE	TRUE		TRUE	True 表示您可以從 HSM 匯出此金鑰。
ID				用於識別金鑰的使用者定義值。
KEY_TYPE				用於識別金鑰的類型 (AES、DESede、一般私密錦綸、EC 或 RSA)。
LABEL				使用者定義的字串可方便您識別 HSM 上的金鑰。為了遵循最佳實務，請為每個金鑰使用唯一標籤，以便日後更容易找到。
LOCAL				表示 HSM 產生的金鑰。

屬性	預設值			備註
	對稱金鑰	金鑰對中的公有金鑰	金鑰對中的私有金鑰	
OBJECT_CLASS				用於識別索引鍵 (SecretKey、PublicKey 或 PrivateKey) 的物件類別。
PRIVATE	TRUE	TRUE	TRUE	True 表示在使用者通過驗證之前，使用者可能無法存取金鑰。為了清楚起見，使用者在通過驗證 AWS CloudHSM 之前無法存取任何金鑰，即使此屬性設定為 FALSE 也是如此。
SIGN	TRUE		TRUE	True 表示您可以使用金鑰來簽署訊息摘要。對於公有金鑰和您已封存的私有金鑰，這通常會設定為 FALSE。

屬性	預設值			備註
	對稱金鑰	金鑰對中的公有金鑰	金鑰對中的私有金鑰	
SIZE				定義金鑰大小的屬性。如需有關支援金鑰大小的詳細資訊，請參閱用戶端 SDK 5 的支援機制。
TOKEN	FALSE	FALSE	FALSE	永久金鑰，會跨叢集中的所有 HSM 進行複寫並包含在備份中。TOKEN = FALSE 代表一個暫時性金鑰，該金鑰會在與 HSM 的連接中斷時自動擦除。
UNWRAP	TRUE		TRUE	True 表示您可以使用金鑰來取消包裝 (匯入) 另一個金鑰。
VERIFY	TRUE	TRUE		True 表示您可以使用金鑰來驗證簽章。對於私有金鑰，這通常被設定為 FALSE。

屬性	預設值			備註
	對稱金鑰	金鑰對中的公有金鑰	金鑰對中的私有金鑰	
WRAP	TRUE	TRUE		True 表示您可以使用該金鑰來包裝另一個金鑰。對於私有金鑰，您通常將此設定為 FALSE。
WRAP_WITH_TRUSTED	FALSE		FALSE	True 表示金鑰只能與 TRUSTED 屬性為 True 的金鑰進行包裝和取消包裝。一旦 金鑰 WRAP_WITH_TRUSTED 設置為 True，該屬性僅為只讀狀態，不能將其設置為 False。如要了解有關信任包裝的資訊，請參閱使用受信任金鑰控制金鑰解除包裝。

> **Note**
>
> 您可以在 PKCS #11 程式庫中獲得更廣泛的屬性支援。如需詳細資訊，請參閱支援的 PKCS #11 屬性。

設定金鑰的屬性

KeyAttributesMap 是類似 Java 映射的物件，您可以使用它來設定金鑰物件的屬性值。KeyAttributesMap 函數的方法類似於用於 Java 映射處理的方法。

若要設定屬性的自訂值，您有兩個選項：

- 使用下表中列出的方法
- 使用本文稍後示範的產生器模式

屬性映射物件支援以下列方法設定屬性：

作業	傳回值	**KeyAttributesMap** 方法
獲取現有金鑰的金鑰屬性的值	物件 (包含值) 或 null	get(keyAttribute)
填入一個金鑰屬性的值	與金鑰屬性相關聯的先前值，如果金鑰屬性沒有映射，則為 null	put(keyAttribute, value)
填入多個金鑰屬性的值	N/A	putAll () keyAttributesMap
從屬性映射中刪除金鑰值對	與金鑰屬性相關聯的先前值，如果金鑰屬性沒有映射，則為 null	remove(keyAttribute)

> **Note**
>
> 您未明確指定的任何屬性都會設定為the section called “支援的屬性”中上表所列的預設值。

設定金鑰對的屬性

使用 Java 類別 KeyPairAttributesMap 來處理金鑰對的金鑰屬性。KeyPairAttributesMap 封裝了兩個 KeyAttributesMap 物件；一個用於公有金鑰，另一個用於私有金鑰。

若要分別為公有金鑰和私有金鑰設定個別屬性，您可以在該金鑰對應的 KeyAttributes 映射物件上使用 put() 方法。使用 getPublic() 方法來擷取公有金鑰的屬性映射，以及使用 getPrivate()

來擷取私有金鑰的屬性映射。使用 `putAll()` 與金鑰對屬性映射作為引數，填入公有金鑰和私有金鑰對的多個金鑰屬性值。

Java AWS CloudHSM 軟體程式庫的程式碼範例

本主題提供有關用戶端 SDK 5 之 Java 程式碼範例的資源和資訊。

必要條件

執行範例之前，您必須先設定環境：

- 安裝並設定 Java 密碼編譯延伸模組 (JCE) 提供者。
- 設定有效的 HSM 使用者名稱和密碼。加密使用者 (CU) 許可足夠執行這些任務。在每個範例中，您的應用程式會使用這些登入資料來登入 HSM。
- 決定如何提供憑證給 JCE 提供者。

程式碼範例

下列程式碼範例會示範如何使用 AWS CloudHSM JCE 提供者來執行基本任務。更多程式碼範例可在上取得GitHub。

- 登入 HSM
- 管理金鑰
- 產生對稱金鑰
- 產生非對稱金鑰
- 使用 AES-GCM 進行加密和解密
- 使用 AES-CTR 進行加密和解密
- 使用 DESede-ECB 加密和解密[1]
- 使用 RSA 金鑰簽署和驗證
- 使用 EC 金鑰簽署和驗證
- 使用支援的金鑰屬性
- 使用 CloudHSM 金鑰存放區

[1] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於處於非 FIP 模式的叢集，在 2023 之後仍然允許使用該叢集。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

AWS CloudHSM JCE 提供者爪哇

使用 JCE 提供者 Javadocs 取得 AWS CloudHSM JCE SDK 中定義的 Java 類型和方法的使用資訊。若要下載最新的 Javadocs AWS CloudHSM，請參閱下載最新版本頁面上的章節。

您可以將 Javadocs 匯入整合式開發環境 (IDE)，或在網頁瀏覽器中檢視 Javadocs。

如 AWS CloudHSM KeyStore 何使用 Java 類別

該 AWS CloudHSM `KeyStore`類提供了一個特殊用途的 PKCS12 密鑰存儲。此金鑰存放區可與您的金鑰資料一起儲存憑證，並將其與在 AWS CloudHSM儲存的金鑰資料連結在一起。此 AWS CloudHSM `KeyStore`類別會實作 Java 密碼編譯延伸模組 (JCE) 的`KeyStore`服務提供者介面 (SPI)。如需有關使用的詳細資訊`KeyStore`，請參閱類別KeyStore。

> **Note**
>
> 由於憑證是公開資訊，並且為了最大限度地提高加密金鑰的儲存容量，因 AWS CloudHSM 此不支援在 HSM 上儲存憑證。

選擇適當的金鑰存放區

AWS CloudHSM Java 加密延伸模組 (JCE) 提供者提供特殊用途的 AWS CloudHSM 端 HSM。KeyStore此 AWS CloudHSM `KeyStore`類別支援將金鑰作業卸載至 HSM、憑證的本機儲存區和憑證型作業。

載入特殊用途 CloudHSM KeyStore ，如下所示：

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

正在初始化 AWS CloudHSM KeyStore

使用登 AWS CloudHSM KeyStore 入 JCE 提供者的相同方式登入。您可以使用環境變數或系統屬性檔案，並且在開始使用 CloudHSM KeyStore 之前，應先登入。如需使用 JCE 提供者登入 HSM 的範例，請參閱登入 HSM。

如有需要，您可以指定一組密碼來加密保存金鑰儲存資料的本機 PKCS12 檔案。創建密 AWS CloudHSM 鑰庫時，您可以設置密碼並在使用加載，set 和 get 方法時提供密碼。

實例化一個新的 CloudHSM KeyStore 對象，如下所示：

```
ks.load(null, null);
```

使用 store 方法將金鑰存放區資料寫入檔案。從那時起，您可以依照以下方式，搭配來源檔案和密碼使用 load 方法載入現有的金鑰存放區：

```
ks.load(inputStream, password);
```

使用 AWS CloudHSM KeyStore

AWS CloudHSM KeyStore 符合 JCE 類別KeyStore規格，並提供下列功能。

- load

  從特定的輸入串流載入金鑰存放區。如果在儲存金鑰存放區時設定了密碼，則必須提供相同的密碼才能成功載入。將這兩個參數設為 Null 以初始化一個新的空金鑰存放區。

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  ks.load(inputStream, password);
  ```

- aliases

  傳回特定的金鑰存放區執行個體中的所有項目別名名稱列舉。結果包括本機儲存在 PKCS12 檔案中的物件，以及常駐於 HSM 的物件。

  範本程式碼：

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();) {
      String label = entry.nextElement();
      System.out.println(label);
  }
  ```

- containsalias

  如果金鑰存放區可存取至少一個有指定別名的物件，則傳回 True。金鑰存放區會檢查本機儲存在 PKCS12 檔案中的物件，以及常駐於 HSM 的物件。

- deleteEntry

  從本機 PKCS12 檔案刪除憑證項目。使用不支援刪除儲存在 HSM 中的 AWS CloudHSM KeyStore 金鑰資料。您可以使用可銷毀介面 destroy 的銷毀方法刪除金鑰。

```
((Destroyable) key).destroy();
```

- `getCertificate`

  如果可用，則傳回與別名相關聯的憑證。如果別名不存在或參考非憑證的物件，則函數會傳回 NULL。

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
Certificate cert = ks.getCertificate(alias);
```

- `getCertificateAlias`

  傳回第一個資料符合特定憑證的金鑰存放區項目名稱 (別名)。

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
String alias = ks.getCertificateAlias(cert);
```

- `getCertificateChain`

  傳回與特定別名相關聯的憑證鏈。如果別名不存在或參考非憑證的物件，則函數會傳回 NULL。

- `getCreationDate`

  傳回由給定別名辨識的項目建立日期。如果沒有建立日期，則函數會傳回憑證生效的日期。

- `getKey`

  GetKey 會傳遞至 HSM，並傳回對應於指定標籤的金鑰物件。`getKey`直接查詢 HSM 時，它可以用於 HSM 上的任何金鑰，無論它是否由. KeyStore

```
Key key = ks.getKey(keyLabel, null);
```

- `isCertificateEntry`

  檢查有特定別名的項目是否代表憑證項目。

- `isKeyEntry`

  檢查有特定別名的項目是否代表金鑰項目。此動作會在 PKCS12 檔案和 HSM 中搜尋別名。

- `setCertificateEntry`

指定特定憑證至特定別名。如果特定的別名已用於識別金鑰或憑證，則會擲出 KeyStoreException。您可以使用 JCE 程式碼取得金鑰物件，然後使用該 KeyStore SetKeyEntry方法將憑證與金鑰產生關聯。

- 有 byte[] 金鑰的 setKeyEntry

  用戶端 SDK 5 目前不支援此 API。

- 有 Key 物件的 setKeyEntry

  指定特定金鑰至特定別名，並將其儲存在 HSM 內。如果該金鑰未儲存在 HSM 內，則該金鑰將做為可擷取的工作階段金鑰匯入至 HSM。

  如果 Key 物件屬於類型 PrivateKey，則必須附有相對應的憑證鏈。

  如果別名已經存在，則 SetKeyEntry 呼叫會擲出 KeyStoreException 並防止覆寫金鑰。如果必須覆寫金鑰，請使用 KMU 或 JCE。

- engineSize

  傳回金鑰存放區中的項目數目。

- store

  以 PKCS12 檔案格式儲存金鑰存放區至特定輸出串流，並使用特定的密碼加以保護。此外，它仍然保留載入的金鑰 (使用 setKey 呼叫的組合)。

將您的 JCE 提供者從用戶端 SDK 3 遷移到用戶端 SDK 5

使用本主題可將您的 JCE 提供者從用戶端 SDK 3 遷移至用戶端 SDK 5。如需移轉的優點，請參閱用戶端 SDK 5 的優點 。

在中 AWS CloudHSM，客戶應用程式會使用用戶 AWS CloudHSM 端軟體開發套件 (SDK) 執行密碼編譯作業。客戶端 SDK 5 是繼續添加新功能和平台支持的主要 SDK。

用戶端 SDK 3 JCE 提供者使用不屬於標準 JCE 規格一部分的自訂類別和 API。JCE 提供商的客戶端 SDK 5 對 JCE 規範抱怨，並且在某些區域與客戶端 SDK 3 向後不兼容。在遷移至用戶端 SDK 5 時，客戶應用程式可能需要進行變更。本節概述成功移轉所需的變更。

若要檢閱所有提供者的移轉指示，請參閱從用戶端 SDK 3 遷移至用戶端 SDK 5。

主題

- 通過解決突破性更改做好
- 遷移至用戶端 SDK 5
- 相關主題

通過解決突破性更改做好

檢閱這些重大變更，並相應地在開發環境中更新您的應用程式。

提供者類別和名稱已變更

發生了什麼變化	它在客戶端 SDK 3 中是什麼	它是什麼在客戶端 SDK 5	範例
提供者類別和名稱	呼叫用戶端 SDK 3 中的 JCE 提供者類別，`CaviumProvider` 並具有提供者名稱`Cavium`。	在用戶端 SDK 5 中，會呼叫「提供者」類別，`CloudHsmProvider` 並具有「提供者」名稱`CloudHSM`。	範例存放庫中提供了如何初始化`CloudHsmProvider` 物件的 AWS CloudHSM GitHub 範例。

顯式登錄已更改，隱式沒有

發生了什麼變化	它在客戶端 SDK 3 中是什麼	它是什麼在客戶端 SDK 5	範例
明確登入	用戶端 SDK 3 會使用`LoginManager` 類別進行明確登入[1]。	在用戶端 SDK 5 中，提`CloudHSM`供者會實`AuthProvider` 作明確登入。`AuthProvider` 是一個標準的 Java 類，並遵循 Java 的慣用方式登錄到提供程序。在 Client SDK 5 中改進了登錄狀態管理，應	如需如何透過用戶端開發套件 5 使用明確登入的範例，請參閱 AWS CloudHSM LoginRunner 範例儲存庫中的 GitHub 範例。

發生了什麼變化	它在客戶端 SDK 3 中是什麼	它是什麼在客戶端 SDK 5	範例
		用程序不再需要在重新連接[2]期間監視和執行登錄。	
隱式登入	隱含登入不需要變更。從客戶端 SDK 3 遷移到客戶端 SDK 5 時，相同的屬性文件和所有環境變量將繼續適用於隱式登錄。		如需如何搭配用戶端 SDK 5 使用隱含登入的範例，請參閱LoginRunner 範例存放庫中的 AWS CloudHSM GitHub 範例。

- [1] 用戶端 SDK 3 程式碼片段：

```
LoginManager lm = LoginManager.getInstance();

lm.login(partition, user, pass);
```

- [2] 客戶端 SDK 5 代碼片段：

```
// 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);
```

如需如何透過 Client SDK 5 使用明確登入的範例，請參閱LoginRunner 範例存放庫中的 AWS CloudHSM GitHub 範例。

金鑰產生已變更

發生了什麼變化	它在客戶端 SDK 3 中是什麼	它是什麼在客戶端 SDK 5	範例
金鑰產生	在用戶端 SDK 3 中，用`Cavium[Key-type]AlgorithmParameterSpec` 於指定金鑰產生參數。如需程式碼片段，請參閱註腳1。	在用戶端 SDK 5 中，用`KeyAttributesMap` 於指定金鑰產生屬性。如需程式碼片段，請參閱註腳2。	如需如何使用`KeyAttributesMap` 產生對稱金鑰的範例，請參閱 AWS CloudHSM Github Symmetric Keys 範例儲存庫中的範例。
產生金鑰配對	在用戶端 SDK 3 中，用`Cavium[Key-type]Algorithmpa rameterSpec` 於指定 key pair 產生參數。如需程式碼片段，請參閱註腳3。	在用戶端 SDK 5 中，`KeyPairAttributesMap` 用來指定這些參數。如需程式碼片段，請參閱註腳4。	如需如何使用`KeyAttributesMap` 來產生非對稱金鑰的範例，請參閱AsymmetricKeys 範例存放庫中的 AWS CloudHSM GitHub 範例。

- [1] 用戶端 SDK 3 金鑰產生程式碼片段：

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
CaviumAESKeyGenParameterSpec aesSpec = new CaviumAESKeyGenParameterSpec(
keySizeInBits,
keyLabel,
isExtractable,
isPersistent);
keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [2] 客戶端 SDK 5 密鑰生成代碼片段：

```
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] 客戶端 SDK 3 key pair 生成代碼片段：：

```
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] 客戶端 SDK 5 key pair 生成代碼片段：

```
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();
```

查找，刪除和引用鍵已更改

尋找已經產生的金鑰 AWS CloudHSM 需要使用 KeyStore. 用戶端 SDK 3 有兩 KeyStore 種類型：`Cavium`和`CloudHSM`. 用戶端 SDK 5 只有一 KeyStore 種類型：`CloudHSM`。

從到移動`CloudHSM` KeyStore 需`Cavium` KeyStore 要變更類 KeyStore 型。此外，用戶端 SDK 3 會使用金鑰控制點來參考金鑰，而用戶端 SDK 5 則使用金鑰標籤。產生的行為變更如下所示。

發生了什麼變化	它在客戶端 SDK 3 中是什麼	它是什麼在客戶端 SDK 5	範例
關鍵參考	在用戶端 SDK 3 中，應用程式會使用金鑰標籤或金鑰控制點來參考 HSM 中的金鑰。他們使用標籤 KeyStore 來查找鍵，或者他們使用控點和創建`CaviumKey` 對象。	在用戶端 SDK 5 中，應用程式可以如 AWS CloudHSM KeyStore 何使用 Java 類別使用按標籤尋找金鑰。若要依控制點尋找金鑰，請使用 AWS CloudHSM `KeyStoreWithAttributes` 與 AWS CloudHSM `KeyReferenceSpec` 。	
尋找多個項目	當使用`getEntry`、或搜尋金鑰時`getKey`，	使用 AWS CloudHSM `KeyStore`和`KeyStore`	

發生了什麼變化	它在客戶端 SDK 3 中是什麼	它是什麼在客戶端 SDK 5	範例
	如果getCertificate 中存在多個具有相同文字的項目 Cavium KeyStore，則只會傳回找到的第一個項目。	ithAttributes ，這個相同的情況將導致拋出異常。若要修正此問題，建議您使用 CloudHSM CLI 中的設定金鑰屬性命令為金鑰設定唯一標籤。或者使用KeyStoreWithAttributes#getKeys 返回與該密鑰匹配的所有鍵。	
尋找所有金鑰	您可以在用Util.findAllKeys() 戶端 SDK 3 中使用尋找 HSM 中的所有金鑰。	客戶端 SDK 5 使得通過使用KeyStoreWithAttributes 類更簡單，更有效地查找密鑰。如果可能，請快取金鑰以將延遲降至最低。如需詳細資訊，請參閱 有效管理應用程式中的金鑰。當您需要從 HSM 擷取所有金鑰時，請使KeyStoreWithAttributes#getKeys 用空白KeyAttributesMap 金鑰。	AWS CloudHSM Github 範例儲存庫中提供了使用KeyStoreWithAttributes 類別尋找金鑰的範例，並在中顯示程式碼片段1。

發生了什麼變化	它在客戶端 SDK 3 中是什麼	它是什麼在客戶端 SDK 5	範例
金鑰刪除	用戶端 SDK 3 用`Util.deleteKey()` 來刪除金鑰。	客戶端 SDK 5 中的`Key`對象實現了允許使用此`Destroyable` 接口的`destroy()` 方法刪除密鑰的接口。	您可以在 CloudHSM Github 範例儲存庫中找到顯示刪除金鑰功能的範例程式碼。中顯示了每個 SDK 的範例程式碼片段2。

- [1] 代碼片段如下所示：

```
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] 刪除客戶端 SDK 3 中的密鑰：

```
Util.deleteKey(key);
```

  刪除客戶端 SDK 5 中的密鑰：

```
((Destroyable) key).destroy();
```

密碼解包操作已更改，其他密碼操作沒有

> **Note**
>
> 無需更改密碼加密/解密/包裝操作。

解除換行作業需要將 Client SDK 3 `CaviumUnwrapParameterSpec` 類別取代為下列其中一個特定於密碼編譯作業的類別。

- GCMUnwrapKeySpec用於AES/GCM/NoPadding展開
- IvUnwrapKeySpec為AESWrap unwrap和 AES/CBC/NoPadding unwrap
- 適用於 RSA OAEP unwrap 的 OAEPUnwrapKeySpec

示例代碼片段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);
```

簽章作業未變更

簽名操作不需要進行任何變更。

遷移至用戶端 SDK 5

請遵循本節中的指示，從用戶端 SDK 3 遷移至用戶端 SDK 5。

> **Note**
>
> 用戶端軟體開發套件 5 目前不支援 Amazon Linux、Ubuntu 18.04 CentOS 6、CentOS 8 及 RHEL 6。如果您目前正在搭配用戶端 SDK 3 使用其中一個平台，則在移轉至用戶端 SDK 5 時，必須選擇不同的平台。

1. 解除安裝用戶端 SDK 3 的 JCE 提供者。

   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. 解除安裝用戶端 SDK 3 的用戶端常駐程式。

   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**
>
> 自訂組態需要再次啟用。

3. 依照中安裝並使用用戶端 SDK 5 的 AWS CloudHSM JCE 提供者的步驟安裝用戶端 SDK JCE 提供者。
4. 客戶端 SDK 5 引入了新的配置文件格式和命令行引導工具。若要啟動用戶端 SDK 5 JCE 提供者，請遵循使用者指南中列出的引導用戶端 SDK指示。
5. 在您的開發環境中，測試您的應用程式。在最終移轉之前，更新現有程式碼以解決您的重大變更。

相關主題

- 的最佳做法 AWS CloudHSM

適用於 JCE 的進階組態

AWS CloudHSM JCE 提供者包含下列進階組態，這些設定不屬於大多數客戶使用的一般組態。

- 連接至多個叢集
- 使用 JCE 擷取金鑰
- 重試適用於 JCE 的組態

使用 JCE 提供者連線到多個叢集

此組態可讓單一用戶端執行個體與多個叢集進行通訊。與單一執行個體僅能與單一叢集進行通訊相比，該組態可節省某些使用案例的成本。該`CloudHsmProvider`類 AWS CloudHSM是 Java 安全提供程序類的實現。這個類的每個實例代表到整個 AWS CloudHSM 集群的連接。您可以將此類別實例化，並將其新增至 Java 安全提供者清單中，以便您可以使用標準的 JCE 類別與其進行互動。

下面範例會將此類別實例化，並將其新增至 Java 安全提供者清單：

```
if (Security.getProvider(CloudHsmProvider.PROVIDER_NAME) == null) {
```

```
    Security.addProvider(new CloudHsmProvider());
}
```

CloudHsmProvider 組態

可以使用下列兩種方式來對 CloudHsmProvider 進行設定：

1. 使用檔案進行設定 (預設組態)
2. 使用程式碼設定

使用檔案進行設定 (預設組態)

當您使用預設建構函數實例化 CloudHsmProvider 時，根據預設，它會在 Linux 的 /opt/cloudhsm/etc/cloudhsm-jce.cfg 路徑中尋找組態檔案。可使用 configure-jce 設定此組態檔案。

使用預設的建構函數所建立的物件將會使用預設的 CloudHSM 提供者名稱 CloudHSM。提供者名稱有助於與 JCE 進行互動，以讓它了解各種作業應使用哪個提供者。以下為使用 CloudHSM 提供者名稱進行加密作業的範例：

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHSM");
```

使用程式碼設定

從用戶端 SDK 5.8.0 版開始，您還可以使用 Java 程式碼來設定 CloudHsmProvider。使用 CloudHsmProviderConfig 類別的物件可以進行設定。您可以使用 CloudHsmProviderConfigBuilder 建置此物件。

CloudHsmProvider 有另一個接受 CloudHsmProviderConfig 物件的建構函數，如下列範例顯示。

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);
```

在此範例中，JCE 提供者的名稱為 CloudHsmCluster1。這是應用程式可用來與 JCE 互動的名稱：

Example

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHsmCluster1");
```

或者，應用程式還可以使用上面建立的提供者物件，讓 JCE 了解要使用該提供者以進行作業：

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider);
```

如果未使用 withClusterUniqueIdentifier 方法指定唯一識別碼，則會為您建立一個隨機產生的提供者名稱。若要取得此隨機產生的識別碼，應用程式可以呼叫 provider.getName() 以取得識別碼。

連接至多個叢集

如上所述，每個 CloudHsmProvider 都代表與 CloudHSM 叢集的連線。如果您想要從同一個應用程式與另一個叢集進行通訊，您可以使用其他叢集的組態建立另一個物件 CloudHsmProvider，然後您可以使用提供者物件或使用提供者名稱與該叢集互動，如下列範例所顯示。

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);
}
```

設定上述兩個提供者 (兩個叢集) 後，您可以使用提供者物件或使用提供者名稱與其進行互動。

擴展此示例，顯示如何交談`cluster1`，您可以使用以下示例進行 AES/GCM/ NoPadding 操作：

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider1);
```

在同一個應用程序中使用提供者名稱在第二個叢集上產生「AES」金鑰，您還可以使用下列範例：

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider2.getName());
```

重試適用於 JCE 的命令

用戶端 SDK 5.8.0 及更新版本具有內建的自動重試策略，該策略將從用戶端重試 HSM 限流操作。當 HSM 因過於忙於執行先前操作而無法接受更多要求而限制操作時，用戶端 SDK 會嘗試重試限流操作 (最多 3 次)，同時以指數形式回退。此自動重試策略可以設定為兩種模式中的其中一種：關閉模式和標準模式。

- 關閉：用戶端 SDK 將不會針對 HSM 的任何限流操作執行任何重試政策。
- 標準：這是用戶端 SDK 5.8.0 及更新版本的預設模式。在此模式下，用戶端 SDK 會以指數回退形式自動重試限流操作。

如需詳細資訊，請參閱 HSM 調節。

將重試命令設定為關閉模式

Linux

將 Linux 上用戶端 SDK 5 重試命令設定為 off

- 您可以使用下列命令將重試組態設定為 off 模式：

```
$ sudo /opt/cloudhsm/bin/configure-jce --default-retry-mode off
```

Windows

將 Windows 上用戶端 SDK 5 重試命令設定為 off

- 您可以使用下列命令將重試組態設定為 off 模式：

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-jce.exe --default-retry-mode
 off
```

使用 JCE 擷取金鑰

Java 密碼編譯延伸 (JCE) 使用允許插入不同密碼編譯實作的架構。 AWS CloudHSM 提供一個這樣的 JCE 提供者，該提供者將密碼編譯作業卸載到 HSM。對於使用儲存在 AWS CloudHSM 中金鑰的其他大多數 JCE 提供者，他們必須將金鑰位元組以純文字形式從 HSM 擷取到機器的記憶體中以供使用。HSM 通常僅允許將金鑰擷取為包裝物件，而非純文字。但是，為了支持提供者間集成用例，AWS CloudHSM 允許選擇加入配置選項以啟用清除密鑰字節的提取。

> **Important**
>
> AWS CloudHSM 每當指定 AWS CloudHSM 提供者或使用 AWS CloudHSM 金鑰物件時，JCE 就會將操作卸載到。如果您預期在 HSM 內進行作業，則您不需要以純文字形式擷取金鑰。當您的應用程式因第三方程式庫或 JCE 提供者的限制而無法使用安全機制 (例如，包裝和取消包裝金鑰) 時，僅需要以純文字形式擷取金鑰。

AWS CloudHSM JCE 提供程序默認允許提取公鑰以使用外部 JCE 提供程序。一律允許使用下列方法：

類別	方法	Format (getEncoded)
EcPublicKey	getEncoded()	X.509
	getW()	N/A
RSA PublicKey	getEncoded()	X.509
	getPublicExponent()	N/A
CloudHsmRsaPrivateCrtKey	getPublicExponent()	N/A

AWS CloudHSM JCE 提供程序默認情況下不允許清除私鑰或秘密密鑰提取密鑰字節。如果您的使用案例需要，您可在下列情況下以純文字形式擷取私用金鑰或秘密金鑰的金鑰位元組：

1. 私用金鑰和秘密金鑰的 EXTRACTABLE 屬性設定為 true。
   - 根據預設，私用金鑰和秘密金鑰的 EXTRACTABLE 屬性設定為 true。EXTRACTABLE 金鑰是允許從 HSM 匯出的金鑰。如需詳細資訊，請參閱 Client SDK 5 支援的 Java 屬性。
2. 私用金鑰和私密金鑰的 WRAP_WITH_TRUSTED 屬性設定為 false。
   - getEncoded、getPrivateExponent 和 getS 不能與無法以純文字形式匯出的私用金鑰搭配使用。WRAP_WITH_TRUSTED 不允許您的私用金鑰以純文字形式匯出 HSM。如需詳細資訊，請參閱使用受信任的金鑰控制金鑰取消包裝。

允許 AWS CloudHSM JCE 提供商從中提取私鑰密鑰 AWS CloudHSM

> **Important**
>
> 變更此組態後，就可以從 HSM 叢集以純文字形式擷取所有 EXTRACTABLE 金鑰位元組。為了獲得更好的安全性，您應考慮使用金鑰包裝方法將金鑰安全地從 HSM 擷取出來。這可防止意外從 HSM 擷取金鑰位元組。

1. 使用下列命令，可在 JCE 中啟用您的私用金鑰或秘密金鑰：

   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. 啟用以純文字形式擷取金鑰後，即可啟用下列方法將私用金鑰擷取至記憶體。

類別	方法	Format (getEncoded)
金鑰	getEncoded()	RAW
EC PrivateKey	getEncoded()	PKCS#8
	getS()	N/A
RSA PrivateCrtKey	getEncoded()	X.509
	getPrivateExponent()	N/A
	getPrimeP()	N/A
	getPrimeQ()	N/A
	getPrimeExponentP ()	N/A
	getPrimeExponentQ ()	N/A
	getCrtCoefficient()	N/A

如果您想還原預設行為，且不允許 JCE 以純文字形式匯出金鑰，請執行下列命令：

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
```

適用於 Microsoft Windows 的 Cryptography API: Next Generation (CNG) 和金鑰儲存提供者 (KSP)

適 AWS CloudHSM 用於視窗的用戶端包含 CNG 和 KSP 提供者。目前，只有用戶端 SDK 3 支援 CNG 和 KSP 提供者。

金鑰儲存提供者 (KSP) 可讓您儲存和擷取金鑰。例如，如果您將 Microsoft Active Directory Certificate Services (AD CS) 角色新增到您的 Windows 伺服器，並選擇為憑證授權機制 (CA) 建立新的私有金鑰，您可以選擇 KSP 來管理金鑰儲存。當您設定 AD CS 角色時，您可以選擇 KSP。如需詳細資訊，請參閱 建立 Windows Server CA。

Cryptography API: Next Generation (CNG) 是一套專用於 Microsoft Windows 作業系統的密碼編譯 API。CNG 可讓開發人員使用密碼編譯技術來確保 Windows 應用程式的安全。在高層次上，CNG 的 AWS CloudHSM 實作提供下列功能：

- 密碼編譯基本指令：可讓您執行基本密碼編譯操作。
- 金鑰匯入和匯出：可讓您匯入和匯出對稱和非對稱金鑰。
- 資料保護 API (CNG DPAPI)：可讓您輕鬆地加密和解密資料。
- 金鑰儲存和擷取：可讓您安全地存放和隔離非對稱金鑰對的私有金鑰。

主題

- 驗證適用於 Windows 的 KSP 和 CNG 提供者
- 視窗 AWS CloudHSM 先決條
- 將 AWS CloudHSM 金鑰與憑證建立關聯
- CNG 提供者的程式碼範例

驗證適用於 Windows 的 KSP 和 CNG 提供者

當您安裝 Windows AWS CloudHSM 用戶端時，會安裝 KSP 和 CNG 提供者。您可以依照在 安裝用戶端 (Windows) 的步驟安裝用戶端。

設定和執行 Windows AWS CloudHSM 用戶端

若要啟動 Windows CloudHSM 用戶端，您必須先符合必要條件。然後更新提供者使用的組態檔案，然後完成以下步驟啟動用戶端。在您首次使用 KSP 和 CNG 提供者時和在叢集中新增或移除 HSM 之後，您需要執行以下步驟。如此一來， AWS CloudHSM 就能夠同步處理資料並維持叢集中所有 HSM 的一致性。

步驟 1：停止用 AWS CloudHSM 戶端

在您更新提供者使用的組態檔之前，請先停止用 AWS CloudHSM 戶端。如果用戶端已停止，執行停止命令並不會造成影響。

- 用於 Windows 用戶端 1.1.2+：

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- 用於 Windows 用戶端 1.1.1 和更早版本：

  在您啟動用 AWS CloudHSM 戶端的命令視窗中使用 Ctrl + C。

步驟 2：更新 AWS CloudHSM 配置文件

此步驟使用-a設定工具的 參數，將叢集的其中一個 HSM 的彈性網路界面 (ENI) IP 地址新增至組態檔案。

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

若要取得叢集中 HSM 的 ENI IP 位址，請瀏覽至 AWS CloudHSM 主控台，選擇叢集，然後選取所需的叢集。您也可以使用DescribeClusters作業、描述叢集命令或指令程式。Get-HSM2Cluster PowerShell僅輸入一個 ENI IP 地址。無論您使用哪個 ENI IP 地址都是如此。

步驟 3：啟動用 AWS CloudHSM 戶端

接下來，啟動或重新啟動 AWS CloudHSM 用戶端。用 AWS CloudHSM 戶端啟動時，會使用其組態檔案中的 ENI IP 位址來查詢叢集。接著會將叢集的所有 HSM 的 ENI IP 地址，新增至叢集資訊檔案。

- 用於 Windows 用戶端 1.1.2+：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 用於 Windows 用戶端 1.1.1 和更早版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

檢查 KSP 和 CNG 提供者

您可以使用以下命令，以判斷哪些提供者已安裝在您的系統上。這些命令列出已註冊的 KSP 和 CNG 提供者。不需要執行 AWS CloudHSM 用戶端。

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -enum
```

```
C:\Program Files\Amazon\CloudHSM>cng_config.exe -enum
```

如要驗證 KSP 和 CNG 提供者是否已安裝在 Windows Server EC2 執行個體上，您會在清單中看到下列項目：

```
Cavium CNG Provider
Cavium Key Storage Provider
```

如果缺少 CNG 提供者，請執行下列命令。

```
C:\Program Files\Amazon\CloudHSM>cng_config.exe -register
```

如果缺少 KSP 提供者，請執行下列命令。

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -register
```

視窗 AWS CloudHSM 先決條

您必須先設定系統上 HSM 的登入認證，才能啟動 Windows 用 AWS CloudHSM 戶端並使用 KSP 和 CNG 提供者。您可以透過 Windows 認證管理員或系統環境變數來設定登入資料。建議您使用 Windows 認證管理員來存放登入資料。此選項適用於用 AWS CloudHSM 戶端 2.0.4 及更新版本。使用環境變數較容易設定，但比起使用 Windows 認證管理員較不安全。

Windows 認證管理員

您可以使用 `set_cloudhsm_credentials` 公用程式或 Windows 認證管理員界面。

- 使用 **`set_cloudhsm_credentials`** 公用程式：

  `set_cloudhsm_credentials` 公用程式包含在您的 Windows 安裝程式中。您可以使用此公用程式，輕鬆地將 HSM 登入資料傳遞到 Windows 認證管理員。如果您想要從來源編譯此公用程式，您可以使用安裝程式中包含的 Python 程式碼。

  1. 前往 `C:\Program Files\Amazon\CloudHSM\tools\` 資料夾。
  2. 使用 CU 使用者名稱和密碼參數執行 `set_cloudhsm_credentials.exe` 檔案。

     ```
     set_cloudhsm_credentials.exe --username <CU USER> --password <CU PASSWORD>
     ```

- 使用認證管理員界面：

  您可以使用認證管理員界面來手動管理您的登入資料。

  1. 若要開啟認證管理員，請在工作列的搜尋方塊中輸入 `credential manager`，然後選取認證管理員。
  2. 選取 Windows 認證以管理 Windows 認證。
  3. 選取新增一般認證並填寫詳細資訊，如下所示：
     - 在網際網路或網路位址中，輸入目標名稱為 `cloudhsm_client`。
     - 在使用者名稱和密碼中，輸入 CU 登入資料。
     - 按一下 OK (確定)。

系統環境變數

您可以設定系統環境變數，用以識別 Windows 應用程式的 HSM 和 加密使用者 (CU)。您可以使用 setx 命令來設定系統環境變數，或以程式設計方式或是在 Windows 系統屬性控制台的進階標籤中設定永久性系統環境變數。

> **Warning**
>
> 當您透過系統環境變數設定登入資料時，密碼會在使用者的系統上以純文字提供。若要解決此問題，請使用 Windows 認證管理員。

設定下列系統環境變數：

n3fips_password=*CU USERNAME*:*CU PASSWORD*

識別 HSM 中的加密使用者 (CU)，並提供所有必要的登入資訊。您的應用程式會以這個 CU 的身分進行驗證和執行。這個應用程式具有此 CU 的許可，並僅可以檢視和管理該 CU 擁有和共用的金鑰。若要建立新 CU，請使用 createUser。若要尋找現有 CU，請使用 listUsers。

例如：

```
setx /m n3fips_password test_user:password123
```

將 AWS CloudHSM 金鑰與憑證建立關聯

您必須先將 AWS CloudHSM 金鑰匯入本機憑證存放區，並將中繼資料與憑證建立關聯 SignTool，才能搭配協力廠商工具 (例如 Microsoft) 使金鑰使用。若要匯入金鑰的中繼資料，請使用包含在 CloudHSM 3.0 以上版本中的 import_key.exe 公用程式。以下步驟提供額外資訊和範例輸出。

步驟 1：匯入您的憑證

在 Windows 中，您應可按兩下憑證，將其匯入至您的本機憑證存放區。

不過，如果按兩下無法執行，請使用 Microsoft Certreq 工具將憑證匯入至憑證管理員。例如：

```
certreq -accept certificatename
```

如果這個動作失敗，且您收到錯誤訊息 `Key not found`，請繼續執行步驟 2。如果憑證出現在金鑰存放區，就表示您已完成任務，不需要進一步的動作。

步驟 2：收集憑證識別資訊

如果上一個步驟沒有成功，您需要建立您的私密金鑰與憑證之間的關聯。不過，在您可以建立關聯之前，您必須先找到憑證的唯一容器名稱和序號。使用公用程式 (例如 certutil) 來顯示所需的憑證資訊。以下來自 certutil 的範例輸出顯示容器名稱和序號。

```
================ 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
```

步驟 3：將 AWS CloudHSM 私密金鑰與憑證建立關聯

若要將金鑰與憑證產生關聯，請先確定啟動用 AWS CloudHSM 戶端精靈。然後，使用 import_key.exe (包含在 CloudHSM 3.0 以上版本中)，建立私有金鑰與憑證之間的關聯。指定憑證時，請使用其簡易容器名稱。以下範例顯示命令和回應。此動作只會複製金鑰的中繼資料；金鑰會保留在 HSM 上。

```
$> import_key.exe -RSA CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c

Successfully opened Microsoft Software Key Storage Provider : 0NCryptOpenKey failed :
 80090016
```

步驟 4：更新憑證存放區

確定 AWS CloudHSM 用戶端常駐程式仍在執行中。然後，使用 certutil 動詞 -repairstore 來更新憑證序號。以下範例顯示命令和輸出。如需有關 -repairstore 動詞的資訊，請參閱 Microsoft 文件。

```
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.
```

更新憑證序號後，您可以在 Windows 上將此憑證和對應的 AWS CloudHSM 私密金鑰與任何協力廠商簽署工具搭配使用。

CNG 提供者的程式碼範例

⚠ ** 僅限範例程式碼 - 不適用於生產用途 **
此範例程式碼僅供說明之用。不要在生產環境中執行此程式碼。

下列範例顯示如何列舉系統上已註冊加密提供者的程式碼範例，以尋找與適用於 Windows 的 CloudHSM 用戶端一起安裝的 CNG 提供者。此範例還示範如何建立非對稱金鑰對，以及如何使用金鑰對來簽署資料。

⚠ Important

執行此範例之前，您必須先設定 HSM 登入資料，如先決條件中所述。如需詳細資訊，請參閱視窗 AWS CloudHSM 先決條。

```
// 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;
}
```

先前的用戶端 SDK (用戶端 SDK 3)

AWS CloudHSM 包括兩個主要的客戶端 SDK 版本：

- 用戶端 SDK 5：這是我們最新預設的用戶端 SDK。如需有關其優點和優勢的資訊，請參閱 用戶端 SDK 5 的優點 。
- 用戶端 SDK 3：這是我們較舊的用戶端 SDK。它包括一組完整的組件，用於基於平台和語言的應用程序兼容性和管理工具。

如需從用戶端 SDK 3 遷移至用戶端 SDK 5 的指示，請參閱從用戶端 SDK 3 遷移至用戶端 SDK 5。

本主題列出用戶端 SDK 3 說明文件。

若要下載，請參閱 下載。

檢查您的用戶端 SDK 版本

Amazon Linux

使用下列命令：

```
rpm -qa | grep ^cloudhsm
```

Amazon Linux 2

使用下列命令：

```
rpm -qa | grep ^cloudhsm
```

CentOS 6

使用下列命令：

```
rpm -qa | grep ^cloudhsm
```

CentOS 7

使用下列命令：

```
rpm -qa | grep ^cloudhsm
```

CentOS 8

使用下列命令：

```
rpm -qa | grep ^cloudhsm
```

RHEL 6

使用下列命令：

```
rpm -qa | grep ^cloudhsm
```

RHEL 7

使用下列命令：

```
rpm -qa | grep ^cloudhsm
```

RHEL 8

使用下列命令：

```
rpm -qa | grep ^cloudhsm
```

Ubuntu 16.04 LTS

使用下列命令：

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 18.04 LTS

使用下列命令：

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 20.04 LTS

使用下列命令：

```
apt list --installed | grep ^cloudhsm
```

Windows Server

使用下列命令：

```
wmic product get name,version
```

用戶端 SDK 元件比較

除了命令列工具之外，用戶端 SDK 3 還包含可從各種基於平台或語言的應用程式對 HSM 進行卸載密碼編譯操作的元件。用戶端 SDK 5 與用戶端 SDK 3 具有同位檢查，但它尚不支援 CNG 和 KSP 提供者。下表旨在比較用戶端 SDK 3 和用戶端 SDK 5 中的元件可用性。

元件	用戶端 SDK 5	用戶端 SDK 3
PKCS #11 程式庫	是	是
JCE 提供者	是	是
OpenSSL 動態引擎	是	是
CNG 和 KSP 提供者		是
CloudHSM 管理公用程式 (CMU) [1]	是	是
金鑰管理公用程式 (KMU) [1]	是	是
設定工具	是	是

[1] CMU 和 KMU 元件包含在 CloudHSM CLI 與用戶端 SDK 5 中。

主題

- 用戶端 SDK 3 支援的平台
- 在 Linux 上升級用戶端 SDK 3
- 適用於 SDK 3 的 PKCS #11 程式庫
- 安裝適用於 OpenSSL 動態引擎的用戶端 SDK 3
- 適用於 JCE 提供者的用戶端 SDK 3

用戶端 SDK 3 支援的平台

用戶端 SDK 3 需要用戶端常駐程式，並提供命令列工具，包括 CloudHSM 管理公用程式 (CMU)、金鑰管理公用程式 (KMU) 和設定工具。

每個版本的 AWS CloudHSM 客戶端 SDK 的基本支持都不同。SDK 中元件的平台通常支援與基礎支援相符，但並非總是如此。若要判斷特定元件的平台支援，請先確定您想要的平台出現在 SDK 的基礎區段中，然後在元件區段中檢查是否有任何排除項或任何其他相關資訊。

平台支援會隨時間變更。舊版 CloudHSM 用戶端 SDK 可能不支援此處列出的所有作業系統。使用版本說明來判斷舊版 CloudHSM 用戶端 SDK 的作業系統支援。如需詳細資訊，請參閱 AWS CloudHSM 用戶端 SDK 的下載。

AWS CloudHSM 僅支援 64 位元作業系統。

內容

Linux 支援

- 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] 不支援 OpenSSL 動態引擎。如需詳細資訊，請參閱 OpenSSL 動態引擎。

[2] 不支援用戶端 SDK 3.3.0 及更高版本。

[3] SDK 3.4 是 Ubuntu 16.04 上最後一個受支援的版本。

[4] SDK 3.4 是 CentOS 8.3+ 上最後一個受支援的版本。

Windows 支援

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

用戶端 SDK 3 的 HSM 相容性

hsm1. 中	hsm2 米。中
與客戶端版本 SDK 3.1.0 及更高版本兼容。	不支援。

元件支援

PKCS #11 程式庫

PKCS #11 程式庫是 Linux 唯一符合 Linux 基礎支援的元件。如需詳細資訊，請參閱 the section called “Linux 支援”。

CloudHSM 管理公用程式 (CMU)

CloudHSM 管理公用程式 (CMU) 命令列工具可協助加密主管管理 HSM 中的使用者。它包括可建立、刪除和列出使用者，以及變更使用者密碼的工具。如需詳細資訊，請參閱 CloudHSM 管理公用程式 (CMU)。

金鑰管理公用程式 (KMU)

金鑰管理公用程式 (KMU) 是一個命令列工具，可協助加密使用者 (CU) 管理硬體安全模組 (HSM) 上的金鑰。如需詳細資訊，請參閱 金鑰管理公用程式 (KMU)。

JCE 提供者

JCE 提供者是 Linux 唯一符合 Linux 基礎支援的元件。如需詳細資訊，請參閱 the section called "Linux 支援"。

- 需要 OpenJDK 1.8

OpenSSL 動態引擎

OpenSSL 動態引擎是 Linux 唯一不符合 Linux 基礎支援的元件。請參閱以下排除項目。

- 需要 OpenSSL 1.0.2 [f+]

不支援的平台：

- CentOS 8
- Red Hat Enterprise Linux (RHEL) 8
- Ubuntu 18.04 LTS

這些平台隨附的 OpenSSL 版本與用戶端 SDK 3 的 OpenSSL 動態引擎不相容。 AWS CloudHSM 使用適用於用戶端 SDK 5 的 OpenSSL 動態引擎支援這些平台。

CNG 和 KSP 提供者

CNG 和 KSP 提供者是與 Windows 基礎支援相符的唯一 Windows 元件。如需更多詳細資訊，請參閱 Windows 支援。

在 Linux 上升級用戶端 SDK 3

使用 AWS CloudHSM Client SDK 3.1 及更新版本時，用戶端精靈的版本和您安裝的任何元件都必須符合才能升級。對於所有 Linux 系統，您必須使用單一命令，以相同版本的 PKCS #11 程式庫、Java 密碼編譯延伸模組 (JCE) 提供者或 OpenSSL 動態引擎來批次升級用戶端常駐程式。這項需求不適用於 Windows 系統，因為 CNG 和 KSP 提供者的二進位檔案已經包含在用戶端常駐程式套件中。

檢查用戶端常駐程式版本

- 在基於 Red Hat 的 Linux 系統 (包括 Amazon Linux 和 CentOS) 上，使用以下命令：

```
rpm -qa | grep ^cloudhsm
```

- 在基於 Debian 的 Linux 系統上，請使用下列命令：

```
apt list --installed | grep ^cloudhsm
```

- 在 Windows 系統上，請使用下列命令：

```
wmic product get name,version
```

主題

必要條件

下載最新版本的 AWS CloudHSM 客戶端守護程序並選擇您的組件。

> **Note**
>
> 您不必安裝所有元件。對於已安裝的每個元件，您必須升級該元件以符合用戶端常駐程式的版本。

最新的 Linux 用戶端常駐程式

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
```

最新的 PKCS #11 程式庫

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
```

最新的 OpenSSL 動態引擎

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
```

最新的 JCE 提供者

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
```

步驟 1：停止用戶端常駐程式

使用以下命令來停止用戶端常駐程式。

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
```

步驟 2：升級用戶端 SDK

下列命令顯示升級用戶端常駐程式和元件所需的語法。執行命令之前，請先移除任何您不想升級的元件。

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>
```

步驟 3：啟動用戶端常駐程式

使用下列命令來啟動用戶端常駐程式。

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

目前尚未提供對 OpenSSL 動態引擎的支援。

適用於 SDK 3 的 PKCS #11 程式庫

PKCS #11 是在硬體安全模組 (HSM) 上執行密碼編譯作業的標準。

如需關於啟動載入的資訊，請參閱 連接至叢集。

主題

- 安裝用戶端 SDK 3 的 PKCS #11 程式庫安裝
- 對 PKCS #11 程式庫進行身分驗證 (用戶端 SDK 3)
- 支援的金鑰類型 (用戶端 SDK 3)
- 支援的機制 (用戶端 SDK 3)
- 支援的 API 操作 (用戶端 SDK 3)
- 支援的金鑰屬性 (用戶端 SDK 3)
- PKCS #11 程式庫 (用戶端 SDK 3) 的程式碼範例

安裝用戶端 SDK 3 的 PKCS #11 程式庫安裝

用戶端 SDK 3 的先決條件

PKCS #11 程式庫需要用 AWS CloudHSM 戶端。

如果您尚未安裝並設定 AWS CloudHSM 用戶端，請立即依照中的步驟執行安裝用戶端 (Linux)。安裝和設定用戶端之後，請使用以下命令來啟動用戶端。

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
```

安裝用戶端 SDK 3 的 PKCS #11 程式庫

下列命令會下載和安裝 PKCS #11 程式庫。

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
```

- 如果您安裝了 PKCS #11 程式庫的 EC2 執行個體中沒有安裝用戶端 SDK 3 的其他元件，則您必須啟動用戶端 SDK 3。您只需要在每個執行個體上使用用戶端 SDK 3 的元件執行一次。
- 您可於下列位置找到 PKCS #11 程式庫檔案：

  Linux 二進位檔案、組態指令碼、憑證和日誌檔案：

```
/opt/cloudhsm/lib
```

對 PKCS #11 程式庫進行身分驗證 (用戶端 SDK 3)

當使用 PKCS #11 程式庫時，您的應用程式會以特定加密使用者 (CU) 在 HSM 中執行。您的應用程式只能檢視和管理 CU 所擁有和共用的金鑰。您可以使用 HSM 中現有的 CU，或建立新的 CU。如需關於管理 CU 的資訊，請參閱使用 CloudHSM CLI 管理 HSM 使用者和使用 CloudHSM 管理公用程式 (CMU) 管理 HSM 使用者。

若要將 CU 指定給 PKCS #11 程式庫，請使用 PKCS #11 C_Login 函數的 pin 參數。對於 AWS CloudHSM，pin 參數具有以下格式：

```
<CU_user_name>:<password>
```

例如，以下命令會將 PKCS #11 程式庫 pin 設定給使用者名稱為 `CryptoUser` 和密碼為 `CUPassword123!` 的 CU。

```
CryptoUser:CUPassword123!
```

支援的金鑰類型 (用戶端 SDK 3)

PKCS #11 程式庫支援下列金鑰類型。

金鑰類型	描述
RSA	產生 2048 位元至 4096 位元的 RSA 金鑰，以 256 位元為單位遞增。
EC	使用 secp224r1 (P-224)、secp256r1 (P-256)、secp256k1 (Blockchain)、secp384r1 (P-384) 和 secp521r1 (P-521) 曲線產生金鑰。
AES	產生 128 位元、192 位元和 256 位元的 AES 金鑰。
DES3 (三重 DES)	產生 192 位元 DES3 金鑰。請參閱下列備註 1 查看即將進行的變更。
GENERIC_SECRET	產生 1 到 64 位元組的一般機密。

- [1] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於處於非 FIP 模式的叢集，在 2023 之後仍然允許使用該叢集。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

支援的機制 (用戶端 SDK 3)

PKCS #11 程式庫支援下列演算法：

- 加密和解密：AES-CBC、AES-CTR、AES-ECB、AES-GCM、DES3-CBC、DES3-ECB、RSA-OAEP 和 RSA-PKCS
- 簽署和驗證：RSA、HMAC 和 ECDSA；無論是否使用雜湊皆不影響
- 雜湊/摘要：SHA1、SHA224、SHA256、SHA384 和 SHA512
- 金鑰包裝：AES 金鑰包裝、[4] AES-GCM、RSA-AES 和 RSA-OAEP
- 金鑰派生：ECDH、[5] SP800-108 CTR KDF

PKCS #11 程式庫機制功能表

PKCS #11 程式庫與 PKCS #11 規格的 2.40 版相容。若要使用 PKCS #11 叫用加密功能，請利用指定機制呼叫函數。下表總結說明 AWS CloudHSM支援的函數和機制組合。

解譯支援的 PKCS #11 機制函數表

✓ 標記表示 AWS CloudHSM 支援該功能的機制。並不是 PKCS #11 規格中所列的所有可能函數都受到支援。✖ 標記表示尚 AWS CloudHSM 未支援給定函數的機制，即使 PKCS #11 標準允許它。空白儲存格表示 PKCS #11 標準不支援指定函數的機制。

支援的 PKCS #11 程式庫機制和函數

Mechanism	函數						
	產生金鑰或金鑰對	簽署和驗證	SR 和 VR	摘要	加密和解密	衍生金鑰	包裝和 UnWrap
CKM_RSA_PKCS_KEY_PAIR_GEN	✓						

Mechanism	函數						
CKM_RSA_X9_31_KEY_PAIR_GEN	✓[2]						
CKM_RSA_X_509		✓			✓		
CKM_RSA_PKCS 請參閱備註 8		✓[1]	✖		✓[1]		✓[1]
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]					

Mechanism	函數						
CKM_SHA1_RSA_PKCS_PSS		✓[3.2]					
CKM_SHA224_RSA_PKCS_PSS		✓[3.2]					
CKM_SHA256_RSA_PKCS_PSS		✓[3.2]					
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]					

Mechanism	函數						
CKM_ECDSA_SHA384		✓[3.2]					
CKM_ECDSA_SHA512		✓[3.2]					
CKM_ECDH1_DERIVE						✓[5]	
CKM_SP800_108_COUNTER_KDF						✓	
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 請參閱 備註 [8]	✓						

Mechanism	函數						
CKM_DES3_CBC 請參閱 備註 8					✓[3.3]		✖
CKM_DES3_CBC_PAD 請參閱 備註 8					✓		✖
CKM_DES3_ECB 請參閱 備註 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]					

Mechanism	函數						
CKM_SHA256				✓[3.1]			
CKM_SHA256_HMAC		✓[3.3]					
CKM_SHA384				✓[3.1]			
CKM_SHA384_HMAC		✓[3.3]					
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]

Mechanism	函數						
CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD							✓[7.1]
CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD							✓[7.1]

機制註釋

- [1] 僅限單一部分操作。
- [2] 此機制的功能與 CKM_RSA_PKCS_KEY_PAIR_GEN 機制完全相同，但在產生 p 和 q 時提供更強大的保證。
- [3.1] 根據客戶端 SDK 以不同的 AWS CloudHSM 方式進行散列。對於用戶端 SDK 3，我們執行雜湊的位置取決於資料大小以及您使用的是單一部分還是多部分操作。

  用戶端 SDK 3 中的單一部分操作

  用戶端 SDK 3 各機制的最大資料集大小如表 3.1 所列。整個雜湊會在 HSM 內計算。不支援大於 16KB 的資料。

  表 3.1，單一部分操作的資料集大小上限

Mechanism	資料大小上限
CKM_SHA_1	16296
CKM_SHA224	16264
CKM_SHA256	16296
CKM_SHA384	16232
CKM_SHA512	16232

多部分操作用戶端 SDK 3

雖支援大於 16 KB 的資料，但資料大小會決定雜湊的發生位置。小於 16 KB 的資料緩衝區會在 HSM 內進行雜湊處理。16 KB 和系統資料大小上限之間的緩衝區會在軟體本機雜湊處理。請記住：雜湊函數不需要加密密碼編譯，因此您可以在 HSM 之外安全地計算。

- [3.2] 根據客戶端 SDK 以不同的 AWS CloudHSM 方式進行散列。對於用戶端 SDK 3，我們執行雜湊的位置取決於資料大小以及您使用的是單一部分還是多部分操作。

  單一部分操作用戶端 SDK 3

  用戶端 SDK 3 各機制的最大資料集大小如表 3.2 所列。不支援大於 16KB 的資料。

  表 3.2，單一部分操作的資料集大小上限

Mechanism	資料大小上限
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

Mechanism	資料大小上限
CKM_ECDSA_SHA256	16296
CKM_ECDSA_SHA384	16232
CKM_ECDSA_SHA512	16232

多部分操作用戶端 SDK 3

雖支援大於 16 KB 的資料，但資料大小會決定雜湊的發生位置。小於 16 KB 的資料緩衝區會在 HSM 內進行雜湊處理。16 KB 和系統資料大小上限之間的緩衝區會在軟體本機雜湊處理。請記住：雜湊函數不需要加密密碼編譯，因此您可以在 HSM 之外安全地計算。

- [3.3] 當使用下列任一機制操作資料時，若資料緩衝區超過資料大小上限，該操作就會導致錯誤。對於這些機制，所有資料處理都必須在 HSM 內進行。下表列出每個機制設定的資料大小上限：

  表 3.3，最大資料集大小

Mechanism	資料大小上限
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] 當執行 AES-GCM 加密時，HSM 不會接受來自應用程式的初始化向量 (IV) 資料。請務必使用其產生的 IV。系統會將 HSM 所提供的 12 位元組 IV 寫入記憶體參考，該記憶體參考是由您提供的

CK_GCM_PARAMS 參數結構 pIV 元素所指向。為了確保使用者不會混淆，在初始化 AES-GCM 加密時，1.1.1 版和更新版本中的 PKCS # 11 開發套件會強制將該 pIV 指向歸零的緩衝區。

- [5] 僅限於用戶端 SDK 3。此機制的實作是為了支援 SSL/TLS 卸載案例，且在 HSM 內只會部分執行。使用此機制前，請參閱 PKCS #11 程序庫的已知問題 中的「問題：在 HSM 中 ECDH 金鑰衍生只會部分執行」。CKM_ECDH1_DERIVE 不支援 secp521r1 (P-521) 曲線。
- [6] 支援下列 CK_MECHANISM_TYPE 和 CK_RSA_PKCS_MGF_TYPE，作為 CKM_RSA_PKCS_OAEP 的 CK_RSA_PKCS_OAEP_PARAMS：
  - CKM_SHA_1 使用 CKG_MGF1_SHA1
  - CKM_SHA224 使用 CKG_MGF1_SHA224
  - CKM_SHA256 使用 CKG_MGF1_SHA256
  - CKM_SHA384 使用 CKM_MGF1_SHA384
  - CKM_SHA512 使用 CKM_MGF1_SHA512
- [7.1] 廠商定義的機制。為了能使用 CloudHSM 廠商定義的機制，編譯期間 PKCS #11 應用程式必須加入 /opt/cloudhsm/include/pkcs11t.h。

  CKM_CLOUDHSM_AES_GCM：這個專屬機制是標準 CKM_AES_GCM 程式設計更安全的替代方案。這將 HSM 生成的 IV 加入至加密文字的開頭，而不是將它寫回加密初始化期間提供的 CK_GCM_PARAMS 結構中。您可以搭配 C_Encrypt、C_WrapKey、C_Decrypt 和 C_UnwrapKey 函數搭配使用此機制。使用此機制時，CK_GCM_PARAMS 結構中的 pIV 變數必須設定為 NULL。與 C_Decrypt 和 C_UnwrapKey 搭配使用此機制時，IV 應至於取消包裝的加密文字之前。

  CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD：AES 金鑰包裝與 PKCS #5 填補

  CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD：AES 金鑰包裝，零填補

  如需有關 AES 金鑰包裝的其他資訊，請參閱 AES 金鑰包裝。
- [8] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於處於非 FIP 模式的叢集，在 2023 之後仍然允許使用該叢集。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

支援的 API 操作 (用戶端 SDK 3)

PKCS #11 程式庫支援下列 PKCS #11 API 操作。

- 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 (僅支援用戶端 SDK 3)
- C_SignRecoverInit (僅支援用戶端 SDK 3)
- C_SignUpdate
- C_UnWrapKey
- C_Verify
- C_VerifyFinal
- C_VerifyInit
- C_VerifyRecover (僅支援用戶端 SDK 3)
- C_VerifyRecoverInit (僅支援用戶端 SDK 3)
- C_VerifyUpdate
- C_WrapKey

支援的金鑰屬性 (用戶端 SDK 3)

金鑰物件可以是公有金鑰、私有金鑰或私密金鑰。系統會透過屬性來指定金鑰物件上允許的動作。金鑰物件建立時，即會一併建立屬性。在您使用 PKCS #11 程式庫時，我們會指派 PKCS #11 標準所指定的預設值。

AWS CloudHSM 不支援 PKCS #11 規格中列出的所有屬性。我們會符合所有支援屬性的規格，並在個別表格中列出這些屬性。

用來建立、修改或複製物件的加密函數 (如 C_CreateObject、C_GenerateKey、C_GenerateKeyPair、C_UnwrapKey 和 C_DeriveKey) 會採用屬性範本做為其中一個參數。如需在建立物件期間傳遞屬性範本的詳細資訊，請參閱透過 PKCS #11 程式庫產生金鑰範例。

PKCS #11 程式庫屬性解譯表

PKCS #11 程式庫表包含金鑰類型不同的屬性清單。它指示一個給定的屬性是否支持與 AWS CloudHSM特定的密鑰類型使用特定的加密功能時。

圖例：

- ✓ 表示 CloudHSM 支援特定金鑰類型的屬性。
- ✖ 表示 CloudHSM 不支援特定金鑰類型的屬性。
- R 表示特定金鑰類型的屬性值設定為唯讀模式。
- S 表示屬性較為敏感，因此無法透過 `GetAttributeValue` 讀取。
- 預設值欄位中的空白儲存格表示屬性沒有獲派指定預設值。

GenerateKeyPair

屬性	金鑰類型				預設值
	EC 私有金鑰	EC 公有金鑰	RSA 私有金鑰	RSA 公有金鑰	
CKA_CLASS	✓	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	✓	
CKA_LABEL	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	True
CKA_TOKEN	✓	✓	✓	✓	False

屬性	金鑰類型				預設值
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✗	✓	✗	✓	False
CKA_DECRYPT	✓	✗	✓	✗	False
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

屬性	金鑰類型				預設值
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	False
CKA_UNWRAP	✓	✖	✓	✖	False
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	✖	✖	✖	✖	

屬性	金鑰類型				預設值
CKA_COEFFICIENT	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	
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

屬性	金鑰類型			預設值
	AES	DES3	一般機密	
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	
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

屬性	金鑰類型			預設值
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	True
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
CKA_ALWAYS_SENSITIVE	✖	✖	✖	

屬性	金鑰類型			預設值
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	

屬性	金鑰類型			預設值
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

CreateObject

屬性	金鑰類型							預設值
	EC 私有金鑰	EC 公有金鑰	RSA 私有金鑰	RSA 公有金鑰	AES	DES3	一般機密	
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

屬性	金鑰類型							預設值
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
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✗	✓	✗	✓	✓	✓	False
CKA_SIGN_RECOVER	✗	✗	✓[3]	✗	✗	✗	✗	False
CKA_VERIFY	✗	✓	✗	✓	✓	✓	✓	False
CKA_VERIFY_RECOVER	✗	✗	✗	✓[4]	✗	✗	✗	
CKA_WRAP	✗	✗	✗	✓	✓	✓	✗	False
CKA_WRAP_TEMPLATE	✗	✓	✗	✓	✓	✓	✗	

屬性	金鑰類型							預設值
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓	False
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	✖	✖	✓	✖	✖	✖	✖	

屬性	金鑰類型							預設值
CKA_PRIME_2	✖	✖	✓	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✓	✖	✖	✖	✖	
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

屬性	金鑰類型					預設值
	EC 私有金鑰	RSA 私有金鑰	AES	DES3	一般機密	
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

屬性	金鑰類型					預設值
CKA_DESTROYABLE	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	✓	✓	False
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	✖	✖	✖	✖	✖	

屬性	金鑰類型					預設值
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	✖	✖	✖	✖	✖	

屬性	金鑰類型					預設值
CKA_CHECK_VALUE	R	R	R	R	R	

DeriveKey

屬性	金鑰類型			預設值
	AES	DES3	一般機密	
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

屬性	金鑰類型			預設值
CKA_DESTROYABLE	✓[1]	✓[1]	✓[1]	True
CKA_SIGN	✓	✓	✓	False
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	✖	✖	✖	

屬性	金鑰類型			預設值
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]	

屬性	金鑰類型			預設值
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

屬性	金鑰類型						
	EC 私有金鑰	EC 公有金鑰	RSA 私有金鑰	RSA 公有金鑰	AES	DES3	一般機密
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	✖	✖	✓	✖	✓	✓	✖

屬性	金鑰類型						
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓
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	✖	✖	✓	✖	✓	✓	✖

屬性	金鑰類型						
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓
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	✖	✖	✖	✖

屬性	金鑰類型						
CKA_EXPONENT_2	✖	✖	S	✖	✖	✖	✖
CKA_PRIVATE_EXPONENT	✖	✖	S	✖	✖	✖	✖
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
CKA_VALUE	S	✖	✖	✖	✓[2]	✓[2]	✓[2]
CKA_VALUE_LEN	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	✖

屬性註釋

- [1] 此屬性受韌體部分支援，且需明確設定為僅限預設值。
- [2] 必要屬性。
- [3] 僅限於用戶端 SDK 3。CKA_SIGN_RECOVER 屬性由 CKA_SIGN 屬性衍生而來。若已設定，該屬性就僅能設定為與 CKA_SIGN 相同的值。若未設定，則該屬性會衍生 CKA_SIGN 的預設值。CloudHSM 只支援以 RSA 為基礎的可復原簽章機制，因此屬性目前僅適用於 RSA 公有金鑰。
- [4] 僅限於用戶端 SDK 3。CKA_VERIFY_RECOVER 屬性由 CKA_VERIFY 屬性衍生而來。若已設定，該屬性就僅能設定為與 CKA_VERIFY 相同的值。若未設定，則該屬性會衍生 CKA_VERIFY 的

預設值。CloudHSM 只支援以 RSA 為基礎的可復原簽章機制，因此此屬性目前僅適用於 RSA 公有金鑰。

修改屬性

有些物件屬性可在物件建立後進行修改，但有些不行。若要修改屬性，請使用來自 cloudhsm_mgmt_util 的 setAttribute 命令。您也可以使用來自 cloudhsm_mgmt_util 的 listAttribute 命令，來衍生屬性和代表這些屬性的常數清單。

下列清單會顯示物件建立後可修改的屬性：

- CKA_LABEL
- CKA_TOKEN

> **Note**
>
> 只有在將工作階段金鑰變更為符記金鑰時，才允許進行修改。使用 key_mgmt_util 中的 setAttribute 命令來變更屬性值。

- CKA_ENCRYPT
- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

> **Note**
>
> 這個屬性支援金鑰衍生。所有公有金鑰的屬性須為 False，不能設定為 True。如果是私密金鑰和 EC 私有金鑰，則該屬性可設定為 True 或 False。

- CKA_TRUSTED

> **Note**
>
> 唯有加密管理員 (CO) 可將這個屬性設定成 `True` 或 `False`。

- `CKA_WRAP_WITH_TRUSTED`

> **Note**
>
> 將此屬性套用於可匯出的資料金鑰，以表明只能使用標記為 `CKA_TRUSTED` 的金鑰包裝此金鑰。一旦設定 `CKA_WRAP_WITH_TRUSTED` 為 true，屬性就會變成唯讀，而且您無法變更或移除屬性。

解譯錯誤代碼

若在範本中指定特定金鑰不支援的屬性，就會導致錯誤。下表包含違反規格時所產生的錯誤代碼：

錯誤代碼	Description
`CKR_TEMPLATE_INCONSISTENT`	當您在屬性範本中指定的屬性符合 PKCS #11 規格，卻不受 CloudHSM 支援時，就會收到此錯誤。
`CKR_ATTRIBUTE_TYPE_INVALID`	當您擷取的屬性值符合 PKCS #11 規格，卻不受 CloudHSM 支援時，就會收到此錯誤。
`CKR_ATTRIBUTE_INCOMPLETE`	當您沒有在屬性範本中指定必要屬性時，就會收到此錯誤。
`CKR_ATTRIBUTE_READ_ONLY`	當您在屬性範本中指定唯讀屬性時，就會收到此錯誤。

PKCS #11 程式庫 (用戶端 SDK 3) 的程式碼範例

上的程式碼範例說 GitHub 明如何使用 PKCS #11 程式庫完成基本工作。

範本程式碼先決條件

執行範例之前，請執行以下步驟來設定您的環境：

- 安裝並設定適用於用戶端 SDK 3 的 PKCS #11 程式庫。
- 設定密碼編譯使用者 (CU)。您的應用程式會使用此 HSM 帳戶在 HSM 上執行程式碼範例。

程式碼範例

PKCS #11 AWS CloudHSM 軟體程式庫的程式碼範例可在上取得。GitHub此儲存庫包括如何使用 PKCS #11 執行一般作業的範例，包括加密、解密、簽署和驗證。

- 產生金鑰 (AES、RSA、EC)
- 列出金鑰屬性
- 使用 AES GCM 加密和解密資料
- 使用 AES_CTR 加密和解密資料
- 使用 3DES 加密和解密資料
- 使用 RSA 簽署和驗證資料
- 使用 HMAC KDF 衍生金鑰
- 使用以 PKCS #5 填補的 AES 包裝和取消包裝金鑰
- 使用不填補的 AES 包裝和取消包裝金鑰
- 使用零填補的 AES 包裝和取消包裝金鑰
- 使用 AES-GCM 包裝和取消包裝金鑰
- 使用 RSA 包裝和取消包裝金鑰

安裝適用於 OpenSSL 動態引擎的用戶端 SDK 3

用戶端 SDK 3 確實需要用戶端常駐程式才能連線到叢集。其支援：

- 產生 2048 位元、3072 位元和 4096 位元金鑰的 RSA 金鑰
- RSA 簽署/驗證。
- RSA 加密/解密。
- 產生隨機數字，經過安全性密碼編譯並通過 FIPS 驗證。

主題

使用用戶端 SDK 3 的 OpenSSL 動態引擎的先決條件

如需關於平台支援的詳細資訊，請參閱 用戶端 SDK 3 支援的平台。

在您可以使用 OpenSSL 的 AWS CloudHSM 動態引擎之前，您需要用 AWS CloudHSM 戶端。

用戶端是與叢集中的 HSM 建立 end-to-end 加密通訊的精靈，而 OpenSSL 引擎會在本機與用戶端進行通訊。若要安裝和設定用 AWS CloudHSM 戶端，請參閱安裝用戶端 (Linux)。使用下列命令來啟動。

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
```

安裝適用於用戶端 SDK 3 的 OpenSSL 動態引擎

下列步驟說明如何安裝和設定 OpenSSL 的 AWS CloudHSM 動態引擎。如需關於升級的詳細資訊，請參閱 升級用戶端 SDK 3。

安裝和設定 OpenSSL 引擎

1. 使用以下命令來下載和安裝 OpenSSL 引擎。

   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
```

應將 OpenSSL 引擎安裝在 `/opt/cloudhsm/lib/libcloudhsm_openssl.so`。

2. 使用下列命令來設定名為 `n3fips_password` 的環境變數，其中包含加密使用者 (CU) 的登入資料。

```
$ export n3fips_password=<HSM user name>:<password>
```

使用適用於用戶端 SDK 3 的 OpenSSL 動態引擎

若要從 OpenSSL 整合的應用程式使用 OpenSSL 的 AWS CloudHSM 動態引擎，請確定您的應用程式使用名為的 OpenSSL 動態引擎。`cloudhsm`動態引擎的共用程式庫位於 `/opt/cloudhsm/lib/libcloudhsm_openssl.so`。

若要從 OpenSSL 命令列使用 OpenSSL 的 AWS CloudHSM 動態引擎，請使用`-engine`選項來指定名為的 OpenSSL 動態引擎。`cloudhsm`例如：

```
$ openssl s_server -cert server.crt -key server.key -engine cloudhsm
```

適用於 JCE 提供者的用戶端 SDK 3

AWS CloudHSM JCE 提供者是根據 Java 密碼編譯延伸 (JCE) 提供者架構建置的提供者實作。JCE 可讓您使用 Java 開發套件 (JDK) 執行密碼編譯操作。在本指南中， AWS CloudHSM JCE 提供者有時稱為 JCE 提供者。使用 JCE 提供者和 JDK 將密碼編譯操作卸載至 HSM。

主題

- 安裝並使用用戶端 SDK 3 的 AWS CloudHSM JCE 提供者
- 用戶端 SDK 3 支援的機制
- 用戶端 SDK 3 的支援 Java 金鑰屬性
- 适用于客戶端 SDK 3 的 Java AWS CloudHSM 軟件庫的代碼示例
- 如 AWS CloudHSM KeyStore 何使用 Java 類別作為客戶端 SDK 3

安裝並使用用戶端 SDK 3 的 AWS CloudHSM JCE 提供者

在您可以使用 JCE 提供者之前，您需要用 AWS CloudHSM 戶端。

用戶端是與叢集中的 HSM 建立 end-to-end 加密通訊的精靈。JCE 提供者會在本機與用戶端進行通訊。如果您尚未安裝並設定用 AWS CloudHSM 戶端套件，請立即依照下列步驟執行安裝用戶端 (Linux)。安裝和設定用戶端之後，請使用以下命令來啟動用戶端。

僅 Linux 和相容的作業系統才支援 JCE 提供者。

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
```

主題

- JCE 提供者中的金鑰管理基礎知識

安裝 JCE 提供者

使用下列命令下載和安裝 JCE 提供者。僅 Linux 和相容的作業系統支援此提供者。

> **Note**
>
> 如需升級，請參閱 升級用戶端 SDK 3。

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
```

當您執行上述命令後，您可以找到以下 JCE 提供者檔案：

- /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

驗證安裝

HSM 上執行基本操作以驗證安裝。

驗證 JCE 提供者安裝

1. (選用) 如果您的環境中尚未安裝 Java，請使用下列命令來安裝。

   Linux (and compatible libraries)

   ```
   $ sudo yum install java-1.8.0-openjdk
   ```

   Ubuntu

   ```
   $ sudo apt-get install openjdk-8-jre
   ```

2. 使用下列命令來設定必要的環境變數。以加密使用者 (CU) 的登入資料取代 *<HSM user name>* 和 *<password>*。

   ```
   $ export LD_LIBRARY_PATH=/opt/cloudhsm/lib
   ```

   ```
   $ export HSM_PARTITION=PARTITION_1
   ```

   ```
   $ export HSM_USER=<HSM user name>
   ```

   ```
   $ export HSM_PASSWORD=<password>
   ```

3. 請使用以下命令執行基本功能測試。如果成功執行，您會看到類似如下的命令輸出。

   ```
   $ 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)
```

提供憑證給 JCE 提供者

HSM 需要先驗證您的 Java 應用程式，應用程式才能使用這些 HSM。每個應用程式可以使用一個工作階段。HSM 會使用明確登入或隱含登入方法，驗證工作階段。

明確登入：此方法可讓您直接在應用程式中提供 CloudHSM 憑證。它使用 `LoginManager.login()` 方法；您可在其中傳遞 CU 使用者名稱、密碼和 HSM 分割區 ID。如需使用明確登入方法的詳細資訊，請參閱登入 HSM 的程式碼範例。

隱含登入：此方法可讓您在新的屬性檔案、系統屬性或環境變數中設定 CloudHSM 憑證。

- 新的屬性檔案：建立名稱為 `HsmCredentials.properties` 的新檔案，並將其新增到您應用程式的 CLASSPATH。檔案應包含以下內容：

```
HSM_PARTITION = PARTITION_1
HSM_USER = <HSM user name>
HSM_PASSWORD = <password>
```

- 系統屬性：執行應用程式時，您可以透過系統屬性設定憑證。以下範例示範兩種不同的作法：

```
$ 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>");
```

- 環境變數：將憑證設定為環境變數。

```
$ export HSM_PARTITION=PARTITION_1
$ export HSM_USER=<HSM user name>
$ export HSM_PASSWORD=<password>
```

如果應用程式未提供登入資料，或者，如果您在 HSM 驗證工作階段之前嘗試操作，就可能無法使用登入資料。在這些情況下，適用於 Java 的 CloudHSM 軟體程式庫會按照以下順序，搜尋登入資料：

1. `HsmCredentials.properties`
2. 系統屬性
3. 環境變數

錯誤處理

使用明確登入會比隱含登入方法更容易處理錯誤。當您使用 `LoginManager` 類別時，您比較能控制應用程式如何處理失敗情況。隱含登入方法會讓處理錯誤時難以判斷到底是登入資料無效，或是 HSM 在驗證工作階段時發生問題。

JCE 提供者中的金鑰管理基礎知識

JCE 提供者的金鑰管理基本概念包括匯入金鑰、匯出金鑰、透過控制代碼載入金鑰或刪除金鑰。如需管理金鑰的詳細資訊，請參閱管理金鑰的程式碼範例。

您也可以在 程式碼範例 中尋找更多 JCE 提供者程式碼範例。

用戶端 SDK 3 支援的機制

如需支援之 Java 密碼編譯架構 (JCA) 介面和引擎類別的相關資訊 AWS CloudHSM，請參閱下列主題。

主題

支援的金鑰

Java AWS CloudHSM 軟體程式庫可讓您產生下列金鑰類型。

- AES：128 位元、192 位元和 256 位元 AES 金鑰。
- DESede：92 位元 3DES 金鑰。請參閱下列備註 1 查看即將進行的變更。
- NIST 曲線 secp256r1 (P-256)、secp384r1 (P-384) 和 secp256k1 (區塊鏈) 的 ECC 金鑰對。
- RSA：2048 位元至 4096 位元 RSA 金鑰，以 256 位元為單位遞增。

除了標準參數，我們也支援產生的每個金鑰使用下列參數。

- Label：金鑰標籤，您可以使用來搜尋金鑰。
- isExtractable ：指示是否可以從 HSM 匯出該金鑰。
- isPersistent ：指出在目前工作階段結束時，HSM 上是否保留金鑰。

> **Note**
>
> Java 程式庫 3.1 版更詳細地提供指定參數的能力。如需詳細資訊，請參閱支援的 Java 屬性。

受支援的密碼

Java 的 AWS CloudHSM 軟體程式庫支援下列演算法、模式和填補組合。

演算法	Mode	填補	備註
AES	CBC	AES/CBC/NoPadding AES/CBC/PKCS5Padding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。
AES	ECB	AES/ECB/NoPadding AES/ECB/PKCS5Padding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。使用轉型 AES。
AES	CTR	AES/CTR/NoPadding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。
AES	GCM	AES/GCM/NoPadding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 、Cipher.WRAP_MODE 和 Cipher.UNWRAP_MODE 。 執行 AES-GCM 加密時，HSM 會忽略請求中的初始化向量 (IV)，並使用自己產生的 IV。操作完成後，您必須呼叫 Cipher.getIV() 以取得 IV。
AESWrap	ECB	AESWrap/ECB/ZeroPadding	實作 Cipher.WRAP_MODE 和

演算法	Mode	填補	備註
		AESWrap/ECB/NoPadding AESWrap/ECB/PKCS5Padding	Cipher.UNWRAP_MODE 。使用轉型 AES。
DESede (三重 DES)	CBC	DESede/CBC/NoPadding DESede/CBC/PKCS5Padding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。 金鑰產生常式接受 168 或 192 位元的大小。不過，在內部所有 DESede 金鑰都是 192 位元。 請參閱下列備註 1 查看即將進行的變更。
DESede (三重 DES)	ECB	DESede/ECB/NoPadding DESede/ECB/PKCS5Padding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。 金鑰產生常式接受 168 或 192 位元的大小。不過，在內部所有 DESede 金鑰都是 192 位元。 請參閱下列備註 1 查看即將進行的變更。

演算法	Mode	填補	備註
RSA	ECB	RSA/ECB/NoPadding RSA/ECB/PKCS1Padding	實作 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。 請參閱下列備註 1 查看即將進行的變更。
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	實作 Cipher.ENCRYPT_MODE 、Cipher.DECRYPT_MODE 、Cipher.WRAP_MODE 和 Cipher.UNWRAP_MODE 。 OAEPPadding 是 OAEP，且填補類型是 SHA-1。

演算法	Mode	填補	備註
RSAAESWrap	ECB	OAEPPADDING	實作 Cipher.WRAP_Mode 和 Cipher.UNWRAP_MODE 。

支援的摘要

Java 的 AWS CloudHSM 軟體程式庫支援下列訊息摘要。

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

> **Note**
>
> 長度在 16 KB 下的資料是在 HSM 上進行雜湊處理，而較大的資料則是在本機軟體中進行雜湊處理。

支援的雜湊訊息驗證碼 (HMAC) 演算法

適用於 Java 的 AWS CloudHSM 軟體程式庫支援下列 HMAC 演算法。

- HmacSHA1
- HmacSHA224
- HmacSHA256
- HmacSHA384
- HmacSHA512

支援的登入/驗證機制

Java AWS CloudHSM 軟體程式庫支援下列類型的簽章和驗證。

RSA 簽章類型

- NONEwithRSA
- SHA1withRSA
- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA
- SHA1withRSA/PSS
- SHA224withRSA/PSS
- SHA256withRSA/PSS
- SHA384withRSA/PSS
- SHA512withRSA/PSS

ECDSA 簽章類型

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

機制註釋

[1] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於非 FIP 模式下的叢集，在 2023 之後仍然允許使用。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

用戶端 SDK 3 的支援 Java 金鑰屬性

本主題說明如何使用 Java 程式庫 3.1 版的專屬擴充功能來設定金鑰屬性。請在以下操作期間，使用此擴充功能來設定支援的金鑰屬性及其值：

- 金鑰產生
- 金鑰匯入

- 金鑰取消包裝

> **Note**
>
> 設定自訂金鑰屬性的擴充功能是選用功能。如果您具有在 Java 程式庫 3.0 版本中運作的程式碼，則不需要修改該程式碼。您建立的金鑰會繼續包含與之前相同的屬性。

主題

了解屬性

您可以使用金鑰屬性來指定金鑰物件上允許的動作，包括公有金鑰、私有金鑰或秘密金鑰。您可以在建立金鑰物件操作期間定義金鑰屬性和值。

但是，Java Cryptography Extension (JCE) 未指定應該如何設定金鑰屬性的值，因此預設情況下會允許大部分動作。相反地，PKCS#11 標準定義了一組具有更嚴格預設值的完整屬性。從 Java 程式庫 3.1 版開始，CloudHSM 提供專屬的擴充功能，可讓您為常用的屬性設定更嚴格的值。

支援的屬性

您可以設定下表中列出之屬性的值。最佳實務是，只為您希望更具有限制性的屬性設定值。如果您未指定值，CloudHSM 會使用下表中指定的預設值。預設值欄位中的空白儲存格表示屬性沒有獲派指定預設值。

屬性	預設值			備註
	對稱金鑰	金鑰對中的公有金鑰	金鑰對中的私有金鑰	
CKA_TOKEN	FALSE	FALSE	FALSE	永久金鑰，會跨叢集中的所有 HSM 進行複

屬性	預設值			備註
				寫並包含在備份中。CKA_TOKEN = FALSE 代表工作階段金鑰，該金鑰只被載入到一個 HSM 上，並在與 HSM 的連接中斷時自動擦除。
CKA_LABEL				使用者定義的字串。它可讓您輕鬆識別 HSM 上的金鑰。
CKA_EXTRACTABLE	TRUE		TRUE	True 表示您可以從 HSM 匯出此金鑰。
CKA_ENCRYPT	TRUE	TRUE		True 表示您可以使用金鑰來加密任何緩衝區。
CKA_DECRYPT	TRUE		TRUE	True 表示您可以使用金鑰來解密任何緩衝區。對於其 CKA_WRAP 設定為 true 的金鑰，通常將此設定為 FALSE。

屬性	預設值			備註
CKA_WRAP	TRUE	TRUE		True 表示您可以使用該金鑰來包裝另一個金鑰。對於私有金鑰，您通常將此設定為 FALSE。
CKA_UNWRAP	TRUE		TRUE	True 表示您可以使用金鑰來取消包裝 (匯入) 另一個金鑰。
CKA_SIGN	TRUE		TRUE	True 表示您可以使用金鑰來簽署訊息摘要。對於公有金鑰和您已封存的私有金鑰，這通常會設定為 FALSE。
CKA_VERIFY	TRUE	TRUE		True 表示您可以使用金鑰來驗證簽章。對於私有金鑰，這通常被設定為 FALSE。

屬性	預設值			備註
CKA_PRIVATE	TRUE	TRUE	TRUE	True 表示在使用者通過驗證之前，使用者可能無法存取金鑰。為了清楚起見，使用者在通過驗證之前無法存取 CloudHSM 上的任何金鑰，即使此屬性設為 FALSE 也一樣。

> **Note**
>
> 您可以在 PKCS #11 程式庫中獲得更廣泛的屬性支援。如需詳細資訊，請參閱支援的 PKCS #11 屬性。

設定金鑰的屬性

`CloudHsmKeyAttributesMap` 是類似 Java 映射的物件，您可以使用它來設定金鑰物件的屬性值。`CloudHsmKeyAttributesMap` 函數的方法類似於用於 Java 映射處理的方法。

若要設定屬性的自訂值，您有兩個選項：

- 使用下表中列出的方法
- 使用本文稍後示範的產生器模式

屬性映射物件支援以下列方法設定屬性：

作業	傳回值	**CloudHSMKeyAttributesMap** 方法
獲取現有金鑰的金鑰屬性的值	物件 (包含值) 或 null	get(keyAttribute)

作業	傳回值	**CloudHSMKeyAttributesMap** 方法
填入一個金鑰屬性的值	與金鑰屬性相關聯的先前值，如果金鑰屬性沒有映射，則為 null	put(keyAttribute, value)
填入多個金鑰屬性的值	N/A	putAll () keyAttributesMap
從屬性映射中刪除金鑰值對	與金鑰屬性相關聯的先前值，如果金鑰屬性沒有映射，則為 null	remove(keyAttribute)

> **Note**
>
> 您未明確指定的任何屬性都會設定為the section called “支援的屬性”中上表所列的預設值。

產生器模式範例

開發人員通常會發現，透過產生器模式利用類別更加方便。舉例來說：

```
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();
```

開發人員也可以利用預先定義的屬性集，作為在金鑰範本中強制執行最佳實務的便利方式。舉例來說：

```
//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
```

設定金鑰對的屬性

使用 Java 類別 CloudHsmKeyPairAttributesMap 來處理金鑰對的金鑰屬性。CloudHsmKeyPairAttributesMap 封裝了兩個 CloudHsmKeyAttributesMap 物件；一個用於公有金鑰，另一個用於私有金鑰。

若要分別為公有金鑰和私有金鑰設定個別屬性，您可以在該金鑰對應的 CloudHsmKeyAttributes 映射物件上使用 put() 方法。使用 getPublic() 方法來擷取公有金鑰的屬性映射，以及使用 getPrivate() 來擷取私有金鑰的屬性映射。使用 putAll() 與金鑰對屬性映射作為引數，填入公有金鑰和私有金鑰對的多個金鑰屬性值。

產生器模式範例

開發人員通常會發現，透過產生器模式來設定金鑰屬性更加方便。例如：

```
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**
>
> 如需有關此專屬延伸功能的詳細資訊，請參閱 Javadoc 封存檔和上的範例。 GitHub若要瀏覽 Javadoc，請下載並展開此封存。

整合練習

若要使用金鑰操作指定金鑰屬性，請依照下列步驟執行：

1. 執行個體化對稱金鑰的 `CloudHsmKeyAttributesMap` 或金鑰對的 `CloudHsmKeyPairAttributesMap`。
2. 使用必要的金鑰屬性和值來定義步驟 1 的屬性物件。
3. 執行個體化對應至特定金鑰類型的 `Cavium*ParameterSpec` 類別，並在其建構函數中傳遞此設定好的屬性物件。
4. 將此 `Cavium*ParameterSpec` 物件傳遞到相應的加密類別或方法中。

如需參考，下表包含支援自定義金鑰屬性的 Cavium*ParameterSpec 類別和方法。

金鑰類型	參數規格類別	範例建構函數
基本類別	CaviumKeyGenAlgorithmParameterSpec	CaviumKeyGenAlgorithmParameterSpec(CloudHsmKeyAttributesMap keyAttributesMap)
DES	CaviumDESKeyGenParameterSpec	CaviumDESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap keyAttributesMap)
RSA	CaviumRSAKeyGenParameterSpec	CaviumRSAKeyGenParameterSpec(int keysize, BigInteger publicExponent, CloudHsmKeyPairAttributesMap keyPairAttributesMap)
秘密	CaviumGenericSecretKeyGenParameterSpec	CaviumGenericSecretKeyGenParameterSpec(int size, CloudHsmKeyAttributesMap keyAttributesMap)
AES	CaviumAESKeyGenParameterSpec	CaviumAESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttribu

金鑰類型	參數規格類別	範例建構函數
		tesMap key AttributesMap)
EC	CaviumECGenParameterSpec	CaviumECGenParameterSpec(String stdName, CloudHsmKeyPairAttributesMap keyPairAttributesMap)

範本程式碼：產生並包裝金鑰

這些簡短的程式碼範例示範兩個不同操作的步驟：金鑰產生和金鑰包裝：

```
// 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);
```

适用于客戶端 SDK 3 的 Java AWS CloudHSM 軟件庫的代碼示例

必要條件

執行範例之前，您必須先設定環境：

- 安裝並設定 Java 密碼編譯延伸模組 (JCE) 提供者和 AWS CloudHSM 用戶端套件。
- 設定有效的 HSM 使用者名稱和密碼。加密使用者 (CU) 許可足夠執行這些任務。在每個範例中，您的應用程式會使用這些登入資料來登入 HSM。
- 決定如何提供憑證給 JCE 提供者。

程式碼範例

下列程式碼範例會示範如何使用 AWS CloudHSM JCE 提供者來執行基本任務。更多程式碼範例可在上取得GitHub。

- 登入 HSM
- 管理金鑰
- 產生 AES 金鑰
- 使用 AES-GCM 進行加密和解密
- 使用 AES-CTR 進行加密和解密
- 使用 D3DES-ECB 加密和解密[請參閱備註 1]
- 使用 AES-GCM 包裝和取消包裝金鑰
- 使用 AES 包裝和取消包裝金鑰
- 使用 RSA 包裝和取消包裝金鑰
- 使用支援的金鑰屬性
- 列舉金鑰存放區中的金鑰
- 使用 CloudHSM 金鑰存放區
- 以多執行緒範例簽署訊息
- 使用 EC 金鑰簽署和驗證

[1] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於處於非 FIP 模式的叢集，在 2023 之後仍然允許使用該叢集。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

如 AWS CloudHSM KeyStore 何使用 Java 類別作為客戶端 SDK 3

此 AWS CloudHSM **`KeyStore`**類別提供特殊用途的 PKCS12 金鑰存放區，可讓您透過 AWS CloudHSM 金鑰工具和 jarsigner 等應用程式存取金鑰。此金鑰存放區可與您的金鑰資料一起儲存憑證，並將其與在 AWS CloudHSM儲存的金鑰資料連結在一起。

> **Note**
>
> 由於憑證是公開資訊，並且為了最大化加密金鑰的儲存容量，因 AWS CloudHSM 此不支援在 HSM 上儲存憑證。

此 AWS CloudHSM `KeyStore`類別會實作 Java 密碼編譯延伸模組 (JCE) 的`KeyStore`服務提供者介面 (SPI)。如需有關使用的詳細資訊`KeyStore`，請參閱類別KeyStore。

選擇適當的金鑰存放區

AWS CloudHSM Java 密碼編譯延伸 (JCE) 提供者隨附預設的傳遞、唯讀金鑰存放區，可將所有交易傳遞至 HSM。此預設金鑰存放區與特殊用途 AWS CloudHSM KeyStore不同。在大多數情況下，您會使用預設值取得較佳的執行階段效能和輸送量。除了將金鑰作業卸載至 HSM 之外，您應該僅針對需要憑證和憑證型作業支援的應用程式使用。 AWS CloudHSM KeyStore

雖然這兩種金鑰存放區都使用 JCE 提供者進行操作，但它們是獨立的實體，且不會彼此交換資訊。

請依照下方步驟說明為您的 Java 應用程式載入預設的金鑰存放區：

```
KeyStore ks = KeyStore.getInstance("Cavium");
```

載入特殊用途 CloudHSM KeyStore ，如下所示：

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

初始化 AWS CloudHSM KeyStore

以登 AWS CloudHSM KeyStore 入 JCE 提供者的相同方式登入。您可以使用環境變數或系統屬性檔案，並且在開始使用 CloudHSM KeyStore 之前，應先登入。如需使用 JCE 提供者登入 HSM 的範例，請參閱登入 HSM。

如有需要，您可以指定一組密碼來加密保存金鑰儲存資料的本機 PKCS12 檔案。創建密 AWS CloudHSM 鑰庫時，您可以設置密碼並在使用加載，set 和 get 方法時提供密碼。

實例化一個新的 CloudHSM KeyStore 對象，如下所示：

```
ks.load(null, null);
```

使用 `store` 方法將金鑰存放區資料寫入檔案。從那時起，您可以依照以下方式，搭配來源檔案和密碼使用 `load` 方法載入現有的金鑰存放區：

```
ks.load(inputStream, password);
```

使用 AWS CloudHSM KeyStore

CloudHSM KeyStore 物件通常是透過第三方應用程式 (例如 Jarsig ner 或金鑰工具) 使用。您也可以直接搭配程式碼存取物件。

AWS CloudHSM KeyStore 符合 JCE 類別KeyStore規格，並提供下列功能。

- `load`

  從特定的輸入串流載入金鑰存放區。如果在儲存金鑰存放區時設定了密碼，則必須提供相同的密碼才能成功載入。將這兩個參數設為 Null 以初始化一個新的空金鑰存放區。

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  ks.load(inputStream, password);
  ```

- `aliases`

  傳回特定的金鑰存放區執行個體中的所有項目別名名稱列舉。結果包括本機儲存在 PKCS12 檔案中的物件，以及常駐於 HSM 的物件。

  範本程式碼：

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();)
   {
    String label = entry.nextElement();
    System.out.println(label);
   }
  ```

- `ContainsAlias`

  如果金鑰存放區可存取至少一個有指定別名的物件，則傳回 True。金鑰存放區會檢查本機儲存在 PKCS12 檔案中的物件，以及常駐於 HSM 的物件。

- `DeleteEntry`

  從本機 PKCS12 檔案刪除憑證項目。使用不支援刪除儲存在 HSM 中的 AWS CloudHSM KeyStore 金鑰資料。您可以使用 CloudHSM 的 key_mgmt_util 工具刪除金鑰。

- `GetCertificate`

  如果可用，則傳回與別名相關聯的憑證。如果別名不存在或參考非憑證的物件，則函數會傳回 NULL。

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  Certificate cert = ks.getCertificate(alias)
  ```

- `GetCertificateAlias`

  傳回第一個資料符合特定憑證的金鑰存放區項目名稱 (別名)。

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  String alias = ks.getCertificateAlias(cert)
  ```

- `GetCertificateChain`

  傳回與特定別名相關聯的憑證鏈。如果別名不存在或參考非憑證的物件，則函數會傳回 NULL。

- `GetCreationDate`

  傳回由給定別名辨識的項目建立日期。如果沒有建立日期，則函數會傳回憑證生效的日期。

- `GetKey`

  GetKey 會傳遞至 HSM，並傳回對應於指定標籤的金鑰物件。`getKey`直接查詢 HSM 時，它可用於 HSM 上的任何金鑰，無論是否由. KeyStore

  ```
  Key key = ks.getKey(keyLabel, null);
  ```

- `IsCertificateEntry`

  檢查有特定別名的項目是否代表憑證項目。

- `IsKeyEntry`

  檢查有特定別名的項目是否代表金鑰項目。此動作會在 PKCS12 檔案和 HSM 中搜尋別名。

- `SetCertificateEntry`

指定特定憑證至特定別名。如果特定的別名已用於識別金鑰或憑證，則會擲出 KeyStoreException。您可以使用 JCE 程式碼取得金鑰物件，然後使用該 KeyStore SetKeyEntry方法將憑證與金鑰產生關聯。

- 有 byte[] 金鑰的 SetKeyEntry

  用戶端 SDK 3 目前不支援此 API。

- 有 Key 物件的 SetKeyEntry

  指定特定金鑰至特定別名，並將其儲存在 HSM 內。如果 Key 物件不是類型 CaviumKey，金鑰會以可擷取的工作階段金鑰匯入 HSM。

  如果 Key 物件屬於類型 PrivateKey，則必須附有相對應的憑證鏈。

  如果別名已經存在，則 SetKeyEntry 呼叫會擲出 KeyStoreException 並防止覆寫金鑰。如果必須覆寫金鑰，請使用 KMU 或 JCE。

- EngineSize

  傳回金鑰存放區中的項目數目。

- Store

  以 PKCS12 檔案格式儲存金鑰存放區至特定輸出串流，並使用特定的密碼加以保護。此外，它仍然保留載入的金鑰 (使用 setKey 呼叫的組合)。

整合第三方應用程式與 AWS CloudHSM

一些使用案例 AWS CloudHSM 涉及整合第三方軟體應用程式與 AWS CloudHSM 叢集中的 HSM。透過整合協力廠商軟體 AWS CloudHSM，您可以達成各種安全性相關目標。下列主題說明如何達成其中一些目標。

主題

- 使用 SSL/TLS 卸載來提高您的網絡服務器安全性 AWS CloudHSM
- 使用 AWS CloudHSM將 Windows Server 設定為憑證授權單位 (CA)
- Oracle 資料庫的透明資料加密 (TDE) 搭配 AWS CloudHSM
- 搭配使 SignTool 用 Microsoft AWS CloudHSM 來簽署檔案
- Java Keytool 和 Jarsigner
- 其他第三方廠商整合

使用 SSL/TLS 卸載來提高您的網絡服務器安全性 AWS CloudHSM

Web 伺服器及其用戶端 (Web 瀏覽器) 可以使用 Secure Sockets Layer (SSL) 或 Transport Layer Security (TLS) 協議確認 Web 伺服器的身分，並建立通過 Internet 發送和接收網頁或其他數據的安全連接。這通常被稱為 HTTPS。Web 伺服器會使用公開-私有金鑰對以及 SSL/TLS 公開金鑰憑證，來和每個用戶端建立 HTTPS 工作階段。此過程涉及大量的 Web 服務器計算，但是您可以將其中一些卸載到您的 AWS CloudHSM 集群中，稱為 SSL 加速。卸載可減少 Web 伺服器的計算負載，並藉由將伺服器的私有金鑰儲存在 HSM 中，提供額外的安全性。

下列主題提供 SSL/TLS 卸載如何 AWS CloudHSM 運作的概觀，以及在下列平台上設定 SSL/TLS 卸載的教學課程。 AWS CloudHSM

對於 Linux，請在 NGINX 或 Apache HTTP 伺服器 Web 伺服器軟件上使用 OpenSSL 動態引擎

對於 Windows，使用 Internet Information Services (IIS) for Windows Server Web 伺服器軟體。

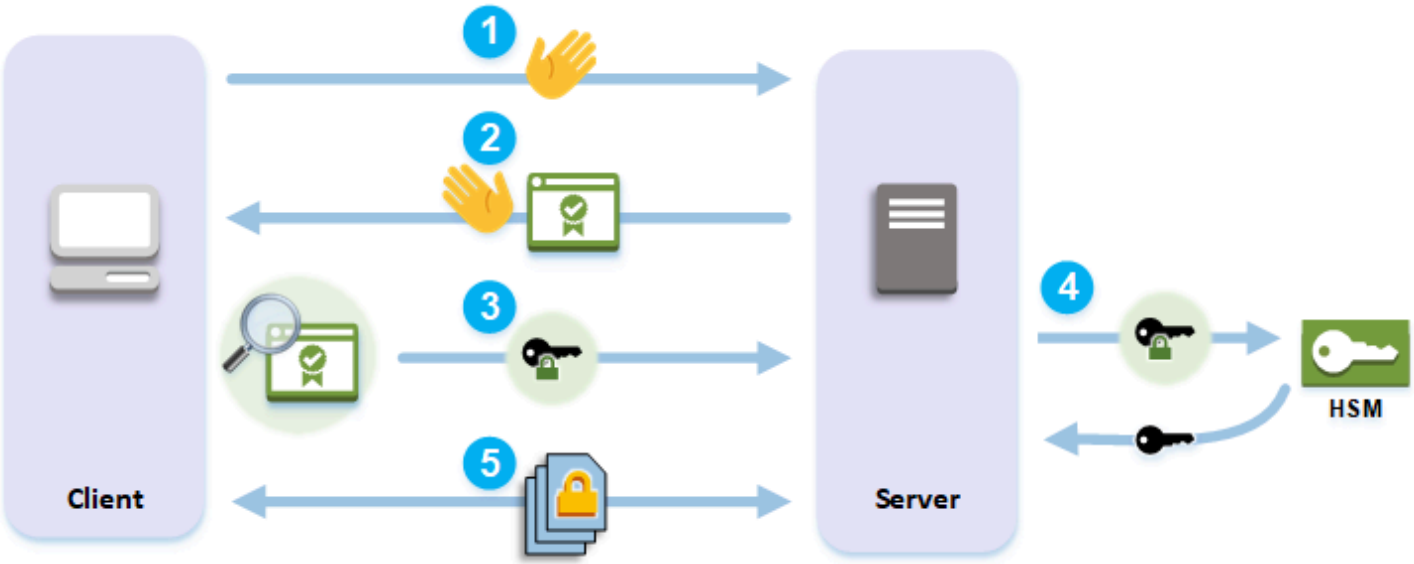
主題

- SSL/TLS 卸載的運作方式 AWS CloudHSM
- Linux 上的 SSL/TLS 卸載
- 在 Windows 上使用带有 CNG 的 IIS 进行 SSL/TLS 卸载
- 新增具有 Elastic Load Balancing (選用) 的負載平衡器

SSL/TLS 卸載的運作方式 AWS CloudHSM

為了建立 HTTPS 連線，您的 Web 伺服器會與用戶端進行交握程序。在這個過程中，伺服器會將一些加密處理卸載到 HSM，如下所示。圖片下方有每個處理步驟的說明。

> **Note**
>
> 以下圖片和程序假設伺服器驗證和金鑰交換使用 RSA。使用 Diffie–Hellman 代替 RSA 時，程序略有不同。



1. 用戶端傳送 hello 訊息到伺服器。
2. 伺服器回應 hello 訊息並傳送伺服器的憑證。
3. 用戶端執行下列動作：
   a. 確認 SSL/TLS 伺服器憑證是由客戶端信任的根憑證簽署。
   b. 從伺服器憑證擷取公開金鑰。
   c. 產生預主秘密，並以伺服器的公開金鑰將它加密。
   d. 將已加密的預主秘密傳送給伺服器。
4. 為了解密用戶端的預主秘密，伺服器會將其傳送給 HSM。HSM 使用 HSM中的私有金鑰來解密預主秘密，然後將預主秘密傳送給服務器。用戶端和伺服器分開各自使用預主秘密和 hello 訊息中的一些資訊來計算主要秘密。

5. 交握程序結束。在剩下來的工作階段中，在用戶端和伺服器之間傳送的所有訊息都會使用主要秘密的衍生產品加密。

若要了解如何使用來設定 SSL/TLS 卸載 AWS CloudHSM，請參閱下列其中一個主題：

- Linux 上的 SSL/TLS 卸載
- 在 Windows 上使用带有 CNG 的 IIS 进行 SSL/TLS 卸载

Linux 上的 SSL/TLS 卸載

有了 AWS CloudHSM，你可以在 Linux 上使用 NGINX，阿帕奇和湯姆貓執行 SSL/TLS 卸載。如需詳細資訊，請參閱下列相關主題。

主題

- 在 Linux 上使用 NGINX 或 Apache 搭配 OpenSSL 進行 SSL/TLS 卸載
- 在 Linux 上使用 Tomcat 與 JSSE 進行 SSL/TLS 卸載

在 Linux 上使用 NGINX 或 Apache 搭配 OpenSSL 進行 SSL/TLS 卸載

本主題提 step-by-step 供在 Linux 網頁伺服器上設定 SSL/TLS 卸載 AWS CloudHSM 的指示。

主題

- 概觀
- 步驟 1：設定先決條件
- 步驟 2：產生或匯入私有金鑰和 SSL/TLS 憑證
- 步驟 3：設定 Web 伺服器
- 步驟 4：啟用 HTTPS 流量並驗證憑證

概觀

在 Linux 上，NGINX 和 Apache HTTP Server Web 伺服器軟體與 OpenSSL 整合以支援 HTTPS。適用於 OpenSSL 之AWS CloudHSM 動態引擎提供的界面，可讓 Web 伺服器軟體使用叢集中的 HSM 進行密碼編譯卸載和金鑰儲存。OpenSSL 引擎是將 Web 伺服器連線到 AWS CloudHSM 叢集的橋樑。

若要完成此教學課程，您必須先選擇是否要在 Linux 上使用 NGINX 或 Apache Web 伺服器軟體。然後，教學課程會向您示範如何執行以下操作：

- 在 Amazon EC2 執行個體上安裝 Web 伺服器軟體。
- 將 Web 伺服器軟體設定為使用儲存在 AWS CloudHSM 叢集中的私有金鑰來支援 HTTPS。
- (選用) 使用 Amazon EC2 建立第二個 Web 伺服器執行個體，并使用 Elastic Load Balancing 建立負載平衡器。使用負載平衡器可將負載分散到多部伺服器，進而提升效能。它也可以在一或多個伺服器失敗時提供備援和高可用性。

當您準備好開始時，請移至步驟 1：設定先決條件。

步驟 1：設定先決條件

不同平台需要不同的先決條件。請使用下方符合您平台的先決條件區段。

主題

- 用戶端 SDK 5 的先決條件
- 用戶端 SDK 3 的先決條件

用戶端 SDK 5 的先決條件

若要設定 Web 伺服器 SSL/TLS 卸載搭配 Client SDK 5，您需要下列項目：

- 具有至少兩個硬體安全模組 (HSM) 的作用中 AWS CloudHSM 叢集

  > **Note**
  >
  > 您可以使用單一 HSM 叢集，但必須先停用用戶端金鑰耐久性。如需詳細資訊，請參閱管理用戶端金鑰耐久性設定和用戶端 SDK 5 設定工具。

- 執行 Linux 作業系統的 Amazon EC2 執行個體，其中已安裝下列軟體：
  - 一個 Web 伺服器 (無論是 NGINX 或 Apache)
  - 適用於用戶端 SDK 5 的 OpenSSL 動態引擎
- 加密使用者 (CU)，擁有及管理 HSM 上 Web 伺服器的私有金鑰。

設定 Linux Web 伺服器執行個體，並在 HSM 上建立 CU

1. 安裝和設定的 OpenSSL 動態引擎。 AWS CloudHSM如需關於安裝 OpenSSL 動態引擎的詳細資訊，請參閱用戶端 SDK 5 的 OpenSSL 動態引擎。
2. 在可存取叢集的 EC2 Linux 執行個體上，安裝 NGINX 或 Apache Web 伺服器：

Amazon Linux

- NGINX

```
$ sudo yum install nginx
```

- Apache

```
$ sudo yum install httpd24 mod24_ssl
```

Amazon Linux 2

- 如需如何在 Amazon Linux 2 上下載最新版本的 NGINX 的資訊，請參閱 NGINX 網站。

  可用於 Amazon Linux 2 的最新版本 NGINX 使用的 OpenSSL 版本比系統版本的 OpenSSL 更新。安裝 NGINX 之後，您需要建立一個符號連結，從 AWS CloudHSM OpenSSL 動態引擎程式庫到此版本的 OpenSSL 預期的位置

```
$ 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

- 如需如何在 CentOS 7 上下載最新版本的 NGINX 的資訊，請參閱 NGINX 網站。

  可用於 CentOS 7 的最新版本 NGINX 使用的 OpenSSL 版本比系統版本 OpenSSL 更新。安裝 NGINX 之後，您需要建立一個符號連結，從 AWS CloudHSM OpenSSL 動態引擎程式庫到此版本的 OpenSSL 預期的位置

```
$ 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

- 如需如何在 Red Hat 7 上下載最新版本的 NGINX 的資訊，請參閱 NGINX 網站。

  可用於 Red Hat 7 的最新版本 NGINX 使用的 OpenSSL 版本比系統版本的 OpenSSL 更新。安裝 NGINX 之後，您需要建立一個符號連結，從 AWS CloudHSM OpenSSL 動態引擎程式庫到此版本的 OpenSSL 預期的位置

  ```
  $ 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

目前尚未提供對 OpenSSL 動態引擎的支援。

3. 使用 CloudHSM CLI 建立 CU。如需關於管理 HSM 使用者的詳細資訊，請參閱使用 CloudHSM CLI 管理 HSM 使用者。

> **Tip**
>
> 保持追蹤 CU 使用者名稱和密碼。之後在為您的 Web 伺服器產生或匯入 HTTPS 私有金鑰和憑證時，您將會需要該資訊。

完成這些步驟之後，請移至 步驟 2：產生或匯入私有金鑰和 SSL/TLS 憑證。

備註

- 若要使用 Security-Enhanced Linux (SELinux) 和 Web 伺服器，您必須在連接埠 2223 上允許傳出 TCP 連線，也就是用戶端 SDK 5 用來與 HSM 通訊的連接埠。
- 若要建立和啟用叢集並授予 EC2 執行個體存取叢集的權限，請完成AWS CloudHSM入門中的步驟。入門提供有關使用一個 HSM 和一個 Amazon EC2 用戶端執行個體建立作用中叢集的 step-by-step 指示。可以使用此用戶端執行個體做為 Web 伺服器。
- 若要避免停用用戶端金鑰耐久性，請在叢集中新增多個 HSM。如需詳細資訊，請參閱 新增 HSM。
- 要連接到用戶端執行個體，可以使用 SSH 或 PuTTY。如需詳細資訊，請參閱 Amazon EC2 文件中的使用 SSH 連接至您的 Linux 執行個體或使用 PuTTY 從 Windows 連接至您的 Linux 執行個體。

用戶端 SDK 3 的先決條件

若要設定 Web 伺服器 SSL/TLS 卸載搭配 Client SDK 3，您需要下列項目：

- 至少具有一個 HSM 的作用中 AWS CloudHSM 叢集。
- 執行 Linux 作業系統的 Amazon EC2 執行個體，其中已安裝下列軟體：
  - 用 AWS CloudHSM 戶端和命令列工具。
  - NGINX 或 Apache Web 伺服器應用程式。
  - 適用於 OpenSSL 的 AWS CloudHSM 動態引擎。
- 加密使用者 (CU)，擁有及管理 HSM 上 Web 伺服器的私有金鑰。

設定 Linux Web 伺服器執行個體，並在 HSM 上建立 CU

1. 完成「開始使用」中的步驟。然後，您將擁有一個包含一個 HSM 和一個 Amazon EC2 用戶端執行個體的作用中叢集。您的 EC2 執行個體將使用命令列工具進行設定。使用此用戶端執行個體做為您的 Web 伺服器。
2. 連接至您的用戶端執行個體。如需詳細資訊，請參閱 Amazon EC2 文件中的使用 SSH 連接至您的 Linux 執行個體或使用 PuTTY 從 Windows 連接至您的 Linux 執行個體。
3. 在可存取叢集的 EC2 Linux 執行個體上，安裝 NGINX 或 Apache Web 伺服器：

   Amazon Linux

   - NGINX

   ```
   $ sudo yum install nginx
   ```

- Apache

```
$ sudo yum install httpd24 mod24_ssl
```

Amazon Linux 2

- NGINX 版本 1.19 是 NGINX 的最新版本，與 Amazon Linux 2 上的 Client SDK 3 引擎兼容。

  如需詳細資訊並下載 NGINX 版本 1.19，請參閱 NGINX 網站。

- Apache

```
$ sudo yum install httpd mod_ssl
```

CentOS 7

- NGINX 版本 1.19 是 NGINX 的最新版本，與 CentOS 7 上的用戶端 SDK 3 引擎兼容。

  如需詳細資訊並下載 NGINX 版本 1.19，請參閱 NGINX 網站。

- Apache

```
$ sudo yum install httpd mod_ssl
```

Red Hat 7

- NGINX 版本 1.19 是 NGINX 的最新版本，與 Red Hat 7 上的用戶端 SDK 3 引擎兼容。

  如需詳細資訊並下載 NGINX 版本 1.19，請參閱 NGINX 網站。

- 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. (選用) 在您的叢集中新增更多 HSM。如需詳細資訊，請參閱 新增 HSM。
5. 使用 cloudhsm_mgmt_util 建立 CU。如需詳細資訊，請參閱 管理 HSM 使用者。保持追蹤 CU 使用者名稱和密碼。之後在為您的 Web 伺服器產生或匯入 HTTPS 私有金鑰和憑證時，您將會需要該資訊。

完成這些步驟之後，請移至 步驟 2：產生或匯入私有金鑰和 SSL/TLS 憑證。

步驟 2：產生或匯入私有金鑰和 SSL/TLS 憑證

若要啟用 HTTPS，您的 Web 伺服器應用程式 (NGINX 或 Apache) 需要私有金鑰和對應的 SSL/TLS 憑證。若要搭配使用網頁伺服器 SSL/TLS 卸載 AWS CloudHSM，您必須將私密金鑰儲存在叢集中的 HSM 中。 AWS CloudHSM 您可採用下列其中一種方式來這樣做：

- 如果您還沒有私有金鑰和對應的憑證，可以在 HSM 中產生私有金鑰。您可以使用私有金鑰建立憑證簽署請求 (CSR)，以用來建立 SSL/TLS 憑證。
- 如果您已有私有金鑰和對應的憑證，可將私有金鑰匯入 HSM。

無論您選擇哪種上述方法，都可以從 HSM 匯出仿造 PEM 私有金鑰，HSM 是 PEM 格式的私有金鑰檔案，其中包含儲存在 HSM 上之私有金鑰的參考 (這不是實際的私有金鑰)。在 SSL/TLS 卸載期間，Web 伺服器會使用仿造的 PEM 私有金鑰檔案識別 HSM 上的私有金鑰。

執行以下任意一項：

- 產生私有金鑰和憑證
- 匯入現有的私有金鑰和憑證

產生私有金鑰和憑證

產生私有金鑰

本節說明如何使用用戶端 SDK 3 的金鑰管理公用程式 (KMU) 產生金鑰對。在 HSM 內部產生金鑰對之後，您可以將其匯出為仿造的 PEM 檔案，然後產生對應的憑證。

使用金鑰管理公用程式 (KMU) 產生的私有金鑰可與用戶端 SDK 3 和用戶端 SDK 5 搭配使用。

安裝和設定金鑰管理公用程式 (KMU)

1. 連接至您的用戶端執行個體。
2. 安裝和設定用戶端 SDK 3。
3. 執行下列命令以啟動用 AWS CloudHSM 戶端。

   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

目前尚未提供對 OpenSSL 動態引擎的支援。

4. 執行以下命令來啟動 key_mgmt_util 命令列工具。

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. 執行以下命令來登入 HSM。以加密使用者 (CU) 的使用者名稱和密碼，取代*<#####>*和*<##>*。

```
Command: loginHSM -u CU -s <user name> -p <password>>
```

產生私有金鑰

根據您的使用案例，您可以產生 RSA 或 EC 金鑰對。執行以下任意一項：

- 在 HSM 上產生 RSA 私有金鑰

  使用 genRSAKeyPair 命令來產生 RSA 金鑰對。此範例會產生模數為 2048、公有指數為 65537，以及標籤為 *tls_rsa_keypair* 的 RSA 金鑰對 。

```
Command: genRSAKeyPair -m 2048 -e 65537 -l tls_rsa_keypair
```

如果命令成功，您應該會看到下列輸出，指出您已成功產生 RSA 金鑰對。

```
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
```

- 在 HSM 上產生 EC 私有金鑰

  使用 genECCKeyPair 命令來產生 EC 金鑰對。本範例會產生一個曲線 ID 為 2 (對應於 NID_X9_62_prime256v1 曲線) 且標籤為 *tls_ec_keypair* 的 EC 金鑰對。

```
Command: genECCKeyPair -i 2 -l tls_ec_keypair
```

  如果命令成功，您應該會看到下列輸出，指出您已成功產生 EC 金鑰對。

```
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
```

匯出仿造的 PEM 私有金鑰檔案

HSM 上擁有私有金鑰後，您必須匯出仿造的 PEM 私有金鑰檔案。此檔案不包含實際的金鑰資料，但可讓 OpenSSL 動態引擎識別 HSM 上的私有金鑰。接著，您可以使用私有金鑰來建立憑證簽署要求 (CSR)，簽署 CSR 來建立憑證。

> **Note**
>
> 使用金鑰管理公用程式 (KMU) 產生的仿造的 PEM 檔案可與用戶端 SDK 3 和用戶端 SDK 5 搭配使用。

找出與您要匯出為仿造的 PEM 的金鑰對應的金鑰控制代碼，然後執行下列命令，以仿造的 PEM 格式匯出私有金鑰，並將其儲存至檔案。以您自己的值取代下列值。

- *<private_key_handle>*：產生的私有金鑰控制代碼。這是上一個步驟的金鑰產生命令之一所產生的控制代碼。在上述範例中，私有金鑰的控制代碼是 8。
- *<web_server_fake_PEM.key>*：將寫入仿造的 PEM 金鑰的檔案名稱。

```
Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
```

Exit (退出)

執行以下命令來停止 key_mgmt_util。

```
Command: exit
```

現在，系統上應該有一個新檔案，該檔案位於前述指令中 *<web_server_fake_PEM.key>* 所指定的路徑。此檔案是仿造的 PEM 私有金鑰檔案。

產生自簽憑證

產生仿造的 PEM 私有金鑰後，您可以使用此檔案來產生憑證簽署要求 (CSR) 和憑證。

在生產環境中，您通常會使用憑證授權單位 (CA) 從 CSR 建立憑證。測試環境不需要 CA。如果您確實使用 CA，請將 CSR 檔案發送給他們，並使用他們在 Web 伺服器中為您提供 HTTPS 的簽名 SSL/TLS 憑證。

作為使用 CA 的替代方法，您可以使用 AWS CloudHSM OpenSSL 動態引擎來建立自我簽署憑證。自簽憑證不受瀏覽器所信任，請勿用於生產環境。可以用於測試環境。

> **Warning**
>
> 自簽憑證應該只用於測試環境。若為生產環境，請使用更安全的方法 (例如憑證授權單位) 來建立憑證。

安裝和設定 OpenSSL 動態引擎

1. 連接至您的用戶端執行個體。
2. 要安裝與設定，請執行下列其中一個動作：
   - the section called “安裝 OpenSSL 動態引擎”
   - the section called “OpenSSL 動態引擎”

產生憑證

1. 取得先前步驟中產生的仿造的 PEM 檔案副本。
2. 建立 CSR

   執行下列命令，以使用 AWS CloudHSM OpenSSL 動態引擎建立憑證簽署要求 (CSR)。以包含仿造 PEM 私有金鑰的檔案名稱取代 *<web_server_fake_PEM.key>*。以包含 CSR 的檔案名稱取代 *<web_server.csr>*。

   `req` 是互動式命令。回應每個欄位。欄位資訊會複製到您的 SSL/TLS 憑證。

   ```
   $ openssl req -engine cloudhsm -new -key <web_server_fake_PEM.key> -
   out <web_server.csr>
   ```

3. 建立自簽憑證

   執行下列命令以使用 AWS CloudHSM OpenSSL 動態引擎，以您的 HSM 上的私密金鑰簽署您的 CSR。這會建立自簽憑證。在命令中，以您自己的值取代下列值。

   - *<web_server.csr>*：包含 CSR 的檔案名稱。
   - *<web_server_fake_PEM.key>*：包含仿造 PEM 私有金鑰的檔案名稱。
   - *<web_server.crt>*：包含 Web 伺服器憑證的檔案名稱。

   ```
   $ openssl x509 -engine cloudhsm -req -days 365 -in <web_server.csr> -
   signkey <web_server_fake_PEM.key> -out <web_server.crt>
   ```

完成這些步驟之後，請移至 步驟 3：設定 Web 伺服器。

匯入現有的私有金鑰和憑證

您在 Web 伺服器上可能已經有用於 HTTPS 的私有金鑰和對應的 SSL/TLS 憑證。若是如此，您可以執行本區段中的步驟，將該金鑰匯入到 HSM：

> **Note**
>
> 有關私有金鑰匯入和用戶端 SDK 兼容性的一些注意事項：
>
> - 匯入現有的私有金鑰需要用戶端 SDK 3。

- 您可以將用戶端 SDK 3 中的私有金鑰與用戶端 SDK 5 搭配使用。
- 適用於用戶端 SDK 3 的 OpenSSL 動態引擎不支援最新的 Linux 平台，但是用戶端 SDK 5 的 OpenSSL 動態引擎支援最新的 Linux 平台。您可以使用用戶端 SDK 3 提供的金鑰管理公用程式 (KMU) 匯入現有的私有金鑰，然後使用該私有金鑰和 OpenSSL 動態引擎與用戶端 SDK 5 的實作，以支援最新 Linux 平台上的 SSL/TLS 卸載。

將現有的私有金鑰匯入到 Client SDK 3 的 HSM

1. 連線到 Amazon EC2 用戶端執行個體。如有必要，請將現有的私有金鑰和憑證複製到執行個體。
2. 安裝和設定用戶端 SDK 3
3. 執行下列命令以啟動用 AWS CloudHSM 戶端。

   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

目前尚未提供對 OpenSSL 動態引擎的支援。

4. 執行以下命令來啟動 key_mgmt_util 命令列工具。

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. 執行以下命令來登入 HSM。以加密使用者 (CU) 的使用者名稱和密碼，取代*<#####>* 和*<##>*。

```
Command: loginHSM -u CU -s <user name> -p <password>
```

6. 執行以下命令，將您的私有金鑰匯入到 HSM。

   a. 執行以下命令，建立僅對目前工作階段而言有效的對稱包裝金鑰。以下顯示命令和輸出。

```
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. 執行以下命令，將現有的私有金鑰匯入到 HSM。以下顯示命令和輸出。以您自己的值取代下列值：

   - *<web_server_existing.key>*：包含您私有金鑰的檔案名稱。
   - *<web_server_imported_key>*：匯入的私有金鑰標籤。

- *<wrapping_key_handle>*：前一個命令所產生的包裝金鑰控制代碼。在上述範例中，包裝金鑰控制代碼是 6。

```
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. 執行以下命令，以仿造 PEM 格式匯出私有金鑰，並將其儲存到檔案。以您自己的值取代下列值。

   - *<private_key_handle>*：匯入的私有金鑰控制代碼。這是上一個步驟的第二個命令所產生的控制代碼。在上述範例中，私有金鑰的控制代碼是 8。
   - *<web_server_fake_PEM.key>*：包含匯出的 PEM 私有金鑰檔案名稱。

```
Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
```

8. 執行以下命令來停止 key_mgmt_util。

```
Command: exit
```

完成這些步驟之後，請移至 步驟 3：設定 Web 伺服器。

步驟 3：設定 Web 伺服器

更新 Web 伺服器軟體組態，以使用 HTTPS 憑證以及您在 上一個步驟中建立的對應仿造 PEM 私有金鑰。請記得在開始之前備份現有的憑證和金鑰。這將完成針對 SSL/TLS 卸載搭配 AWS CloudHSM設定您的 Linux Web 伺服器軟體。

完成以下其中一個區段中的步驟。

主題

- 設定 NGINX Web 伺服器

- 設定 Apache Web 伺服器

設定 NGINX Web 伺服器

使用此區段可在支援的平台上設定 NGINX。

更新 NGINX 的 Web 伺服器組態

1. 連接至您的用戶端執行個體。
2. 執行以下命令，為 Web 伺服器憑證和仿造 PEM 私有金鑰建立所需的目錄。

```
$ sudo mkdir -p /etc/pki/nginx/private
```

3. 執行以下命令，將您的 Web 伺服器憑證複製到所需的位置。以 Web 伺服器憑證的名稱取代 *<web_server.crt>*。

```
$ sudo cp <web_server.crt> /etc/pki/nginx/server.crt
```

4. 執行以下命令，將您的仿造 PEM 私有金鑰複製到所需的位置。以包含仿造 PEM 私有金鑰的檔案名稱取代 *<web_server_fake_PEM.key>*。

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/nginx/private/server.key
```

5. 執行以下命令來變更檔案的擁有權，使得名為 nginx 的使用者可讀取這些檔案。

```
$ sudo chown nginx /etc/pki/nginx/server.crt /etc/pki/nginx/private/server.key
```

6. 執行以下命令來備份 `/etc/nginx/nginx.conf` 檔案。

```
$ sudo cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

7. 更新 NGINX 組態。

> **Note**
>
> 每個叢集可以在所有 NGINX Web 伺服器上支持最多 1000 個 NGINX 工作者程序。

Amazon Linux

使用文字編輯器來編輯 /etc/nginx/nginx.conf 檔案。這需要 Linux 根許可。在檔案上方，新增下列行：

- 如果使用用戶端 SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 如果使用用戶端 SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然後將以下內容新增至檔案的 TLS 區段：

```
# 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

使用文字編輯器來編輯 /etc/nginx/nginx.conf 檔案。這需要 Linux 根許可。在檔案上方，新增下列行：

- 如果使用用戶端 SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 如果使用用戶端 SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然後將以下內容新增至檔案的 TLS 區段：

```
# 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

使用文字編輯器來編輯 /etc/nginx/nginx.conf 檔案。這需要 Linux 根許可。在檔案上方，新增下列行：

- 如果使用用戶端 SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 如果使用用戶端 SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然後將以下內容新增至檔案的 TLS 區段：

```
# 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

使用文字編輯器來編輯 /etc/nginx/nginx.conf 檔案。這需要 Linux 根許可。在檔案上方，新增下列行：

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然後將以下內容新增至檔案的 TLS 區段：

```
# 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

使用文字編輯器來編輯 /etc/nginx/nginx.conf 檔案。這需要 Linux 根許可。在檔案上方，新增下列行：

- 如果使用用戶端 SDK 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 如果使用用戶端 SDK 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然後將以下內容新增至檔案的 TLS 區段：

```
# 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

使用文字編輯器來編輯 /etc/nginx/nginx.conf 檔案。這需要 Linux 根許可。在檔案上方，新增下列行：

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然後將以下內容新增至檔案的 TLS 區段：

```
# 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

使用文字編輯器來編輯 /etc/nginx/nginx.conf 檔案。這需要 Linux 根許可。在檔案上方，新增下列行：

```
ssl_engine cloudhsm;
    env n3fips_password;
```

然後將以下內容新增至檔案的 TLS 區段：

```
# 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

使用文字編輯器來編輯 /etc/nginx/nginx.conf 檔案。這需要 Linux 根許可。在檔案上方，新增下列行：

```
ssl_engine cloudhsm;
    env CLOUDHSM_PIN;
```

然後將以下內容新增至檔案的 TLS 區段：

```
# 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

使用文字編輯器來編輯 /etc/nginx/nginx.conf 檔案。這需要 Linux 根許可。在檔案上方，新增下列行：

```
ssl_engine cloudhsm;
    env CLOUDHSM_PIN;
```

然後將以下內容新增至檔案的 TLS 區段：

```
# 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

目前尚未提供對 OpenSSL 動態引擎的支援。

儲存檔案。

8. 備份 systemd 組態檔案，然後設定 EnvironmentFile 路徑。

Amazon Linux

不需要採取行動。

Amazon Linux 2

1. 備份 nginx.service 檔案。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 在文字編輯器中開啟 /lib/systemd/system/nginx.service 檔案，然後在 [Service] (服務) 區段中，新增以下路徑：

```
EnvironmentFile=/etc/sysconfig/nginx
```

CentOS 7

不需要採取行動。

CentOS 8

1. 備份 nginx.service 檔案。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 在文字編輯器中開啟 /lib/systemd/system/nginx.service 檔案，然後在 [Service] (服務) 區段中，新增以下路徑：

```
EnvironmentFile=/etc/sysconfig/nginx
```

Red Hat 7

不需要採取行動。

Red Hat 8

1. 備份 nginx.service 檔案。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 在文字編輯器中開啟 /lib/systemd/system/nginx.service 檔案，然後在 [Service] (服務) 區段中，新增以下路徑：

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 16.04

1. 備份 nginx.service 檔案。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 在文字編輯器中開啟 /lib/systemd/system/nginx.service 檔案，然後在 [Service] (服務) 區段中，新增以下路徑：

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 18.04

1. 備份 nginx.service 檔案。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 在文字編輯器中開啟 /lib/systemd/system/nginx.service 檔案，然後在 [Service] (服務) 區段中，新增以下路徑：

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 20.04 LTS

1. 備份 nginx.service 檔案。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 在文字編輯器中開啟 /lib/systemd/system/nginx.service 檔案，然後在 [Service] (服務) 區段中，新增以下路徑：

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 22.04 LTS

目前尚未提供對 OpenSSL 動態引擎的支援。

9. 檢查 /etc/sysconfig/nginx 檔案是否存在，然後執行以下其中一項：

- 如果檔案存在，請執行下列命令來備份檔案：

```
$ sudo cp /etc/sysconfig/nginx /etc/sysconfig/nginx.backup
```

- 如果檔案不存在，請開啟文字編輯器，然後建立在 /etc/sysconfig/ 資料夾中建立名為 nginx 的檔案。

10. 設定 NGINX 環境。

> **Note**
>
> 用戶端 SDK 5 引入了用於存儲 CU 憑證的 CLOUDHSM_PIN 環境變量。

Amazon Linux

在文字編輯器中開啟 /etc/sysconfig/nginx 檔案。這需要 Linux 根許可。新增加密使用者 (CU) 憑證：

- 如果使用用戶端 SDK 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用用戶端 SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

儲存檔案。

Amazon Linux 2

在文字編輯器中開啟 /etc/sysconfig/nginx 檔案。這需要 Linux 根許可。新增加密使用者 (CU) 憑證：

- 如果使用用戶端 SDK 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用用戶端 SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

儲存檔案。

CentOS 7

在文字編輯器中開啟 /etc/sysconfig/nginx 檔案。這需要 Linux 根許可。新增加密使用者 (CU) 憑證：

- 如果使用用戶端 SDK 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用用戶端 SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

儲存檔案。

CentOS 8

在文字編輯器中開啟 /etc/sysconfig/nginx 檔案。這需要 Linux 根許可。新增加密使用者 (CU) 憑證：

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

儲存檔案。

Red Hat 7

在文字編輯器中開啟 /etc/sysconfig/nginx 檔案。這需要 Linux 根許可。新增加密使用者 (CU) 憑證：

- 如果使用用戶端 SDK 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用用戶端 SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

儲存檔案。

Red Hat 8

在文字編輯器中開啟 /etc/sysconfig/nginx 檔案。這需要 Linux 根許可。新增加密使用者 (CU) 憑證：

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

儲存檔案。

Ubuntu 16.04 LTS

在文字編輯器中開啟 /etc/sysconfig/nginx 檔案。這需要 Linux 根許可。新增加密使用者 (CU) 憑證：

```
n3fips_password=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

儲存檔案。

Ubuntu 18.04 LTS

在文字編輯器中開啟 /etc/sysconfig/nginx 檔案。這需要 Linux 根許可。新增加密使用者 (CU) 憑證：

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

儲存檔案。

Ubuntu 20.04 LTS

在文字編輯器中開啟 /etc/sysconfig/nginx 檔案。這需要 Linux 根許可。新增加密使用者 (CU) 憑證：

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

儲存檔案。

Ubuntu 22.04 LTS

目前尚未提供對 OpenSSL 動態引擎的支援。

11. 啟動 NGINX 伺服器。

Amazon Linux

在文字編輯器中開啟 /etc/sysconfig/nginx 檔案。這需要 Linux 根許可。新增加密使用者 (CU) 憑證：

```
$ sudo service nginx start
```

Amazon Linux 2

停止任何正在運行的 NGINX 進程

```
$ sudo systemctl stop nginx
```

重新載入 systemd 組態以取得最新的變更

```
$ sudo systemctl daemon-reload
```

啟動 NGINX 進程

```
$ sudo systemctl start nginx
```

CentOS 7

停止任何正在運行的 NGINX 進程

```
$ sudo systemctl stop nginx
```

重新載入 systemd 組態以取得最新的變更

```
$ sudo systemctl daemon-reload
```

啟動 NGINX 進程

```
$ sudo systemctl start nginx
```

CentOS 8

停止任何正在運行的 NGINX 進程

```
$ sudo systemctl stop nginx
```

重新載入 systemd 組態以取得最新的變更

```
$ sudo systemctl daemon-reload
```

啟動 NGINX 進程

```
$ sudo systemctl start nginx
```

Red Hat 7

停止任何正在運行的 NGINX 進程

```
$ sudo systemctl stop nginx
```

重新載入 systemd 組態以取得最新的變更

```
$ sudo systemctl daemon-reload
```

啟動 NGINX 進程

```
$ sudo systemctl start nginx
```

Red Hat 8

停止任何正在運行的 NGINX 進程

```
$ sudo systemctl stop nginx
```

重新載入 systemd 組態以取得最新的變更

```
$ sudo systemctl daemon-reload
```

啟動 NGINX 進程

```
$ sudo systemctl start nginx
```

Ubuntu 16.04 LTS

停止任何正在運行的 NGINX 進程

```
$ sudo systemctl stop nginx
```

重新載入 systemd 組態以取得最新的變更

```
$ sudo systemctl daemon-reload
```

啟動 NGINX 進程

```
$ sudo systemctl start nginx
```

Ubuntu 18.04 LTS

停止任何正在運行的 NGINX 進程

```
$ sudo systemctl stop nginx
```

重新載入 systemd 組態以取得最新的變更

```
$ sudo systemctl daemon-reload
```

啟動 NGINX 進程

```
$ sudo systemctl start nginx
```

Ubuntu 20.04 LTS

停止任何正在運行的 NGINX 進程

```
$ sudo systemctl stop nginx
```

重新載入 systemd 組態以取得最新的變更

```
$ sudo systemctl daemon-reload
```

啟動 NGINX 進程

```
$ sudo systemctl start nginx
```

Ubuntu 22.04 LTS

目前尚未提供對 OpenSSL 動態引擎的支援。

12. (選用) 將平台設定為在啟動時啟動 NGINX。

Amazon Linux

```
$ sudo chkconfig nginx on
```

Amazon Linux 2

```
$ sudo systemctl enable nginx
```

CentOS 7

不需要採取行動。

CentOS 8

```
$ sudo systemctl enable nginx
```

Red Hat 7

不需要採取行動。

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

目前尚未提供對 OpenSSL 動態引擎的支援。

在更新您的 Web 伺服器組態之後，請移至 步驟 4：啟用 HTTPS 流量並驗證憑證。

設定 Apache Web 伺服器

使用此區段可在支援的平台上設定 Apache。

更新 Apache 的 Web 伺服器組態

1. 連線到 Amazon EC2 用戶端執行個體。
2. 為平台定義憑證和私有金鑰的預設位置。

Amazon Linux

在 /etc/httpd/conf.d/ssl.conf 檔案中，請確定這些值存在：

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Amazon Linux 2

在 /etc/httpd/conf.d/ssl.conf 檔案中，請確定這些值存在：

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

CentOS 7

在 /etc/httpd/conf.d/ssl.conf 檔案中，請確定這些值存在：

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

CentOS 8

在 /etc/httpd/conf.d/ssl.conf 檔案中，請確定這些值存在：

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 7

在 /etc/httpd/conf.d/ssl.conf 檔案中，請確定這些值存在：

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 8

在 /etc/httpd/conf.d/ssl.conf 檔案中，請確定這些值存在：

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Ubuntu 16.04 LTS

在 /etc/apache2/sites-available/default-ssl.conf 檔案中，請確定這些值存在：

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 18.04 LTS

在 /etc/apache2/sites-available/default-ssl.conf 檔案中，請確定這些值存在：

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 20.04 LTS

在 /etc/apache2/sites-available/default-ssl.conf 檔案中，請確定這些值存在：

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 22.04 LTS

目前尚未提供對 OpenSSL 動態引擎的支援。

3. 將 Web 伺服器憑證複製到平台所需的位置。

Amazon Linux

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

以 Web 伺服器憑證的名稱取代 *<web_server.crt>*。

Amazon Linux 2

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

以 Web 伺服器憑證的名稱取代 *<web_server.crt>*。

CentOS 7

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

以 Web 伺服器憑證的名稱取代 *<web_server.crt>*。

CentOS 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

以 Web 伺服器憑證的名稱取代 *<web_server.crt>*。

Red Hat 7

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

以 Web 伺服器憑證的名稱取代 *<web_server.crt>*。

Red Hat 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

以 Web 伺服器憑證的名稱取代 *<web_server.crt>*。

Ubuntu 16.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

以 Web 伺服器憑證的名稱取代 *<web_server.crt>*。

Ubuntu 18.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

以 Web 伺服器憑證的名稱取代 *<web_server.crt>*。

Ubuntu 20.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

以 Web 伺服器憑證的名稱取代 *<web_server.crt>*。

Ubuntu 22.04 LTS

目前尚未提供對 OpenSSL 動態引擎的支援。

4. 將仿造 PEM 私有金鑰複製到平台所需的位置。

Amazon Linux

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

以包含仿造 PEM 私有金鑰的檔案名稱取代 *<web_server_fake_PEM.key>*。

Amazon Linux 2

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

以包含仿造 PEM 私有金鑰的檔案名稱取代 *<web_server_fake_PEM.key>*。

CentOS 7

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

以包含仿造 PEM 私有金鑰的檔案名稱取代 *<web_server_fake_PEM.key>*。

CentOS 8

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

以包含仿造 PEM 私有金鑰的檔案名稱取代 *<web_server_fake_PEM.key>*。

Red Hat 7

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

以包含仿造 PEM 私有金鑰的檔案名稱取代 *<web_server_fake_PEM.key>*。

Red Hat 8

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

以包含仿造 PEM 私有金鑰的檔案名稱取代 *<web_server_fake_PEM.key>*。

Ubuntu 16.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

以包含仿造 PEM 私有金鑰的檔案名稱取代 *<web_server_fake_PEM.key>*。

Ubuntu 18.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

以包含仿造 PEM 私有金鑰的檔案名稱取代 *<web_server_fake_PEM.key>*。

Ubuntu 20.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

以包含仿造 PEM 私有金鑰的檔案名稱取代 *<web_server_fake_PEM.key>*。

Ubuntu 22.04 LTS

目前尚未提供對 OpenSSL 動態引擎的支援。

5. 如果平台需要，請更改這些檔案的所有權。

Amazon Linux

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

提供讀取許可給名為 apache 的使用者。

Amazon Linux 2

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

提供讀取許可給名為 apache 的使用者。

CentOS 7

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

提供讀取許可給名為 apache 的使用者。

CentOS 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

提供讀取許可給名為 apache 的使用者。

Red Hat 7

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

提供讀取許可給名為 apache 的使用者。

Red Hat 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

提供讀取許可給名為 apache 的使用者。

Ubuntu 16.04 LTS

不需要採取行動。

Ubuntu 18.04 LTS

不需要採取行動。

Ubuntu 20.04 LTS

不需要採取行動。

Ubuntu 22.04 LTS

目前尚未提供對 OpenSSL 動態引擎的支援。

6. 為平台設定 Apache 指令。

Amazon Linux

找到此平台的 SSL 檔案：

```
/etc/httpd/conf.d/ssl.conf
```

該文件包含定義伺服器應該如何運行的 Apache 指令。指令顯示在左側，後面接著一個值。使用文字編輯器編輯此檔案。這需要 Linux 根許可。

使用這些值更新或輸入下列指令：

```
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
```

儲存檔案。

Amazon Linux 2

找到此平台的 SSL 檔案：

```
/etc/httpd/conf.d/ssl.conf
```

該文件包含定義伺服器應該如何運行的 Apache 指令。指令顯示在左側，後面接著一個值。使用文字編輯器編輯此檔案。這需要 Linux 根許可。

使用這些值更新或輸入下列指令：

```
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
```

儲存檔案。

CentOS 7

找到此平台的 SSL 檔案：

```
/etc/httpd/conf.d/ssl.conf
```

該文件包含定義伺服器應該如何運行的 Apache 指令。指令顯示在左側，後面接著一個值。使用文字編輯器編輯此檔案。這需要 Linux 根許可。

使用這些值更新或輸入下列指令：

```
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
```

儲存檔案。

CentOS 8

找到此平台的 SSL 檔案：

```
/etc/httpd/conf.d/ssl.conf
```

該文件包含定義伺服器應該如何運行的 Apache 指令。指令顯示在左側，後面接著一個值。使用文字編輯器編輯此檔案。這需要 Linux 根許可。

使用這些值更新或輸入下列指令：

```
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
```

儲存檔案。

Red Hat 7

找到此平台的 SSL 檔案：

```
/etc/httpd/conf.d/ssl.conf
```

該文件包含定義伺服器應該如何運行的 Apache 指令。指令顯示在左側，後面接著一個值。使用文字編輯器編輯此檔案。這需要 Linux 根許可。

使用這些值更新或輸入下列指令：

```
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
```

儲存檔案。

Red Hat 8

找到此平台的 SSL 檔案：

```
/etc/httpd/conf.d/ssl.conf
```

該文件包含定義伺服器應該如何運行的 Apache 指令。指令顯示在左側，後面接著一個值。使用文字編輯器編輯此檔案。這需要 Linux 根許可。

使用這些值更新或輸入下列指令：

```
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
```

儲存檔案。

Ubuntu 16.04 LTS

找到此平台的 SSL 檔案：

```
/etc/apache2/mods-available/ssl.conf
```

該文件包含定義伺服器應該如何運行的 Apache 指令。指令顯示在左側，後面接著一個值。使用文字編輯器編輯此檔案。這需要 Linux 根許可。

使用這些值更新或輸入下列指令：

```
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
```

儲存檔案。

啟用 SSL 模組和預設 SSL 網站組態：

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 18.04 LTS

找到此平台的 SSL 檔案：

```
/etc/apache2/mods-available/ssl.conf
```

該文件包含定義伺服器應該如何運行的 Apache 指令。指令顯示在左側，後面接著一個值。使用文字編輯器編輯此檔案。這需要 Linux 根許可。

使用這些值更新或輸入下列指令：

```
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
```

儲存檔案。

啟用 SSL 模組和預設 SSL 網站組態：

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 20.04 LTS

找到此平台的 SSL 檔案：

```
/etc/apache2/mods-available/ssl.conf
```

該文件包含定義伺服器應該如何運行的 Apache 指令。指令顯示在左側，後面接著一個值。使用文字編輯器編輯此檔案。這需要 Linux 根許可。

使用這些值更新或輸入下列指令：

```
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
```

儲存檔案。

啟用 SSL 模組和預設 SSL 網站組態：

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 22.04 LTS

目前尚未提供對 OpenSSL 動態引擎的支援。

7. 為平台設定環境值檔案。

Amazon Linux

不需要採取行動。環境值要填入 `/etc/sysconfig/httpd`

Amazon Linux 2

開啟 httpd 服務檔案：

```
/lib/systemd/system/httpd.service
```

將下列內容加入到 `[Service]` 區段：

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 7

開啟 httpd 服務檔案：

```
/lib/systemd/system/httpd.service
```

將下列內容加入到 `[Service]` 區段：

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 8

開啟 httpd 服務檔案：

```
/lib/systemd/system/httpd.service
```

將下列內容加入到 [Service] 區段：

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 7

開啟 httpd 服務檔案：

```
/lib/systemd/system/httpd.service
```

將下列內容加入到 [Service] 區段：

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 8

開啟 httpd 服務檔案：

```
/lib/systemd/system/httpd.service
```

將下列內容加入到 [Service] 區段：

```
EnvironmentFile=/etc/sysconfig/httpd
```

Ubuntu 16.04 LTS

不需要採取行動。環境值要填入 /etc/sysconfig/httpd

Ubuntu 18.04 LTS

不需要採取行動。環境值要填入 /etc/sysconfig/httpd

Ubuntu 20.04 LTS

不需要採取行動。環境值要填入 /etc/sysconfig/httpd

Ubuntu 22.04 LTS

目前尚未提供對 OpenSSL 動態引擎的支援。

8. 在儲存平台環境變數的檔案中，設定包含加密使用者 (CU) 憑證的環境變數：

Amazon Linux

使用文字編輯器編輯 /etc/sysconfig/httpd。

- 如果使用用戶端 SDK 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用用戶端 SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

Amazon Linux 2

使用文字編輯器編輯 /etc/sysconfig/httpd。

- 如果使用用戶端 SDK 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用用戶端 SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

CentOS 7

使用文字編輯器編輯 /etc/sysconfig/httpd。

- 如果使用用戶端 SDK 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用用戶端 SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

CentOS 8

使用文字編輯器編輯 /etc/sysconfig/httpd。

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

Red Hat 7

使用文字編輯器編輯 /etc/sysconfig/httpd。

- 如果使用用戶端 SDK 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用用戶端 SDK 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

Red Hat 8

使用文字編輯器編輯 /etc/sysconfig/httpd。

```
CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

> **Note**
>
> 用戶端 SDK 5 引入了用於存儲 CU 憑證的 CLOUDHSM_PIN 環境變量。

Ubuntu 16.04 LTS

使用文字編輯器編輯 /etc/apache2/envvars。

```
export n3fips_password=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

Ubuntu 18.04 LTS

使用文字編輯器編輯 /etc/apache2/envvars。

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

> **Note**
>
> 用戶端 SDK 5 引入了用於存儲 CU 憑證的 CLOUDHSM_PIN 環境變量。在用戶端 SDK 3 中，將 CU 憑證儲存在 n3fips_password 環境變數中。用戶端 SDK 5 支援這兩個環境變數，但建議使用 CLOUDHSM_PIN。

Ubuntu 20.04 LTS

使用文字編輯器編輯 /etc/apache2/envvars。

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

以 CU 憑證取代*<CU #####>*和*<##>*。

> **Note**
>
> 用戶端 SDK 5 引入了用於存儲 CU 憑證的 CLOUDHSM_PIN 環境變量。在用戶端 SDK 3 中，將 CU 憑證儲存在 n3fips_password 環境變數中。用戶端 SDK 5 支援這兩個環境變數，但建議使用 CLOUDHSM_PIN。

Ubuntu 22.04 LTS

目前尚未提供對 OpenSSL 動態引擎的支援。

9. 啟動 Apache Web 伺服器。

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

目前尚未提供對 OpenSSL 動態引擎的支援。

10. (選用) 將平台設定為在啟動時啟動 Apache。

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

目前尚未提供對 OpenSSL 動態引擎的支援。

在更新您的 Web 伺服器組態之後，請移至 步驟 4：啟用 HTTPS 流量並驗證憑證。

步驟 4：啟用 HTTPS 流量並驗證憑證

在您將 Web 伺服器設定為 SSL/TLS 卸載使用之後 AWS CloudHSM，請將 Web 伺服器執行個體新增至允許輸入 HTTPS 流量的安全性群組。這可讓用戶端 (例如 Web 瀏覽器) 與 Web 伺服器建立 HTTPS 連線。然後建立 HTTPS 連接到您的 Web 服務器，並驗證它使用的是您為 SSL/TLS 卸載配置的證書。AWS CloudHSM

主題

- 啟用傳入 HTTPS 連線
- 驗證 HTTPS 是否使用您已設定的憑證

啟用傳入 HTTPS 連線

若要從用戶端 (例如 Web 瀏覽器) 連接到 Web 伺服器，請建立允許傳入 HTTPS 連接的安全群組。特別是應該允許連接埠 443 上的傳入 TCP 連線。將此安全群組指派到您的 Web 伺服器。

建立 HTTPS 的安全群組並將其指派至您的 Web 伺服器

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在導覽窗格中選擇安全群組。
3. 選擇建立安全群組。
4. 對於 Create Security Group (建立安全群組)，執行下列動作：
   a. 對於 Security group name (安全群組名稱)，輸入您要建立之安全群組的名稱。
   b. (選用) 輸入您要建立之安全群組的描述。
   c. 對於 VPC，選擇包含 Web 伺服器 Amazon EC2 執行個體的 VPC。
   d. 選取 Add Rule (新增規則)。
   e. 對於類型，從下拉式視窗中選取 HTTPS。
   f. 對於來源，輸入來源位置。
   g. 選擇建立安全群組。
5. 在導覽窗格中，選擇執行個體。
6. 選取 Web 伺服器執行個體旁的核取方塊。
7. 選取頁面頂端的動作下拉式選單。選取安全性，然後選取變更安全群組。
8. 對於關聯的安全群組，請選取搜尋方塊，然後選取您為 HTTPS 建立之安全群組。然後選擇新增安全群組。
9. 選取 Save (儲存)。

驗證 HTTPS 是否使用您已設定的憑證

將 Web 伺服器新增至安全群組之後，您可以驗證 SSL/TLS 卸載是否正在使用自簽憑證。若要這樣做，您可以使用 Web 瀏覽器或使用 OpenSSL s_client 之類的工具。

使用 Web 瀏覽器驗證 SSL/TLS 卸載

1. 使用 Web 瀏覽器來透過伺服器的公有 DNS 名稱或 IP 地址連接到您的 Web 伺服器。請確定網址列中的 URL 開頭為 https://。例如 **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**。

> **Tip**
>
> 您可以使用 DNS 服務 (例如 Amazon Route 53)，將網站的網域名稱 (例如，https://www.example.com/) 路由到 Web 伺服器。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》或 DNS 服務文件中的將流量路由到 Amazon EC2 執行個體。

2. 使用您的 Web 瀏覽器來檢視 Web 伺服器憑證。如需詳細資訊，請參閱下列內容：

   - 若為 Mozilla Firefox，請參閱 Mozilla 技術支援網站上的檢視憑證。
   - 若為 Google Chrome，請參閱 Google Web 開發人員工具網站上的了解安全問題。

   其他 Web 瀏覽器可能有類似的功能，可供您用來檢視 Web 伺服器憑證。

3. 確保 SSL/TLS 憑證是您設定 Web 伺服器所要使用的憑證。

使用 OpenSSL s_client 來驗證 SSL/TLS 卸載

1. 執行以下 OpenSSL 命令來使用 HTTPS 連接至 Web 伺服器。以 Web 伺服器的公有 DNS 名稱或 IP 地址來取代*<#####>*。

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> 您可以使用 DNS 服務 (例如 Amazon Route 53)，將網站的網域名稱 (例如，https://www.example.com/) 路由到 Web 伺服器。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》或 DNS 服務文件中的將流量路由到 Amazon EC2 執行個體。

2. 確保 SSL/TLS 憑證是您設定 Web 伺服器所要使用的憑證。

您現在有透過 HTTPS 而受到保護的網站。Web 伺服器的私密金鑰會儲存在 AWS CloudHSM 叢集中的 HSM 中。

若要新增負載平衡器，請參閱 新增具有 Elastic Load Balancing (選用) 的負載平衡器。

在 Linux 上使用 Tomcat 與 JSSE 進行 SSL/TLS 卸載

本主題提供 step-by-step 使用 Java 安全通訊端延伸 (JSSE) 搭配 JCE SDK 來設定 SSL/TLS 卸載的指示。 AWS CloudHSM

主題

概觀

在中 AWS CloudHSM，Tomcat 網頁伺服器可以在 Linux 上運作以支援 HTTPS。 AWS CloudHSM JCE SDK 提供了可與 JSSE（Java 安全套接字擴展）一起使用的接口，以便在此類 Web 服務器中使用 HSM。 AWS CloudHSM JCE 是將 JSSE 連接到 AWS CloudHSM HSM 叢集的橋接器。JSSE 是適用於 Secure Sockets Layer (SSL) 和 Transport Layer Security (TLS) 通訊協定的 Java API。

步驟 1：設定先決條件

請遵循這些先決條件，在 Linux 上使用具有 SSL/TLS 卸載 AWS CloudHSM 的 Tomcat 網頁伺服器。您必須符合這些先決條件，才能透過用戶端 SDK 5 和 Tomcat Web 伺服器來設定 Web 伺服器 SSL/TLS 卸載。

> **Note**
>
> 不同平台需要不同的先決條件。請務必遵循適用於平台的正確安裝步驟。

必要條件

- 執行 Linux 作業系統且安裝了 Tomcat Web 伺服器的 Amazon EC2 執行個體。
- 加密使用者 (CU)，擁有及管理 HSM 上 Web 伺服器的私有金鑰。
- 具有至少兩個硬體安全性模組 (HSM) 的作用中 AWS CloudHSM 叢集，這些模組已安裝並設定用戶端 SDK 5 的 JCE。

> **Note**
>
> 您可以使用單一 HSM 叢集，但必須先停用用戶端金鑰耐久性。如需詳細資訊，請參閱管理用戶端金鑰耐久性設定和用戶端 SDK 5 設定工具。

如何滿足先決條件

1. 在至少具有兩個硬體安全性模組 (HSM) 的作用中 AWS CloudHSM 叢集 AWS CloudHSM 上安裝和設定 JCE。如需關於安裝的詳細資訊，請參閱用戶端 SDK 5 的 JCE。
2. 在可以存取 AWS CloudHSM 叢集的 EC2 Linux 執行個體上，依照 Apache Tomcat 指示下載並安裝 Tomcat 網頁伺服器。
3. 使用 CloudHSM CLI 建立加密使用者 (CU)。如需關於管理 HSM 使用者的詳細資訊，請參閱使用 CloudHSM CLI 管理 HSM 使用者。

   > **Tip**
   >
   > 保持追蹤 CU 使用者名稱和密碼。之後在為您的 Web 伺服器產生或匯入 HTTPS 私有金鑰和憑證時，您將會需要該資訊。

4. 要使用 Java Keytool 設置 JCE，請按照 使用用戶端 SDK 5 與 Java Keytool 和 Jarsigner 整合 中的說明進行操作。

完成這些步驟之後，請移至 步驟 2：產生或匯入私有金鑰和 SSL/TLS 憑證。

備註

- 若要使用 Security-Enhanced Linux (SELinux) 和 Web 伺服器，您必須在連接埠 2223 上允許傳出 TCP 連線，也就是用戶端 SDK 5 用來與 HSM 通訊的連接埠。
- 若要建立和啟用叢集並授予 EC2 執行個體存取叢集的權限，請完成AWS CloudHSM入門中的步驟。本節提供使用一個 HSM 和一個 Amazon EC2 用戶端執行個體建立作用中叢集的 step-by-step 說明。可以使用此用戶端執行個體做為 Web 伺服器。
- 若要避免停用用戶端金鑰耐久性，請在叢集中新增多個 HSM。如需詳細資訊，請參閱 新增 HSM。
- 要連接到用戶端執行個體，可以使用 SSH 或 PuTTY。如需詳細資訊，請參閱 Amazon EC2 文件中的使用 SSH 連接至您的 Linux 執行個體或使用 PuTTY 從 Windows 連接至您的 Linux 執行個體。

步驟 2：產生或匯入私有金鑰和 SSL/TLS 憑證

若要啟用 HTTPS，Tomcat Web 伺服器應用程式需要私有金鑰和對應 SSL/TLS 憑證。若要搭配使用網頁伺服器 SSL/TLS 卸載 AWS CloudHSM，您必須將私密金鑰儲存在叢集中的 HSM 中。 AWS CloudHSM

> **Note**
>
> 如果您還沒有私有金鑰和對應的憑證，可以在 HSM 中產生私有金鑰。您可以使用私有金鑰建立憑證簽署請求 (CSR)，以用來建立 SSL/TLS 憑證。

您可以建立一個本機 AWS CloudHSM KeyStore 檔案，其中包含 HSM 上私密金鑰和相關聯憑證的參考。在 SSL/TLS 卸載期間，您的 Web 伺服器會使用該 AWS CloudHSM KeyStore 檔案來識別 HSM 上的私密金鑰。

主題

- 產生私有金鑰
- 產生自簽憑證。

產生私有金鑰

本節說明如何使用 JDK 產生金鑰配對。 KeyTool 在 HSM 內部產生 key pair 之後，您可以將其匯出為 KeyStore 檔案，然後產生對應的憑證。

根據您的使用案例，您可以產生 RSA 或 EC 金鑰對。下列步驟說明如何產生 RSA 金鑰對。

使用中的**genkeypair**指 KeyTool 令產生 RSA key pair

1. 用特定數據替換以下*<VARIABLES>*後，請使用下列命令產生名為 `jsse_keystore.keystore` 的金鑰存放區檔案，該檔案將具有 HSM 上私有金鑰的參考。

```
$ 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>***：您要生成金鑰存放區檔案的路徑。
- ***<UNIQUE ALIAS FOR KEYS>***：這是用於唯一識別 HSM 上的金鑰。此別名將被設置為金鑰的 LABEL 屬性。
- ***<KEY PASSWORD>***：我們會將您金鑰的參考儲存在本機金鑰儲存檔案中，而此密碼會保護該本機參考。
- ***<KEYSTORE PASSWORD>***：這是本機金鑰存放區檔案的密碼。
- ***<JSSE KEYSTORE NAME>***：金鑰存放區檔案的名稱。
- ***<CERT DOMAIN NAME>***：X.500 辨別名稱。
- ***<KEY ALGORITHM>***：生成金鑰對的金鑰算法（例如，RSA 和 EC）。
- ***<KEY SIZE>***：要產生金鑰對的金鑰大小 (例如，2048、3072 和 4096)。
- ***<SIGN ALGORITHM>***：要產生金鑰對的金鑰大小 (例如，SHA1withRSA、SHA224withRSA、SHA256withRSA、SHA384withRSA 和 SHA512withRSA)。

2. 若要確認命令是否成功，請輸入下列命令，並確認您是否已成功產生 RSA 金鑰對。

```
$ ls <PATH>/<JSSE KEYSTORE NAME>.keystore
```

產生自簽憑證。

產生私有金鑰與金鑰存放區檔案後，您可以使用此檔案來產生憑證簽署要求 (CSR) 和憑證。

在生產環境中，您通常會使用憑證授權單位 (CA) 從 CSR 建立憑證。測試環境不需要 CA。如果您確實使用 CA，請將 CSR 檔案發送給他們，並使用他們在 Web 伺服器中為您提供 HTTPS 的簽名 SSL/TLS 憑證。

作為使用 CA 的替代方法，您可以使用建 KeyTool 立自我簽署憑證。自簽憑證不受瀏覽器所信任，請勿用於生產環境。可以用於測試環境。

> ⚠ Warning
>
> 自簽憑證應該只用於測試環境。若為生產環境，請使用更安全的方法 (例如憑證授權單位) 來建立憑證。

產生憑證

1. 取得先前步驟中產生的金鑰存放區檔案的副本。
2. 執行下列命令以使用 KeyTool 建立憑證簽署要求 (CSR)。

```
$ 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**
>
> 憑證簽署請求的輸出檔案為 `certreq.csr`。

簽署憑證

- 用特定數據替換以下*<VARIABLES>*後，請執行下列命令，以在 HSM 上的私有金鑰簽署 CSR。這會建立自簽憑證。

```
$ 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` 是使用別名私有金鑰的簽署憑證。

在金鑰存放區中匯入憑證

- 用特定數據替換以下*<VARIABLES>*後，請執行下列命令，將已簽署的憑證匯入為受信任的憑證。此步驟會將憑證儲存在別名所識別的金鑰存放區項目中。

```
$ 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>
```

將憑證轉換為 PEM

- 執行下列命令，將已簽署的憑證檔案 (.crt) 轉換為 PEM。PEM 檔案將用於從 http 用戶端發送請求。

```
$ openssl x509 -inform der -in certificate.crt -out certificate.pem
```

完成這些步驟之後，前往步驟 3：設定 Web 伺服器。

步驟 3：設定 Tomcat Web 伺服器

更新 Web 伺服器軟體組態，以使用 HTTPS 憑證以及您在 上一個步驟中建立的對應 PEM 檔案。請記得在開始之前備份現有的憑證和金鑰。這將完成針對 SSL/TLS 卸載搭配 AWS CloudHSM設定您的 Linux Web 伺服器軟體。如需詳細資訊，請參閱 Apache Tomcat 9 組態參考。

停止伺服器

- 用特定數據替換以下*<VARIABLES>*後，請運行以下命令以在更新組態之前停止 Tomcat 伺服器。

```
$ /<TOMCAT_DIRECTORY>/bin/shutdown.sh
```

  - ***<TOMCAT_DIRECTORY>***：Tomcat 安裝目錄。

更新 Tomcat 類別路徑

1. 連接至您的用戶端執行個體。
2. 找到 Tomcat 安裝資料夾。
3. 用特定數據替換下面的*<VARIABLES>*後，使用下面的命令向 Tomcat 類路徑添加 Java 庫和 CloudHSM Java 路徑 (位於 Tomcat/bin/catalina.sh 檔案)。

```
$ 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 程式庫位置。
- ***<TOMCAT PATH>***：Tomcat 安裝資料夾。

在伺服器組態中新增 HTTPS 連接器。

1. 前往 Tomcat 安裝資料夾。
2. 用特定數據替換下面的*<VARIABLES>*後，請使用下列命令新增 HTTPS 連接器，以使用必要條件中產生的憑證：

```
$ 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>***：金鑰存放區檔案所在的目錄。
- ***<JSSE KEYSTORE NAME>***：金鑰存放區檔案的名稱。
- ***<KEYSTORE PASSWORD>***：這是本機金鑰存放區檔案的密碼。
- ***<KEY PASSWORD>***：我們會將您金鑰的參考儲存在本機金鑰儲存檔案中，而此密碼會保護該本機參考。
- ***<UNIQUE ALIAS FOR KEYS>***：這是用於唯一識別 HSM 上的金鑰。此別名將被設置為金鑰的 LABEL 屬性。
- ***<TOMCAT PATH>***：Tomcat 資料夾的路徑。

啟動伺服器

- 用特定數據替換以下*<VARIABLES>*後，使用以下命令啟動 Tomcat 伺服器：

```
$ /<TOMCAT DIRECTORY>/bin/startup.sh
```

> **Note**
>
> ***<TOMCAT DIRECTORY>***是您的 Tomcat 安裝目錄的名稱。

在更新您的 Web 伺服器組態之後，請前往 步驟 4：啟用 HTTPS 流量並驗證憑證。

步驟 4：啟用 HTTPS 流量並驗證憑證

在您將 Web 伺服器設定為 SSL/TLS 卸載使用之後 AWS CloudHSM，請將 Web 伺服器執行個體新增至允許輸入 HTTPS 流量的安全性群組。這可讓用戶端 (例如 Web 瀏覽器) 與 Web 伺服器建立 HTTPS 連線。然後建立 HTTPS 連接到您的 Web 服務器，並驗證它使用的是您為 SSL/TLS 卸載配置的證書。AWS CloudHSM

主題

- 啟用傳入 HTTPS 連線
- 驗證 HTTPS 是否使用您已設定的憑證

啟用傳入 HTTPS 連線

若要從用戶端 (例如 Web 瀏覽器) 連接到 Web 伺服器，請建立允許傳入 HTTPS 連接的安全群組。特別是應該允許連接埠 443 上的傳入 TCP 連線。將此安全群組指派到您的 Web 伺服器。

建立 HTTPS 的安全群組並將其指派至您的 Web 伺服器

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在導覽窗格中選擇安全群組。
3. 選擇建立安全群組。
4. 對於 Create Security Group (建立安全群組)，執行下列動作：
    a. 對於 Security group name (安全群組名稱)，輸入您要建立之安全群組的名稱。
    b. (選用) 輸入您要建立之安全群組的描述。
    c. 對於 VPC，選擇包含 Web 伺服器 Amazon EC2 執行個體的 VPC。
    d. 選取 Add Rule (新增規則)。
    e. 對於類型，從下拉式視窗中選取 HTTPS。
    f. 對於來源，輸入來源位置。
    g. 選擇建立安全群組。
5. 在導覽窗格中，選擇執行個體。
6. 選取 Web 伺服器執行個體旁的核取方塊。
7. 選取頁面頂端的動作下拉式選單。選取安全性，然後選取變更安全群組。

8. 對於關聯的安全群組，請選取搜尋方塊，然後選取您為 HTTPS 建立之安全群組。然後選擇新增安全群組。
9. 選取 Save (儲存)。

驗證 HTTPS 是否使用您已設定的憑證

將 Web 伺服器新增至安全群組之後，您可以驗證 SSL/TLS 卸載是否正在使用自簽憑證。若要這樣做，您可以使用 Web 瀏覽器或使用 OpenSSL s_client 之類的工具。

使用 Web 瀏覽器驗證 SSL/TLS 卸載

1. 使用 Web 瀏覽器來透過伺服器的公有 DNS 名稱或 IP 地址連接到您的 Web 伺服器。請確定網址列中的 URL 開頭為 https://。例如 **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**。

   > **Tip**
   >
   > 您可以使用 DNS 服務 (例如 Amazon Route 53)，將網站的網域名稱 (例如，https://www.example.com/) 路由到 Web 伺服器。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》或 DNS 服務文件中的將流量路由到 Amazon EC2 執行個體。

2. 使用您的 Web 瀏覽器來檢視 Web 伺服器憑證。如需詳細資訊，請參閱下列內容：

   - 若為 Mozilla Firefox，請參閱 Mozilla 技術支援網站上的檢視憑證。
   - 若為 Google Chrome，請參閱 Google Web 開發人員工具網站上的了解安全問題。

   其他 Web 瀏覽器可能有類似的功能，可供您用來檢視 Web 伺服器憑證。
3. 確保 SSL/TLS 憑證是您設定 Web 伺服器所要使用的憑證。

使用 OpenSSL s_client 來驗證 SSL/TLS 卸載

1. 執行以下 OpenSSL 命令來使用 HTTPS 連接至 Web 伺服器。以 Web 伺服器的公有 DNS 名稱或 IP 地址來取代*<#####>*。

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> 您可以使用 DNS 服務 (例如 Amazon Route 53)，將網站的網域名稱 (例如，https://www.example.com/) 路由到 Web 伺服器。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》或 DNS 服務文件中的將流量路由到 Amazon EC2 執行個體。

2. 確保 SSL/TLS 憑證是您設定 Web 伺服器所要使用的憑證。

您現在有透過 HTTPS 而受到保護的網站。Web 伺服器的私密金鑰會儲存在 AWS CloudHSM 叢集中的 HSM 中。

若要新增負載平衡器，請參閱 新增具有 Elastic Load Balancing (選用) 的負載平衡器。

在 Windows 上使用帶有 CNG 的 IIS 进行 SSL/TLS 卸载

本教學課程提供 step-by-step 如何在 Windows 網頁伺服器上設定 SSL/TLS 卸載 AWS CloudHSM 的指示。

主題

概觀

在 Windows 上，Internet Information Services (IIS) for Windows Server Web 伺服器應用程式原本就支援 HTTPS。適用於 Microsoft 加密 API 之AWS CloudHSM 金鑰儲存供應商 (KSP)：下一代 (CNG) 提供的界面，可讓 IIS 在您的叢集中使用 HSM，進行密碼編譯卸載和金鑰儲存。 AWS CloudHSM KSP 是將 IIS 連接到 AWS CloudHSM 叢集的橋接器。

此教學課程會讓您了解如何執行以下操作：

- 在 Amazon EC2 執行個體上安裝 Web 伺服器軟體。
- 將 Web 伺服器軟體設定為使用儲存在 AWS CloudHSM 叢集中的私有金鑰來支援 HTTPS。

- (選用) 使用 Amazon EC2 建立第二個 Web 伺服器執行個體，并使用 Elastic Load Balancing 建立負載平衡器。使用負載平衡器可將負載分散到多部伺服器，進而提升效能。它也可以在一或多個伺服器失敗時提供備援和高可用性。

當您準備好開始時，請移至步驟 1：設定先決條件。

步驟 1：設定先決條件

若要設定網頁伺服器 SSL/TLS 卸載 AWS CloudHSM，您需要下列項目：

- 至少具有一個 HSM 的作用中 AWS CloudHSM 叢集。
- 執行 Windows 作業系統的 Amazon EC2 執行個體，其中已安裝下列軟體：
  - 適用於視窗的 AWS CloudHSM 用戶端軟體。
  - Internet Information Services (IIS) for Windows Server。
- 加密使用者 (CU)，擁有及管理 HSM 上 Web 伺服器的私有金鑰。

> **Note**
>
> 本教學課程使用 Microsoft Windows Server 2016。也支援 Microsoft Windows Server 2012，但不支援 Microsoft Windows Server 2012 R2。

設定 Windows 伺服器執行個體，並在 HSM 上建立 CU

1. 完成「開始使用」中的步驟。當您啟動 Amazon EC2 用戶端時，請選擇 Windows Server 2016 或 Windows Server 2012 AMI。當您完成這些步驟，便擁有至少包含一個 HSM 的作用中叢集。您也有一個執行 Windows 伺服器且已安裝適用於 Windows 的 AWS CloudHSM 用戶端軟體的 Amazon EC2 用戶端執行個體。
2. (選用) 在您的叢集中新增更多 HSM。如需詳細資訊，請參閱 新增 HSM。
3. 連接至 Windows 伺服器。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到您的執行個體。
4. 使用 CloudHSM CLI 建立加密使用者 (CU)。保持追蹤 CU 使用者名稱和密碼。您需要它們來完成下一個步驟。

> **Note**
>
> 如需如何建立使用者的詳細資訊，請參閱使用 CloudHSM CLI 管理 HSM 使用者。

5. 使用您在先前步驟中建立的 CU 使用者名稱和密碼，設定 HSM 的登入資料。
6. 在步驟 5 中，如果您使用 Windows 身份證明管理員來設定 HSM 認證，請psexec.exe從下載以 SysInternals 以 NT 授權單位\ SYSTEM 身分執行下列命令：

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
 --username <USERNAME> --password <PASSWORD>
```

以 HSM 憑證取代*<#####>*和*<##>*。

在 Windows 伺服器上安裝 IIS

1. 如果您尚未這麼做，請連接至您的 Windows 伺服器。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到您的執行個體。
2. 在 Windows 伺服器上，啟動 Server Manager (伺服器管理員)。
3. 在 Server Manager (伺服器管理員) 儀表板中，選擇 Add roles and features (新增角色和功能)。
4. 閱讀 Before you begin (開始之前) 資訊，然後選擇 Next (下一步)。
5. 對於 Installation Type (安裝類型)，選擇 Role-based or feature-based installation (角色型或功能型安裝)。然後選擇下一步。
6. 對於 Server Selection (伺服器選項)，選擇 Select a server from the server pool (從伺服器集區選取伺服器)。然後選擇下一步。
7. 對於 Server Roles (伺服器角色)，執行下列動作：
   a. 選取 Web Server (IIS) (Web 伺服器 (IIS))。
   b. 對於 Add features that are required for Web Server (IIS) (新增 Web 伺服器 (IIS) 需要的功能)，選擇 Add Features (新增功能)。
   c. 選擇 Next (下一步) 來完成伺服器角色的選取。
8. 針對 Features (功能)，接受預設。然後選擇下一步。
9. 閱讀 Web Server Role (IIS) (Web 伺服器角色 (IIS)) 資訊。然後選擇下一步。
10. 對於 Select role services (選取角色服務)，接受預設值或變更為偏好的設定。然後選擇下一步。

11. 針對 Confirmation (確認)，閱讀確認資訊。然後選擇 Install (安裝)。
12. 完成安裝之後，請選擇 Close (關閉)。

完成這些步驟之後，請移至 步驟 2：建立憑證簽署要求 (CSR) 和憑證。

步驟 2：建立憑證簽署要求 (CSR) 和憑證

若要啟用 HTTPS，您的 Web 伺服器需要 SSL/TLS 憑證和對應私有金鑰。若要搭配使用 SSL/TLS 卸載 AWS CloudHSM，請將私密金鑰儲存在叢集中的 HSM 中。 AWS CloudHSM 為此，請使用適用於 Microsoft 密碼編譯 API 的AWS CloudHSM 金鑰儲存供應商 (KSP)：下一代 (CNG) 建立憑證簽署請求 (CSR)。然後，將 CSR 提供給憑證授權機構 (CA)，其會簽署 CSR 以產生憑證。

主題

- 建立 CSR
- 取得已簽署的憑證並匯入它

建立 CSR

使用視窗伺服器上的 AWS CloudHSM KSP 來建立企業社會責任。

若要建立 CSR

1. 如果您尚未這麼做，請連接至您的 Windows 伺服器。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到您的執行個體。
2. 使用下列命令啟動用 AWS CloudHSM 戶端常駐程式。

   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

- 用於 Windows 用戶端 1.1.2+：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 用於 Windows 用戶端 1.1.1 和更早版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

3. 在 Windows 伺服器上，使用文字編輯器來建立名為 `IISCertRequest.inf` 的憑證要求檔案。下列示範範例 `IISCertRequest.inf` 檔案的內容。如需您可在檔案中指定之區段、金鑰和數值的詳細資訊，請參閱 Microsoft 文件。請勿變更 `ProviderName` 值。

```
[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. 使用 Windows certreq 命令，從您在前一個步驟中建立的 IISCertRequest.inf 檔案建立 CSR。以下範例會將 CSR 儲存至名為 IISCertRequest.csr 的檔案。如果您為憑證要求檔案使用不同的檔案名稱，請以適當的檔案名稱取代 *IIS CertRequest .inf*。您可以選擇性地將 *IIS CertRequest .csr* 取代為 CSR 檔案的不同檔案名稱。

```
C:\>certreq -new IISCertRequest.inf IISCertRequest.csr
        SDK Version: 2.03

CertReq: Request Created
```

   IISCertRequest.csr 檔案包含您的 CSR。您需要此 CSR 才能取得已簽署的憑證。

取得已簽署的憑證並匯入它

在生產環境中，您通常會使用憑證授權單位 (CA) 從 CSR 建立憑證。測試環境不需要 CA。如果您使用 CA，請將 CSR 檔案 (IISCertRequest.csr) 傳送至其中，並使用 CA 來建立已簽署的 SSL/TLS 憑證。

除了使用 CA，您也可以使用 OpenSSL 之類的工具來建立自簽憑證。

> **Warning**
>
> 自簽憑證不受瀏覽器所信任，請勿用於生產環境。可以用於測試環境。

以下程序示範如何建立自簽憑證，並使用它來簽署 Web 伺服器 CSR。

建立自簽憑證

1. 使用下列 OpenSSL 命令來建立私有金鑰。您可以選擇性地將 *SelfSignedCA.key* 取代為包含私密金鑰的檔案名稱。

```
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. 使用以下 OpenSSL 命令，以使用您在前一個步驟中建立之私有金鑰來建立自簽憑證。這是互動式命令。請閱讀畫面上的指示，並依照提示操作。將 *SelfSignedCA.key* 取代為包含私密金鑰的檔案名稱 (如果不同的話)。您可以選擇性地以檔案名稱取代 *SelfSignedCA.CRT*，以包含您的自我簽署憑證。

```
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 []:
```

使用自簽憑證來簽署您的 Web 伺服器 CSR

- 使用以下 OpenSSL 命令來使用私有金鑰和自簽憑證以簽署 CSR。將以下名稱取代為包含對應資料之檔案的名稱 (若不同的話)。
  - *IIS CertRequest .csr* — 包含網頁伺服器 CSR 的檔案名稱
  - *SelfSignedCa.crt* — 包含您自我簽署憑證的檔案名稱
  - *SelfSignedCA.key* — 包含您私密金鑰的檔案名稱
  - *IISCert.crt*：包含 Web 伺服器已簽署憑證之檔案的名稱

```
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:
```

在完成前一個步驟之後，您有一個適用於 Web 伺服器的自簽憑證 (IISCert.crt) 和自簽憑證 (SelfSignedCA.crt)。當您具有這些檔案，請移至步驟 3：設定 Web 伺服器。

步驟 3：設定 Web 伺服器

更新您的 IIS網站組態，來使用您在前一個步驟結束時建立的 HTTPS 憑證。這將完成針對 SSL/TLS 卸載搭配 AWS CloudHSM設定您的 Windows Web 伺服器軟體 (IIS)。

如果已使用自簽憑證來簽署 CSR，則您必須先將自簽憑證匯入至 Windows 受信任的根認證授權單位。

將自簽憑證匯入至 Windows 受信任的根認證授權單位

1. 如果您尚未這麼做，請連接至您的 Windows 伺服器。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到您的執行個體。
2. 將自簽憑證複製到 Windows 伺服器。
3. 在 Windows 伺服器上，開啟 Control Panel (控制台)。
4. 針對 Search Control Panel (搜尋控制台)，輸入 **certificates**。然後選擇 Manage computer certificates (管理電腦憑證)。
5. 在 Certificates - Local Computer (憑證 - 本機電腦) 視窗中，按兩下 Trusted Root Certification Authorities (受信任的根認證授權單位)。
6. 用滑鼠右鍵按一下 Certificates (憑證)，然後選擇 All Tasks (所有工作)、Import (匯入)。
7. 在 Certificate Import Wizard (憑證匯入精靈) 中，選擇 Next (下一步)。
8. 選擇 Browse (瀏覽)，然後尋找並選取您的自簽憑證。如果您已遵循本教學課程前一個步驟中的指示來建立自簽憑證，則您的自簽憑證即名為 SelfSignedCA.crt。選擇 Open (開啟)。
9. 選擇下一步。

10. 對於 Certificate Store (憑證存放區)，選擇 Place all certificates in the following store (將所有憑證放入以下存放區)。然後，確保選取 Trusted Root Certification Authorities (受信任的根認證授權單位) 做為 Certificate store(憑證存放區)。
11. 選擇 Next (下一步) ，然後選擇 Finish (完成)。

更新 IIS 網站組態

1. 如果您尚未這麼做，請連接至您的 Windows 伺服器。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到您的執行個體。
2. 啟動 AWS CloudHSM 用戶端常駐程式。
3. 將 Web 伺服器的已簽署憑證 (這是您在本教學課程前一個步驟結束時建立的憑證) 複製到 Windows 伺服器。
4. 在您的 Windows 伺服器上，使用 Windows certreq 命令，來接受已簽署的憑證，如以下範例所示。將 *IISCert.crt* 取代為包含 Web 伺服器的已簽署憑證之檔案的名稱。

```
C:\>certreq -accept IISCert.crt
        SDK Version: 2.03
```

5. 在 Windows 伺服器上，啟動 Server Manager (伺服器管理員)。
6. 在 Server Manager (伺服器管理員) 儀表板右上角，選擇 Tools (工具)、Internet Information Services (IIS) Manager (Internet Information Services (IIS) 管理員)。
7. 在 Internet Information Services (IIS) Manager (Internet Information Services (IIS) 管理員) 視窗中，按兩下您的伺服器名稱。然後按兩下 Sites (網站)。選取網站。
8. 選取 SSL Settings (SSL 設定)。然後，在視窗右側選擇 Bindings (繫結)。
9. 在 Site Bindings (網站繫結) 視窗中，選擇 Add (新增)。
10. 對於 Type (類型)，選擇 https (https)。對於 SSL certificate (SSL 憑證)，選擇您在本教學課程前一個步驟結束時建立的 HTTPS 憑證。

> **Note**
>
> 如果您在此次憑證繫結中遇到錯誤，請重新啟動您的伺服器並重試此步驟。

11. 選擇確定。

在更新網站組態之後，請移至步驟 4：啟用 HTTPS 流量並驗證憑證。

步驟 4：啟用 HTTPS 流量並驗證憑證

在您設定 SSL/TLS 卸載的 Web 伺服器之後 AWS CloudHSM，請將 Web 伺服器執行個體新增至允許輸入 HTTPS 流量的安全性群組。這可讓用戶端 (例如 Web 瀏覽器) 與 Web 伺服器建立 HTTPS 連線。然後建立 HTTPS 連接到您的 Web 服務器，並驗證它使用的是您為 SSL/TLS 卸載配置的證書。AWS CloudHSM

主題

啟用傳入 HTTPS 連線

若要從用戶端 (例如 Web 瀏覽器) 連接到 Web 伺服器，請建立允許傳入 HTTPS 連接的安全群組。特別是應該允許連接埠 443 上的傳入 TCP 連線。將此安全群組指派到您的 Web 伺服器。

建立 HTTPS 的安全群組並將其指派至您的 Web 伺服器

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在導覽窗格中選擇安全群組。
3. 選擇建立安全群組。
4. 對於 Create Security Group (建立安全群組)，執行下列動作：
    a. 對於 Security group name (安全群組名稱)，輸入您要建立之安全群組的名稱。
    b. (選用) 輸入您要建立之安全群組的描述。
    c. 對於 VPC，選擇包含 Web 伺服器 Amazon EC2 執行個體的 VPC。
    d. 選取 Add Rule (新增規則)。
    e. 對於類型，從下拉式視窗中選取 HTTPS。
    f. 對於來源，輸入來源位置。
    g. 選擇建立安全群組。
5. 在導覽窗格中，選擇執行個體。
6. 選取 Web 伺服器執行個體旁的核取方塊。
7. 選取頁面頂端的動作下拉式選單。選取安全性，然後選取變更安全群組。
8. 對於關聯的安全群組，請選取搜尋方塊，然後選取您為 HTTPS 建立之安全群組。然後選擇新增安全群組。

9. 選取 Save (儲存)。

驗證 HTTPS 是否使用您已設定的憑證

將 Web 伺服器新增至安全群組之後，您可以驗證 SSL/TLS 卸載是否正在使用自簽憑證。若要這樣做，您可以使用 Web 瀏覽器或使用 OpenSSL s_client 之類的工具。

使用 Web 瀏覽器驗證 SSL/TLS 卸載

1. 使用 Web 瀏覽器來透過伺服器的公有 DNS 名稱或 IP 地址連接到您的 Web 伺服器。請確定網址列中的 URL 開頭為 https://。例如 **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**。

   > **Tip**
   >
   > 您可以使用 DNS 服務 (例如 Amazon Route 53)，將網站的網域名稱 (例如，https://www.example.com/) 路由到 Web 伺服器。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》或 DNS 服務文件中的將流量路由到 Amazon EC2 執行個體。

2. 使用您的 Web 瀏覽器來檢視 Web 伺服器憑證。如需詳細資訊，請參閱下列內容：

   - 若為 Mozilla Firefox，請參閱 Mozilla 技術支援網站上的檢視憑證。
   - 若為 Google Chrome，請參閱 Google Web 開發人員工具網站上的了解安全問題。

   其他 Web 瀏覽器可能有類似的功能，可供您用來檢視 Web 伺服器憑證。

3. 確保 SSL/TLS 憑證是您設定 Web 伺服器所要使用的憑證。

使用 OpenSSL s_client 來驗證 SSL/TLS 卸載

1. 執行以下 OpenSSL 命令來使用 HTTPS 連接至 Web 伺服器。以 Web 伺服器的公有 DNS 名稱或 IP 地址來取代*<#####>*。

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> 您可以使用 DNS 服務 (例如 Amazon Route 53)，將網站的網域名稱 (例如，https://www.example.com/) 路由到 Web 伺服器。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》或 DNS 服務文件中的將流量路由到 Amazon EC2 執行個體。

2. 確保 SSL/TLS 憑證是您設定 Web 伺服器所要使用的憑證。

您現在有透過 HTTPS 而受到保護的網站。Web 伺服器的私密金鑰會儲存在 AWS CloudHSM 叢集中的 HSM 中。

若要新增負載平衡器，請參閱 新增具有 Elastic Load Balancing (選用) 的負載平衡器。

新增具有 Elastic Load Balancing (選用) 的負載平衡器

對一個 Web 伺服器設定 SSL/TLS 卸載之後，您可以建立更多個 Web 伺服器和 Elastic Load Balancing 負載平衡器，將 HTTPS 流量路由到 Web 伺服器。負載平衡器可以平衡兩個或更多個伺服器之間的流量，以降低個別 Web 伺服器的負載。還可以提高網站的可用性，因為負載平衡器會監控 Web 伺服器的運作狀態，只將流量路由到運作狀態良好的伺服器。如果 Web 伺服器故障，負載平衡器會自動停止路由流量。

主題

- 針對第二個 Web 伺服器建立子網路
- 建立第二個 Web 伺服器
- 建立負載平衡器

針對第二個 Web 伺服器建立子網路

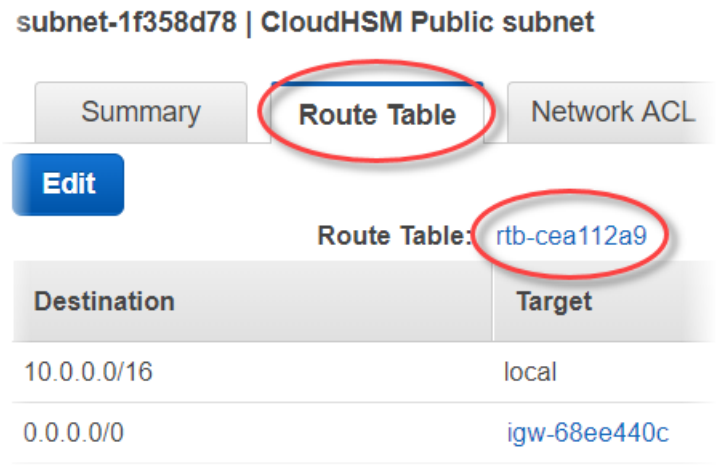
在建立其他 Web 伺服器之前，您需要在包含現有 Web 伺服器和 AWS CloudHSM 叢集的相同 VPC 中建立新子網路。

建立新的子網路

1. 開啟 Amazon VPC 主控台的子網路區段。
2. 選擇 Create Subnet (建立子網路)。
3. 在 Create Subset (建立子網路) 對話方塊中，執行下列動作：

a. 在 Name tag (名稱標籤) 中，輸入子網路的名稱。
b. 對於 VPC，請選擇包含現有 Web 伺服器和 AWS CloudHSM 叢集的 AWS CloudHSM VPC。
c. 在 Availability Zone (可用區域) 中，選擇一個不是現有 Web 伺服器所在的可用區域。
d. 在 IPv4 CIDR 區塊中，輸入要用於子網路 CIDR 區塊。例如，輸入 **`10.0.10.0/24`**。
e. 選擇 Yes, Create (是，建立)。

4. 選取包含現有 Web 伺服器之公有子網路旁的核取方塊。這與您在先前步驟中建立的公有子網路不同。
5. 在內容窗格中，選擇路由表標籤。然後選擇路由表的連結。



6. 選取路由表旁的核取方塊。
7. 選擇子網路關聯標籤。然後選擇 Edit (編輯)。
8. 選取您先前在此程序中建立之公用子網路旁的核取方塊。然後選擇 Save (儲存)。

建立第二個 Web 伺服器

完成下列步驟，使用與現有 Web 伺服器相同的組態來建立第二個 Web 伺服器。

建立第二個 Web 伺服器

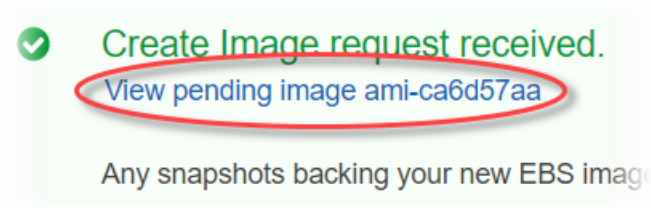
1. 開啟 Amazon EC2 主控台的執行個體區段。
2. 選取現有 Web 伺服器執行個體旁的核取方塊。
3. 依序選擇 Actions (動作)、Image (映像)、Create Image (建立映像)。
4. 在 Create Image (建立映像) 對話方塊中，執行下列動作：

a. 在 Image Name (映像名稱) 中，輸入映像的名稱。

b. 在 Image description (映像描述) 中，輸入映像的描述。

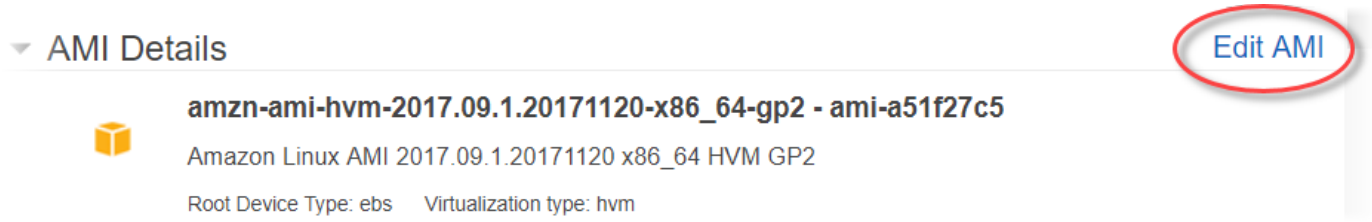
c. 選擇 Create Image (建立映像)。這個動作會重新啟動現有的 Web 伺服器。

d. 選擇檢視待處理映像 ami-***<AMI ID>***連結。



在狀態欄，注意您的映像狀態。當您的映像狀態是 available (可用) 時 (這可能需要幾分鐘)，請移至下一個步驟。

5. 在導覽窗格中，選擇執行個體。
6. 選取現有 Web 伺服器旁的核取方塊。
7. 選擇 Actions (動作)，然後選擇 Launch More Like This (啟動更多類似項目)。
8. 選擇 Edit AMI (編輯 AMI)。



9. 在左側導覽窗格中選擇我的 AMI。然後清除搜尋方塊中的文字。
10. 在 Web 伺服器映像旁，選擇 Select (選取)。
11. 選擇是，我想要繼續此 AMI (***<####>*** - ami-***<AMI ID>***)。
12. 選擇下一步。
13. 選取執行個體類型，然後選擇 Next: Configure Instance Details (下一步：設定執行個體的詳細資訊)。
14. 在 Step 3: Configure Instance Details (步驟 3：設定執行個體詳細資訊) 中，執行下列動作：

    a. 在 Network (網路) 中，選擇包含現有 Web 伺服器的 VPC。

    b. 在 Subnet (子網路) 中，選擇您針對第二個 Web 伺服器建立的公有子網路。

    c. 在 Auto-assign Public IP (自動指派公有 IP) 中，選擇 Enable (啟用)。

    d. 依喜好變更剩餘的執行個體詳細資訊。然後選擇 Next: Add Storage (下一步：新增儲存體)。
15. 依喜好變更儲存設定。然後選擇 Next: Add Tags (下一步：新增標籤)。
16. 依喜好新增或編輯標籤。然後選擇 Next: Configure Security Group (下一步：設定安全群組)。

17. 在 Step 6: Configure Security Group (步驟 6：設定安全群組) 中，執行下列動作：

    a. 在 Assign a security group (指派安全群組) 中，選擇 Select an existing security group (選取現有的安全群組)。

    b. 選取名為 cloudhsm-***<## ID>***-sg 的安全群組旁邊的核取方塊。 AWS CloudHSM 在建立叢集時代表您建立此安全群組。您必須選擇此安全群組，Web 伺服器執行個體才能連接到叢集內的 HSM。

    c. 選取允許傳入 HTTPS 流量之安全群組旁的核取方塊。您先前已建立此安全群組。

    d. (選用) 選取允許從您的網路傳入 SSH (Linux) 或 RDP (Windows) 流量之安全群組旁的核取方塊。也就是說，安全群組必須允許連接埠 22 (適用於 Linux 上的 SSH) 或連接埠 3389 (適用於 Windows 上的 RDP) 上的傳入 TCP 流量。否則，您無法連接到用戶端執行個體。如果您沒有像這樣的安全群組，則必須建立一個安全群組，並於稍後指派給用戶端執行個體。

    選擇 Review and Launch (檢閱和啟動)。

18. 檢閱您的執行個體詳細資訊，然後選擇 Launch (啟動)。

19. 選擇是否以現有的金鑰對啟動執行個體、建立新的金鑰對，或在沒有金鑰對的情況下啟動執行個體。

    - 若要使用現有的金鑰對，請執行下列動作：
      1. 選擇 Choose an existing key pair (選擇現有金鑰對)。
      2. 在 Select a key pair (選取金鑰對) 中，選擇要使用的金鑰對。
      3. 選取我知道我能夠使用所選取的私有金鑰檔案 (***<########>***.pem)，也知道如果沒有此檔案，我將無法登入我的執行個體) 旁的核取方塊。
    - 若要建立新的金鑰對，請執行下列動作：
      1. 選擇 Create a new key pair (建立新的金鑰對)。
      2. 在 Key pair name (金鑰對名稱) 中，輸入金鑰對名稱。
      3. 選擇 Download Key Pair (下載金鑰對)，然後將私有金鑰檔案儲存到安全又可存取的位置。

      > ⚠ Warning
      >
      > 此後，您就無法再次下載私有金鑰檔案。如果現在不下載私有金鑰檔案，您將無法存取用戶端執行個體。

    - 若要在沒有金鑰對的情況下啟動執行個體，請執行下列動作：
      1. 選擇 Proceed without a key pair (不使用金鑰對而繼續)。

2. 選取 I acknowledge that I will not be able to connect to this instance unless I already know the password built into this AMI (我知道除非我已擁有內建於此 AMI 的密碼，否則我將無法連接至此執行個體) 旁的核取方塊。

選擇 Launch Instances (啟動執行個體)。

建立負載平衡器

完成以下步驟來建立 Elastic Load Balancing 負載平衡器，將 HTTPS 流量路由到您的 Web 伺服器。

建立負載平衡器

1. 開啟 Amazon EC2 主控台中的負載平衡器區段。
2. 選擇 Create Load Balancer (建立負載平衡器)。
3. 在 Network Load Balancer (網路負載平衡器) 區段中，選擇 Create (建立)。
4. 在 Step 1: Configure Load Balancer (步驟 1：設定負載平衡器) 中，執行下列動作：
   a. 在 Name (名稱) 中，輸入您要建立之負載平衡器的名稱。
   b. 在接聽程式區段中，將負載平衡器連接埠的值改為 **443**。
   c. 在 Availability Zones (可用區域) 區段的 VPC 中，選擇包含您的 Web 伺服器的 VPC。
   d. 在 Availability Zones (可用區域) 區段中，選擇包含您 Web 伺服器的子網路。
   e. 選擇 Next: Configure Routing (下一步：設定路由)。
5. 在 Step 2: Configure Routing (步驟 2：設定路由) 中，執行下列動作：
   a. 在 Name (名稱) 中，輸入您要建立之目標群組的名稱。
   b. 將連接埠的值改為 **443**。
   c. 選擇 Next: Register Targets (下一步：註冊目標)。
6. 在 Step 3: Register Targets (步驟 3：註冊目標) 中，執行下列動作：
   a. 在執行個體區段中，選取您 Web 伺服器執行個體旁的核取方塊。然後選擇 Add to registered (新增至已註冊)。
   b. 選擇下一步：檢閱。
7. 檢閱負載平衡器詳細資訊，然後選擇 Create (建立)。
8. 已成功建立負載平衡器時，請選擇 Close (關閉)。

當您完成前面的步驟後，Amazon EC2 主控台會顯示您的 Elastic Load Balancing 負載平衡器。

當負載平衡器的狀態為作用中，您可以確認負載平衡器是否在運作。也就是說，您可以確認負載平衡器是否將 HTTPS 流量傳送到 Web 伺服器 (使用 SSL/TLS 卸載搭配 AWS CloudHSM)。若要這樣做，您可以使用 Web 瀏覽器或 OpenSSL s_client 之類的工具。

使用 Web 瀏覽器來確認負載平衡器是否在運作

1. 在 Amazon EC2 主控台，尋找您剛建立之負載平衡器的 DNS 名稱。然後選取 DNS 名稱並將其複製。
2. 透過 Web 瀏覽器 (例如 Mozilla Firefox 或 Google Chrome)，使用負載平衡器的 DNS 名稱來連接到負載平衡器。請確定網址列中的 URL 開頭為 https://。

   > **Tip**
   >
   > 您可以使用 DNS 服務 (例如 Amazon Route 53)，將網站的網域名稱 (例如，https://www.example.com/) 路由到 Web 伺服器。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》或 DNS 服務文件中的將流量路由到 Amazon EC2 執行個體。

3. 使用您的 Web 瀏覽器來檢視 Web 伺服器憑證。如需詳細資訊，請參閱下列內容：

   - 若為 Mozilla Firefox，請參閱 Mozilla 技術支援網站上的檢視憑證。
   - 若為 Google Chrome，請參閱 Google Web 開發人員工具網站上的了解安全問題。

   其他 Web 瀏覽器可能有類似的功能，可供您用來檢視 Web 伺服器憑證。
4. 請確定憑證是您將 Web 伺服器設定為要使用的憑證。

使用 OpenSSL s_client 來驗證負載平衡器是否在運作

1. 使用以下 OpenSSL 命令使用 HTTPS 連接負載平衡器。以負載平衡器的 DNS 名稱取代 *<DNS ##>*。

```
openssl s_client -connect <DNS name>:443
```

> **Tip**
>
> 您可以使用 DNS 服務 (例如 Amazon Route 53)，將網站的網域名稱 (例如，https://www.example.com/) 路由到 Web 伺服器。如需詳細資訊，請參閱《Amazon Route 53 開發人員指南》或 DNS 服務文件中的將流量路由到 Amazon EC2 執行個體。

2. 請確定憑證是您將 Web 伺服器設定為要使用的憑證。

您現在擁有一個使用 HTTPS 保護的網站，並將網頁伺服器的私密金鑰儲存在 AWS CloudHSM 叢集中的 HSM 中。您的網站有兩個 Web 伺服器和一個負載平衡器，有助於提升效率和可用性。

使用 AWS CloudHSM將 Windows Server 設定為憑證授權單位 (CA)

在公有金鑰基礎設施 (PKI) 中，憑證授權單位 (CA) 是發行數位憑證的受信任實體。這些數位憑證透過公有金鑰加密和數位簽章，將公有金鑰繫結至身分識別 (個人或組織)。若要操作 CA，您必須保護簽署 CA 所發行憑證的私有金鑰，以維護信任。您可以將私有金鑰儲存在 AWS CloudHSM 叢集中的 HSM 中，並使用 HSM 執行密碼編譯簽署作業。

在本教學課程中，您 AWS CloudHSM 將使用 Windows 伺服器並設定 CA。您會在 Windows 伺服器上安裝適用於 Windows 的 AWS CloudHSM 用戶端軟體，然後將 Active Directory Certificate Services (AD CS) 角色新增到您的 Windows Server。設定此角色時，您可以使用 AWS CloudHSM 金鑰儲存區提供者 (KSP) 在 AWS CloudHSM 叢集上建立並儲存 CA 的私密金鑰。KSP 是將 Windows 伺服器連接到 AWS CloudHSM 叢集的橋接器。在最後一個步驟，您會使用 Windows Server CA 來簽署憑證簽署要求 (CSR)。

如需詳細資訊，請參閱下列主題：

主題

- Windows Server CA 步驟 1：設定先決條件
- Windows Server CA 步驟 2：建立 Windows Server CA 與 AWS CloudHSM
- 視窗伺服器 CA 步驟 3：使用您的視窗伺服器 CA 簽署憑證簽署要求 (CSR) AWS CloudHSM

Windows Server CA 步驟 1：設定先決條件

若要使用將 Windows 伺服器設定為憑證授權單位 (CA) AWS CloudHSM，您需要下列項目：

- 至少具有一個 HSM 的作用中 AWS CloudHSM 叢集。
- 執行 Windows 伺服器作業系統且已安裝適用於 Windows 的 AWS CloudHSM 用戶端軟體的 Amazon EC2 執行個體。本教學課程使用 Microsoft Windows Server 2016。
- 一位密碼編譯使用者 (CU)，擁有及管理 HSM 上的 CA 私有金鑰。

若要以下列方式設定 Windows 伺服器 CA 的必要條件 AWS CloudHSM

1. 完成「開始使用」中的步驟。當您啟動 Amazon EC2 用戶端時，請選擇 Windows Server AMI。本教學課程使用 Microsoft Windows Server 2016。當您完成這些步驟，便擁有至少包含一個 HSM 的作用中叢集。您也有一個執行 Windows 伺服器且已安裝適用於 Windows 的 AWS CloudHSM 用戶端軟體的 Amazon EC2 用戶端執行個體。
2. (選用) 在您的叢集中新增更多 HSM。如需詳細資訊，請參閱 新增 HSM。
3. 連接至您的用戶端執行個體。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到您的執行個體。
4. 透過使用 CloudHSM CLI 管理 HSM 使用者或使用 CloudHSM 管理公用程式 (CMU) 管理 HSM 使用者來建立加密使用者 (CU)。保持追蹤 CU 使用者名稱和密碼。您需要它們來完成下一個步驟。
5. 使用您在先前步驟中建立的 CU 使用者名稱和密碼，設定 HSM 的登入資料。
6. 在步驟 5 中，如果您使用 Windows 身份證明管理員來設定 HSM 認證，請`psexec.exe`從下載以 SysInternals 以 NT 授權單位\ SYSTEM 身分執行下列命令：

   ```
   psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
    --username <USERNAME> --password <PASSWORD>
   ```

   以 HSM 憑證取代*<#####>*和*<##>*。

若要使用建立視窗伺服器 CA AWS CloudHSM，請移至建立 Windows Server CA。

Windows Server CA 步驟 2：建立 Windows Server CA 與 AWS CloudHSM

若要建立 Windows Server CA，您可以將 Active Directory Certificate Services (AD CS) 角色新增到您的 Windows Server。新增此角色時，您可以使用 AWS CloudHSM 金鑰儲存區提供者 (KSP) 在 AWS CloudHSM 叢集上建立並儲存 CA 的私密金鑰。

> **Note**
>
> 建立您的 Windows Server CA 時，您可以選擇建立根 CA 或次級 CA。您一般會根據組織的公有金鑰基礎設施和安全政策的設計進行此決策。本教學課程說明如何建立根 CA 以簡化程序。

將 AD CS 角色新增至您的 Windows Server 和建立 CA 的私有金鑰

1. 如果您尚未這麼做，請連接至您的 Windows 伺服器。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到您的執行個體。
2. 在 Windows 伺服器上，啟動 Server Manager (伺服器管理員)。
3. 在 Server Manager (伺服器管理員) 儀表板中，選擇 Add roles and features (新增角色和功能)。
4. 閱讀 Before you begin (開始之前) 資訊，然後選擇 Next (下一步)。
5. 對於 Installation Type (安裝類型)，選擇 Role-based or feature-based installation (角色型或功能型安裝)。然後選擇下一步。
6. 對於 Server Selection (伺服器選項)，選擇 Select a server from the server pool (從伺服器集區選取伺服器)。然後選擇下一步。
7. 對於 Server Roles (伺服器角色)，執行下列動作：
   a. 選取 Active Directory Certificate Services (Active Directory Certificate Services)。
   b. 對於 Add features that are required for Active Directory Certificate Services (新增 Active Directory Certificate Services 需要的功能，選擇 Add Features (新增功能)。
   c. 選擇 Next (下一步) 來完成伺服器角色的選取。
8. 對於 Features (功能)，接受預設值，然後選擇 Next (下一步)。
9. 對於 AD CS (AD CS)，執行下列動作：
   a. 選擇下一步。
   b. 選取 Certification Authority (憑證授權單位)，然後選擇 Next (下一步)。
10. 對於 Confirmation (確認)，閱讀確認資訊，然後選擇 Install (安裝)。請勿關閉視窗。
11. 選擇反白顯示的 Configure Active Directory Certificate Services on the destination server (在目的地伺服器上設定 Active Directory Certificate Services) 連結。
12. 對於 Credentials (登入資料)，驗證或變更顯示的登入資料。然後選擇下一步。
13. 對於 Role Services (角色服務)，選取 Certification Authority (憑證授權單位)。然後選擇下一步。
14. 對於 Setup Type (設定類型)，選取 Standalone CA (獨立 CA)。然後選擇下一步。

15. 對於 CA Type (CA 類型)，選取 Root CA (根 CA)。然後選擇下一步。

> **Note**
>
> 您可以根據公有金鑰基礎設施和組織安全政策的設計，選擇建立根 CA 或次級 CA。本教學課程說明如何建立根 CA 以簡化程序。

16. 對於 Private Key (私有金鑰)，選取 Create a new private key (建立新的私有金鑰)。然後選擇下一步。
17. 對於 Cryptography (加密法)，執行下列動作：

    a. 對於 Select a cryptographic provider (選取密碼編譯供應商)，從功能表選擇其中一個 Cavium Key Storage Provider (Cavium 金鑰儲存供應商) 選項。這些是 AWS CloudHSM 金鑰儲存供應商。例如，您可以選擇 RSA#Cavium Key Storage Provider (RSA#Cavium 金鑰儲存供應商)。

    b. 對於 Key length (金鑰長度)，選擇其中一個金鑰長度選項。

    c. 對於 Select the hash algorithm for signing certificates issued by this CA (選取用於簽署這個 CA 發出之憑證的雜湊演算法)，請選擇其中一個雜湊演算法選項。

    選擇下一步。
18. 對於 CA Name (CA 名稱)，執行下列動作：

    a. (選用) 編輯常用的名稱。

    b. (選用) 輸入辨別名稱尾碼。

    選擇下一步。
19. 對於 Validity Period (有效期間)，以年、月、週或天指定時段。然後選擇下一步。
20. 對於 Certificate Database (憑證資料庫)，您可以接受預設值，或選擇變更資料庫和資料庫日誌的位置。然後選擇下一步。
21. 對於 Confirmation (確認)，檢閱您的 CA 的相關資訊，然後選擇 Configure (設定)。
22. 選擇 Close (關閉)，然後再次選擇 Close (關閉)。

您現在有一個視窗伺服器 CA AWS CloudHSM。若要了解如何使用您的 CA 來簽署憑證簽署要求 (CSR)，請移至簽署 CSR。

視窗伺服器 CA 步驟 3：使用您的視窗伺服器 CA 簽署憑證簽署要求 (CSR) AWS CloudHSM

您可以搭配使用 Windows 伺服器 CA AWS CloudHSM 來簽署憑證簽署要求 (CSR)。若要完成以下步驟，您需要有效的 CSR。有幾種方式可建立 CSR，包括：

- 使用 OpenSSL
- 使用 Windows Server Internet Information Services (IIS) Manager
- 在 Microsoft Management Console 中使用憑證嵌入式管理單元
- 在 Windows 上使用 certreq 命令列公用程式

建立 CSR 的步驟已超出本教學課程的範圍。當您有 CSR 時，您可以向 Windows Server CA 簽署此 CSR。

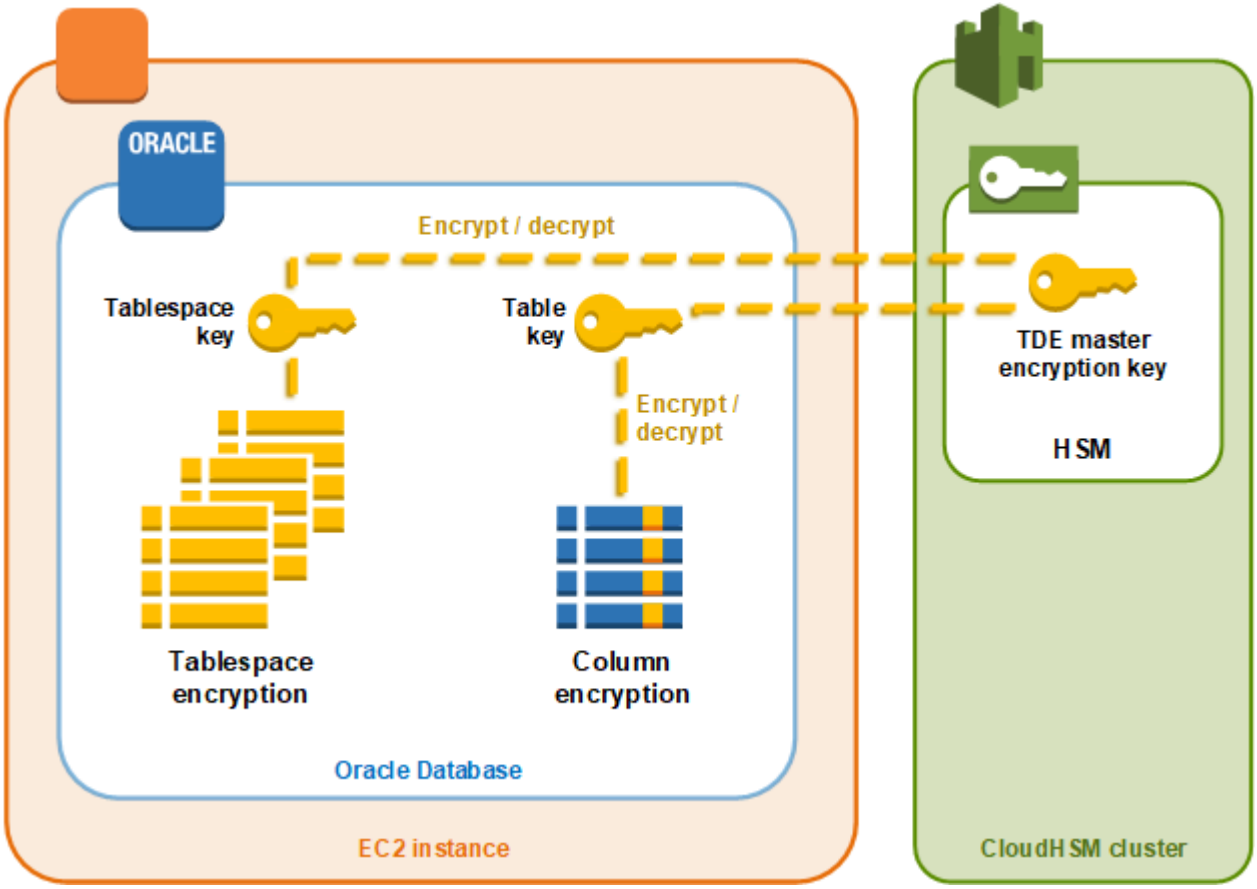
向 Windows Server CA 簽署 CSR

1. 如果您尚未這麼做，請連接至您的 Windows 伺服器。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到您的執行個體。
2. 在 Windows 伺服器上，啟動 Server Manager (伺服器管理員)。
3. 在伺服器管理員儀表板右上角，選擇 Tools (工具)、Certification Authority (憑證授權單位)。
4. 在 Certification Authority (憑證授權單位) 視窗中，選擇您的電腦名稱。
5. 從 Action (執行) 功能表中，選擇 All Tasks (所有工作)、Submit new request (提交新要求)。
6. 選取您的 CSR 檔案，然後選擇 Open (開啟)。
7. 在Certification Authority (憑證授權單位)視窗中，按兩下 Pending Requests (擱置要求)。
8. 選取擱置要求。然後，從 Action (執行) 功能表中，選擇 All Tasks (所有工作)、Issue (發行)。
9. 在 Certification Authority (憑證授權單位) 視窗中，按兩下 Issued Requests (已發出的要求)，以檢視已簽署的憑證。
10. (選用) 若要將已簽署的憑證匯出到檔案，請完成以下步驟：
    a. 在 Certification Authority (憑證授權單位) 視窗中，按兩下憑證。
    b. 選擇 Details (詳細資料) 標籤，然後選擇 Copy to File (複製到檔案)。
    c. 依照憑證匯出精靈中的指示進行。

您現在有一個 Windows 伺服器 CA AWS CloudHSM，以及由視窗伺服器 CA 簽署的有效憑證。

Oracle 資料庫的透明資料加密 (TDE) 搭配 AWS CloudHSM

透明資料加密 (TDE) 可用來加密資料庫檔案。使用 TDE，資料庫軟體將資料存儲在磁盤上之前對其進行加密。資料庫資料表資料欄或資料表空間中的資料會使用資料表金鑰或資料表空間金鑰加密。Oracle 的資料庫軟體的某些版本提供 TDE。在 Oracle TDE 中，這些金鑰會使用 TDE 主加密金鑰加密。您可以將 TDE 主要加密金鑰儲存在叢集中的 HSM 中，以達到更高的安全性。 AWS CloudHSM



在這個解決方案中，您使用在 Amazon EC2 執行個體上安裝的 Oracle 資料庫。Oracle 資料庫會與適用於 PKCS #11 的AWS CloudHSM 軟體程式庫整合，可將 TDE 主金鑰存放在您叢集中的 HSM。

Important

- 建議在 Amazon EC2 執行個體上安裝 Oracle 資料庫。

執行下列步驟以完成 Oracle TDE 與 AWS CloudHSM的整合。

若要設定 Oracle TDE 整合 AWS CloudHSM

1. 遵循 設定先決條件 中的步驟準備您的環境。
2. 請遵循中的步驟，設定資料庫將「Oracle 資料庫」設定為與 AWS CloudHSM 叢集整合。

使用 Oracle TDE AWS CloudHSM：設定先決條件

若要完成與 Oracle TDE 整合 AWS CloudHSM，您需要下列項目：

- 至少具有一個 HSM 的作用中 AWS CloudHSM 叢集。
- 執行 Amazon Linux 作業系統的 Amazon EC2 執行個體，其中已安裝下列軟體：
  - 用 AWS CloudHSM 戶端和命令列工具。
  - PKCS AWS CloudHSM #11 的軟體程式庫。
  - 甲骨文數據庫。 AWS CloudHSM 支持甲骨文 TDE 集成。用戶端 SDK 5.6 及更高版本支援適用於 Oracle 資料庫 19c 的 Oracle TDE。客戶端 SDK 3 支援適用於 Oracle 資料庫 11g 和 12c 版的 Oracle TDE。
- 加密使用者 (CU)，擁有及管理叢集中 HSM 上的 TDE 主加密金鑰。

完成以下步驟來設定所有先決條件。

若要設定 Oracle TDE 整合的先決條件，請執行下列步驟： AWS CloudHSM

1. 完成「開始使用」中的步驟。完成之後，您將擁有一個作用中叢集，且其中有一個 HSM。您也會有一個執行 Amazon Linux 作業系統的 Amazon EC2 執行個體。 AWS CloudHSM 客戶端和命令行工具也將被安裝和配置。
2. (選用) 在您的叢集中新增更多 HSM。如需詳細資訊，請參閱 新增 HSM。
3. 連接到 Amazon EC2 用戶端執行個體，並執行下列動作：
   a. 安裝 PKCS AWS CloudHSM #11 的軟體程式庫。
   b. 安裝 Oracle 資料庫。如需詳細資訊，請參閱 Oracle 資料庫文件。用戶端 SDK 5.6 及更高版本支援適用於 Oracle 資料庫 19c 的 Oracle TDE。客戶端 SDK 3 支援適用於 Oracle 資料庫 11g 和 12c 版的 Oracle TDE。
   c. 使用 cloudhsm_mgmt_util 命令列工具來在您的叢集上建立加密使用者 (CU)。如需關於建立 CU 的詳細資訊，請參閱如何使用 CMU 管理 HSM 使用者和 管理 HSM 使用者。

完成這些步驟之後，您可以 設定資料庫。

Oracle TDE 使用 AWS CloudHSM：設定資料庫並產生主要加密金鑰

若要將 Oracle TDE 與您的 AWS CloudHSM 叢集整合，請參閱下列主題：

1. 更新 Oracle 資料庫組態，使用叢集中的 HSM 做為外部安全模組。如需外部安全模組的相關資訊，請參閱 Oracle Database 進階安全指南中的透明資料加密簡介。
2. 在叢集的 HSM 上產生 Oracle TDE 主加密金鑰。

更新 Oracle 資料庫組態

若要更新 Oracle 資料庫組態來使用叢集中的 HSM 做為外部安全模組，請完成以下步驟。如需外部安全模組的相關資訊，請參閱 Oracle Database 進階安全指南中的透明資料加密簡介。

更新 Oracle 組態

1. 連線到 Amazon EC2 用戶端執行個體。您已將 Oracle 資料庫安裝在此執行個體上。
2. 建立名為 `sqlnet.ora` 之檔案的備份複本。關於此檔案的位置，請參閱 Oracle 文件。
3. 使用文字編輯器來編輯名為 `sqlnet.ora` 的檔案。新增以下這一行。如果檔案中有一行以 `encryption_wallet_location` 開頭，請換成以下這一行。

   ```
   encryption_wallet_location=(source=(method=hsm))
   ```

   儲存檔案。
4. 執行下列命令，以建立「Oracle 資料庫」預期在其中尋找 AWS CloudHSM PKCS #11 軟體程式庫之程式庫檔案的目錄。

   ```
   sudo mkdir -p /opt/oracle/extapi/64/hsm
   ```
5. 執行下列命令，將 PKCS #11 檔案的 AWS CloudHSM 軟體程式庫複製到您在上一個步驟中建立的目錄。

   ```
   sudo cp /opt/cloudhsm/lib/libcloudhsm_pkcs11.so /opt/oracle/extapi/64/hsm/
   ```

> **Note**
>
> /opt/oracle/extapi/64/hsm 目錄必須僅包含一個程式庫檔案。移除該目錄中存在的任何其他檔案。

6. 執行以下命令來變更 /opt/oracle 目錄的擁有權及其中一切設定。

```
sudo chown -R oracle:dba /opt/oracle
```

7. 啟動 Oracle 資料庫。

產生 Oracle TDE 主加密金鑰

若要在叢集中 HSM 上產生 Oracle TDE 主金鑰，請完成以下程序中的步驟。

產生主金鑰

1. 使用下列命令開啟 Oracle SQL*Plus。當系統提示時，請輸入您安裝 Oracle 資料庫時所設定的系統密碼。

```
sqlplus / as sysdba
```

> **Note**
>
> 對於用戶端 SDK 3，每次產生主金鑰時，都必須設定 CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE 環境變數。此變數僅用於產生主金鑰。如需詳細資訊，請參閱整合第三方應用程式的已知問題中的「問題：Oracle 在產生主金鑰期間設定 PCKS #11 屬性 CKA_MODIFIABLE，但 HSM 不支援此屬性」。

2. 執行 SQL 陳述式來建立主加密金鑰，如下列範例所示。請使用與 Oracle 資料庫版本對應的陳述式。以加密使用者 (CU) 的使用者名稱取代 *<CU #####>*。以 CU 密碼取代*<##>*。

> **Important**
>
> 只執行一次下列命令。每次執行此命令都會建立新的主加密金鑰。

- 若為 Oracle 資料庫第 11 版，請執行下列 SQL 陳述式。

```
SQL> alter system set encryption key identified by "<CU user name>:<password>";
```

- 若為 Oracle 資料庫第 12 版和第 19c 版，請執行下列 SQL 陳述式。

```
SQL> administer key management set key identified by "<CU user
 name>:<password>";
```

如果回應是 `System altered` 或 `keystore altered`，則表示您已成功產生和設定 Oracle TDE 的主金鑰。

3. (選用) 執行以下命令來驗證 Oracle 錢包的狀態。

```
SQL> select * from v$encryption_wallet;
```

如果錢包未開啟，請使用下列其中一個命令來開啟。以加密使用者 (CU) 的名稱取代 *<CU #####>*。以 CU 密碼取代*<##>*。

- 若為 Oracle 11，請執行下列命令來開啟錢包。

```
SQL> alter system set encryption wallet open identified by "<CU user
 name>:<password>";
```

若要手動關閉錢包，請執行下列命令。

```
SQL> alter system set encryption wallet close identified by "<CU user
 name>:<password>";
```

- 若為 Oracle 12 和 Oracle 19c，請執行下列命令來開啟錢包。

```
SQL> administer key management set keystore open identified by "<CU user
 name>:<password>";
```

若要手動關閉錢包，請執行下列命令。

```
SQL> administer key management set keystore close identified by "<CU user
 name>:<password>";
```

搭配使 SignTool 用 Microsoft AWS CloudHSM 來簽署檔案

在加密和公開金鑰基礎設施 (PKI) 中，數位簽章是用來確認資料已由可信任的實體傳送。此簽章也代表傳輸中的資料未經過篡改。簽章是一種由寄件者的私有金鑰產生的加密雜湊。接收者可以使用寄件者的公開金鑰解密資料的雜湊簽章，以驗證資料的完整性。因此，應該由寄件者負責維護數位憑證。數位憑證代表寄件者的私有金鑰擁有權，並可將解密所需的公有金鑰提供給接收者。只要私密金鑰是寄件者所擁有，簽章就可以信任。 AWS CloudHSM 提供安全的 FIPS 140-2 第 3 級驗證硬體，讓您透過獨家單一租用戶存取來保護這些金鑰。

許多組織都使用 Microsoft SignTool，這是一種可簽署、驗證和時間戳記檔案的命令列工具，以簡化程式碼簽章程序。您可以使用安全地儲存金鑰配對，直到需要金鑰配對為止 SignTool，從而建立可輕鬆自動化的工作流程 AWS CloudHSM 來簽署資料。

下列主題提供如何 SignTool 搭配使用的概觀 AWS CloudHSM：

主題

Microsoft SignTool AWS CloudHSM 步驟 1：設置先決條件

若要搭配使 SignTool 用 Microsoft AWS CloudHSM，您需要下列項目：

- 執行 Windows 作業系統的 Amazon EC2 用戶端執行個體。
- 憑證授權單位 (CA)；不論是自己維護，或由第三方供應商建立。
- 與 EC2 執行個體位於相同虛擬公有雲 (VPC) 中的作用中 AWS CloudHSM 叢集。叢集必須至少包含一個 HSM。
- 擁有和管理 AWS CloudHSM 叢集中金鑰的加密使用者 (CU)。
- 未簽署的檔案或可執行檔。
- Microsoft Windows 軟體開發套件 (SDK)。

若要設定搭 AWS CloudHSM 配 Windows 搭配使用的先決條件 SignTool

1. 請依照本指南中入門一節的說明，來啟動 Windows EC2 執行個體與 AWS CloudHSM 叢集。

2. 如果您想要託管自己的 Windows 伺服器 CA，請遵循將 Windows 伺服器設定為憑證授權單位中的步驟 1 和 2 AWS CloudHSM。否則，請繼續使用公開信任的第三方 CA。
3. 在您的 Windows EC2 執行個體上，下載並安裝下列其中一個版本的 Microsoft Windows 軟體開發套件：

   - Microsoft Windows SDK 10
   - Microsoft Windows SDK 8.1
   - Microsoft Windows SDK 7

   `SignTool` 可執行檔是 Windows SDK Signing Tools for Desktop Apps 安裝功能的一部分。如果您不需要其他功能，可以省略其安裝。預設安裝位置為：

   ```
   C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\<CPU
    architecture>\signtool.exe
   ```

您現在可以使用 Microsoft 視窗 SDK、您的 AWS CloudHSM 叢集和 CA 來建立簽署憑證。

Microsoft SignTool 與 AWS CloudHSM 步驟 2：創建一個簽名證書

現在您已將 Windows SDK 下載到您的 EC2 執行個體上，即可使用它來產生憑證簽署請求 (CSR)。CSR 是未簽署的憑證，最終會傳遞到您的 CA 以進行簽署。在這個範例中，我們使用 Windows SDK 隨附的 `certreq` 可執行檔，來產生 CSR。

使用 **`certreq`** 可執行檔來產生 CSR

1. 如果您尚未連線至 Windows EC2 執行個體，請先進行連線。如需詳細資訊，請參閱 Amazon EC2 使用者指南中的 Connect 到您的執行個體。
2. 建立名為 `request.inf` 的檔案，其中包含以下各行。將 `Subject` 資訊取代為您組織的相關資訊。如需每個參數的說明，請參閱 Microsoft 文件。

   ```
   [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. 執行 `certreq.exe`。在此範例中，我們將 CSR 儲存為 `request.csr`。

```
certreq.exe -new request.inf request.csr
```

在內部，AWS CloudHSM 叢集上會產生新的 key pair 組，並使用該組的私密金鑰來建立 CSR。

4. 將 CSR 提交給您的 CA。如果您使用的是 Windows Server CA，請依照下列步驟進行：

   a. 輸入下列命令以開啟 CA 工具：

   ```
   certsrv.msc
   ```

   b. 在新的視窗中，以滑鼠右鍵按一下 CA 伺服器的名稱。選擇 All Tasks (所有任務)，然後選擇 Submit new request (提交新請求)。

   c. 導覽至 `request.csr` 的位置，然後選擇 Open (開啟)。

   d. 展開伺服器 CA 功能表，導覽至待處理的請求資料夾。用滑鼠右鍵按一下您剛建立的請求，並在 All Tasks (所有任務) 下方選擇 Issue (發行)。

   e. 現在，導覽至 Issued Certificates (發行的憑證) 資料夾 (位於 Pending Requests (待處理的請求) 資料夾上方)。

   f. 選擇 Open (開啟) 以檢視憑證，然後選擇 Details (詳細資訊) 標籤。

   g. 選擇 Copy to File (複製到檔案) 來啟動 [憑證匯出精靈]。將 DER 編碼的 X.509 檔案以 `signedCertificate.cer` 名稱儲存到安全的位置。

   h. 結束 CA 工具，並使用下列命令，將憑證檔案移至 Windows 的個人憑證存放區中。其他應用程式即可使用此憑證。

   ```
   certreq.exe -accept signedCertificate.cer
   ```

您現在可以使用匯入的憑證來簽署檔案。

Microsoft SignTool 與 AWS CloudHSM 步驟 3：簽署一個文件

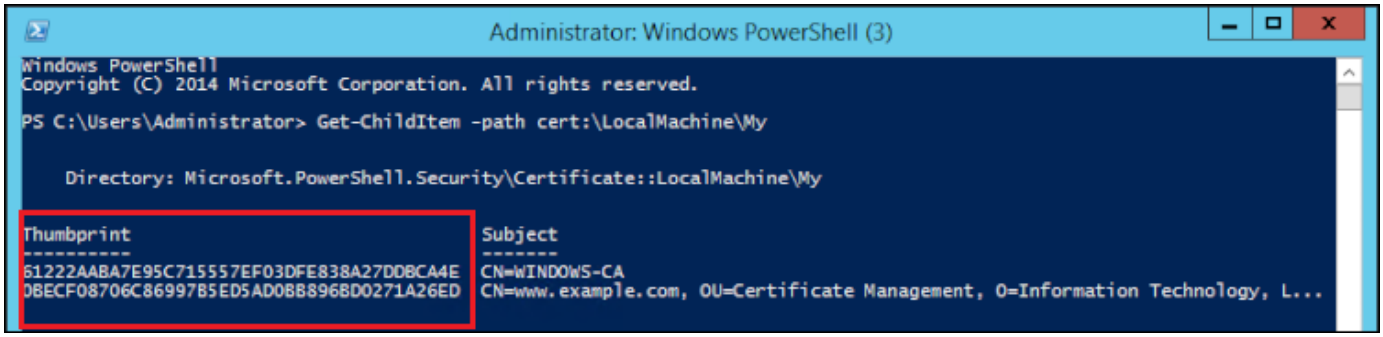
您現在可以使用 SignTool 並匯入的憑證來簽署範例檔案。若要執行此作業，您需要知道憑證的 SHA-1 雜湊或「指紋」。指紋是用來確保 SignTool僅使用經過 AWS CloudHSM驗證的憑證。在這個例子中，我們用 PowerShell 來獲取證書的哈希值。您也可以使用 CA 的 GUI 或 Windows 軟體開發套件的 `certutil` 可執行檔。

取得憑證的指紋，並使用它來簽署檔案

1. 以管理員 PowerShell 身份打開並運行以下命令：

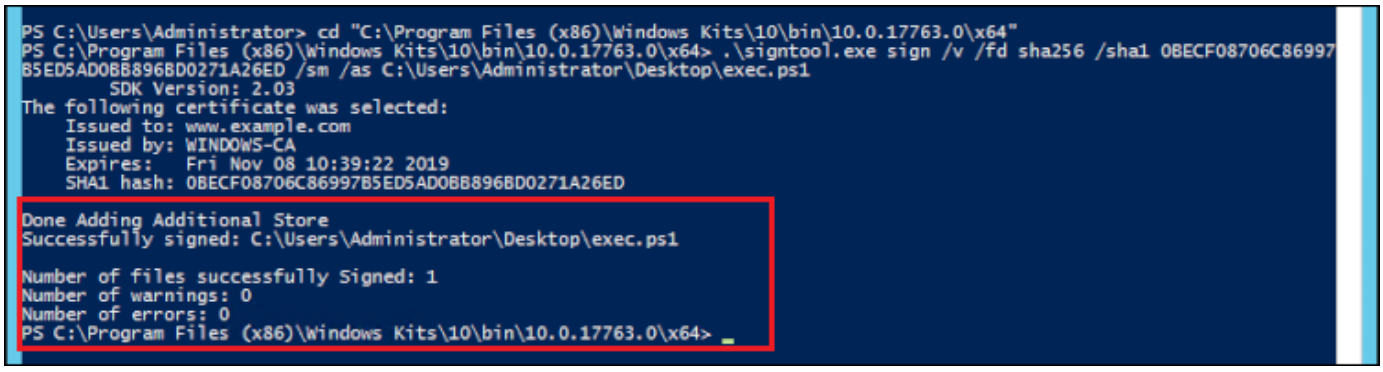
   ```
   Get-ChildItem -path cert:\LocalMachine\My
   ```

   複製傳回的 Thumbprint。

2. 導覽至包含 PowerShell 的目錄`SignTool.exe`。預設位置為 `C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64`。
3. 最後，執行以下命令簽署您的檔案。如果命令成功，會 PowerShell 傳回成功訊息。

   ```
   signtool.exe sign /v /fd sha256 /sha1 <thumbprint> /sm C:\Users\Administrator
   \Desktop\<test>.ps1
   ```

4. (選用) 若要驗證檔案上的簽章，請使用下列命令：

```
signtool.exe verify /v /pa C:\Users\Administrator\Desktop\<test>.ps1
```

Java Keytool 和 Jarsigner

AWS CloudHSM 通過客戶端 SDK 3 和客戶端 SDK 5 提供了與 Java 密鑰工具和 Jarsigner 實用程序的集成。這些工具具體的使用步驟會因您目前下載的用戶端 SDK 版本而有所不同：

- 使用用戶端 SDK 5 與 Java Keytool 和 Jarsigner 整合
- 使用用戶端 SDK 3 與 Java Keytool 和 Jarsigner 整合

使用用戶端 SDK 5 與 Java Keytool 和 Jarsigner 整合

AWS CloudHSM 金鑰存放區是一種特殊用途的 JCE 金鑰存放區，可透過第三方工具 (例如和) 利用與 HSM 上金鑰相關聯的憑證。`keytool` `jarsigner` AWS CloudHSM 不會將憑證儲存在 HSM 上，因為憑證是公開的非機密資料。金 AWS CloudHSM 鑰存放區會將憑證儲存在本機檔案中，並將憑證對應至 HSM 上對應的金鑰。

當您使用金 AWS CloudHSM 鑰存放區產生新金鑰時，本機金鑰存放區檔案中不會產生任何項目 — 金鑰會在 HSM 上建立。同樣的，當您使用 AWS CloudHSM 金鑰存放區搜尋金鑰時，搜尋會傳遞至 HSM。當您將憑證儲存在 AWS CloudHSM 金鑰存放區中時，提供者會驗證 HSM 上是否存在 key pair 與對應別名的 key pair，然後將提供的憑證與對應的金鑰配對產生關聯。

主題

- 必要條件
- 使用 AWS CloudHSM 密鑰存儲與密鑰工具
- 使用 AWS CloudHSM 金鑰存放區與 Jarsigner
- 已知問題

必要條件

若要使用 AWS CloudHSM 金鑰存放區，您必須先初始化並設定 AWS CloudHSM JCE SDK。

步驟 1：安裝 JCE

要安裝 JCE（包括 AWS CloudHSM 客戶端的先決條件），請按照安裝 Java 庫的步驟進行操作。

步驟 2：將 HSM 登入憑證新增至環境變數

設定環境變數以包含您的 HSM 登入憑證。

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 JCE 提供多種登入選項。若要將 AWS CloudHSM 金鑰存放區與第三方應用程式搭配使用，您必須搭配環境變數使用隱含登入。如果您想要透過應用程式程式碼使用明確登入，您必須使用 AWS CloudHSM 金鑰存放區建置自己的應用程式。如需其他資訊，請參閱使用 AWS CloudHSM 金鑰存放區的文章。

步驟 3：註冊 JCE 提供商

若要在 Java CloudProvider 組態中註冊 JCE 提供者，請依照下列步驟執行：

1. 在 Java 安裝中開啟 `java.security` 組態檔案進行編輯。
2. 在 `java.security` 組態檔案中，新增 `com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider` 為最後一個提供者。例如，如果 `java.security` 檔案中有九個供應商，請將下列提供商新增為區段中的最後一個供應商。

   ```
   security.provider.10=com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider
   ```

> **Note**
>
> 將 AWS CloudHSM 提供者新增為較高優先順序可能會對您的系統效能造成負面影響，因為 AWS CloudHSM 提供者會優先處理可能會安全卸載到軟體的作業。最佳作法是永遠指定您要用於作業的提供者，無論是軟體提供者還是以軟體為基礎的提供者。 AWS CloudHSM

> **Note**
>
> 使用 Keytool 和 AWS CloudHSM 金鑰存放區產生金鑰時，指定 `-providerName`、`-providerclass` 和 `-providerpath` 命令列選項可能會出現錯誤。

使用 AWS CloudHSM 密鑰存儲與密鑰工具

Keytool 是常見的金鑰和憑證任務的常用命令列公用程式。完整的 Keytool 教學並不在 AWS CloudHSM 文件範圍之內。本文說明在透過 AWS CloudHSM 金鑰存放區 AWS CloudHSM 做為信任根使用時，您應該搭配各種 keytool 函式使用的特定參數。

將 keytool 與 AWS CloudHSM 金鑰存放區搭配使用時，請為任何 keytool 命令指定下列引數：

Linux

```
-storetype CLOUDHSM -J-classpath< '-J/opt/cloudhsm/java/*'>
```

Windows

```
-storetype CLOUDHSM -J-classpath<'-J"C:\Program Files\Amazon\CloudHSM\java\*"'>
```

如果要使用金鑰存放區建立新金鑰存放區檔 AWS CloudHSM 案，請參閱使用 AWS CloudHSM KeyStore。若要要使用現有的金鑰存放區，請使用金鑰存放區引數指定其名稱 (包含路徑) 至 Keytool。如果您在 keytool 指令中指定不存在的金鑰存放區檔案，則金鑰存放區會建立新的 AWS CloudHSM 金鑰存放區檔案。

使用 Keytool 建立新金鑰

您可以使用 Keytool 產生由 AWS CloudHSM的 JCE SDK 支援的 RSA、AES 和 DESede 類型金鑰。

> **Important**
>
> 通過 keytool 生成的密鑰在軟件中生成，然後導入到一個可提取 AWS CloudHSM 的，持久的密鑰。

我們強烈建議您不要在 Keytool 中產生不可匯出的金鑰，然後又匯入相對應的憑證至金鑰存放區。如果您透過 keytool 和 Jarsigner 使用可擷取的 RSA 或 EC 金鑰，則提供者會從匯出金鑰，然後在本機使用金鑰進行簽署作業。 AWS CloudHSM

如果您有多個用戶端執行個體連線到 AWS CloudHSM 叢集，請注意，在一個用戶端執行個體的金鑰存放區匯入憑證不會自動讓憑證可用於其他用戶端執行個體。若要在每個用戶端執行個體上註冊金鑰和相關憑證，您必須執行 Java 應用程式，如 the section called “使用 Keytool 產生 CSR” 所述。或者，您可以在一個用戶端上進行必要的變更，並將產生的金鑰存放區檔案複製到其他每個用戶端執行個體。

範例 1：產生帶對稱 AES-256 金鑰，並將其儲存在工作目錄中名為「my_keystore.store」的金鑰存放區檔案中。將 *<secret label>* 取代為唯一的標籤。

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\*"'
```

範例 2：產生 RSA 2048 金鑰對，並將其儲存在工作目錄中名為「my_keystore.store」的金鑰存放區檔案中。將 *<RSA key pair label>* 取代為唯一的標籤。

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\*"'
```

您可以在 Java 程式庫中找到支援的簽章演算法清單。

使用 Keytool 刪除金鑰

金 AWS CloudHSM 鑰存放區不支援刪除金鑰。您可以使用可銷毀介面的銷毀方法刪除金鑰。

```
((Destroyable) key).destroy();
```

使用 Keytool 產生 CSR

如果您使用 OpenSSL 動態引擎，您可以在產生憑證簽署要求 (CSR) 時獲得最大的彈性。以下命令使用 Keytool 來產生具有別名 `my-key-pair` 的金鑰對適用 CSR。

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**
>
> 若要使用 Keytool 的金鑰對，該金鑰對必須再指定的金鑰存放區檔案中有項目。如果您要使用非 Keytool 產生的金鑰對，則必須將金鑰和憑證中繼資料匯入至金鑰存放區。如需匯入金鑰存放區資料的說明，請參閱 the section called “使用 keytool 將中繼憑證和根憑證匯入 AWS CloudHSM 金鑰存放區 ”。

使用 keytool 將中繼憑證和根憑證匯入 AWS CloudHSM 金鑰存放區

若要匯入 CA 憑證，您必須在新匯入的憑證上啟用完整憑證鏈的驗證。以下為命令的範例。

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\*"'
```

如果您將多個用戶端執行個體連接到 AWS CloudHSM 叢集，則在一個用戶端執行個體的金鑰存放區匯入憑證不會自動讓憑證可用於其他用戶端執行個體。您必須在每個用戶端執行個體上匯入憑證。

使用 keytool 從 AWS CloudHSM 密鑰存儲中刪除證書

下列範例顯示命令如何從 Java Keytool 金鑰存放區庫刪除憑證。

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\*"'
```

如果您將多個用戶端執行個體連接到 AWS CloudHSM 叢集，刪除一個用戶端執行個體金鑰存放區上的憑證並不會自動從其他用戶端執行個體移除憑證。您必須在每個用戶端執行個體上刪除憑證。

使用 keytool 將工作證書導入 AWS CloudHSM 密鑰存儲中

憑證簽署要求 (CSR) 簽署後，您就可以將其匯入 AWS CloudHSM 金鑰存放區，並與適當的金鑰對建立關聯。以下是命令範例。

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\*"'
```

別名應為金鑰存放區中具有關聯憑證的金鑰對。如果金鑰是非 Keytool 產生，或是在不同的用戶端執行個體上產生，您就必須先將金鑰和憑證中繼資料匯入金鑰存放區。

憑證鏈必須是可驗證的。如果您無法驗證憑證，則可能需要將簽署 (憑證授權單位) 憑證匯入金鑰存放區，以便驗證該鏈結。

使用 Keytool 匯出憑證

下列範例會產生二進位 X.509 格式的憑證。若要匯出人類可讀的憑證，請新增 `-rfc` 至 `-exportcert` 命令。

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\*"'
```

使用 AWS CloudHSM 金鑰存放區與 Jarsigner

Jarsigner 是一種流行的命令行實用程序，用於使用安全地存儲在 HSM 上的密鑰簽名 JAR 文件。完整的 Jarsigner 教學並不在 AWS CloudHSM 文件範圍之內。本節說明您應該使用的 Jarsigner 參數，透過 AWS CloudHSM 金鑰存放區以信任的根目錄來簽署和驗證簽名。 AWS CloudHSM

設定金鑰和憑證

在您可以使用 Jarsigner 簽署 JAR 文件之前，請確保您已經設定或完成以下步驟：

1. 遵循 AWS CloudHSM 金鑰存放區必要條件中的指引。
2. 設定您的簽署金鑰以及應儲存在目前伺服器或用戶端執行個體的金 AWS CloudHSM 鑰存放區中的相關憑證和憑證鏈結。在上建立金鑰， AWS CloudHSM 然後將關聯的中繼資料匯入金 AWS CloudHSM 鑰存放區。如果您要使用 Keytool 設定金鑰和憑證，請參閱 the section called “使用 Keytool 建立新金鑰”。如果您使用多個用戶端執行個體來簽署 JAR，請建立金鑰並匯入憑證鏈。然後將生成的金鑰存放區文件複製到每個用戶端執行個體。如果您經常產生新的金鑰，您可能會發現將憑證個別匯入至每個用戶端執行個體更容易。
3. 整個憑證鏈必須是可驗證的。若要讓憑證鏈結可驗證，您可能需要將 CA 憑證和中繼憑證新增至 AWS CloudHSM 金鑰存放區。有關使用 Java 程式碼驗證憑證鏈的說明，請參閱 the section called

"使用 AWS CloudHSM 和 Jarsigner 簽署 JAR 檔案" 中的程式碼片段。如果需要，您也可以使用 Keytool 匯入憑證。如需使用 Keytool 的說明，請參閱 the section called "使用 keytool 將中繼憑證和根憑證匯入 AWS CloudHSM 金鑰存放區 "。

使用 AWS CloudHSM 和 Jarsigner 簽署 JAR 檔案

使用以下命令來簽署 JAR 檔案：

Linux;

對於 OpenJDK 8

```
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>
```

適用於 OpenJDK 11、第 17 OpenJDK 和第 21 OpenJDK

```
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

對於 OpenJDK8

```
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>
```

適用於 OpenJDK 11、第 17 OpenJDK 和第 21 OpenJDK

```
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>
```

使用以下命令來驗證已簽署的 JAR：

Linux

對於 OpenJDK8

```
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>
```

適用於 OpenJDK 11、第 17 OpenJDK 和第 21 OpenJDK

```
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

對於 OpenJDK 8

```
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>
```

適用於 OpenJDK 11、第 17 OpenJDK 和第 21 OpenJDK

```
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>
```

已知問題

1. 不支持使用 Keytool 和 Jarsigner 的 EC 金鑰。

使用用戶端 SDK 3 與 Java Keytool 和 Jarsigner 整合

AWS CloudHSM 金鑰存放區是一種特殊用途的 JCE 金鑰存放區，可透過第三方工具 (例如和) 利用與 HSM 上金鑰相關聯的憑證。`keytool` `jarsigner` AWS CloudHSM 不會將憑證儲存在 HSM 上，因為憑證是公開的非機密資料。金 AWS CloudHSM 鑰存放區會將憑證儲存在本機檔案中，並將憑證對應至 HSM 上對應的金鑰。

當您使用金 AWS CloudHSM 鑰存放區產生新金鑰時，本機金鑰存放區檔案中不會產生任何項目 — 金鑰會在 HSM 上建立。同樣的，當您使用 AWS CloudHSM 金鑰存放區搜尋金鑰時，搜尋會傳遞至

HSM。當您將憑證儲存在 AWS CloudHSM 金鑰存放區中時，提供者會驗證 HSM 上是否存在 key pair 與對應別名的 key pair，然後將提供的憑證與對應的金鑰配對產生關聯。

主題

必要條件

若要使用 AWS CloudHSM 金鑰存放區，您必須先初始化並設定 AWS CloudHSM JCE SDK。

步驟 1：安裝 JCE

要安裝 JCE（包括 AWS CloudHSM 客戶端的先決條件），請按照安裝 Java 庫的步驟進行操作。

步驟 2：將 HSM 登入憑證新增至環境變數

設定環境變數以包含您的 HSM 登入憑證。

```
export HSM_PARTITION=PARTITION_1
export HSM_USER=<HSM user name>
export HSM_PASSWORD=<HSM password>
```

> **Note**
>
> CloudHSM JCE 提供多種登入選項。若要將 AWS CloudHSM 金鑰存放區與第三方應用程式搭配使用，您必須搭配環境變數使用隱含登入。如果您想要透過應用程式程式碼使用明確登入，您必須使用 AWS CloudHSM 金鑰存放區建置自己的應用程式。如需其他資訊，請參閱使用 AWS CloudHSM 金鑰存放區的文章。

步驟 3：註冊 JCE 提供商

要註冊 JCE 提供程序，請在 Java CloudProvider 配置中。

1. 在 Java 安裝中開啟 java.security 組態檔案進行編輯。
2. 在 java.security 組態檔案中，新增 `com.cavium.provider.CaviumProvider` 為最後一個提供商。例如，如果 java.security 檔案中有九個供應商，請將下列提供商新增為區段中的最後一個供應商。如果將 Cavium 提供商新增為優先順序較高的提供商，可能會對您的系統效能帶來負面影響。

   ```
   security.provider.10=com.cavium.provider.CaviumProvider
   ```

   > **Note**
   >
   > 進階使用者可能習慣在使用 keytool 時指定 `-providerName`、`-providerclass` 和 `-providerpath` 命令列選項，而不是更新安全性組態檔案。如果您嘗試在使用 AWS CloudHSM 密鑰存儲生成密鑰時指定命令行選項，則會導致錯誤。

使用 AWS CloudHSM 密鑰存儲與密鑰工具

Keytool 是 Linux 系統中常見的金鑰和憑證任務的常用命令列執行程序。完整的 Keytool 教學並不在 AWS CloudHSM 文件範圍之內。本文說明在透過 AWS CloudHSM 金鑰存放區 AWS CloudHSM 做為信任根使用時，您應該搭配各種 keytool 函式使用的特定參數。

將 keytool 與 AWS CloudHSM 金鑰存放區搭配使用時，請為任何 keytool 命令指定下列引數：

```
-storetype CLOUDHSM \
  -J-classpath '-J/opt/cloudhsm/java/*' \
  -J-Djava.library.path=/opt/cloudhsm/lib
```

如果要使用金鑰存放區建立新的金鑰存放區檔 AWS CloudHSM 案，請參閱使用 AWS CloudHSM KeyStore。若要要使用現有的金鑰存放區，請使用金鑰存放區引數指定其名稱 (包含路徑) 至 Keytool。如果您在 keytool 指令中指定不存在的金鑰存放區檔案，則金鑰存放區會建立新的 AWS CloudHSM 金鑰存放區檔案。

使用 Keytool 建立新金鑰

您可以使用密鑰工具來生成由 AWS CloudHSM的 JCE SDK 支持的任何類型的密鑰。請參閱 Java 程式庫的支援金鑰文章中金鑰和長度的完整清單。

> **Important**
>
> 通過 keytool 生成的密鑰在軟件中生成，然後導入到一個可提取 AWS CloudHSM 的，持久的密鑰。

直接在 HSM 上建立不可擷取的金鑰，然後將它們與 keytool 或 Jarsigner 搭配使用的指示，會顯示在使用金鑰存放區註冊預先存在的金鑰中的程式碼範例中。 AWS CloudHSM 我們強烈建議您不要在 Keytool 中產生不可匯出的金鑰，然後又匯入相對應的憑證至金鑰存放區。如果您透過 keytool 和 jarsigner 使用可擷取的 RSA 或 EC 金鑰，則提供者會從匯出金鑰，然後在本機使用金鑰進行簽署作業。 AWS CloudHSM

如果您有多個用戶端執行個體連接到 CloudHSM 叢集，請注意，在一個用戶端執行個體的金鑰存放區上匯入憑證不會自動讓憑證可於其他用戶端執行個體使用。若要在每個用戶端執行個體上註冊金鑰和相關憑證，您必須執行 Java 應用程式，如使用 Keytool 產生 CSR中所述。或者，您可以在一個用戶端上進行必要的變更，並將產生的金鑰存放區檔案複製到其他每個用戶端執行個體。

範例 1：產生帶對稱 AES-256 金鑰，並將其儲存在工作目錄中名為「my_keystore.store」的金鑰存放區檔案中。將 *<secret label>* 取代為唯一的標籤。

```
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/
```

範例 2：產生 RSA 2048 金鑰對，並將其儲存在工作目錄中名為「my_keystore.store」的金鑰存放區檔案中。將 *<RSA key pair label>* 取代為唯一的標籤。

```
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/
```

範例 3：產生 p256 ED 金鑰，並將其儲存在工作目錄中名為「my_keystore.store」的金鑰存放區檔案中。將 *<ec key pair label>* 取代為唯一的標籤。

```
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/
```

您可以在 Java 程式庫中找到支援的簽章演算法清單。

使用 Keytool 刪除金鑰

金 AWS CloudHSM 鑰存放區不支援刪除金鑰。若要刪除鍵，您必須使用 AWS CloudHSM的指令行工具的`deleteKey`功能deleteKey。

使用 Keytool 產生 CSR

如果您使用 OpenSSL 動態引擎，您可以在產生憑證簽署要求 (CSR) 時獲得最大的彈性。以下命令使用 Keytool 來產生具有別名 `my-key-pair` 的金鑰對適用 CSR。

```
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**
>
> 若要使用 Keytool 的金鑰對，該金鑰對必須再指定的金鑰存放區檔案中有項目。如果您要使用非 Keytool 產生的金鑰對，則必須將金鑰和憑證中繼資料匯入至金鑰存放區。如需匯入金鑰儲存資料的指示，請參閱使用 Keytool 將中繼憑證和根憑證匯入 AWS CloudHSM 金鑰存放區。

使用 keytool 將中繼憑證和根憑證匯入 AWS CloudHSM 金鑰存放區

若要匯入 CA 憑證，您必須在新匯入的憑證上啟用完整憑證鏈的驗證。以下為命令的範例。

```
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/
```

如果您將多個用戶端執行個體連接到 AWS CloudHSM 叢集，則在一個用戶端執行個體的金鑰存放區匯入憑證不會自動讓憑證可用於其他用戶端執行個體。您必須在每個用戶端執行個體上匯入憑證。

使用 keytool 從 AWS CloudHSM 密鑰存儲中刪除證書

下列範例顯示命令如何從 Java Keytool 金鑰存放區庫刪除憑證。

```
keytool -delete -alias mydomain -keystore \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

如果您將多個用戶端執行個體連接到 AWS CloudHSM 叢集，刪除一個用戶端執行個體金鑰存放區上的憑證並不會自動從其他用戶端執行個體中移除憑證。您必須在每個用戶端執行個體上刪除憑證。

使用 keytool 將工作證書導入 AWS CloudHSM 密鑰存儲

憑證簽署要求 (CSR) 簽署後，您就可以將其匯入 AWS CloudHSM 金鑰存放區，並與適當的金鑰對建立關聯。以下是命令範例。

```
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/
```

別名應為金鑰存放區中具有關聯憑證的金鑰對。如果金鑰是非 Keytool 產生，或是在不同的用戶端執行個體上產生，您就必須先將金鑰和憑證中繼資料匯入金鑰存放區。如需匯入憑證中繼資料的指示，請參閱使 AWS CloudHSM 用金鑰存放區註冊預先存在的金鑰中的程式碼範例。

憑證鏈必須是可驗證的。如果您無法驗證憑證，則可能需要將簽署 (憑證授權單位) 憑證匯入金鑰存放區，以便驗證該鏈結。

使用 Keytool 匯出憑證

下列範例會產生二進位 X.509 格式的憑證。若要匯出人類可讀的憑證，請新增 `-rfc` 至 `-exportcert` 命令。

```
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/
```

使用 AWS CloudHSM 金鑰存放區與業者

Jarsigner 是一種流行的命令行實用程序，用於使用安全地存儲在 HSM 上的密鑰簽名 JAR 文件。完整的 Jarsigner 教學並不在 AWS CloudHSM 文件範圍之內。本節說明您應該使用的 Jarsigner 參數，透過 AWS CloudHSM 金鑰存放區以信任的根目錄來簽署和驗證簽名。 AWS CloudHSM

設定金鑰和憑證

在您可以使用 Jarsigner 簽署 JAR 文件之前，請確保您已經設定或完成以下步驟：

1. 遵循 AWS CloudHSM 金鑰存放區必要條件中的指引。
2. 設定您的簽署金鑰以及應儲存在目前伺服器或用戶端執行個體的金 AWS CloudHSM 鑰存放區中的相關憑證和憑證鏈結。在上建立金鑰， AWS CloudHSM 然後將關聯的中繼資料匯入金 AWS CloudHSM 鑰存放區。使用在金鑰存放區註冊預先存在的金鑰中的 AWS CloudHSM程式碼範例，將中繼資料匯入金鑰存放區。如果您要使用 Keytool 設定金鑰和憑證，請參閱 使用 Keytool 建立新金鑰。如果您使用多個用戶端執行個體來簽署 JAR，請建立金鑰並匯入憑證鏈。然後將生成的金鑰存放區文件複製到每個用戶端執行個體。如果您經常產生新的金鑰，您可能會發現將憑證個別匯入至每個用戶端執行個體更容易。
3. 整個憑證鏈必須是可驗證的。若要讓憑證鏈結可驗證，您可能需要將 CA 憑證和中繼憑證新增至 AWS CloudHSM 金鑰存放區。有關使用 Java 代碼驗證證書鏈的說明，請參閱使用 AWS CloudHSM 和 Jarsigner 簽署 JAR 文件中的代碼片段。如果需要，您也可以使用 Keytool 匯入憑證。如需使用 keytool 的指示，請參閱使用 Keytool 將中繼憑證和根憑證匯入 AWS CloudHSM 金鑰存放區。

使用 AWS CloudHSM 和 Jarsigner 簽署 JAR 檔案

使用以下命令來簽署 JAR 檔案：

```
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>
```

使用以下命令來驗證已簽署的 JAR：

```
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>
```

已知問題

以下清單提供已知問題的最新清單。

- 使用 keytool 生成密鑰時，提供程序配置中的第一個提供程序不能是 CaviumProvider。
- 使用 Keytool 產生金鑰時，系統會使用安全性組態檔案中的第一個 (受支援的) 提供商來產生金鑰。這通常是軟體提供商。然後，產生的金鑰會指定別名，並在金鑰新增程序期間以永久性 (Token) 金鑰的形式匯入 AWS CloudHSM HSM。
- 將 keytool 與 AWS CloudHSM 金鑰存放區搭配使用時，請勿在指`-providerName`令行上指定`-providerclass`、或`-providerpath`選項。按照金鑰存放區先決條件中的說明，在安全提供商檔案中指定這些選項。
- 當透過 Keytool 和 Narsigner 使用不可擷取的 EC 金鑰時，SunEC 提供商必須從 java.security 檔案的提供商清單中移除/停用。如果您透過 keytool 和 Jarsigner 使用可擷取的 EC 金鑰，則提供者會從 AWS CloudHSM HSM 匯出金鑰位元，並在本機使用金鑰進行簽署作業。我們不建議您搭配 Keytool 或 Jarsigner 使用可匯出的金鑰。

在密鑰存儲中註冊預先存在的 AWS CloudHSM 密鑰

為了獲得屬性和標籤的最大安全性和彈性，建議您使用 key_mgmt_util 產生簽署金鑰。您也可以使用 Java 應用程式來產生 AWS CloudHSM中的金鑰。

下節提供程式碼範例，示範如何在 HSM 上產生新 key pair，並使用匯入金鑰存放區的現有金鑰來註冊該金 AWS CloudHSM 鑰。匯入的金鑰可以與第三方工具 (如 Keytool 和 Jarsigner) 搭配使用。

若要使用既有的金鑰，請修改程式碼範例以依標籤查詢金鑰，而不是產生新金鑰。在上 GitHub的 KeyUtilitiesRunner.java 範例中提供了用於按標籤查找索引鍵的範例程式碼。

> **Important**
>
> AWS CloudHSM 使用本機金鑰存放區註冊儲存的金鑰不會匯出金鑰。註冊金鑰時，金鑰存放區會註冊金鑰的別名 (或標籤)，並建立本機儲存憑證物件與 AWS CloudHSM金鑰對之間的關聯。只要金鑰對建立為不可匯出，金鑰位元就不會離開 HSM。

```
                    //
// 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());
        }
    }

}
```

其他第三方廠商整合

數個協力廠商支援 AWS CloudHSM 作為信任的根源。這表示您可以在 CloudHSM 叢集建立和存放基礎金鑰時使用您選擇的軟體解決方案。因此，您的工作負載 AWS 可以依賴 CloudHSM 的延遲、可用性、可靠性和彈性優勢。以下清單包含支援 CloudHSM 的第三方廠商。

> **Note**
>
> AWS 不為任何第三方供應商背書或擔保。

- Hashicorp Vault 是一種秘密管理工具旨在提供組織之間的協作和控管。它支持 AWS Key Management Service 並 AWS CloudHSM 作為額外保護的信任根源。
- Thycotic Secrets Server 可協助客戶管理特殊權限帳戶間的敏感登入資料。它支持 AWS CloudHSM 作為信任的根源。
- P6R 的 KMIP 介面卡可讓您透過標準的 KMIP 介面使用您的 AWS CloudHSM 執行個體。
- PrimeKey EJBCA 是 PKI 的一種流行的開源解決方案。它允許您使用安全地創建和存儲密鑰對 AWS CloudHSM。
- Box KeySafe 為許多具有嚴格安全性、隱私權和法規遵循要求的組織提供雲端內容的加密金鑰管理。客戶可以透 AWS CloudHSM 過 AWS KMS 自訂 KeySafe 金鑰存放區直接 AWS Key Management Service 或間接進一步保護金鑰。
- Insyde 軟體支援 AWS CloudHSM 作為韌體簽署的信任根源。
- F5 大 IP LTM AWS CloudHSM 作為信任的根源支持。
- Cloudera Navigator Key HSM 可讓您使用 CloudHSM 叢集，來為 Cloudera Navigator Key Trustee 伺服器建立和存放金鑰。
- Venafi 信任保護平台透過 AWS CloudHSM 金鑰產生和保護功能為 TLS、SSH 和程式碼簽署提供全面的機器身分管理。

監控 AWS CloudHSM

除了用戶端開發套件內建的記錄功能之外，您還可以使用 AWS CloudTrail Amazon CloudWatch 日誌和 Amazon CloudWatch 來監控 AWS CloudHSM。

用戶端 SDK 日誌

使用 Client SDK 記錄來監視您建立之應用程式的診斷和疑難排解資訊。

CloudTrail

用 CloudTrail 於監控 AWS 帳戶中的所有 API 呼叫，包括您用來建立和刪除叢集、硬體安全性模組 (HSM) 和資源標籤所進行的呼叫。

CloudWatch 日誌

使用 CloudWatch 記錄來監控 HSM 執行個體的記錄，其中包括建立和刪除 HSM 使用者、變更使用者密碼、建立和刪除金鑰等事件。

CloudWatch

用 CloudWatch 於即時監控叢集的健全狀況。

主題

使用用戶端 SDK 日誌

您可以擷取用戶端 SDK 產生的記錄檔。 AWS CloudHSM 提供使用用戶端 SDK 3 和用戶端 SDK 5 記錄的實作。

主題

用戶端 SDK 5 記錄

用戶端 SDK 5 日誌包含以元件命名的檔案中每個元件的資訊。您可以使用用戶端 SDK 5 的設定工具來設定每個元件的記錄。

如果您沒有指定檔案的位置，系統會將日誌寫入預設位置：

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
```

如需關於如何設定用戶端 SDK 5 記錄的資訊，請參閱用戶端 SDK 5 設定工具

用戶端 SDK 3 記錄

用戶端 SDK 3 記錄檔包含來自 AWS CloudHSM 用戶端常駐程式的詳細資訊。日誌的位置取決於您執行用戶端常駐程式的 Amazon EC2 用戶端執行個體的作業系統。

Amazon Linux

在 Amazon Linux 中，用 AWS CloudHSM 戶端日誌會寫入名為的檔案`/opt/cloudhsm/run/cloudhsm_client.log`。您可以使用 logrotate 或類似的工具來輪換和管理這些日誌。

Amazon Linux 2

在 Amazon Linux 2 中， AWS CloudHSM 用戶端日誌會收集並存放在日誌中。您可以使用 journalctl 來檢視和管理這些日誌。例如，使用下列命令檢視用 AWS CloudHSM 戶端記錄檔。

```
journalctl -f -u cloudhsm-client
```

CentOS 7

在 CentOS 7 中， AWS CloudHSM 用戶端記錄會被收集並儲存在日誌中。您可以使用 journalctl 來檢視和管理這些日誌。例如，使用下列命令檢視用 AWS CloudHSM 戶端記錄檔。

```
journalctl -f -u cloudhsm-client
```

CentOS 8

在 CentOS 8 中， AWS CloudHSM 用戶端記錄會被收集並儲存在日誌中。您可以使用 journalctl 來檢視和管理這些日誌。例如，使用下列命令檢視用 AWS CloudHSM 戶端記錄檔。

```
journalctl -f -u cloudhsm-client
```

RHEL 7

在 RHEL 7 中， AWS CloudHSM 用戶端記錄會收集並儲存在日誌中。您可以使用 journalctl 來檢視和管理這些日誌。例如，使用下列命令檢視用 AWS CloudHSM 戶端記錄檔。

```
journalctl -f -u cloudhsm-client
```

RHEL 8

在 RHEL 8 中， AWS CloudHSM 用戶端記錄會收集並儲存在日誌中。您可以使用 journalctl 來檢視和管理這些日誌。例如，使用下列命令檢視用 AWS CloudHSM 戶端記錄檔。

```
journalctl -f -u cloudhsm-client
```

Ubuntu 16.04

在 Ubuntu 16.04 中， AWS CloudHSM 用戶端記錄檔會收集並儲存在日誌中。您可以使用 journalctl 來檢視和管理這些日誌。例如，使用下列命令檢視用 AWS CloudHSM 戶端記錄檔。

```
journalctl -f -u cloudhsm-client
```

Ubuntu 18.04

在 Ubuntu 18.04 中， AWS CloudHSM 用戶端記錄檔會收集並儲存在日誌中。您可以使用 journalctl 來檢視和管理這些日誌。例如，使用下列命令檢視用 AWS CloudHSM 戶端記錄檔。

```
journalctl -f -u cloudhsm-client
```

Windows

- 用於 Windows 用戶端 1.1.2+：

  AWS CloudHSM 用戶端記錄檔會寫入程式`cloudhsm.log`檔 AWS CloudHSM 案資料夾 (`C:\Program Files\Amazon\CloudHSM\`) 中的檔案。每個記錄檔名稱的尾碼都會加上時間戳記，指出 AWS CloudHSM 用戶端啟動的時間。

- 用於 Windows 用戶端 1.1.1 和更早版本：

  用戶端日誌不會被寫入檔案。記錄檔會顯示在命令提示字元或您啟動 AWS CloudHSM 用戶端的 PowerShell 視窗中。

使用 AWS CloudTrail 和 AWS CloudHSM

AWS CloudHSM 與 (提供中的使用者 AWS CloudTrail、角色或服務所採取的動作記錄) 的 AWS 服務整合 AWS CloudHSM。 CloudTrail 擷取 AWS CloudHSM 作為事件的所有 API 呼叫。擷取的呼叫包括來自 AWS CloudHSM 主控台的呼叫和 AWS CloudHSM API 作業的程式碼呼叫。如果您建立追蹤，您可以啟用持續交付 CloudTrail 事件到 Amazon S3 儲存貯體，包括 AWS CloudHSM. 如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail 主控台中最近的事件。使用收集的資訊 CloudTrail，您可以判斷提出的要求 AWS CloudHSM、提出要求的 IP 位址、提出要求的人員、提出要求的時間，以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail 用者指南。如需 AWS CloudHSM API 作業的完整清單，請參閱 AWS CloudHSM API 參考中的動作。

AWS CloudHSM 中的資訊 CloudTrail

CloudTrail 在您創建 AWS 帳戶時，您的帳戶已啟用。當活動發生在中時 AWS CloudHSM，該活動會與事件歷史記錄中的其他 AWS 服務 CloudTrail 事件一起記錄在事件中。您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱檢視具有事 CloudTrail 件記錄的事件。

如需 AWS 帳戶中持續記錄事件 (包括的事件) AWS CloudHSM，請建立追蹤。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。根據預設，當您在主控台建立追蹤記錄時，追蹤記錄會套用到所有 AWS 區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。如需詳細資訊，請參閱下列內容：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔並從多個帳戶接收 CloudTrail記錄檔

CloudTrail 記錄所有 AWS CloudHSM 作業，包括唯讀作業，例如`DescribeClusters`和`ListTags`，以及管理作業`InitializeCluster`，例如`CreatHsm`、和`DeleteBackup`。

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 要求是使用根使用者登入資料還是 AWS Identity and Access Management (IAM) 使用者登入資料提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱CloudTrail 使 userIdentity 元素。

瞭解 AWS CloudHSM 記錄檔項目

追蹤是一種組態，可讓事件以日誌檔的形式傳遞到您指定的 Amazon S3 儲存貯體。 CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時

間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範 AWS CloudHSM `CreateHsm`動作的 CloudTrail 記錄項目。

```
{
    "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"
}
```


使用 Amazon CloudWatch 日誌和 AWS CloudHSM 稽核日誌

當您帳戶中的 HSM 收到來自命令列工具或軟體程式庫的 AWS CloudHSM 指令時，它會以稽核記錄表單記錄命令的執行情況。HSM 稽核日誌包含所有用戶端啟動的管理命令，包含建立和刪除 HSM、登入和登出 HSM 以及管理使用者和金鑰的命令。這些日誌提供已變更 HSM 狀態的可靠動作記錄。

AWS CloudHSM 收集您的 HSM 稽核日誌，並代表您將其傳送至 Amazon CloudWatch 日誌。您可以使用 CloudWatch 日誌的功能來管理 AWS CloudHSM 稽核日誌，包括搜尋和篩選日誌，以及將日誌資料匯出到 Amazon S3。您可以在 Amazon CloudWatch 主控台中使用 HSM 稽核日誌，或使用AWS CLI和 CloudWatch 日誌開發套件中的CloudWatch 日誌命令。

主題

- HSM 稽核記錄的運作方式
- 檢視記錄檔中的 HSM 稽核記錄 CloudWatch
- 解譯 HSM 稽核日誌
- HSM 稽核日誌參考

HSM 稽核記錄的運作方式

稽核記錄會在所有 AWS CloudHSM 叢集中自動啟用。它無法停用或關閉，也沒有任何設定可以阻 AWS CloudHSM 止將記錄檔匯出至 CloudWatch 記錄檔。每個日誌事件都具有一個時間戳記和序列號碼，指出事件的順序並協助您偵測任何日誌竄改。

每個 HSM 執行個體都會產生自身的日誌。各種 HSM 的稽核日誌 (甚至是位於相同叢集中的 HSM) 都可能不同。例如，只有每個叢集中的第一個 HSM 會記錄 HSM 的初始化。初始化事件不會出現在從備份複製之 HSM 的日誌中。同樣地，當您建立金鑰時，產生金鑰的 HSM 會記錄金鑰產生事件。其他位於叢集中的 HSM 則會在透過同步接收到金鑰時記錄事件。

AWS CloudHSM 收集日誌並將其發佈到您帳戶中的 CloudWatch 日誌中。若要代表您與 Lo CloudWatch gs 服務通訊，請 AWS CloudHSM 使用服務連結角色。與角色相關聯的 IAM 政策僅 AWS CloudHSM 允許執行將稽核記錄傳送至 CloudWatch 記錄所需的工作。

> **Important**
>
> 若您在 2018 年 1 月 20 日前建立叢集，並且尚未建立連接的服務連結角色，您必須手動建立該角色。這對於從 AWS CloudHSM 叢集 CloudWatch 接收稽核記錄是必要的。如需有關服務連結角色建立的詳細資訊，請參閱了解服務連接角色，以及 IAM 使用者指南的建立一個服務連接角色。

檢視記錄檔中的 HSM 稽核記錄 CloudWatch

Amazon CloudWatch Logs 會將稽核日誌組織到日誌群組中，並在日誌群組中將其組織到日誌串流中。每個記錄項目都是一個事件。 AWS CloudHSM 為每個叢集建立一個記錄群組，並為叢集中的每個 HSM 建立一個記錄資料流。您不需要建立任何 CloudWatch 記錄元件或變更任何設定。

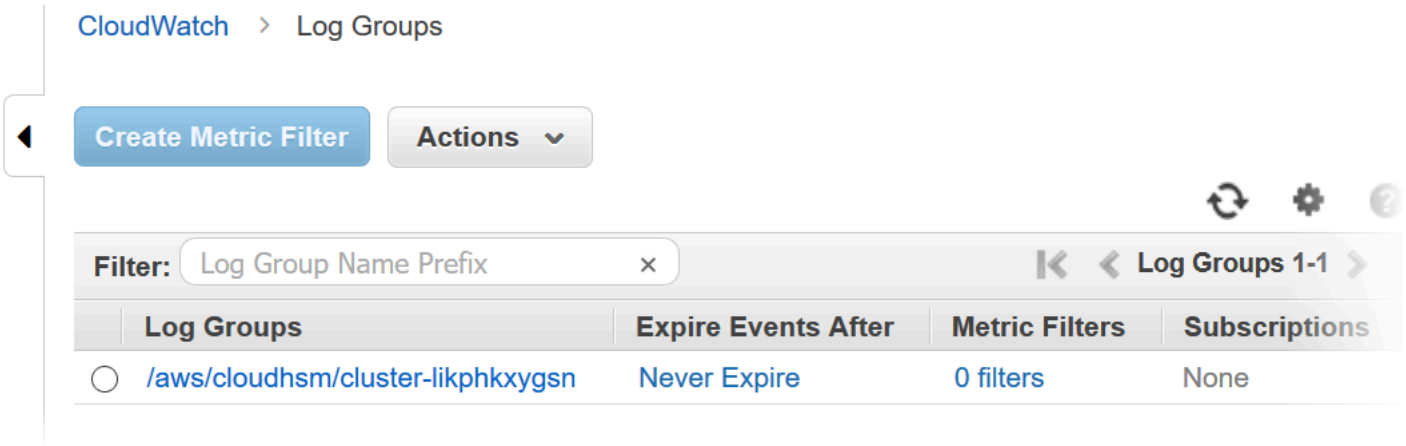
- 日誌群組名稱是 `/aws/cloudhsm/<cluster ID>`；例如 `/aws/cloudhsm/cluster-likphkxygsn`。在 AWS CLI 或 PowerShell 命令中使用記錄群組名稱時，請務必以雙引號括住該名稱。
- 日誌串流名稱是 HSM ID；例如 `hsm-nwbbiqbj4jk`。

  一般而言，每個 HSM 都有一個日誌串流。不過，任何變更 HSM ID 的動作 (例如 HSM 故障及更換時) 都會建立新的日誌串流。

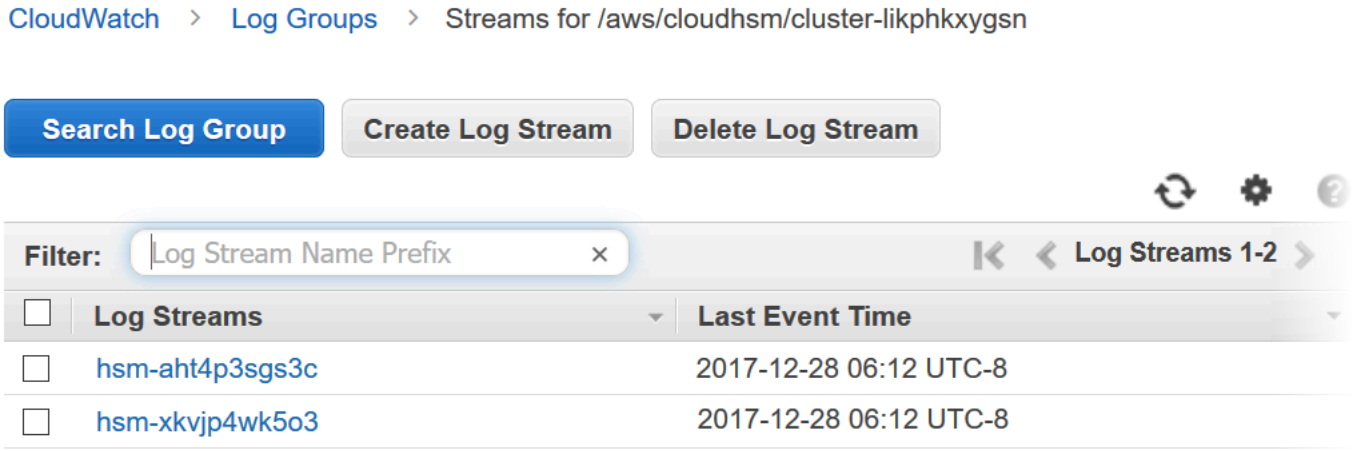
如需有關 CloudWatch 日誌概念的詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南中的概念。

您可以從中的 [記錄] 頁面 AWS Management Console、 CloudWatch 記錄指令 PowerShell程式或記錄 SDK 中的 [CloudWatch 記錄檔] 命令 AWS CLI，檢視 HSM 的稽核CloudWatch 記錄。CloudWatch 如需指示，請參閱 Amazon CloudWatch 日誌使用者指南中的檢視日誌資料。

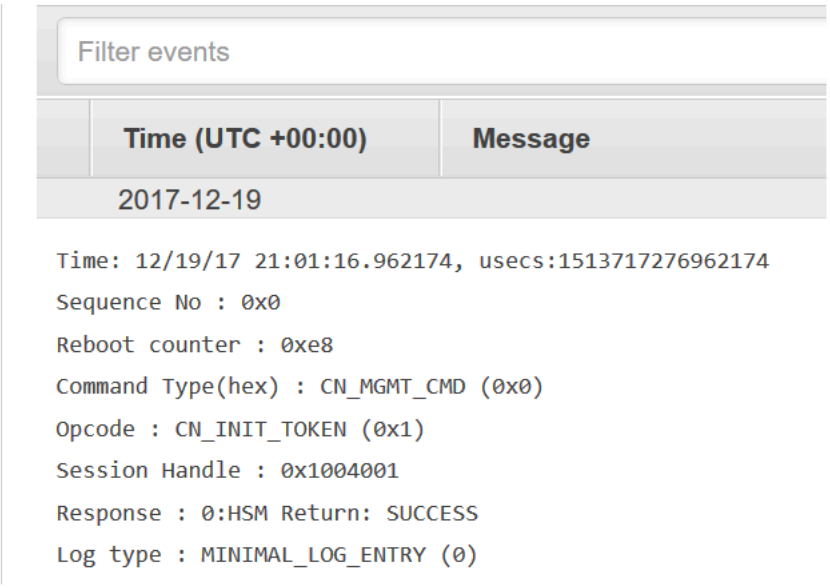
例如，下圖顯示 AWS Management Console中 `cluster-likphkxygsn` 叢集的日誌群組。



當您選擇叢集日誌群組名稱時，可以檢視叢集中每個 HSM 的日誌串流。下圖顯示 `cluster-likphkxygsn` 叢集中 HSM 的日誌串流。

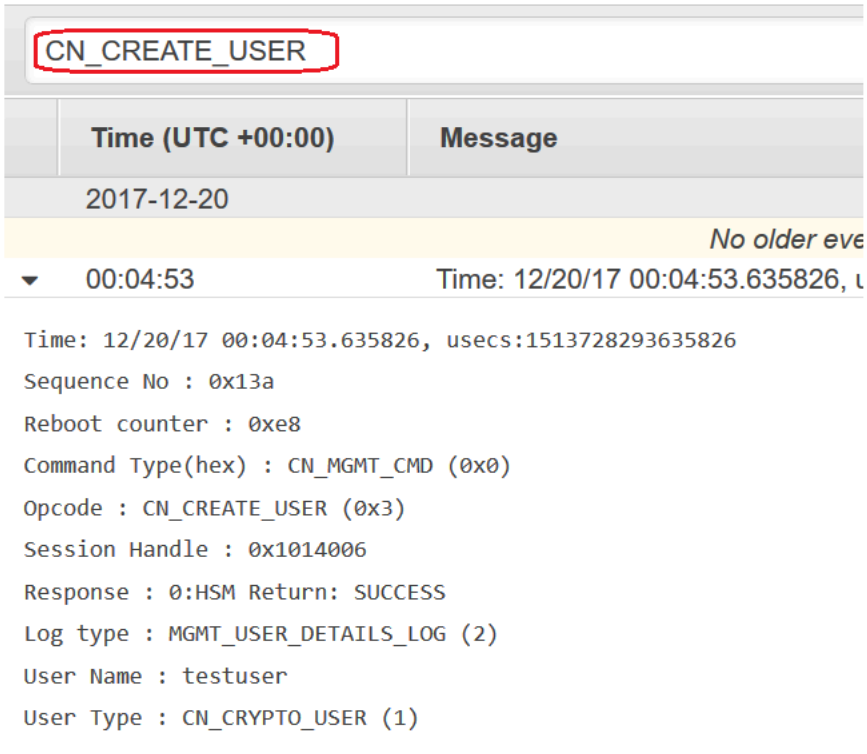


當您選擇 HSM 日誌串流名稱時，可以檢視稽核日誌中的事件。例如，此事件 (序號為 0x0 且 Opcode 為 `CN_INIT_TOKEN`) 一般是每個叢集中第一個 HSM 的第一個事件。它記錄叢集中 HSM 的初始化。



您可以使用 CloudWatch 記錄檔中的所有許多功能來管理稽核記錄。例如，您可以使用篩選事件功能來尋找事件中的特定文字，例如 CN_CREATE_USER 0pcode。

若要尋找所有未包含指定文字的事件，請在文字前面新增減號 (-)。例如，若要尋找未包含 CN_CREATE_USER 的事件，請輸入 -CN_CREATE_USER。



解譯 HSM 稽核日誌

HSM 稽核日誌中的事件具有標準欄位。有些事件類型具有額外的欄位，可擷取事件相關的實用資訊。例如，使用者登入和使用者管理事件會包含使用者名稱和使用者的使用者類型。金鑰管理命令則包含金鑰控點。

有數個欄位可提供特別重要的資訊。Opcode 會識別記錄的管理命令。Sequence No 則會識別日誌串流中的事件，指出記錄該事件的順序。

例如，下列範例事件為 HSM 日誌串流中的第二個事件 (Sequence No: 0x1)。它會顯示 HSM 產生密碼加密金鑰，即其啟動常式的一部分。

```
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)
```

下列欄位適用於稽核記錄中的每個 AWS CloudHSM 事件。

時間

事件發生的時間，其時區為 UTC 時區。時間會以可供人閱讀的時間格式，以及單位為微秒的 Unix 時間顯示。

重新開機計數器

32 位元的持久性序數計數器，會在 HSM 硬體重新開機時遞增。

日誌串流中的所有事件都具有相同的重新開機計數器值。但是，重新開機計數器對日誌串流來說可能並非唯一，因為相同叢集中不同 HSM 執行個體的計數都可能不同。

序號

64 位元的序數計數器，會在發生每個日誌事件時遞增。每個日誌串流中的第一個事件都具有 0x0 的序號。Sequence No 的值之間不應會有任何間隙。序號在日誌串流中是唯一的。

命令類型

代表命令種類的十六進位值。 AWS CloudHSM 日誌串流中命令的命令類型為 CN_MGMT_CMD (0x0) 或 CN_CERT_AUTH_CMD (0x9)。

Opcode

識別執行的管理命令。如需 AWS CloudHSM 稽核記錄中的Opcode值清單，請參閱HSM 稽核日誌參考。

工作階段控點

識別執行命令及記錄事件的工作階段。

回應

記錄針對管理命令的回應。您可以針對 SUCCESS 和 ERROR 值搜尋 Response 欄位。

日誌類型

指示記錄命令之記 AWS CloudHSM 錄檔的記錄類型。

- MINIMAL_LOG_ENTRY (0)
- MGMT_KEY_DETAILS_LOG (1)
- MGMT_USER_DETAILS_LOG (2)
- GENERIC_LOG

稽核日誌事件的範例

日誌串流中的事件會記錄 HSM 從建立到刪除的歷史記錄。您可以使用日誌檢閱您 HSM 的生命週期，洞見其操作。當您解譯事件時，請注意表示管理命令或動作的 Opcode，以及指出事件順序的 Sequence No。

主題

範例：初始化叢集中的第一個 HSM

每個叢集中第一個 HSM 的稽核日誌串流都與叢集中其他 HSM 的日誌串流有相當大的差異。每個叢集中第一個 HSM 的稽核日誌會記錄其建立和初始化。叢集中其他從備份產生之 HSM 的日誌會從登入事件開始。

> **Important**
>
> 下列初始化項目不會出現在 CloudHSM 稽核記錄功能發行之前初始化的叢集記錄中 (2018 年 8 月 30 CloudWatch 日)。如需詳細資訊，請參閱文件歷史記錄。

下列範例事件會出現在叢集中第一個 HSM 的日誌串流內。日誌中的第一個事件 (帶有 Sequence No 0x0 的事件) 代表初始化 HSM 的命令 (CN_INIT_TOKEN)。回應表示命令成功 (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)
```

此範例日誌串流中的第二個事件 (Sequence No 0x1) 會記錄建立 HSM 使用之密碼加密金鑰的命令 (CN_GEN_PSWD_ENC_KEY)。

這是每個叢集中第一個 HSM 很典型的過程。因為相同叢集中的後續 HSM 都是第一個 HSM 的複本，他們會使用相同的密碼加密金鑰。

```
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)
```

此範例日誌串流中的第三個事件 (Sequence No 0x2) 為建立「應用裝置使用者 (AU)」???，即 AWS CloudHSM 服務。涉及 HSM 使用者的事件會包含使用者名稱和使用者類型的額外欄位。

```
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)
```

此範例日誌串流中的第四個事件 (Sequence No 0x3) 會記錄 CN_INIT_DONE 事件，即完成 HSM 初始化。

```
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)
```

您可以依循啟動過程中的剩餘事件。這些事件可能包含幾項登入和登出事件，以及金鑰加密金鑰 (KEK) 的產生。下列事件會記錄變更「前加密主管 (PRECO)」???密碼的命令。此命令會啟用叢集。

```
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)
```

登入及登出事件

當解譯您的稽核日誌時，請注意記錄使用者登入和登出 HSM 的事件。這些事件可協助您判斷從登入到登出命令之間，執行管理命令的使用者序列為何。

例如，此日誌項目會記錄名為 admin 之加密主管的登入。序號 (0x0) 指出這是此日誌串流中的第一個事件。

當使用者登入 HSM 時，叢集中的其他 HSM 也會記錄使用者的登入事件。初始登入事件之後不久，您便可以在叢集中其他 HSM 的日誌串流中找到相對應的登入事件。

```
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)
```

以下範例事件會記錄 admin 加密主管登出。序號 (0x2) 指出這是此日誌串流中的第三個事件。

若登入的使用者在沒有登出的情況下關閉工作階段，日誌串流便會包含一個 CN_APP_FINALIZE 或關閉工作階段事件 (CN_SESSION_CLOSE)，而非 CN_LOGOUT 事件。與登入事件不同，此登出事件通常只會由執行命令的 HSM 記錄。

```
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)
```

若登入嘗試因為使用者名稱無效而失敗，則 HSM 會記錄一項 CN_LOGIN 事件，其中包含登入命令中提供的使用者名稱和類型。回應會顯示錯誤訊息 157，解釋使用者名稱不存在。

```
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)
```

若登入嘗試因為密碼無效而失敗，則 HSM 會記錄一項 CN_LOGIN 事件，其中包含登入命令中提供的使用者名稱和類型。回應會顯示帶有 RET_USER_LOGIN_FAILURE 錯誤碼的錯誤訊息。

```
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)
```

範例：建立和刪除使用者

此範例會顯示加密管理員 (CO) 建立和刪除使用者時記錄的日誌事件。

第一個事件記錄 CO (admin) 登入 HSM。序號為 0x0 表示此為日誌串流中的第一個事件。事件中也包含了登入使用者的名稱和類型。

```
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)
```

日誌串流中的下一個事件 (序號為 0x1) 則記錄了 CO 建立新的加密使用者 (CU)。事件中也包含了新使用者的名稱和類型。

```
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)
```

然後，CO 會建立另一個加密管理員 (alice)。序號指出此動作是接續在前一個動作之後，期間沒有任何介入動作。

```
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)
```

稍後，名為 admin 的 CO 登入並刪除名為 alice 的加密管理員。HSM 記錄 CN_DELETE_USER 事件。事件中也包含了遭刪除之使用者的名稱和類型。

```
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)
```

範例：建立和刪除金鑰對

此範例顯示您建立及刪除金鑰對時，於 HSM 稽核日誌中記錄的事件。

下列事件會記錄名為 crypto_user 的加密使用者 (CU) 登入 HSM。

```
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)
```

接著，CU 會產生一組金鑰對 (CN_GENERATE_KEY_PAIR)。私有金鑰具有金鑰控制代碼 131079。公有金鑰具有金鑰控制代碼 131078。

```
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
```

CU 立即刪除了金鑰對。CN_DESTROY_OBJECT 事件會記錄公有金鑰的刪除 (131078)。

```
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
```

接著，第二個 CN_DESTROY_OBJECT 事件會記錄私有金鑰的刪除 (131079)。

```
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
```

最後，CU 登出。

```
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)
```

範例：產生及同步金鑰

此範例顯示在具有多個 HSM 的叢集中建立金鑰的效果。在其中一個 HSM 上產生的金鑰，會從 HSM 以遮罩物件擷取，然後以遮罩物件插入其他 HSM。

> **Note**
>
> 用戶端工具可能會無法成功同步金鑰。或是命令可能會包含 min_srv 參數，限定將金鑰同步至指定數量的 HSM。在任一情況下， AWS CloudHSM 服務都會將金鑰同步處理至叢集中的其他 HSM。由於 HSM 僅會在其日誌中記錄用戶端的管理命令，因此伺服器端的同步不會記錄在 HSM 的日誌中。

首先請考慮接收及執行命令的 HSM 日誌串流。日誌串流會針對 HSM ID hsm-abcde123456 予以命名，但 HSM ID 未出現在日誌事件中。

首先，testuser 加密使用者 (CU) 登入 hsm-abcde123456 HSM。

```
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)
```

CU 執行exSymKey命令來產生對稱金鑰。hsm-abcde123456 HSM 會產生金鑰控點為 262152 的對稱金鑰。HSM 會在其日誌中記錄 CN_GENERATE_KEY 事件。

```
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-abcde123456 之日誌串流中的下一個事件會記錄金鑰同步處理程序中的第一個步驟。從 HSM 將新的金鑰 (金鑰控點 262152) 擷取為遮罩物件。

```
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
```

現在請考慮 HSM hsm-zyxwv987654 (相同叢集中的另一個 HSM) 的日誌串流。此日誌串流也包含 testuser CU 的登入事件。時間值會在使用者登入 hsm-abcde123456 HSM 之後短暫發生。

```
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)
```

此 HSM 的這個日誌串流沒有 CN_GENERATE_KEY 事件。但它的事件會記錄與此 HSM 的金鑰同步處理。CN_INSERT_MASKED_OBJECT_USER 事件會記錄將金鑰 262152 接收為遮罩物件。現在，金鑰 262152 存在於叢集的兩個 HSM 上。

```
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 使用者登出時，此 CN_LOGOUT 事件只會顯示在收到命令之 HSM 的日誌串流中。

範例：匯出金鑰

此範例顯示加密使用者 (CU) 從含多個 HSM 的叢集匯出金鑰時所記錄的稽核日誌事件。

下列事件會記錄 CU (testuser) 登入 key_mgmt_util。

```
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)
```

CU 會執行exSymKey指令來匯出金鑰 7 (256 位元 AES 金鑰)。此命令使用金鑰 6 (HSM 上的 256 位元 AES 金鑰) 做為包裝金鑰。

接收此命令的 HSM 會記錄金鑰 7 (即將匯出的金鑰) 的 CN_WRAP_KEY 事件。

```
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
```

然後，HSM 會記錄包裝金鑰 (金鑰 6) 的 CN_NIST_AES_WRAP 事件。金鑰會予以包裝後立即予以解除包裝，但 HSM 只會記錄一個事件。

```
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 命令會將匯出的金鑰寫入至檔案，但不會變更 HSM 上的金鑰。因此，叢集中其他 HSM 的日誌內沒有對應事件。

範例：匯入金鑰

此範例顯示您將金鑰匯入至叢集中的 HSM 時所記錄的稽核日誌事件。在此範例中，加密使用者 (CU) 使用imSymKey指令將 AES 金鑰匯入 HSM。此命令使用金鑰 6 做為包裝金鑰。

接收這些命令的 HSM 會先記錄金鑰 6 (包裝金鑰) 的 CN_NIST_AES_WRAP 事件。

```
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
```

接著，HSM 會記錄代表匯入操作的 CN_UNWRAP_KEY 事件。匯入的金鑰會獲指派金鑰控點 11。

```
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
```

產生或匯入新的金鑰時，用戶端工具會自動嘗試同步處理新的金鑰與叢集中的其他 HSM。在此情況下，將金鑰 11 從 HSM 擷取為遮罩物件時，HSM 會記錄 CN_EXTRACT_MASKED_OBJECT_USER 事件。

```
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
```

叢集中其他 HSM 的日誌串流會反映收到新匯入的金鑰。

例如，此事件會記錄在相同叢集中不同 HSM 的日誌串流內。此 CN_INSERT_MASKED_OBJECT_USER 事件會記錄收到代表金鑰 11 的遮罩物件。

```
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
```

範例：共用和取消共用金鑰

此範例顯示當加密使用者 (CU) 與其他加密使用者共用或停止共用 ECC 私有金鑰所記錄的稽核日誌事件。CU 可使用 shareKey 命令，並提供金鑰控制代碼、使用者 ID，及 1 值以共用金鑰，或 0 值以停止共用金鑰。

在下列範例中，接收命令的 HSM 會記錄 CM_SHARE_OBJECT 事件，其表示共用操作。

```
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 稽核日誌參考

AWS CloudHSM 在稽核記錄事件中記錄 HSM 管理命令。每個事件都有一個操作碼 (Opcode) 值，以識別發生的動作和其回應。您可以使用 Opcode 值來搜尋、排序和篩選日誌。

下表定義 AWS CloudHSM 稽核記錄中的Opcode值。

操作碼 (Opcode)	描述
使用者登入：這些事件包含使用者名稱和使用者類型。	
CN_LOGIN (0xd)	使用者登入
CN_LOGOUT (0xe)	使用者登出
CN_APP_FINALIZE	與 HSM 的連線已關閉。此連線中的任何工作階段金鑰或法定權杖都會遭到刪除。
CN_CLOSE_SESSION	與 HSM 的工作階段已關閉。此工作階段中的任何工作階段金鑰或法定權杖都已刪除。
使用者管理：這些事件包含使用者名稱和使用者類型。	
CN_CREATE_USER (0x3)	建立加密使用者 (CU)
CN_CREATE_CO	建立加密主管 (CO)
CN_DELETE_USER	刪除使用者
CN_CHANGE_PSWD	變更使用者密碼

操作碼 (Opcode)	描述
CN_SET_M_VALUE	為使用者動作設定法定驗證 (M，共 N 個)
CN_APPROVE_TOKEN	核准使用者動作的法定驗證 Token
CN_DELETE_TOKEN	刪除一個或多個仲裁令牌
CN_GET_TOKEN	請求簽名令牌以啟動仲裁操作
金鑰管理：這些事件包含金鑰控制代碼。	
CN_GENERATE_KEY	產生對稱金鑰
CN_GENERATE_KEY_PAIR (0x19)	產生非對稱 key pair
CN_CREATE_OBJECT	匯入公有金鑰 (不含包裝)
CN_MODIFY_OBJECT	設定金鑰屬性
CN_DESTROY_OBJECT (0x11)	刪除工作階段金鑰
CN_TOMBSTONE_OBJECT	刪除令牌密鑰
CN_SHARE_OBJECT	共用或取消共用金鑰
CN_WRAP_KEY	匯出金鑰 (wrapKey) 的加密複本
CN_UNWRAP_KEY	匯入金鑰 (unwrapKey) 的加密複本
CN_DERIVE_KEY	從現有密鑰導出對稱密鑰
CN_NIST_AES_WRAP	使用 AES 密鑰加密或解密密鑰
CN_INSERT_MASKED_OBJECT_USER	插入具有叢集中其他 HSM 屬性的加密金鑰。
CN_EXTRACT_MASKED_OBJECT_USER	包裝/加密具有來自HSM 屬性的金鑰，以便傳送到叢集中的另一個 HSM。
Back up HSMs	
CN_BACKUP_BEGIN	開始備份程序

操作碼 (Opcode)	描述
CN_BACKUP_END	完成備份程序
CN_RESTORE_BEGIN	從備份開始還原
CN_RESTORE_END	從備份完成還原過程
Certificate-Based Authentication	
CN_CERT_AUTH_STORE_CERT	儲存叢集憑證
HSM Instance Commands	
CN_INIT_TOKEN (0x1)	啟動 HSM 初始化程序
CN_INIT_DONE	HSM 初始化程序已完成
CN_GEN_KEY_ENC_KEY	產生金鑰加密金鑰 (KEK)
CN_GEN_PSWD_ENC_KEY (0x1d)	產生密碼加密金鑰 (PEK)
HSM crypto commands	
CN_FIPS_RAND	產生符合 FIPS 標準的隨機數字

取得的 CloudWatch 指標 AWS CloudHSM

用 CloudWatch 於即時監控 AWS CloudHSM 叢集。指標可以按區域、叢集 ID 或叢集 ID 和 HSM ID 分類。

AWS/CloudHSM 命名空間包含下列指標：

指標	描述
HsmUnhealthy	HSM 執行個體未正確執行。 AWS CloudHSM 會自動替換運作狀態不良的執行個體。在我們更換 HSM 同時，您可以選擇主動擴展叢集大小以降低效能影響。

指標	描述
HsmTemperature [1]	硬體處理器的結合溫度。溫度到達攝氏 110 度時系統會關閉。
HsmKeysSessionOccupied	HSM 執行個體使用的工作階段金鑰數目。
HsmKeysTokenOccupied	HSM 執行個體和叢集使用的權杖金鑰數目。
HsmSslCtxsOccupied [1]	目前為 HSM 執行個體建立的 end-to-end 加密通道數目。最多可達 2,048 個通路。
HsmSessionCount	HSM 執行個體的連線數目。最多可達 2,048 個。根據預設，用戶端精靈會設定為在一個 end-to-end 加密通道下開啟每個 HSM 執行個體的兩個工作階段。AWS CloudHSM 此外，HSM 最多可開啟 2 個連線，以監控 HSM 的健全狀況。
HsmUsersAvailable	可以建立的額外使用者數目。這等於使用者的最大數目 (列於 HsmUsersMax) 減去迄今為止建立的使用者。
HsmUsersMax [1]	可以在 HSM 執行個體上建立的使用者數目上限。目前這個數目是 1,024。
InterfaceEth2OctetsInput [1]	迄今為止累計傳入 HSM 的流量總和。
InterfaceEth2OctetsOutput [1]	迄今為止累計傳出 HSM 的流量總和。

- [1] 此量度不適用於 hsm2m。

AWS CloudHSM 性能

若是生產叢集，您應該至少讓兩個 HSM 執行個體分散在一個區域的不同可用區域中。我們建議您測試叢集的負載，以確定您應預期的尖峰負載，然後在叢集中再新增一個 HSM 以確保高可用性。若是需要新產生金鑰持久性的應用程式，建議您至少讓三個 HSM 執行個體分散在一個區域的不同可用區域中。

效能資料

AWS CloudHSM 叢集的效能會根據特定的工作負載而有所不同。若要提高效能，您可以將其他 HSM 執行個體新增至叢集中。效能可能會因 EC2 執行個體上的組態、資料大小和其他應用程式負載而有所不同。我們鼓勵對您的應用程式進行負載測試，以判斷擴展需求。

下表顯示在具有 hsm1.medium 執行個體的 EC2 執行個體上執行的常見加密演算法的近似效能。

適用於 hsm1. 中型的效能資料

作業	雙 HSM 叢集 [1]	三個 HSM 叢集 [2]	六個 HSM 叢集 [3]
RSA 2048 位元符號	每秒 2,000 次運算	每秒 3,000 次運算	每秒 5,000 次運算
EC P256 符號	每秒 500 次運算	每秒 750 次運算	每秒 1,500 次運算

- [1] 雙 HSM 叢集，其中 Java 多執行緒應用程式在一個 c4.large EC2 執行個體上執行，其中一個 HSM 位於與 EC2 執行個體相同的 AZ 中。
- [2] 三個 HSM 叢集，其中 Java 多執行緒應用程式在一個 c4.large EC2 執行個體上執行，其中一個 HSM 位於與 EC2 執行個體相同的 AZ 中。
- [3] 六個 HSM 叢集，其中 Java 多執行緒應用程式在一個 c4.large EC2 執行個體上執行，其中兩個 HSM 位於與 EC2 執行個體相同的可用區域中。

HSM 調節

當您的工作負載超過叢集的 HSM 容量時，您會收到錯誤訊息，指出 HSM 忙碌或得到調節。有關發生這種情況時該怎麼做的詳細信息，請參閱 HSM 調節

中的安全性 AWS CloudHSM

雲安全 AWS 是最高的優先級。身為 AWS 客戶，您可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。

安全是 AWS 與您之間共同承擔的責任。共同責任模型將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 — AWS 負責保護在 AWS 雲端中執行 AWS 服務的基礎架構。 AWS 還為您提供可以安全使用的服務。要了解適用於的合規計劃 AWS CloudHSM，請參閱合規計劃的 AWS 服務範圍合規計劃。
- 雲端中的安全性 — 您的責任取決於您使用的 AWS 服務。您也必須對其他因素負責，包括資料的機密性、您公司的要求和適用法律和法規。

本文件可協助您瞭解如何在使用時套用共同責任模型 AWS CloudHSM。下列主題說明如何設定 AWS CloudHSM 以符合安全性與合規性目標。您也會學到如何使用其他可協助您監控和保護 AWS CloudHSM 資源的 AWS 服務。

目錄

資料保護 AWS CloudHSM

AWS 共用責任模型適用於中的資料保護 AWS CloudHSM。如此模型中所述， AWS 負責保護執行所有 AWS 雲端. 您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，我們建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 登入資料並設定個別使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源進行通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用設定 API 和使用者活動記錄 AWS CloudTrail。
- 使用 AWS 加密解決方案以及其中的所有默認安全控制 AWS 服務。
- 使用進階的受管安全服務 (例如 Amazon Macie)，協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取時需要經 AWS 過 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如名稱欄位。這包括當您使用主控台、API AWS CloudHSM 或 AWS SDK 時 AWS 服務 使用或其他使用時。 AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

靜態加密

從 HSM AWS CloudHSM 進行備份時，HSM 會在將資料傳送到之前加密其資料。 AWS CloudHSM資料會使用唯一、短暫的加密金鑰進行加密。如需詳細資訊，請參閱 AWS CloudHSM 叢集備份。

傳輸中加密

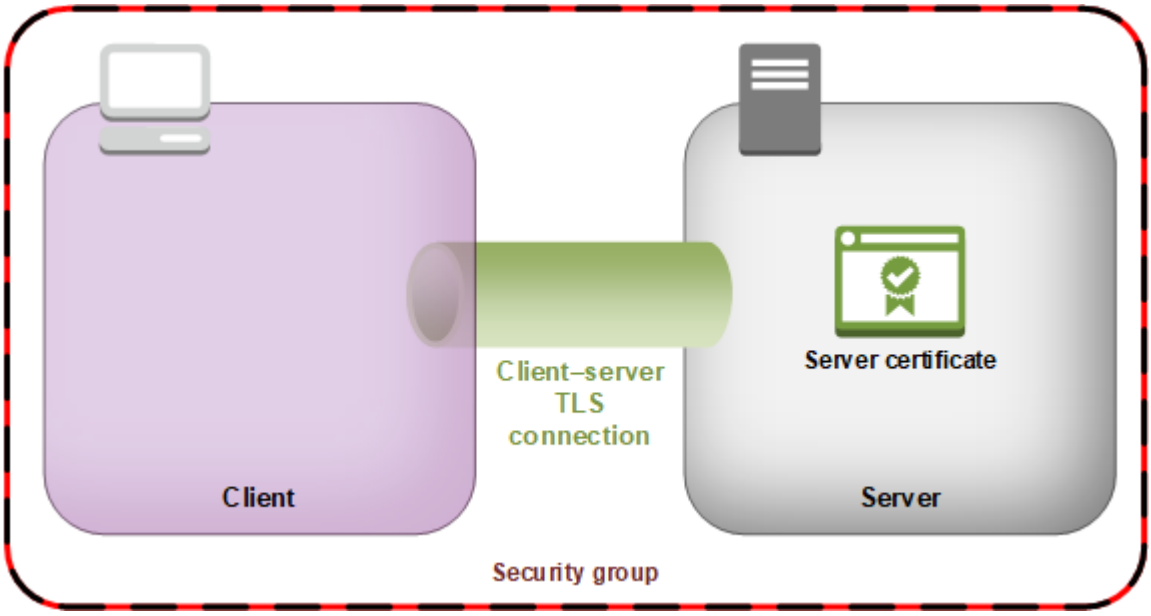
叢集中 AWS CloudHSM 用戶端與 HSM 之間的通訊會從端到端加密。只有您的用戶端和 HSM 能解密此通訊。如需詳細資訊，請參閱 E nd-to-end 加密。

AWS CloudHSM 用戶端 end-to-end 加密

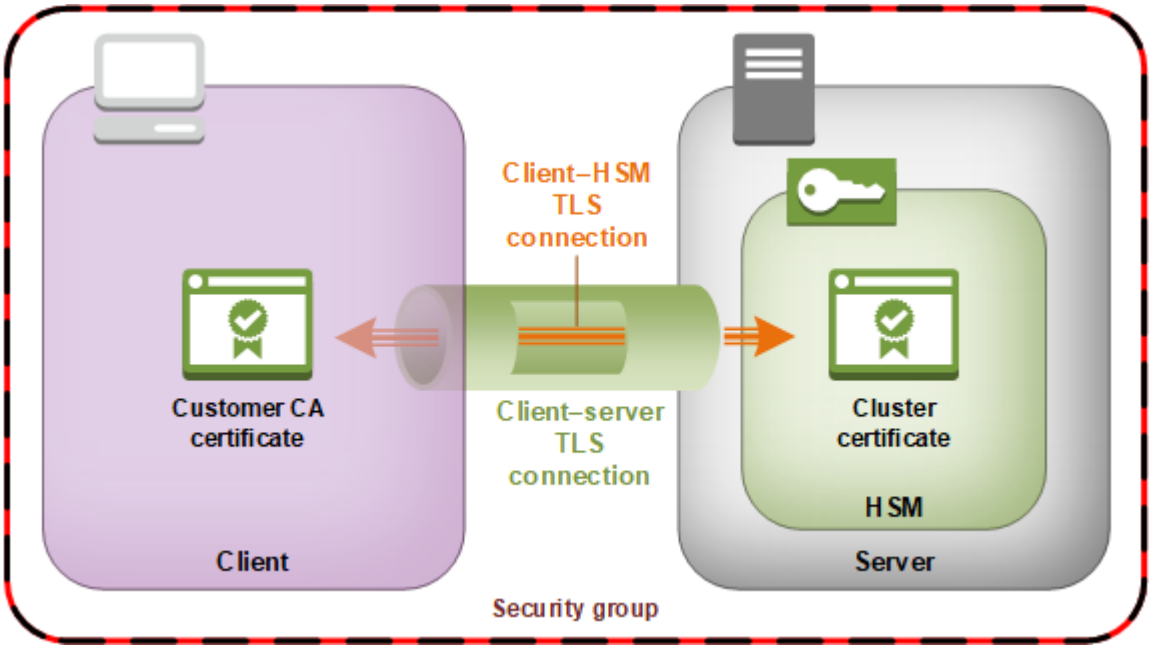
會將用戶端執行個體與您叢集中 HSM 之間的通訊進行端對端加密。只有您的用戶端和 HSM 可以解密通訊。

下列程序說明用戶端如何與 HSM 建立 end-to-end 加密通訊。

1. 您的用戶端會與託管 HSM 硬體的伺服器建立 Transport Layer Security (TLS) 連線。您叢集的安全群組僅會允許來自安全群組中用戶端執行個體對伺服器的傳入流量。用戶端也會檢查伺服器的憑證，以確保它是受信任的伺服器。



2. 下一步，用戶端會與 HSM 硬體建立加密連接。HSM 擁有您使用自己的憑證授權單位 (CA) 簽署的叢集憑證，而用戶端會有 CA 的根憑證。在建立用戶端 HSM 的加密連線之前，用戶端會對其根憑證驗證 HSM 的叢集憑證。只有在用戶端成功驗證 HSM 受信任時，才會建立連接。



叢集備份的安全性

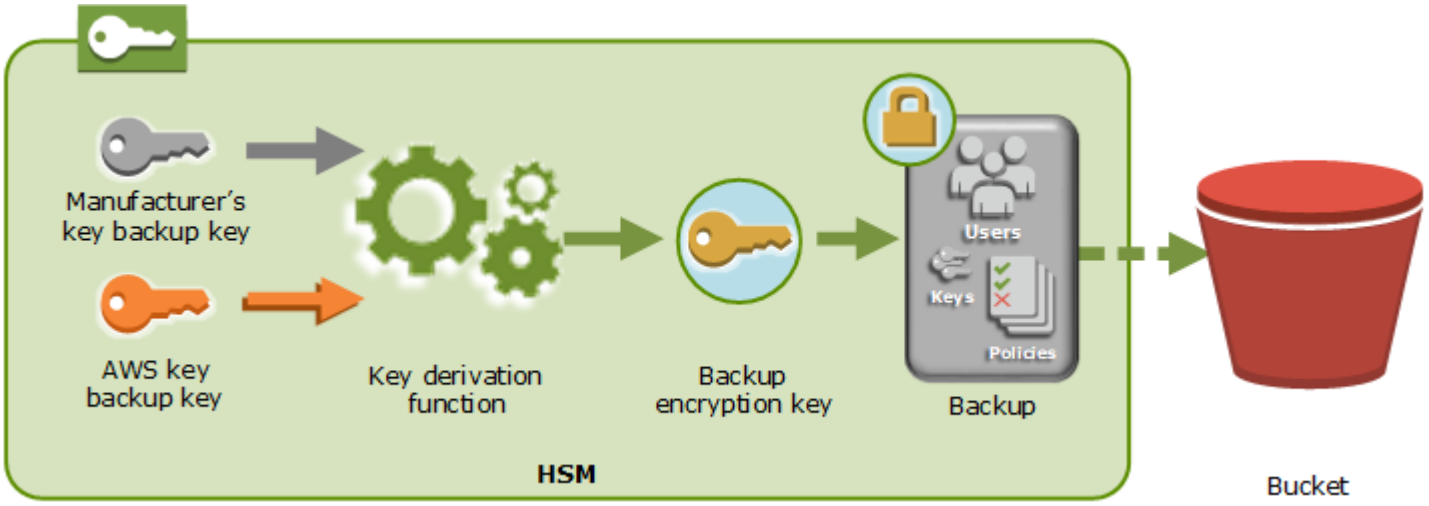
從 HSM AWS CloudHSM 進行備份時，HSM 會先加密其所有資料，然後再傳送到該資料。 AWS CloudHSM資料永遠不會以純文字格式離開 HSM。此外，無法解密備份， AWS 因為 AWS 無法存取用於解密備份的金鑰。

為了加密其資料，HSM 使用唯一、暫時性的加密金鑰，也就是暫時性備份金鑰 (EBK)。EBK 是在進行備份時 AWS CloudHSM ，HSM 內部產生的 AES 256 位元加密金鑰。HSM 會產生 EBK，然後用它來加密 HSM 的資料，使用符合 NIST 特刊 800-38F、經 FIPS 核准的 AES 金鑰包裝方法。然後 HSM 會將 AWS CloudHSM加密的資料提供給。加密的資料包含 EBK 加密的複本。

若要加密 EBK，HSM 會使用另一個加密金鑰，稱為持久性備份金鑰 (PBK)。PBK 也是 AES 256 位元加密金鑰。若要產生 PBK，HSM 會在符合 NIST 特刊 800-108 的計數器模式中使用經 FIPS 核准的金鑰衍生函數 (KDF)。此 KDF 的輸入包括下列項目：

- 製造商金鑰備份金鑰 (MKBK)，由硬體製造商永久內嵌在 HSM 硬體中。
- AWS 金鑰備份金鑰 (AKBK)，安全地安裝在 HSM 最初由設定時。 AWS CloudHSM

下圖彙總加密程序。備份加密金鑰代表持久性備份金鑰 (PBK) 和暫時性備份金鑰 (EBK)。



AWS CloudHSM 可以將備份還原到僅由同一製造商製造的 AWS擁有的 HSM 上。因為每個備份包含來自原始 HSM 的所有使用者、金鑰和組態，還原的 HSM 會包含與原始 HSM 相同的保護和存取控制。還原的資料會將還原之前已在 HSM 上的所有其他資料覆寫。

備份僅包含加密的資料。在服務將備份存放到 Amazon S3 之前，該服務會使用 AWS Key Management Service (AWS KMS) 再次加密備份。

的身分識別與存取管理 AWS CloudHSM

AWS 使用安全登入資料來識別您並授予您對 AWS 資源的存取權。您可以使用 AWS Identity and Access Management (IAM) 的功能，允許其他使用者、服務和應用程式完全或以有限的方式使用您的 AWS 資源。您可以在不共用您的安全登入資料的情況下執行這項操作。

預設情況下，IAM 使用者沒有可建立、檢視或修改 AWS 資源的許可。若要允許 IAM 使用者存取資源 (例如負載平衡器)，並執行工作，您可以：

1. 建立 IAM 政策以准許 IAM 使用者使用他們所需的特定資源和 API 動作。
2. 將政策連接到 IAM 使用者所屬的 IAM 使用者或群組。

將政策連接到使用者或使用者群組時，政策會允許或拒絕使用者在特定資源上執行特定任務的許可。

例如，您可以使用 IAM 在 AWS 帳戶下建立使用者與群組。IAM 使用者可以是個人、系統或應用程式。然後，您需要使用 IAM 政策，將許可授予使用者和群組在指定資源上執行特定動作。

使用 IAM 政策授予許可

將政策連接到使用者或使用者群組時，政策會允許或拒絕使用者在特定資源上執行特定任務的許可。

IAM 政策為包含一或多個陳述式的 JSON 文件。每個陳述式的結構，如下列範例所示。

```
{
  "Version": "2012-10-17",
  "Statement":[{
    "Effect": "effect",
    "Action": "action",
    "Resource": "resource-arn",
    "Condition": {
      "condition": {
        "key":"value"
      }
    }
  }]
}
```

- Effect：Effect 可以是 `Allow` 或 `Deny`。根據預設，IAM 使用者沒有使用資源和 API 動作的許可，因此所有請求均會遭到拒絕。明確允許覆寫預設值。明確拒絕覆寫任何允許。
- Action：Action 是您授予或拒絕許可的特定 API 動作。如需有關指定動作的詳細資訊，請參閱 適用於的 API 動作 AWS CloudHSM。
- 資源 — 受動作影響的資源。 AWS CloudHSM 不支援資源層級權限。您必須使用 * 萬用字元來指定所有 AWS CloudHSM 資源。
- Condition：您可以選擇性地使用條件來控制政策何時生效。如需詳細資訊，請參閱 條件鍵 AWS CloudHSM。

如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/。

適用於的 API 動作 AWS CloudHSM

在 IAM 政策聲明的「動作」元素中，您可以指定 AWS CloudHSM 提供的任何 API 動作。您必須以小寫字串 `cloudhsm:` 做為動作名稱的字首，如下列範例所示。

```
"Action": "cloudhsm:DescribeClusters"
```

若要在單一陳述式中指定多個動作，請將它們括在方括號中，並以逗號分隔，如下列範例所示。

```
"Action": [
    "cloudhsm:DescribeClusters",
    "cloudhsm:DescribeHsm"
]
```

您也可以使用 * 萬用字元指定多個動作。下列範例會指定開頭為 AWS CloudHSM 的所有 API 動作名稱`List`。

```
"Action": "cloudhsm:List*"
```

若要指定的所有 API 動作 AWS CloudHSM，請使用 * 萬用字元，如下列範例所示。

```
"Action": "cloudhsm:*"
```

如需的 API 動作清單 AWS CloudHSM，請參閱AWS CloudHSM 動作。

條件鍵 AWS CloudHSM

在建立政策時，您可以指定控制政策生效時機的條件。每個條件都包含一或多個索引鍵/值對。有全球條件金鑰和服務特定的條件金鑰。

AWS CloudHSM 沒有服務特定的內容金鑰。

如需有關全域條件金鑰的詳細資訊，請參閱《IAM 使用者指南》中的 AWS 全域條件上下文金鑰。

預先定義的 AWS 受管政策 AWS CloudHSM

AWS 建立的受管政策會針對常用案例授予必要的許可。您可以根據 IAM 使用者對於 AWS CloudHSM 需要的存取權，將這些政策連接到 IAM 使用者：

- AWSCloudHSMFullAccess— 授予使用 AWS CloudHSM 功能所需的完整訪問權限。
- AWSCloudHSMReadOnlyAccess— 授予 AWS CloudHSM 功能的唯讀存取權限。

客戶管理的政策 AWS CloudHSM

我們建議您為 AWS CloudHSM 其建立僅包含執行所需權限的 IAM 管理員群組 AWS CloudHSM。將具有適當許可的政策連接至此群組。視需要將 IAM 使用者新增至群組。您新增的每個使用者都會繼承系統管理員群組的政策。

此外，我們建議您根據使用者需要的許可來建立其他使用者群組。這可確保只有受信任的使用者才能存取重要的 API 動作。例如，您可以建立一個使用者群組，用來授予叢集和 HSM 的唯讀存取權。由於此群組不允許使用者刪除叢集或 HSM，因此不受信任的使用者無法影響生產工作負載的可用性。

隨著新的 AWS CloudHSM 管理功能隨著時間的推移而增加，您可以確保只有受信任的使用者能夠立即獲得存取權。透過在建立時指派有限的許可給政策，您可以稍後手動為其指派新的功能許可。

以下是的範例政策 AWS CloudHSM。如需有關如何建立政策並將其附于 IAM 使用者群組的資訊，請參閱《IAM 使用者指南》中在 JSON 標籤上建立政策。

範例

- 唯讀許可
- 進階使用者許可
- 管理許可

Example 範例：唯讀許可

此政策允許存取 `DescribeClusters` 和 `DescribeBackups` API 動作。它也包含特定 Amazon EC2 API 動作的其他許可。不過，此政策不允許使用者刪除叢集或 HSM。

```
{
   "Version": "2012-10-17",
   "Statement": {
      "Effect": "Allow",
      "Action": [
         "cloudhsm:DescribeClusters",
         "cloudhsm:DescribeBackups",
         "cloudhsm:ListTags"
      ],
```

```
      "Resource": "*"
   }
}
```

Example 範例：進階使用者許可

此原則允許存取 AWS CloudHSM API 動作的子集。其中也包含特定 Amazon EC2 動作的其他許可。不過，此政策不允許使用者刪除叢集或 HSM。您必須包含`iam:CreateServiceLinkedRole`動作，才能 AWS CloudHSM 在您的帳戶中自動建立AWSServiceRoleForCloudHSM服務連結角色。此角色可 AWS CloudHSM 記錄事件。如需詳細資訊，請參閱 服務連結角色 AWS CloudHSM。

> **Note**
>
> 如需查看每個 API 的特定許可，請參閱《服務授權參考》中的適用於 AWS CloudHSM的操作、資源和條件金鑰。

```
{
   "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 範例：管理許可

此原則允許存取所有 AWS CloudHSM API 動作，包括刪除 HSM 和叢集的動作。其中也包含特定 Amazon EC2 動作的其他許可。您必須包含`iam:CreateServiceLinkedRole`動作，才能 AWS CloudHSM 在您的帳戶中自動建立AWSServiceRoleForCloudHSM服務連結角色。此角色可 AWS CloudHSM 記錄事件。如需詳細資訊，請參閱 服務連結角色 AWS CloudHSM。

```
{
   "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":"*"
   }
}
```

服務連結角色 AWS CloudHSM

您先前建立用來客戶管理的政策 AWS CloudHSM包含`iam:CreateServiceLinkedRole`動作的 IAM 政策。 AWS CloudHSM 定義名為AWSServiceRoleForCloudHSM的服務連結角色。角色由預先定義，AWS CloudHSM 並包含代表您呼叫其他 AWS 服務所 AWS CloudHSM 需的權限。此角色可讓您更輕鬆地設定服務，因為您不需要手動新增角色政策和信任政策許可。

角色政策 AWS CloudHSM 允許您建立 Amazon CloudWatch 日誌群組和記錄串流，並代表您寫入日誌事件。您可以在下面和 IAM 主控台中檢視此角色政策。

```
{
    "Version": "2018-06-12",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

AWSServiceRoleForCloudHSM角色的信任原則允 AWS CloudHSM 許擔任該角色。

```
{
  "Version": "2018-06-12",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudhsm.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
```

```
  ]
}
```

建立服務連結角色 (自動)

AWS CloudHSM 如果您在建立 AWS CloudHSM 系統管理員群組時所定義的權限中包含`iam:CreateServiceLinkedRole`動作，則會在您建立叢集時建立AWSServiceRoleForCloudHSM角色。請參閱客戶管理的政策 AWS CloudHSM。

如果您已經有一或多個叢集，而只想新增AWSServiceRoleForCloudHSM角色，則可以使用主控台、建立叢集命令或 CreateClusterAPI 作業來建立叢集。然後使用主控台、刪除叢集命令或 DeleteClusterAPI 作業將其刪除。建立新叢集會建立服務連結角色，並將此角色套用到您帳戶中的所有叢集。或者，您可以手動建立角色。如需詳細資訊，請參閱下一節。

> **Note**
>
> 如果您只是為了新增AWSServiceRoleForCloudHSM角色而建立叢集，則不需要執行建立叢集中開始使用 AWS CloudHSM所述的所有步驟。

建立服務連結角色 (手動)

您可以使用 IAM 主控 AWS CLI台或 API 建立AWSServiceRoleForCloudHSM角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立服務連結角色。

編輯服務連結角色

AWS CloudHSM 不允許您編輯AWSServiceRoleForCloudHSM角色。例如，建立角色後，您無法變更其名稱，因為可能有各種實體依名稱來參考該角色。您也無法變更角色政策。但是，您可以使用 IAM 來編輯角色描述。如需更多資訊，請參閱《IAM 使用者指南》中的編輯服務連結角色。

刪除 服務連結角色

只要已套用服務連結角色的叢集仍然存在，您就無法刪除此服務連結角色。若要刪除角色，您必須先刪除叢集中的每個 HSM，然後刪除叢集。您的帳戶中的每個叢集都必須刪除。然後，您可以使用 IAM 主控 AWS CLI台或 API 刪除角色。如需刪除叢集的詳細資訊，請參閱刪除 AWS CloudHSM 叢集。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

合規

對於處於 FIPS 模式的叢集， AWS CloudHSM 提供符合 PCI-PIN、PCI-3DS 和 SOC2 相容性要求的 FIPS 核准 HSM。 AWS CloudHSM 也可讓客戶選擇非 FIP 模式的叢集。如需各項認證和合規性要求適用的詳細資訊，請參閱AWS CloudHSM 叢集模式和 HSM 類型。

仰賴 FIPS 驗證的 HSM 可協助您符合雲端資料安全性的企業、合約和法規遵循要求。 AWS

FIPS 140-2 合規

美國聯邦資訊處理標準 (FIPS) 第 140-2 號公報中，美國政府的安全標準具體說明密碼模組的安全要求，以保護敏感資訊。HSM1. 中型 HSM 類型由 AWS CloudHSM FIPS 140-2 第 3 級認證 (憑證 #4218)。如需詳細資訊，請參閱硬體的 FIPS 驗證。

PCI DSS 合規

支付卡產業資料安全標準 (PCI DSS) 是一種由 PCI 安全標準協會管理的專屬資訊安全標準。所提供的 HSM AWS CloudHSM 符合 PCI DSS 的規定。

PCI PIN 合規

PCI PIN 針對傳輸、處理和管理個人識別號碼 (PIN) 資料、自動櫃員機和 point-of-sale (POS) 終端機進行交易時所使用的資訊，提供安全要求和評估標準。自 2023 年 1 月起，所提供的 hsm1. 中型 HSM AWS CloudHSM 已符合 PCI 密碼相容。如需詳細資訊，請參閱文章《AWS CloudHSM 現已通過 PCI PIN 認證》https://aws.amazon.com/blogs/security/aws-cloudhsm-is-now-pci-pin-certified/。

PCI-3DS 合規性

PCI 3DS (或三個網域安全、3-D 安全) 為 EMV 3D 安全電子商務支付提供數據的安全性。PCI 3DS 為在線購物提供了另一層安全性。HSM1. 中型 HSM 提供的 HSM 類型符合 PCI-3DS 標準 AWS CloudHSM 。

SOC2

SOC2 是協助服務組織展示其雲端和資料中心安全控制的架構。AWS CloudHSM 已在關鍵領域實作 SOC2 控制，以遵循可信任的服務原則。如需詳細資訊，請參閱《AWS SOC 常見問答集頁面》。

AWS CloudHSM PCI-PIN 碼法規遵循常見問題

PCI PIN 針對傳輸、處理和管理個人識別號碼 (PIN) 資料、自動櫃員機和 point-of-sale (POS) 終端機進行交易時所使用的資訊，提供安全要求和評估標準。

客戶可透過 AWS Artifact (可隨選存取 AWS 合規報告的自助入口網站) 取得 PCI-PIN 合規聲明文件 (AOC) 和責任摘要。如需詳細資訊，請在 AWS 管理主控台登入 AWS Artifact，或在《AWS Artifact 入門指南》中進一步了解。

常見問答集

問：什麼是合規聲明文件和責任摘要？

合規性證明（AOC）由合格的 PIN 評估人（QPA）證明 AWS CloudHSM 符合 PCI-PIN 標準中的適用控制項生產。職責彙總矩陣會說明各自職責的控制項 AWS CloudHSM 及其客戶。

問：如何取得合規 AWS CloudHSM 性證明？

客戶可透過 AWS Artifact (自助入口網站) 取得 PCI-PIN 合規聲明文件 (AOC)。AWS Artifact是一個可隨需存取 AWS 合規報告的自助入口網站。如需詳細資訊，請在 AWS 管理主控台登入 AWS Artifact，或在《AWS Artifact 入門指南》中進一步了解。

問：如何了解我負責哪些 PCI PIN 控制項？

如需詳細資訊，請參閱 AWS AWS CloudHSM PCI PIN 合規 Package 中的「PCI PIN 碼責任摘要」，客戶可透過 AWS Artifact (可隨選存取 AWS 合規報告的自助入口網站) 取得。如需詳細資訊，請在 AWS 管理主控台登入 AWS Artifact，或在《AWS Artifact 入門指南》中進一步了解。

問：身為 AWS CloudHSM 客戶，我是否可以仰賴 PCI-PIN 碼合規性驗證 (AOC)？

客戶必須管理自己的 PCI-PIN 合規性。您需要透過合格的 PIN 評估員 (QPA) 進行正式的 PCI-PIN 證明程序，以驗證您的付款工作負載是否符合所有 PCI-PIN 控制項/需求。但是，對於 AWS 負責的控制，您的 QPA 可以依賴合規 AWS CloudHSM 證明 (AOC) 而無需進一步測試。

問：是否 AWS CloudHSM 負責與金鑰管理生命週期相關的 PCI-PIN 需求？

AWS CloudHSM 負責 HSM 的實體裝置生命週期。客戶負責 PCI-PIN 標準中的金鑰管理生命週期需求。

問：哪些 AWS CloudHSM 控制項符合 PCI-PIN 標準？

AOC 總結了 QPA 評估的 AWS CloudHSM 控制措施。客戶可透過 AWS Artifact 取得 PCI-PIN 責任摘要。AWS Artifact 是一個可隨需存取 AWS 合規報告的自助入口網站。

問：是否 AWS CloudHSM 支持密碼轉換和 DUKPT 等付款功能？

否， AWS CloudHSM 提供一般用途 HSM。隨著時間的推移，我們可能會提供付款功能。雖然此服務並未直接執行付款功能，但 AWS CloudHSM PCI PIN 碼合規證明可讓客戶在其執行服務時，達到自己的 PCI 合規性。 AWS CloudHSM如果您有興趣在工作負載中使用 AWS 付款加密服務，請參閱部落格「使用 AWS 付款加密技術將付款處理移至雲端」。

棄用通知

不時，為了保持符合 FIPS 140，PCI-DSS，PCI-PIN，PCI-3DS 和 SOC2 的要求，可能 AWS CloudHSM 會棄用功能。此頁面會列出目前套用的變更。

FIPS 140 合規性：2024 機制棄用

美國國家標準技術研究院 (NIST) [1] 建議，在 2023 年 12 月 31 日之後不允許對三重 DES (DESede、3DES、DES3) 加密以及 RSA 金鑰包裝和取消包裝 (採用 PKCS #1 v1.5 的填充方式) 的支援。因此，我們的聯邦資訊處理標準 (FIPS) 模式叢集中對這些支援將於 2024 年 1 月 1 日結束。在非 FIP 模式下的叢集仍然 Support 這些功能。

本指南適用於下列密碼編譯操作：

- 三重 DES 金鑰產生
  - CKM_DES3_KEY_GEN 對於 PKCS #11 程式庫
  - DESede JCE 提供商的註冊機
  - genSymKey 與 -t=21 用于 KMU 中
- 使用三重 DES 金鑰進行加密 (注意：允許解密操作)
  - 對於 PKCS #11 程式庫：CKM_DES3_CBC 加密、CKM_DES3_CBC_PAD 加密和 CKM_DES3_ECB 加密
  - 對於 JCE 提供商：DESede/CBC/PKCS5Padding 加密、DESede/CBC/NoPadding 加密、DESede/ECB/Padding 加密和 DESede/ECB/NoPadding 加密
- 使用 PKCS #1 v1.5 填充進行 RSA 金鑰包裝、取消包裝、加密和解密
  - CKM_RSA_PKCS 為 PKCS #11 SDK 進行包裝、取消包裝、加密和解密
  - RSA/ECB/PKCS1Padding 包裝、取消包裝、加密和解密 JCE SDK
  - wrapKey、unWrapKey 與 -m 12 用於 KMU (注意：12 是機制 RSA_PKCS 的值)

[1] 有關此變更的詳細信息，請參閱《過渡使用加密算法和密鑰長度》中的表 1 和表 5。

韌性 AWS CloudHSM

AWS 全球基礎架構是圍繞區 AWS 域和可用區域建立的。 AWS 區域提供多個實體分離和隔離的可用區域，這些區域透過低延遲、高輸送量和高度備援的網路連線。透過可用區域，您可以設計與操作的應用程式和資料庫，在可用區域之間自動容錯移轉而不會發生中斷。可用區域的可用性、容錯能力和擴展能力，均較單一或多個資料中心的傳統基礎設施還高。

如需區域和可用區域的相關 AWS 資訊，請參閱AWS 全域基礎結構。如需支援彈性之 AWS CloudHSM 功能的詳細資訊，請參閱叢集高可用性和負載平衡。

基礎結構安全 AWS CloudHSM

作為受管服務， AWS CloudHSM 受 Amazon Web Services：安 AWS 全流程概觀白皮書中所述的全球網路安全程序保護。

您可以使用 AWS 已發佈的 API 呼叫透 AWS CloudHSM 過網路進行存取。此外，請求必須使用存取金鑰 ID 和與 IAM 主體相關聯的私密存取金鑰來簽署。或者，您可以透過 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

網路隔離

虛擬私有雲端 (VPC) 是 AWS 雲端中您自己邏輯隔離區域中的虛擬網路。您可以在 VPC 的私有子網路中建立叢集。您可以在建立 VPC 時建立私有子網路。如需詳細資訊，請參閱 建立虛擬私有雲端 (VPC)。

建立 HSM 時，請在子網路中 AWS CloudHSM 放置 elastic network interface (ENI)，以便與 HSM 互動。如需詳細資訊，請參閱 叢集架構。

AWS CloudHSM 建立安全群組，允許叢集中 HSM 之間的入站和輸出通訊。您可以使用此安全群組，讓 EC2 執行個體與叢集中的 HSM 通訊。如需詳細資訊，請參閱 設定用戶端 Amazon EC2 執行個體安全群組。

使用者的授權

使用時 AWS CloudHSM，在 HSM 上執行的作業需要經過驗證的 HSM 使用者的認證。如需詳細資訊，請參閱 the section called “了解 HSM 使用者”。

AWS CloudHSM 和 VPC 端點

您可以在 VPC 和 AWS CloudHSM 建立介面 VPC 端點之間建立私人連線。介面端點採用這項技術 AWS PrivateLink，可讓您在沒有網際網路閘道、NAT 裝置、VPN 連線或 AWS Direct Connect 連線的情況下私有存取 AWS CloudHSM API。VPC 中的執行個體不需要公有 IP 位址即可與 AWS CloudHSM API 通訊。您的 VPC 與 AWS CloudHSM 之間的網路流量都會在 Amazon 網路的範圍內。

每個介面端點都是由您子網路中的一或多個彈性網路介面表示。

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的介面虛擬私人雲端端點 (AWS PrivateLink)。

AWS CloudHSM VPC 端點的考量

在為其設定介面 VPC 端點之前 AWS CloudHSM，請務必先查看 Amazon VPC 使用者指南中的界面端點屬性和限制。

- AWS CloudHSM 支援從您的 VPC 呼叫其所有 API 動作。

為 AWS CloudHSM建立介面 VPC 端點

您可以使用 Amazon VPC 主控台或 AWS Command Line Interface ()AWS CLI建立 AWS CloudHSM 服務的 VPC 端點。如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的建立介面端點。

若要為其建立 VPC 端點 AWS CloudHSM，請使用下列服務名稱：

```
com.amazonaws.region.cloudhsmv2
```

例如，在美國西部 (奧勒岡) 區域 (us-west-2) ，服務名稱為：

```
com.amazonaws.us-west-2.cloudhsmv2
```

若要更容易使用 VPC 端點，您可以為 VPC 端點啟用私人 DNS 主機名稱。如果您選取「啟用私人 DNS 名稱」選項，則標準 AWS CloudHSM DNS 主機名稱 (`https://cloudhsmv2.<region>.amazonaws.com`) 會解析為您的 VPC 端點。

此選項可讓您更輕鬆使用 VPC 端點。 AWS SDK 和預設 AWS CLI 使用標準 AWS CloudHSM DNS 主機名稱，因此您不需要在應用程式和命令中指定 VPC 端點 URL。

如需詳細資訊，請參閱《Amazon VPC 使用者指南》中的透過介面端點存取服務。

建立 VPC 端點原則 AWS CloudHSM

您可以將端點政策連接至控制 AWS CloudHSM存取權限的 VPC 端點。此政策會指定下列資訊：

- 可執行動作的主體。
- 可執行的動作。
- 可供執行動作的資源。

如需詳細資訊，請參閱 Amazon VPC 使用者指南中的使用 VPC 端點控制對服務的存取。

範例：用於動作的 VPC 端點原則 AWS CloudHSM

以下是的端點策略範例 AWS CloudHSM。連接至端點時，此策略會授與所有資源上所有主參與者所列 AWS CloudHSM 動作的存取權。的身分識別與存取管理 AWS CloudHSM如需其他 AWS CloudHSM 動作及其對應的 IAM 許可，請參閱。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "cloudhsm:DescribeBackups",
            "cloudhsm:DescribeClusters",
            "cloudhsm:ListTags",
         ],
         "Resource":"*"
      }
   ]
}
```

更新中的管理 AWS CloudHSM

AWS 會管理韌體。韌體由第三方維護，而且必須由 NIST 針對 FIPS 140-2 第 3 級合規進行評估。只能安裝由 FIPS 金鑰 (AWS 沒有存取權) 以密碼編譯方式簽署的韌體。

疑難排 AWS CloudHSM

如果您在使用中遇到問題 AWS CloudHSM，下列主題可協助您解決這些問題。

主題

已知問題

AWS CloudHSM 有以下已知問題。選擇主題以進一步了解。

主題

- 整合第三方應用程式的已知問題

所有 HSM 執行個體的已知問題

下列問題會影響所有使用 AWS CloudHSM 者，不論使用者是否使用 key_mgmt_util 命令列工具、PKCS #11 SDK、JCE SDK 或 OpenSSL SDK。

主題

- 問題：AES 金鑰包裝使用 PKCS #5 填補，而不是使用零填補的金鑰包裝標準合規實作。
- 問題：用戶端常駐程式的組態檔案中至少要有一個有效的 IP 地址，才能成功連接到叢集。
- 問題：可以 AWS CloudHSM 使用用戶端 SDK 3 雜湊和簽署的資料上限為 16 KB
- 問題：無法將匯入的金鑰指定為不可匯出。
- 問題：已移除 wrapKey 和 key_mgmt_util 中的 unWrapKey指令的預設機制
- 問題：如果在您的叢集中有單一 HSM，HSM 容錯移轉無法正常運作。
- 問題：如果您在短期內超過您叢集中 HSM 的金鑰容量，用戶端會進入未處理的錯誤狀態。
- 問題：不支援使用 800 位元組以上的 HMAC 金鑰進行摘要操作。
- 問題：隨用戶端 SDK 3 散發的 client_info 工具會刪除選用輸出引數所指定的路徑內容
- 問題：在容器化環境中使用 --cluster-id 引數執行 SDK 5 設定工具時收到錯誤。
- 問題：您收到錯誤「無法從提供的 pfx 檔案建立憑證/金鑰。錯誤: NotPkcs8 吋

問題：AES 金鑰包裝使用 PKCS #5 填補，而不是使用零填補的金鑰包裝標準合規實作。

此外，不支援無填補和零填補的金鑰包裝。

- 影響：如果您在 AWS CloudHSM中使用此演算法進行包裝和展開，則不會產生任何影響。但是，包裝的金鑰 AWS CloudHSM 無法在預期符合無填補規格的其他 HSM 或軟體中解包。這是因為在標準合規取消包裝期間，系統可能會在金鑰資料後加上 8 個位元組的資料填補。外部包裝的金鑰無法正確解包至 AWS CloudHSM 實體中。
- 因應措施：若要將在 AWS CloudHSM 執行個體上使用含 PKCS #5 填補的「AES 金鑰包裝」包裝的金鑰在外部取消包裝，在嘗試使用該金鑰前請先去除額外的填補。您可以在檔案編輯器裁剪額外的位元組，或是只將金鑰的位元組複製到程式碼中的新緩衝區。
- 解決方案狀態：在 3.1.0 用戶端和軟體版本中， AWS CloudHSM 為 AES 金鑰包裝提供符合標準的選項。如需詳細資訊，請參閱AES 金鑰包裝。

問題：用戶端常駐程式的組態檔案中至少要有一個有效的 IP 地址，才能成功連接到叢集。

- 影響：如果您刪除叢集中的每個 HSM，然後新增另一個 HSM，該 HSM 會得到新 IP 地址，而用戶端協助程式會繼續在原來的 IP 地址搜尋您的 HSM。
- 因應措施：如果您執行間歇性的工作負載，建議您使用CreateHsm函IpAddress數中的引數將 elastic network interface (ENI) 設定為其原始值。請注意，ENI 為可用區域 (AZ) 專用。另一個替代的做法是，刪除 `/opt/cloudhsm/daemon/1/cluster.info` 檔案，然後將用戶端組態重設為新 HSM 的 IP 地址。您也可以使用 `client -a <IP address>` 命令。如需詳細資訊，請參閱安裝和設定 AWS CloudHSM 用戶端 (Linux) 或安裝和設定 AWS CloudHSM 用戶端 (Windows)。

問題：可以 AWS CloudHSM 使用用戶端 SDK 3 雜湊和簽署的資料上限為 16 KB

- 解決狀態：會將小於 16KB 的資料繼續傳送至 HSM 進行雜湊處理。我們已新增可在本機、軟體、大小在 16KB 與 64KB 之間的資料進行雜湊處理的功能。如果資料緩衝區大於 64KB，用戶端 SDK 5 將會明確失敗。您必須將用戶端和 SDK 更新為大於 5.0.0 或更高版本，才能從此修正中受益。

問題：無法將匯入的金鑰指定為不可匯出。

- 解決狀態： 已修正此問題。您的部分無須採取任何動作，即可受益於修正。

問題：已移除 wrapKey 和 key_mgmt_util 中的 unWrapKey指令的預設機制

- 解析度：使用 wrapKey 或指 unWrapKey 令時，您必須使用-m選項來指定機制。如需詳細資訊，請參閱 wrapKey 或unWrapKey文章中的範例。

問題：如果在您的叢集中有單一 HSM，HSM 容錯移轉無法正常運作。

- 影響：如果您叢集中的單一 HSM 執行個體失去連線，即使之後還原 HSM 執行個體，用戶端也將不會與此執行個體連線。
- 因應措施：我們建議至少在任一生產叢集中執行兩個 HSM 執行個體。如果您使用此組態，您將不會受到此問題影響。如果是單一 HSM 叢集，請退回用戶端協助程式，以還原連線。
- 解決狀態：此狀態已在 AWS CloudHSM 用戶端 1.1.2 版本中解決。您必須升級到此用戶端，以受益於此修正。

問題：如果您在短期內超過您叢集中 HSM 的金鑰容量，用戶端會進入未處理的錯誤狀態。

- 影響：用戶端遇到未處理的錯誤狀態時，用戶端會凍結且必須重新啟動。
- 因應措施：請測試您的傳輸量，以確保您並非以用戶端無法處理的速率建立工作階段金鑰。您可以新增 HSM 至叢集，或減慢工作階段金鑰的建立速度，藉以降低速率。
- 解決狀態：此狀態已在 AWS CloudHSM 用戶端 1.1.2 版本中解決。您必須升級到此用戶端，以受益於此修正。

問題：不支援使用 800 位元組以上的 HMAC 金鑰進行摘要操作。

- 影響：800 位元組以上的 HMAC 金鑰可以在 HSM 上產生或匯入 HSM。然而，如果您透過 JCE 或 key_mgmt_util 在摘要操作中使用較大的金鑰，此次操作將會失敗。請注意，如果您使用 PKCS11，HMAC 金鑰的大小限制為 64 位元組。
- 因應措施：如果您將 HMAC 金鑰用於 HSM 上的摘要操作，請大小小於 800 位元組。
- 解決狀態：此時並無。

問題：隨用戶端 SDK 3 散發的 client_info 工具會刪除選用輸出引數所指定的路徑內容

- 影響：指定輸出路徑下的所有現有檔案和子目錄可能永久遺失。
- 因應措施：使用 `client_info` 工具時，請勿使用選用引數 `-output` *`path`*。
- 解決狀態：此狀態已在用戶端 3.3.2 版本中解決。您必須升級到此用戶端，以受益於此修正。

問題：在容器化環境中使用 **`--cluster-id`** 引數執行 SDK 5 設定工具時收到錯誤。

使用 --叢集-id 引數搭配使用 [設定工具] 時，您會收到下列錯誤：

```
No credentials in the property bag
```

執行個體中繼資料服務版本 2 (IMDSv2) 的更新造成此錯誤。如需詳細資訊，請參閱 IMDSv2 文件。

- 影響：此問題會影響在容器化環境中在 SDK 5.5.0 及更新版本上執行設定工具的使用者，以及使用 EC2 執行個體中繼資料提供憑證。
- 因應措施：將 PUT 回應躍點限制設定為至少兩個。如需如何執行此動作的指引，請參閱設定執行個體中繼資料選項。

問題：您收到錯誤「無法從提供的 pfx 檔案建立憑證/金鑰。錯誤: NotPkcs8 吋

- 影響：如果使用者的私密金鑰不是 PKCS8 格式，則使用憑證和私密金鑰重新設定 SSL 的 SDK 5.11.0 使用者將會失敗。
- 因應措施：您可以使用 openssl 命令將自訂 SSL 私密金鑰轉換為 PKCS8 格式：`openssl pkcs8 -topk8 -inform PEM -outform PEM -in ssl_private_key -out ssl_private_key_pkcs8`
- 解決方案狀態：用戶端 SDK 5.12.0 發行版本中已解決此問題。您必須升級至此用戶端版本或更新版本，才能從此修正中獲益。

hsm2m 執行個體的已知問題

下列問題會影響所有 hsm2m.medium 執行個體。

主題

- 問題：由於 PBKDF2 反覆運算增加，登入延遲增加
- 問題：用戶端 SDK 5.12.0 及更早版本使用嘗試設定金鑰的受信任屬性的 CO 將失敗

問題：由於 PBKDF2 反覆運算增加，登入延遲增加

- 影響：為了提高安全性，hsm2m.medium 會在登入要求期間執行 60,000 次以密碼為基礎的金鑰衍生函數 2 (PBKDF2) 反覆運算，而 hsm1.medium 則為 1,000 次。這種增加可能會導致每個登入要求的延遲時間增加最多 2 秒 (2 秒)。

  AWS CloudHSM 用戶端 SDK 的預設逾時時間為 20 秒。登入要求可能會逾時並導致錯誤。
- 因應措施：如果可能，請在相同應用程式中序列化登入要求，以避免在登入期間延長延遲。
- 解決方案狀態：未來版本的 Client SDK 會增加登入要求的預設逾時，以解決這種延遲的延遲。

問題：用戶端 SDK 5.12.0 及更早版本使用嘗試設定金鑰的受信任屬性的 CO 將失敗

- 影響：任何嘗試設定金鑰信任屬性的 CO 使用者都會收到錯誤訊息，指出此情況`User type should be CO or CU`。
- 解決方案：未來版本的用戶端 SDK 將解決此問題。更新將在我們的用戶指南中註釋。文件歷史紀錄

PKCS #11 程序庫的已知問題

主題

問題：PKCS #11 程式庫 3.0.0 版中的 AES 金鑰包裝不會在使用前驗證 IV。

如果您指定的 IV 長度小於 8 個位元組，便會在使用之前利用無法預測的位元組進行填補。

> **Note**
>
> 這只會影響包含 `CKM_AES_KEY_WRAP` 機制的 `C_WrapKey`。

- 影響：如果您在 PKCS #11 程序庫 3.0.0 版中提供的 IV 長度小於 8 個位元組。您便可能無法取消包裝金鑰。
- 因應措施：
  - 我們強烈建議您升級到 PKCS #11 程式庫 3.0.1 或更新版本，這些新版本會在 AES 金鑰包裝期間強制要求 IV 長度。修改您的包裝程式碼以傳遞 NULL IV，或是指定預設 IV `0xA6A6A6A6A6A6A6A6`。如需詳細資訊，請參閱 AES 金鑰包裝的非合規長度自訂 IV。

  - 如果您已搭配長度小於 8 個位元組的 IV 使用 PKCS #11 程式庫 3.0.0 包裝任何金鑰，請聯絡我們以取得支援。
- 解決狀態：此問題已在 PKCS #11 程式庫 3.0.1 中解決。如要使用 AES 金鑰包裝來包裝金鑰，請指定 NULL 或長度為 8 個位元組的 IV。

問題：PKCS #11 SDK 2.0.4 和先前的版本一律會針對 AES 金鑰包裝和取消包裝使用 **0xA6A6A6A6A6A6A6A6** 預設 IV。

使用者提供的 IV 會在無提示的情況下遭到忽略。

> **Note**
>
> 這只會影響包含 CKM_AES_KEY_WRAP 機制的 C_WrapKey。

- Impact: (影響：)
  - 如果您使用 PKCS #11 程式庫 2.0.4 或先前版本及使用者提供的 IV，您的金鑰會使用預設 IV 0xA6A6A6A6A6A6A6A6 進行包裝。
  - 如果您使用 PKCS #11 程式庫 3.0.0 或更新版本及使用者提供的 IV，您的金鑰則會使用使用者提供的 IV 進行包裝。
- Workarounds: (因應措施：)
  - 如要取消包裝使用 PKCS #11 程式庫 2.0.4 或先前版本包裝的金鑰，請使用預設 IV 0xA6A6A6A6A6A6A6A6。
  - 如要取消包裝使用 PKCS #11 程式庫 3.0.0 及更新版本包裝的金鑰，請使用使用者提供的 IV。
- Resolution status: (解決狀態：) 我們強烈建議您修改您的包裝和取消包裝程式碼，以傳遞 NULL IV，或是指定預設 IV 0xA6A6A6A6A6A6A6A6。

問題：既未支援也未處理 **CKA_DERIVE** 屬性。

- 解決狀態：我們已實作修正接受金鑰 CKA_DERIVE (如果設定為 FALSE)。在我們開始將金鑰衍生函數支援新增至 AWS CloudHSM前，不支援將 CKA_DERIVE 設定為 TRUE。您必須將用戶端和程式庫更新至 1.1.1 版或更高版本，才能受益於修正。

問題：既未支援也未處理 **CKA_SENSITIVE** 屬性。

- 解決狀態：我們已實作可以接受和正確遵守 CKA_SENSITIVE 屬性的修正。您必須將用戶端和程式庫更新至 1.1.1 版或更高版本，才能受益於修正。

問題：不支援分段雜湊和簽署。

- 影響：不會實作 C_DigestUpdate 和 C_DigestFinal。C_SignFinal 也不會實作，非 NULL 緩衝區會失敗並發生 CKR_ARGUMENTS_BAD 錯誤。
- 因應措施：在應用程式中雜湊資料，並 AWS CloudHSM 僅用於簽署雜湊。
- 解決狀態：我們正在進行修復，讓用戶端和程式庫能夠正確實作分段雜湊。更新會公告於 AWS CloudHSM 論壇和版本歷史頁面中。

問題：**C_GenerateKeyPair** 不是以符合標準的方式處理私有範本中的 **CKA_MODULUS_BITS** 或 **CKA_PUBLIC_EXPONENT**。

- 影響：當私有範本包含 CKA_MODULUS_BITS 或 CKA_PUBLIC_EXPONENT，C_GenerateKeyPair 應該傳回 CKA_TEMPLATE_INCONSISTENT。但它反而產生私有金鑰，將所有使用欄位設為 FALSE。無法使用金鑰。
- 因應措施：建議您的應用程式除了錯誤碼，也檢查使用欄位值。
- 解決狀態：我們正在實作修正，讓系統在使用錯誤的私有金鑰範本時傳回適當的錯誤訊息。會在版本歷史頁面中公告更新的 PKCS #11 程式庫。

問題：使用 **CKM_AES_GCM** 機制時，**C_Encrypt** 和 **C_Decrypt** API 操作的緩衝區不能超過 16 KB。

AWS CloudHSM 不支援多部分 AES-GCM 加密。

- 影響：您不能使用 CKM_AES_GCM 機制來加密大於 16 KB 的資料。
- 因應措施：您可以使用替代機制，例如CKM_AES_CBC、CKM_AES_CBC_PAD，或者您可以將資料分成幾個部分，然後使用個AES_GCM別方式加密每個部分。如果您正在使用AES_GCM，則必須管理數據的劃分和後續加密。 AWS CloudHSM 不會為您執行多部分 AES-GCM 加密。請注意，FIPS 要求在 HSM 上產生初始化向量 (IV)。AES-GCM因此，AES-GCM 加密資料之每個片段的 IV 都不同。
- 解決狀態：我們正在進行修復，讓程式庫在資料緩衝區過大時明確失敗。我們會針對 C_EncryptUpdate 和 C_DecryptUpdate API 操作傳回 CKR_MECHANISM_INVALID。我們正在

評估支援較大緩衝區、而不需要仰賴分段加密的替代做法。更新將在 AWS CloudHSM 論壇和版本歷史頁面上公佈。

問題：在 HSM 中橢圓曲線 Diffie-Hellman (ECDH) 金鑰衍生只會部分執行。

您的 EC 私有金鑰仍一律保留在 HSM 中，但金鑰衍生程序會分成多個步驟執行。因此，在用戶端會有每個步驟產生的中繼結果。

- 影響：在用戶端 SDK 3 中，使用該`CKM_ECDH1_DERIVE`機制衍生的金鑰會先在用戶端上使用，然後再匯入 HSM。然後將金鑰控制代碼傳回給到您的應用程式。
- 因應措施：如果您在 AWS CloudHSM中實作 SSL/TLS 卸載，此限制可能不是問題。如果您的應用程式要求金鑰需隨時符合 FIPS，請考慮使用其他不倚賴 ECDH 金鑰衍生的通訊協定。
- 解決狀態：我們正在開發可完全在 HSM 內執行 ECDH 金鑰衍生的選項。當更新的實作可供使用時，即會在版本歷史記錄頁面中公告。

問題：在諸如 CentOs6 和 RHEL 6 之類的 EL6 平台上驗證節 256k1 簽名失敗

這是因為 CloudHSM PKCS#11 程式庫會使用 OpenSSL 來驗證 EC 曲線資料，藉此避免在驗證操作初始化期間進行網路呼叫。由於 EL6 平台上的預設 OpenSSL 套件並不支援 Secp256k1，初始化作業會失敗。

- 影響：無法在 EL6 平台上進行 Secp256k1 簽章驗證，驗證呼叫會失敗並出現 `CKR_HOST_MEMORY` 錯誤。
- 因應措施：若 PKCS#11 應用程式需要驗證 secp256k1 簽章，則建議使用 Amazon Linux 1 或任何 EL7 平台。或者，您可將 OpenSSL 套件升級至支援 secp256k1 曲線的版本。
- 解決狀態：我們正在實作修正，以便在無法驗證本機曲線的情況下切換回 HSM。版本歷史記錄頁面中會公告更新的 PKCS#11 程式庫。

問題：不正確的函數調用序列會給出未定義的結果而不是失敗

- 影響：如果您調用不正確的函數序列，即使個別函數調用傳回成功，最終結果也不正確。例如，解密的資料可能與原始純文本不匹配，否則簽名可能無法驗證。此問題會影響單一和多部分作業。

  函數序列不正確的範例：

  - `C_EncryptInit/C_EncryptUpdate` 後跟 `C_Encrypt`
  - `C_DecryptInit/C_DecryptUpdate` 後跟 `C_Decrypt`

  - C_SignInit/C_SignUpdate 後跟 C_Sign
  - C_VerifyInit/C_VerifyUpdate 後跟 C_Verify
  - C_FindObjectsInit 後跟 C_FindObjectsInit
- 因應措施：您的應用程式應在符合 PKCS #11 規格的情況下，針對單一和多部分作業使用正確的函數調用順序。在此情況下，您的應用程式不應該依賴 CloudHSM PKCS #11 程式庫來傳回錯誤。

問題：SDK 5 不支援唯讀工作階段

- 問題：C_OpenSession 的 SDK 5 不支援使用開啟唯讀工作階段。
- 影響：如果您嘗試在未提供的 CKF_RW_SESSION 情況下調用 C_OpenSession，調用將失敗並顯示錯誤 CKR_FUNCTION_FAILED。
- 因應措施：開啟工作階段時，您必須將 CKF_SERIAL_SESSION | CKF_RW_SESSION 旗標傳遞至 C_OpenSession 函數調用。

問題：**cryptoki.h** 標頭檔案僅限 Windows

- 問題：在 Linux 上使用 AWS CloudHSM 用戶端 SDK 5 版本 5.0.0 到 5.4.0 時，標頭檔案/opt/cloudhsm/include/pkcs11/cryptoki.h僅與視窗作業系統相容。
- 影響：嘗試在 Linux 作業系統的應用程式中包含此標頭檔案時，可能會遇到問題。
- 解析狀態：升級至 AWS CloudHSM 用戶端 SDK 5 5.4.1 或更新版本，其中包含此標頭檔案的 Linux 相容版本。

JCE 開發套件的已知問題

主題

- 問題：使用 GetKey 作業的用戶端 SDK 5 記憶體洩漏

問題：使用非對稱金鑰對時，即使未明確建立或匯入金鑰，仍會看到已佔用的金鑰容量

- 影響：此問題可能導致 HSM 意外用盡金鑰空間，並在您的應用程式使用標準 JCE 金鑰物件 (而非 CaviumKey 物件) 進行加密操作時發生。當您使用標準 JCE 金鑰物件時，CaviumProvider 會隱含地將該金鑰匯入 HSM 做為工作階段金鑰，且不會刪除此金鑰，直到應用程式結束為止。因此，金鑰會在應用程式執行時不斷累積，可能導致您的 HSM 用盡可用的金鑰空間，進而凍結您的應用程式。
- 因應措施：使用 CaviumSignature 類別、 CaviumCipher 類別、 CaviumMac 類別或 CaviumKeyAgreement 類別時，您應提供金鑰做為 CaviumKey，而非標準 JCE 金鑰物件。

  您可以使用 ImportKey 類手動將普通金鑰轉換為 CaviumKey，然後在操作完成後手動刪除該金鑰。
- 解決狀態：我們正在更新 CaviumProvider 以正確管理隱含匯入。修正可供使用時即會在版本歷史頁面中公告。

問題：JCE KeyStore 是唯讀的

- 影響：無法將 HSM 不支援的物件類型存放在 JCE 金鑰存放區中。特別是您無法在金鑰存放區中存放憑證。這會防止與 jarsigner 之類之工具 (其會預期在金鑰存放區中尋找憑證) 之間的互通性。
- 因應措施：您可以重新處理程式碼，從本機檔案或從 S3 儲存貯體位置 (而不是從金鑰存放區) 載入憑證。
- 解決狀態：我們會新增在金鑰存放區中儲存憑證的支援。此功能可供使用時，即會在版本歷史頁面中公告。

問題：AES-GCM 加密的緩衝區不能超過 16,000 位元組。

此外，也不支援分段的 AES-GCM 加密。

- 影響：您無法使用 AES-GCM 來加密大於 16,000 位元組的資料。
- 因應措施：您可以使用替代機制 (例如 AES-CBC)，或者可以將資料切分成數段，並個別加密每一個部分。如果切分資料，您必須管理切分後的加密文字及其解密。由於 FIPS 規定 AES-GCM 的初始化向量 (IV) 要在 HSM 上產生，AES-GCM 加密之每個資料片段的 IV 都不同。
- 解決狀態：我們正在進行修復，讓程式庫在資料緩衝區過大時明確失敗。我們正在評估支援較大緩衝區、而不需要仰賴分段加密的替代做法。更新會公告於 AWS CloudHSM 論壇和版本歷史頁面中。

問題：在 HSM 中橢圓曲線 Diffie-Hellman (ECDH) 金鑰衍生只會部分執行。

您的 EC 私有金鑰仍一律保留在 HSM 中，但金鑰衍生程序會分成多個步驟執行。因此，在用戶端會有每個步驟產生的中繼結果。您可在 Java 程式碼範例中查看 ECDH 金鑰衍生範例。

- 影響：用戶端 SDK 3 將 ECDH 功能新增至 JCE。當您使用`KeyAgreement`類別來衍生時 SecretKey，它會先在用戶端上使用，然後再匯入至 HSM。然後將金鑰控制代碼傳回給到您的應用程式。
- 因應措施：如果您要在中實作 SSL/TLS 卸載 AWS CloudHSM，則此限制可能不是問題。如果您的應用程式要求金鑰需隨時符合 FIPS，請考慮使用其他不倚賴 ECDH 金鑰衍生的通訊協定。
- 解決狀態：我們正在開發可完全在 HSM 內執行 ECDH 金鑰衍生的選項。如果可用，我們將在版本歷史記錄頁面中公告更新的實作。

問題： KeyGenerator 並 KeyAttribute 錯誤地將密鑰大小參數解釋為字節數而不是位

當使用KeyGenerator 類的`init`函數或AWS CloudHSM KeyAttribute 枚舉的`SIZE`屬性生成密鑰時，API 錯誤地期望參數是密鑰字節的數量，而應該是密鑰位的數量。

- 影響：用戶端 SDK 版本 5.4.0 至 5.4.2 錯誤地預期金鑰大小會以位元組形式提供給指定的 API。
- 因應措施：如果使用用戶端 SDK 版本 5.4.0 到 5.4.2，請先將金鑰大小從位元轉換為位元組，然後再使用 KeyGenerator 類別或 KeyAttribute 列舉產生使用 AWS CloudHSM JCE 提供者產生金鑰。
- 解析狀態：將用戶端 SDK 版本升級至 5.5.0 或更新版本，其中包含修正程式，可在使用 KeyGenerator 類別或 KeyAttribute 列舉產生金鑰時正確預期金鑰大小 (位元)。

問題：用戶端 SDK 5 會發出警告「發生了非法的反射式存取作業」

當您將用戶端 SDK 5 與 Java 11 搭配使用時，CloudHSM 會擲回下列 Java 警告：

```
```
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
```
```

```
```

這些警告沒有任何影響。我們知道這個問題，並正在努力解決這個問題。不需要解決方案或因應措施。

問題：JCE 工作階段集區已用盡

影響：看到下列訊息後，您可能無法在 JCE 中執行作業：

```
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
```

因應措施：

- 如果您遇到影響，請重新啟動 JCE 應用程式。
- 執行作業時，您可能需要先完成 JCE 作業，才能遺失對作業的參照。

> **Note**
>
> 視作業而定，可能需要完成方法。

作業	完成方法
加密	在加密或解密模式下的 `doFinal()` 在包裝模式下的 `wrap()` 在取消包裝模式下的 `unwrap()`
KeyAgreement	`generateSecret()` 或 `generateSecret(String)`
KeyPairGenerator	`generateKeyPair()` 、`genKeyPair()` 或 `reset()`
KeyStore	無需任何方法
MAC	`doFinal()` 或 `reset()`

作業	完成方法
MessageDigest	`digest()` 或 `reset()`
SecretKeyFactory	無需任何方法
SecureRandom	無需任何方法
簽章	在簽署模式下的 `sign()` 在驗證模式中下的 `verify()`

解決狀態：我們已在用戶端 SDK 5.9.0 及更新版本中解決了此問題。若要修正此問題，請將您的用戶端 SDK 升級至下列其中一個版本。

問題：使用 GetKey 作業的用戶端 SDK 5 記憶體洩漏

- 影響：用戶端 SDK 5.10.0 版及更早版本中，API `getKey` 作業在 JCE 中發生記憶體洩漏。如果您在應用程式中多次使用 `getKey` API，則會增加記憶體成長，進而增加應用程式中的記憶體佔用量。隨著時間的推移，這可能會導致節流錯誤或需要重新啟動應用程式。
- 因應措施：我們建議升級至用戶端 SDK 5.11.0。如果無法執行此操作，建議您不要在應用程式中多次呼叫 `getKey` API。而是盡可能重複使用先前`getKey`操作中先前返回的密鑰。
- 解決狀態：將用戶端 SDK 版本升級至 5.11.0 或更新版本，其中包含此問題的修正程式。

OpenSSL 動態引擎的已知問題

這些是 OpenSSL 動態引擎的已知問題

主題

- 問題： AWS CloudHSM OpenSSL 動態引擎與開 OpenSSL v3.x 的 FIPS 提供者不相容

問題:你無法在 RHEL 6 和 Cent AWS CloudHSM OpenSSL s6 上安裝動態引擎

- 影響：OpenSSL 動態引擎僅支援 OpenSSL 1.0.2[f+]。在預設情況下，RHEL 6 和 CentOS 6 會隨附 OpenSSL 1.0.1。
- 因應措施：將 RHEL 6 和 CentOS 6 上的 OpenSSL 程式庫升級至 1.0.2[f+] 版。

問題：預設僅支援將 RSA 卸載到 HSM。

- 影響：為了發揮最大效能，並未將程式庫設定為卸載額外功能 (例如產生亂數或 EC-DH 操作)。
- 因應措施：如果您需要卸載額外操作，請透過支援案例聯絡我們。
- 解決狀態：我們正在新增對程式庫的支援，以透過組態檔案來設定卸載選項。更新可供使用時即會在版本歷史頁面中公告。

問題：在 HSM 上不支援使用金鑰進行內含 OAEP 填補的 RSA 加密和解密。

- 影響：任何使用 OAEP 填補對 RSA 加密和解密的呼叫都會失敗，並顯示錯誤。 divide-by-zero 這是因為 OpenSSL 動態引擎是在本機使用仿造的 PEM 檔案呼叫操作，而不是將操作卸載到 HSM 所造成。
- 因應措施：您可以使用 PKCS #11 程式庫 或 JCE 提供者 執行此程序。
- 解決狀態：我們正在新增對程式庫的支援，以讓您正確卸載此操作。更新可供使用時即會在版本歷史頁面中公告。

問題：只會將 RSA 和 ECC 金鑰的私有金鑰產生卸載到 HSM。

對於任何其他金鑰類型，OpenSSL AWS CloudHSM 引擎不會用於呼叫處理。而是改用本機 OpenSSL 引擎。這會在軟體中以本機的方式產生金鑰。

- 影響：容錯移轉並無提示，因此沒有任何跡象可告知您是否尚未收到在 HSM 上安全產生的金鑰。如果金鑰是由 OpenSSL 在本機的軟體中產生，您會看到包含 "`.....++++++`" 字串的輸出追蹤。將此操作卸載到 HSM 時，就不會有此追蹤。因為金鑰不是在 HSM 中產生或存放，將無法供日後使用。
- 因應措施：只針對 OpenSSL 引擎支援的金鑰類型使用 OpenSSL 引擎。對於所有其他金鑰類型，請在應用程式中使用 PKCS #11 或 JCE，或在 CLI `key_mgmt_util` 中使用。

問題：你無法在 RHEL 8、CentOS 8 或 Ubuntu 18.04 LTS 上安裝用戶端 SDK 3 的 OpenSSL 動態引擎

- 影響：在預設情況下，RHEL 8、CentOS 8 和 Ubuntu 18.04 LTS 所提供的 OpenSSL 版本不相容於用戶端 SDK 3 的 OpenSSL 動態引擎。
- 因應措施：使用提供 OpenSSL 動態引擎支援的 Linux 平台。如需關於支援的平台的詳細資訊，請參閱支援的平台。
- 解決方案狀態： AWS CloudHSM 支援這些平台搭配 OpenSSL 動態引擎 (適用於用戶端 SDK 5)。如需詳細資訊，請參閱支援的平台和 OpenSSL 動態引擎。

問題：SHA-1 在 RHEL 9 上簽署並驗證已淘汰 (9.2 以上)

- 影響：針對密碼編譯目的使用 SHA-1 訊息摘要已在 RHEL 9 (9.2+) 中淘汰。因此，使用 OpenSSL 動態引擎使用 SHA-1 簽署並驗證作業將會失敗。
- 因應措施：如果您的案例需要使用 SHA-1 來簽署/驗證現有或協力廠商的加密簽章，請參閱增強 RHEL 安全性：瞭解 RHEL 9 (9.2+) 和 RHEL 9 (9.2+) 版本說明的 SHA-1 淘汰，以取得進一步的詳細資訊。

問題： AWS CloudHSM OpenSSL 動態引擎與開 OpenSSL v3.x 的 FIPS 提供者不相容

- 影響：如果您在啟用 OpenSSL 3.x 版的 FIPS 提供者時嘗試使用 AWS CloudHSM OpenSSL 動態引擎，您將會收到錯誤訊息。
- 因應措施：若要將 AWS CloudHSM OpenSSL 動態引擎與 OpenSSL 3.x 版搭配使用，請確定已設定「預設」提供者。在 OpenSSL 網站上閱讀更多有關預設提供者的資訊。

執行 Amazon Linux 2 的 Amazon EC2 執行個體的已知問題

問題：Amazon Linux 2 版本 2018.07 使用目前與開發**ncurses**套件不相容的更新套件 (版本 6) AWS CloudHSM

您會看到在執行 AWS CloudHSM雲端時傳回下列錯誤，或是執行下列錯誤：

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util: error while loading shared libraries:
 libncurses.so.5: cannot open shared object file: No such file or directory
```

- 影響：在 Amazon Linux 2 版本 2018.07 上執行的執行個體將無法使用所有 AWS CloudHSM 公用程式。
- 因應措施：在 Amazon Linux 2 EC2 執行個體上發出以下命令，安裝受支援的 `ncurses` 套件 (版本 5)：

```
sudo yum update && yum install ncurses-compat-libs
```

- 解決狀態：此狀態已在 AWS CloudHSM 用戶端 1.1.2 版本中解決。您必須升級到此用戶端，以受益於此修正。

整合第三方應用程式的已知問題

問題：用戶端 SDK 3 不支援產生主金鑰時 Oracle 設定 PKCS #11 屬性 **`CKA_MODIFIABLE`**。

此限制是在 PKCS #11 程式庫中定義的。如需詳細資訊，請參閱支援的 PKCS #11 屬性上的註釋 1。

- 影響：Oracle 主金鑰建立失敗。
- 因應措施：在建立新的主金鑰時，將特殊環境變數 `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE` 設定為 TRUE。只有在產生主金鑰時才需要此環境變數，您不需要將此環境變數用於其他任何用途。例如，您會將此變數用於您建立的第一個主金鑰，接著如果您想要輪換主金鑰版本，才會再次使用此環境變數。如需詳細資訊，請參閱產生 Oracle TDE 主加密金鑰。
- 解決狀態：我們正在改進 HSM 韌體，以完全支援 CKA_MODIFIABLE 屬性。更新將在 AWS CloudHSM 論壇和版本歷史頁面上公佈

用戶端 SDK 3 金鑰同步失敗

在 Client SDK 3 中，如果用戶端同步處理失敗， AWS CloudHSM 會盡最大努力回應以清除可能已建立 (而且現在不需要) 的任何不需要的金鑰。此程序包括立即移除不需要的金鑰材料，或標記不需要的材料以供日後移除。在這兩種情況下，解決方案都不需要您採取任何行動。在極少數情況下 AWS CloudHSM 無法移除且無法標記不需要的關鍵材料，您必須刪除關鍵材料。

問題：您嘗試產生符記金鑰、匯入或取消包裝作業，並看到 tombstone 失敗的錯誤。

```
2018-12-24T18:28:54Z liquidSecurity ERR: print_node_ts_status:
[create_object_min_nodes]Key: 264617 failed to tombstone on node:1
```

原因：無 AWS CloudHSM 法刪除並標記不需要的密鑰材料。

解決方案：叢集中的 HSM 包含未標示為不需要的不需要金鑰材料。您必須手動移除金鑰材料。若要手動刪除不想要的金鑰材料，請使用 key_mgmt_util (KMU) 或 PKCS #11 程式庫或 JCE 提供者中的 API。如需詳細資訊，請參閱 deleteKey 或 用戶端 SDK。

若要讓 Token 金鑰更持久，在用戶端同步處理設定中指定的 HSM 數目下限上無法成功的金鑰建立作業 AWS CloudHSM 失敗。如需詳細資訊，請參閱 AWS CloudHSM 中的金鑰同步。

用戶端 SDK 3：使用 pkpspeed 工具驗證 HSM 效能

本主題說明如何使用用戶端 SDK 3 驗證 HSM 效能。

若要驗證 AWS CloudHSM 叢集中 HSM 的效能，您可以使用用戶端 SDK 3 隨附的 pkpspeed (Linux) 或 pkpspeed_阻止工具。pkpspeed 工具會在理想的條件下執行，並直接調用 HSM 來執行作業，而不需要透過 PKCS11 等的開發套件。建議您單獨對應用程式進行負載測試，以判斷擴展需求。我們不建議運行以下測試：隨機（I），ModExp （R）和 EC 點 mul（Y）。

如需有關在 Linux EC2 執行個體上安裝此用戶端的詳細資訊，請參閱安裝和設定 AWS CloudHSM 用戶端 (Linux)。如需有關在 Windows 執行個體上安裝此用戶端的詳細資訊，請參閱安裝和設定 AWS CloudHSM 用戶端。

安裝並設定 AWS CloudHSM 用戶端之後，請執行下列命令來啟動它。

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

- 用於 Windows 用戶端 1.1.2+：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 用於 Windows 用戶端 1.1.1 和更早版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

如果您已經安裝用戶端軟體，您可能需要下載並安裝最新版以取得 pkpspeed。您可以在 Linux 的 `/opt/cloudhsm/bin/pkpspeed` 或 Windows 的 `C:\Program Files\Amazon\CloudHSM\` 中找到 pkpspeed 工具。

若要使用 pkpspeed，請執行 pkpspeed 命令或 pkpspeed_blocking.exe，並指定 HSM 上加密使用者 (CU) 的使用者名稱和密碼。然後設定要使用的選項，同時考慮以下建議。

測試建議

- 若要測試 RSA 簽署的效能和驗證操作，請選擇 RSA_CRT 加密 (在 Linux 中) 或選項 B (在 Windows 中)。不要選擇 RSA (Windows 中的選項 A)。加密的效果相同，但 RSA_CRT 的效能最佳。

- 從少量的執行緒開始。若要測試 AES 效能，一個執行緒通常足以顯示最大的效能。若要測試 RSA 效能 (RSA_CRT)，三個或四個執行緒通常足夠。

pkpspeed 工具的可設定選項

- FIPS 模式：始終處 AWS CloudHSM 於 FIPS 模式（有關詳細信息，請參閱AWS CloudHSM 常見問題解答）。這可以透過使用 AWS CloudHSM 者指南中所述的 CLI 工具進行驗證，並執行指示 FIPS 模式狀態的指 getHSMInfo 令。
- 測試類型 (封鎖與非封鎖)：這指定了如何以線程方式執行作業。您很可能會使用非封鎖獲得更好的數目。這是因為他們利用執行緒和並行。
- 執行緒數目：執行測試的執行緒數目。
- 執行測試的時間 (秒) (最大值 = 600)：pkpspeed 會產生以「作業/秒」為單位測量的結果，並針對執行測試的每秒報告此值。例如，如果測試執行 5 秒鐘，輸出可能會像下列範例值：
  - `OPERATIONS/second 821/1`
  - `OPERATIONS/second 833/1`
  - `OPERATIONS/second 845/1`
  - `OPERATIONS/second 835/1`
  - `OPERATIONS/second 837/1`

可以使用 pkpspeed 工具執行的測試

- AES GCM：測試 AES GCM 模式加密。
- 基本 3DES CBC：測試 3DES CBC 模式加密。請參閱下列備註 1 查看即將進行的變更。
- 基本 AES：測試 AES CBC/ECB 加密。
- 摘要：測試雜湊摘要。
- ECDSA 簽署：測試 ECDSA 簽署。
- ECDSA 驗證：測試 ECDSA 驗證。
- FIPS 隨機：測試 FIPS 相容隨機數的產生 (注意：這只能在封鎖模式中使用)。
- HMAC：測試 HMAC。
- 隨機：此測試不相關，因為我們使用的是 FIPS 140-2 HSM。
- RSA 非 CRT 與 RSA_CRT：測試 RSA 簽署並驗證操作。
- RSA OAEP 加密：測試 RSA OAEP 加密。

- RSA OAEP 解密：測試 RSA OAEP 解密。
- RSA 私有解密非 CRT：測試 RSA 私有金鑰加密 (非優化)。
- RSA 私有金鑰解密 CRT：測試 RSA 私有金鑰加密 (優化)。
- RSA PSS 簽署：測試 RSA PSS 簽署。
- RSA PSS 驗證：測試 RSA PSS 驗證。
- RSA 公有金鑰加密：測試 RSA 公有金鑰加密。

RSA 公有金鑰加密，RSA 私有解密非 CRT 和 RSA 私有金鑰解密 CRT 還將提示使用者回答以下問題：

```
Do you want to use static key [y/n]
```

如果輸入 y，則會將預先計算的金鑰匯入 HSM。

如果輸入 n，則會產生新的金鑰。

[1] 根據 NIST 指引，在 2023 年之後，FIPS 模式下的叢集不允許這樣做。對於處於非 FIP 模式的叢集，在 2023 之後仍然允許使用該叢集。如需詳細資訊，請參閱 FIPS 140 合規性：2024 機制棄用。

範例

以下範例示範在測試 HSM 的 RSA 和 AES 操作效能時，您在 pkpspeed (Linux) 或 pkpspeed_blocking (Windows) 中可以選擇的選項。

Example ：使用 pkpspeed 來測試 RSA 效能

您可以在 Windows、Linux 和相容的作業系統上執行這個範例。

Linux

在 Linux 和相容的作業系統上，使用這些指示。

```
/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 ：使用 pkpspeed 來測試 AES 效能

Linux

在 Linux 和相容的作業系統上，使用這些指示。

```
/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/...
```

用戶端 SDK 5 使用者包含不一致的值

`user list` 命令會傳回叢集中所有使用者和使用者屬性的清單。如果任何使用者的屬性值為「不一致」，表示此使用者不會在您的叢集之間同步處理。這表示使用者在叢集中的不同 HSM 上具有不同的屬性。根據不一致屬性，可以採取不同的修復步驟。

以下資料表包含解決單一使用者不一致之處的步驟。如果單一使用者有多個不一致之處，請從上到下依照下列步驟來解決這些問題。如果有多個不一致的使用者，請為每個使用者瀏覽此清單，完全解決該使用者的不一致之處，然後再繼續下一個使用者。

> **Note**
>
> 要執行這些步驟，您最好以管理員身份登錄。如果您的管理員帳戶不一致，請執行以下步驟使用管理員身分登錄並重複這些步驟，直到所有屬性都一致為止。管理員帳戶一致後，您可以繼續使用該管理員同步叢集中的其他使用者。

屬性不一致	使用者清單輸出範例	影響	復原方法

使用者「角色」「不一致」	{
	"username":
	"test_user",
	"role":
	"**inconsistent** ",
	
	"locked":
	"false",
	"mfa": [],
	"cluster-
	coverage":
	"full"
	}
		此使用者是某些 HSM CryptoUser 的使用者，而且是其他 HSM 的管理員。如果兩個 SDK 同時嘗試使用不同角色建立相同的使用者，就可能會發生這種情況。您必須移除此使用者，然後以所需的角色重新建立該使用者。	1. 以管理員身分登錄。
			2. 刪除所有 HSM 上的使用者：
			
			user delete --username **<user's name>** --role admin
			
			user delete --username **<user's name>** --role crypto-user
			3. 建立具有所需角色的使用者：
			
			user create --username **<user's name>** --role **<desired role>**

使用者「叢集覆蓋範圍」「不一致」	{ "username": "test_user",	此使用者存在於叢集中的 HSM 子集上。如果 user create 部分成功或 user delete 部分	如果使用者不應該存在，請依照下列步驟執行：

屬性不一致	使用者清單輸出範例	影響	復原方法
	"role": "crypto-user", "locked": "false", "mfa": [], "cluster-coverage": "**inconsistent** "}	成功，就可能發生這種情況。 您必須完成之前的作業，無論是建立或從叢集中移除此使用者。	1. 以管理員身分登錄。 2. 執行此命令： user delete --username**<user's name>** --role admin 3. 現在，執行以下命令： user delete --username**<user's name>** --role crypto-user 如果使用者應該存在，請依照下列步驟執行： 1. 以管理員身分登錄。 2. 執行以下命令： user create --username **<user's name>** --role **<desired role>**

屬性不一致	使用者清單輸出範例	影響	復原方法

使用者「鎖定」參數為「不一致」或「真」	{
	"username": "test_user",
	"role": "crypto-user",
	"locked": **inconsistent**,
	"mfa": [],
	"cluster-coverage": "full"
	}
		此使用者在 HSM 的子集上遭到鎖定。
		
		如果使用者使用錯誤的密碼且只連線到叢集中的 HSM 子集，就會發生這種情況。
		
		您必須變更使用者的憑證，使其在叢集中保持一致。
			如果使用者已啟用 MFA，請依照下列步驟執行：
			
			1. 以管理員身分登錄。
			2. 執行下列命令，以暫時停用 MFA：
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			3. 變更使用者的密碼，以便他們可以登入所有 HSM：
			
			user change-password --username **<user's name>** --role **<desired role>**
			
			如果應為使用者啟用 MFA，請依照下列步驟執行：
			
			1. 讓用戶登錄並重新啟用 MFA (這將要求他們簽署字符並在 PEM 檔案中提供其公有金鑰)：

屬性不一致	使用者清單輸出範例	影響	復原方法
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** —token <File>

屬性不一致	使用者清單輸出範例	影響	復原方法

MFA 狀態為「不一致」	{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [
	{
	"strategy":
	"token-sign",
	"status":
	"**inconsistent** "
	}
	],
	"cluster-coverage":
	"full"
	}
		此使用者在叢集中的不同 HSM 上具有不同的 MFA 旗標。
		
		如果 MFA 作業只在 HSM 的子集上完成，就會發生這種情況。
		
		您必須重設使用者的密碼，並允許他們重新啟用 MFA。
			如果使用者已啟用 MFA，請依照下列步驟執行：
			
			1. 以管理員身分登錄。
			2. 執行下列命令，以暫時停用 MFA：
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			3. 然後，您還需要變更使用者的密碼，以便他們可以登入所有 HSM：
			
			user change-password --username **<user's name>** --role **<desired role>**
			
			如果應為使用者啟用 MFA，請依照下列步驟執行：
			
			1. 讓用戶登錄並重新啟用 MFA (這將要求他們簽署字符並在 PEM 檔案中提供其公有金鑰)：

屬性不一致	使用者清單輸出範例	影響	復原方法
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** —token <File>

金鑰可用性檢查期間看到錯誤

問題：HSM 傳回下列錯誤：

```
Key <KEY HANDLE> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

原因：金鑰可用性檢查會尋找在罕見但可能的情況下可能遺失的金鑰。此錯誤通常發生在僅具有一個 HSM 的叢集中，或在具有兩個 HSM 的叢集中，其中一個 HSM 正在取代的期間內發生。在這些情況下，以下客戶操作可能會導致上述錯誤：

- 使用產生對稱金鑰 或之類的命令生成了一個新密鑰關鍵 generate-asymmetric-pair 。
- 列出金鑰 作業已開始。
- SDK 的新執行個體已啟動。

> **Note**
>
> OpenSSL 經常會分配開發套件的新執行個體。

解決方案/建議：從下列動作中進行選擇，以避免發生此錯誤：

- 在設定工具的設定檔案中，使用 --disable-key-availability-check 參數將金鑰可用性設定為 false。如需詳細資訊，請參閱設定工具之 參數 區段。
- 如果使用具有兩個 HSM 的叢集，請避免使用提示錯誤的作業，但在初始化程式碼期間除外。
- 將叢集中的 HSM 數量增加到至少三個。

使用 JCE 擷取金鑰

getEncoded，或 get getPrivateExponent S 返回空

getEncoded、getPrivateExponent 和 getS 將傳回 null，因為它們預設為停用狀態。若要啟用它們，請參閱 使用 JCE 擷取金鑰。

如果 getEncoded、getPrivateExponent 和 getS 在啟用後傳回 Null，金鑰不符合正確的先決條件。如需詳細資訊，請參閱 使用 JCE 擷取金鑰。

getEncoded getPrivateExponent，或 getS 在 HSM 之外的返回密鑰字節

您或有權存取您系統的人已啟用清除金鑰擷取功能。如需詳細資訊，包括如何將此組態重設為預設停用狀態，請參閱下列頁面。

- 使用 JCE 擷取金鑰
- 保護金鑰並從 HSM 擷取金鑰

HSM 調節

當您的工作負載超過叢集的 HSM 容量時，您會收到錯誤訊息，指出 HSM 忙碌或得到調節。發生這種情況時，您可能會看到 HSM 輸送量減少或拒絕請求的速率提高。此外，HSM 可能會傳送下列忙碌錯誤。

適用於用戶端 SDK 5

- 在 PKCS11 中，忙碌錯誤會映射至 CKR_FUNCTION_FAILED。發生此錯誤的原因有多種，但如果 HSM 限流造成此錯誤，則日誌中會出現下列日誌行：
  - `[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`
- 在 JCE 中，忙碌錯誤映射到 `com.amazonaws.cloudhsm.jce.jni.exception.InternalException: Unexpected`

`error with the Provider: The HSM could not queue the request for processing.`

- 其他 SDK 的忙碌錯誤打印出以下消息：`Received error response code from Server. Response Code: 187`。

適用於用戶端 SDK 3

- 在 PKCS11 中，忙碌錯誤會映射到 `CKR_OPERATION_ACTIVE` 錯誤。
- 在 JCE 中，忙碌錯誤映射至 `CFM2Exception`，狀態為 `0xBB (187)`。應用程式可以使用 `CFM2Exception` 的 `getStatus()` 函數來檢查 HSM 傳回的狀態。
- 其他 SDK 忙碌錯誤將打印出以下消息：`HSM Error: HSM is already busy generating the keys(or random bytes) for another request.`

解析度

您可以完成下列其中一個或多個動作解決這些問題：

- 在應用程式層中針對已拒絕的 HSM 作業新增重試命令。在啟用重試命令之前，請確定叢集的大小已適當符合尖峰負載。

> **Note**
>
> 對於用戶端 SDK 5.8.0 及更新版本，預設會開啟重試命令。如需有關每個 SDK 重試命令設定的詳細資訊，請參閱 用戶端 SDK 5 設定工具的進階組態。

- 依照 在叢集中新增或移除 HSM AWS CloudHSM 中的指示，將更多 HSM 新增至叢集。

> **Important**
>
> 我們建議您測試叢集的負載，以確定您應預期的尖峰負載，然後在叢集中再新增一個 HSM 以確保高可用性。

讓 HSM 使用者在叢集中的各 HSM 間保持同步

若要管理 HSM 的使用者，您可以使用稱為雲的 AWS CloudHSM 命令列工具。它只會與在工具組態檔案中的 HSM 通訊。它不會注意到不在組態檔案之叢集中的其他 HSM。

AWS CloudHSM 在叢集中的所有其他 HSM 之間同步處理 HSM 上的金鑰，但不會同步處理 HSM 的使用者或原則。使用 cloudhsm_mgmt_util 來管理 HSM 使用者時，這些使用者變更可能只會影響一些叢集的 HSM，也就是 cloudhsm_mgmt_util 組態檔案中的 HSM。在叢集中跨 HSM AWS CloudHSM 同步金鑰時，這可能會造成問題，因為擁有金鑰的使用者可能不存在於叢集中的所有 HSM 上。

為了避免這些問題，請在管理使用者之前，先編輯 cloudhsm_mgmt_util 組態檔案。如需更多詳細資訊，請參閱 ???。

與叢集的連線中斷

設定 AWS CloudHSM 用戶端時，您已提供叢集中第一個 HSM 的 IP 位址。此 IP 位址會儲存在用 AWS CloudHSM 戶端的設定檔中。用戶端啟動時，它會嘗試連接到這個 IP 地址。如果無法啟動 (例如，因為 HSM 故障或您已刪除 HSM)，您可能會看到如下錯誤：

```
LIQUIDSECURITY: Daemon socket connection error
```

```
LIQUIDSECURITY: Invalid Operation
```

若要解決這些錯誤，請將組態檔案中的 IP 地址，更新為叢集中作用中可存取 HSM 的 IP 地址。

更新用 AWS CloudHSM 戶端的組態檔

1. 使用下列其中一種方法來尋找叢集中的作用中 HSM 的 IP 地址。

   - 在 主控台中，檢視叢集詳細資訊頁面上的AWS CloudHSM HSM 標籤。
   - 使用 AWS Command Line Interface (AWS CLI) 發出指describe-clusters令。

   您在後續步驟需要此 IP 地址。
2. 使用以下命令來停止用戶端。

   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

- 用於 Windows 用戶端 1.1.2+：

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- 用於 Windows 用戶端 1.1.1 和更早版本：

  在您啟動用 AWS CloudHSM 戶端的命令視窗中使用 Ctrl + C。

3. 使用以下命令來更新用戶端的組態檔，以提供您在先前步驟找到的 IP 地址。

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

4. 使用下列命令來啟動 用戶端。

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

- 用於 Windows 用戶端 1.1.2+：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 用於 Windows 用戶端 1.1.1 和更早版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

中缺少 AWS CloudHSM 稽核記錄 CloudWatch

如果您在 2018 年 1 月 20 日之前建立了叢集，您將需要手動設定服務連結的角色，以啟用該叢集稽核日誌的交付。如需如何在 HSM 叢集上啟用服務連結角色的指示，請參閱《IAM 使用者指南》中的了解服務連結角色以及建立服務連結角色。

具有不相容長度的自訂 IV，適用於 AES 金鑰包裝

此故障診斷主題可協助您判斷應用程式是否產生無法復原的包裝金鑰。如果您受到此問題的影響，請使用本主題來解決問題。

主題

- 確定代碼是否生成不可復原的包裝金鑰。
- 如果代碼生成不可復原的包裝金鑰，則必須採取的動作

確定代碼是否生成不可復原的包裝金鑰。

只有當您符合以下所有條件時，您才會受到影響：

條件	如何知道？
應用程式使用 PKCS #11 程式庫	PKCS #11 程式庫會以 `libpkcs11.so` 檔案的形式安裝在 `/opt/cloudhsm/lib` 資料夾中。用 C 語言編寫的應用程序通常直接使用 PKCS #11 程式庫，而用 Java 編寫的應用程序可能通過 Java 抽象層間接使用程式庫。如果您使用的是 Windows，您不會受到影響，因為 PKCS #11 程式庫目前不適用於 Windows。

條件	如何知道？
您的應用程式專門使用 3.0.0 版的 PKCS #11 程式庫	如果您收到來自 AWS CloudHSM 團隊的電子郵件，您可能正在使用 PKCS #11 程式庫的 3.0.0 版。 若要檢查應用程式執行個體上的軟體版本，請使用以下指令： `rpm -qa | grep ^cloudhsm`
您可以使用 AES 金鑰包裝來包裝金鑰	AES 金鑰包裝意味著您使用 AES 金鑰來包裝一些其他金鑰。對應的機制名稱為 CKM_AES_KEY_WRAP 。它與函數 C_WrapKey 一起使用。其他使用初始化向量 (IV) 的 AES 包裝機制 (例如 CKM_AES_GCM 和 CKM_CLOUDHSM_AES_GCM) 不受此問題影響。進一步瞭解函數和機制。
調用 AES 金鑰包裝時指定自定義 IV，並且此 IV 的長度小於 8	AES 金鑰包裝通常使用如下 CK_MECHANISM 結構初始化： `CK_MECHANISM mech = {CKM_AES_KEY_WRAP, IV_POINTER, IV_LENGTH};` 只有在以下情況下此問題才適用於您： • IV_POINTER 為 NULL • IV_LENGTH 小於 8 個位元組

如果您不符合上述所有條件，則可以立即停止閱讀。您包裝的金鑰可以正確解開包裝，並且此問題不會影響您。否則，請參閱 the section called “如果代碼生成不可復原的包裝金鑰，則必須採取的動作”。

如果代碼生成不可復原的包裝金鑰，則必須採取的動作

您應該採取以下三個步驟：

1. 立即將 PKCS #11 程式庫升級到較新的版本
   - 適用於 Amazon Linux、CentOS 6 及 RHEL 6 的最新 PKCS #11 程式庫
   - 適用於 Amazon Linux 2、CentOS 7 及 RHEL 7 的最新 PKCS #11 程式庫
   - 適用於 Ubuntu 16.04 LTS 的最新 PKCS #11 程式庫
2. 更新您的軟體以使用符合標準的 IV

   強烈建議您遵循我們的範例程式碼，並直接指定 NULL IV，這會導致 HSM 使用符合標準的預設 IV。或者，您也可以明確指定 IV 與 0xA6A6A6A6A6A6A6A6 對應的 8 IV 長度。不建議使用任何其他 IV 進行 AES 金鑰包裝，並且會在未來版本的 PKCS #11 程式庫中明確禁用用於 AES 金鑰包裝的自定義 IV。

   正確指定 IV 的範例程式碼會出現在上的 aes_W rap .c 中。 GitHub
3. 識別和恢復現有包裝的金鑰

   您應該識別使用 3.0.0 版的 PKCS #11 程式庫包裝的任何金鑰，然後聯絡支援人員以尋求協助 (https://aws.amazon.com/support) 以復原這些金鑰。

> **Important**
>
> 此問題只會影響以 3.0.0 版的 PKCS #11 程式庫包裝的金鑰。您可以使用舊版 PKCS #11 程式庫 (2.0.4 和較低編號的套件) 或更新版本 (3.0.1 和更高編號的套件) 來包裝金鑰。

解決叢集建立失敗的問題

當您建立叢集時，會建 AWS CloudHSM 立 AWSServiceRoleForCloudHSM 服務連結角色 (如果角色不存在)。如果 AWS CloudHSM 無法建立服務連結角色，您嘗試建立叢集可能會失敗。

此主題說明如何解決最常見的問題，使得您可以成功建立叢集。您只需要建立此角色一次。一旦在您的帳戶中建立服務連結角色，您可以使用任何支援的方法來建立額外的叢集和管理它們。

下列區段提供建議來對與服務連結角色相關的叢集建立錯誤進行故障排除。如果您嘗試這些建議，但仍無法建立叢集，請聯絡 AWS Support。如需 AWSServiceRoleForCloudHSM 服務連結角色的詳細資訊，請參閱服務連結角色 AWS CloudHSM。

主題

- 新增遺失的許可

- 手動建立服務連結角色
- 使用非聯合身分使用者

新增遺失的許可

若要建立服務連結角色，使用者必須具有 iam:CreateServiceLinkedRole 許可。如果建立叢集的 IAM 使用者沒有此權限，叢集建立程序會在您的 AWS 帳戶中建立服務連結角色時失敗。

當許可遺失而造成失敗時，錯誤訊息會包含下列文字。

```
This operation requires that the caller have permission to call
 iam:CreateServiceLinkedRole to create the CloudHSM Service Linked Role.
```

若要解決這個錯誤，請對要建立叢集 IAM 使用者授與 AdministratorAccess 許可，或將 iam:CreateServiceLinkedRole 許可新增至使用者的 IAM 政策。如需說明，請參閱將許可新增至新的或現有的使用者。

然後，再次嘗試建立叢集。

手動建立服務連結角色

您可以使用 IAM 主控台、CLI 或 API 建立 AWSServiceRoleForCloudHSM服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立服務連結角色。

使用非聯合身分使用者

認證來源於以外的 AWS同盟使用者可以執行非同盟使用者的許多工作。不過，AWS 不會允許使用者從聯合端點發出 API 呼叫來建立服務連結角色。

若要解決這個問題，請使用 iam:CreateServiceLinkedRole 許可建立一個非聯合身分使用者，或是為現有的非聯合身分使用者授與 iam:CreateServiceLinkedRole 許可。然後，讓該使用者透過 ???建立叢集 AWS CLI。如此會在您的帳戶中建立服務連結角色。

一旦建立服務連結角色，您可以刪除非聯合身分使用者建立的叢集 (如果您想要的話)。刪除叢集不會影響角色。之後，任何具有必要權限的使用者 (包括聯合身分使用者) 都可以在您的帳戶中建立 AWS CloudHSM 叢集。

若要驗證角色是否已建立，請在 https://console.aws.amazon.com/iam/ 開啟 IAM 主控台，然後選擇角色。或是在 AWS CLI中使用 IAM 獲取角色命令。

```
$  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"
    }
}
```

擷取用戶端組態日誌

AWS CloudHSM 提供用戶端 SDK 3 和用戶端 SDK 5 的工具，以收集有關您環境的資訊，以便 Sup AWS port 對問題進行疑難排解。

主題

- 用戶端 SDK 5 支援工具
- 用戶端 SDK 3 支援工具

用戶端 SDK 5 支援工具

指令碼會擷取下列資訊：

- 用戶端 SDK 5 元件的組態檔案
- 可用日誌檔案

- 作業系統的目前版本
- 套件資訊

執行用戶端 SDK 5 的資訊工具

用戶端 SDK 5 包含每個元件的用戶端支援工具，但所有工具的功能都相同。執行工具來建立包含所有收集資訊的輸出檔案。

這些工具使用這樣的語法：

```
[ pkcs11 | dyn | jce ]_info
```

例如，若要從執行 PKCS #11 程式庫的 Linux 主機收集支援資訊，並讓系統寫入預設目錄，您可以執行下列命令：

```
/opt/cloudhsm/bin/pkcs11_info
```

該工具在 /tmp 目錄中創建輸出檔案。

PKCS #11 library

在 Linux 上收集 PKCS #11 程式庫的支援資料

- 使用支援工具收集資料。

  ```
  /opt/cloudhsm/bin/pkcs11_info
  ```

在 Windows 上收集 PKCS #11 程式庫的支援資料

- 使用支援工具收集資料。

  ```
  C:\Program Files\Amazon\CloudHSM\bin\pkcs11_info.exe
  ```

OpenSSL Dynamic Engine

收集 Linux 上的 OpenSSL 動態引擎支援資料

- 使用支援工具收集資料。

```
/opt/cloudhsm/bin/dyn_info
```

JCE provider

在 Linux 上收集 JCE 提供者的支援資料

- 使用支援工具收集資料。

```
/opt/cloudhsm/bin/jce_info
```

在 Windows 上收集 JCE 提供者的支援資料

- 使用支援工具收集資料。

```
C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe
```

從無伺服器環境擷取日誌

若要針對無伺服器環境 (例如 Fargate 或 Lambda) 進行設定，建議您將 AWS CloudHSM 記錄類型設定為。`term`設定為後`term`，無伺服器環境就可以輸出到 CloudWatch。

若要從中取得用戶端日誌 CloudWatch，請參閱 Amazon 日誌使用者指南中的使用日誌群組和 CloudWatch 日誌串流。

用戶端 SDK 3 支援工具

指令碼會擷取下列資訊：

- 作業系統及其目前版本
- 來自 `cloudhsm_client.cfg`、`cloudhsm_mgmt_util.cfg` 和 `application.cfg` 檔案的用戶端組態資訊
- 來自平台特定位置的用戶端記錄檔
- 使用 cloudhsm_mgmt_util 的叢集和 HSM資訊
- OpenSSL 資訊
- 目前的用戶端和組建版本

- 安裝程式版本

執行用戶端 SDK 3 的資訊工具

指令碼會建立包含所有收集資訊的輸出檔案。指令碼會在 /tmp 目錄中建立輸出檔案。

Linux: `/opt/cloudhsm/bin/client_info`

Windows: `C:\Program Files\Amazon\CloudHSM\client_info`

> **Warning**
>
> 此指令碼存在用戶端 SDK 3 版本 3.1.0 到 3.3.1 的已知問題。強烈建議您升級至 3.3.2 版，其中包含此問題的修正程式。使用此工具之前，請參閱已知問題頁面以獲取更多資訊。

AWS CloudHSM 配額

配額 (先前稱為限制) 是資 AWS 源的指派值。以下配額適用於每個 AWS 區域和 AWS 帳戶的 AWS CloudHSM 資源。預設配額是套用的初始值 AWS，這些值列在下表中。可調整的配額可增加超過預設配額。

Service Quotas

資源	預設配額	是否可調整？
叢集	4	是
HSM	6	是
每個叢集的 HSM 數目	28	否

建議的配額提高要求方式是開啟服務配額主控台。在主控台中，選擇您的服務和配額，然後提交您的要求。如需詳細資訊，請參閱服務配額文件。

以下「系統配額」表格中的配額無法調整。

系統配額

資源	中等名額	中等學生名額
每個叢集的金鑰數目上限	3,300	總共有 16,666 個金鑰，具有非對稱金鑰，最多可達 3,333 個
每個叢集的使用者數目上限	1,024	1,024
使用者名稱的長度上限	31 個字元	31 個字元
必要密碼長度	8 至 32 個字元	8 至 32 個字元
每個叢集的同時用戶端連線數目上限 [1]	900	900
每個應用程式的 PKCS #11 工作階段數目上限	1,024	1,024

[1] 用戶端 SDK 3 的用戶端連線是用戶端常駐程式。對於客戶端 SDK 5，用戶端連接是一個應用程序。

如需更多詳細資訊，請參閱 系統資源。

系統資源

系統資源配額是用 AWS CloudHSM 戶端在執行時允許使用的配額。

檔案描述項是作業系統的機制，用以識別和管理每個處理程序開啟的檔案。

CloudHSM 用戶端協助程式會使用檔案描述項來管理應用程式與用戶端之間的連線，以及用戶端與伺服器之間的連線。

預設情況下，CloudHSM 用戶端組態將分配 3000 個檔案描述項。此預設值旨在用戶端協助程式與 HSM 之間產生最佳的工作階段和執行緒容量。

在極罕見的情況下，如果您在資源受限的環境中執行用戶端，可能需要更改這些預設值。

> **Note**
>
> 如果變更這些值，您的 CloudHSM 用戶端效能可能會受到影響，且/或您的應用程式可能會變得無法運作。

1. 編輯 `/etc/security/limits.d/cloudhsm.conf` 檔案。

```
#
# DO NOT EDIT THIS FILE
#
hsmuser soft nofile 3000
hsmuser hard nofile 3000
```

2. 視需要編輯數值。

> **Note**
>
> `soft` 配額必須小於或等於 `hard` 配額。

3. 重新啟動 CloudHSM 用戶端協助程式程序。

> **Note**
>
> 此組態選項不適用於 Microsoft Windows 平台。

AWS CloudHSM 用戶端 SDK 的下載

下載

2021 年 3 月， AWS CloudHSM 發布了客戶端 SDK 5.0.0 版，該版本引入了具有不同需求，功能和平台支持的全新客戶端 SDK。

用戶端 SDK 5 完全支援生產環境，並提供與用戶端 SDK 3 相同的元件和支援層級，但對 CNG 和 KSP 提供者的支援除外。如需詳細資訊，請參閱 用戶端 SDK 元件比較。

> **Note**
>
> 如需每個用戶端 SDK 支援哪些平台的資訊，請參閱用戶端 SDK 5 支援的平台和用戶端 SDK 3 支援的平台。

最新版本

本節包含最新版本的用戶端 SDK。

用戶端 SDK 5 發行版本：版本 5.12.0

Amazon Linux 2

在 x86_64 架構上下載適用於 Amazon Linux 2 的 5.12.0 版本軟件：

- PKCS #11 程式庫 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL 動態引擎 (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE 提供者 (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

在 ARM64 架構上下載適用於 Amazon Linux 2 的 5.12.0 版本軟件：

- PKCS #11 程式庫 (SHA256 checksum c28a1f27e23e6ab1550dab6a353c6c9338a391a84d57f4ac99a1a3a9810c753f)
- OpenSSL 動態引擎 (SHA256 checksum 7d2e864c31c13f55443c1b1d04589fbdd4558fe103954de4384691e2c429a872)

- JCE 提供者 (SHA256 checksum e9a35eb87b2f257c47fb083d286deb835da45858b2d89759ca7d5bb4ef747b4b)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 28b6f918912b5c63bf10018824b642a805b309c21947a1d0ebbdc44647e80554)

Amazon Linux 2023

在 x86_64 架構下載適用於 Amazon 2023 的 5.12.0 版軟件：

- PKCS #11 程式庫 (SHA256 checksum 02801365cba449c5238a4e5ad3df1ddf7edd00ade976f47e956e885286503f3f)
- OpenSSL 動態引擎 (SHA256 checksum 0abed69a7c6acaafdaabdcc5fab7d56611ffd94f5480cade6f8beace9aeae056)
- JCE 提供者 (SHA256 checksum 3d5d9a903d3a216eca40f92dbb0b4030b7a86ad7ceee8d62241c97a6e1881e25)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum f96671d882b862033bba0b3633448dc6a26e45a25063e29b79a5cd4b7fc4945c)

在 ARM64 架構上下載適用於 Amazon 2023 的版本 5.12.0 軟件：

- PKCS #11 程式庫 (SHA256 checksum 53d05006b46bda8e9c1dd76e8307a780bfe0a67b10a9a87723c97f94e29f5b8e)
- OpenSSL 動態引擎 (SHA256 checksum ec1cca8e01b3303ff9473eeef6b33dc85b6affac7a47387b098905f9f2fc85ba)
- JCE 提供者 (SHA256 checksum c828ae56f46233215b9f35798b5859ebdac962af442acbc457081c3baaa44f11)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum ddd5dcd68d01f4fafaf13dc0b4ddcf98e3731ed51bdd51f85535b29353644a9f)

CentOS 7 (7.8+)

在 x86_64 架構下載適用於 CentOS 7 的 5.12.0 版軟件：

- PKCS #11 程式庫 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL 動態引擎 (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE 提供者 (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 7 (7.8+)

在 64 架構上下載適用於 RHEL 7 的 5.12.0 版軟體：

- PKCS #11 程式庫 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL 動態引擎 (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE 提供者 (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 8 (8.3+)

在 64 架構上下載適用於 RHEL 8 的 5.12.0 版軟體：

- PKCS #11 程式庫 (SHA256 checksum 6e51e95122fd0991278888287f0c408808b26fb5f1196c46168477b9090fc478)
- OpenSSL 動態引擎 (SHA256 checksum 1f1d52ff7af6c537d8cfeb5973c691a9d90a518accd685ff9b66cd78daf98928)
- JCE 提供者 (SHA256 checksum 156944607de987d6b39bd8a2d21ccd294c01377a9e35f9f15f8b0f4c8bb90033)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 351e802f79dd2d0b5f7d23bb74c146be05e5169b603c9aace24189094a45a35d)

RHEL 9 (9.2+)

在 64 架構上下載適用於 RHEL 9 的 5.12.0 版軟體：

- PKCS #11 程式庫 (SHA256 checksum d1b2f4ac7e6e0c18e788512e7726bc68b571d99a1442ce2f2e80f4b0f9956266)
- OpenSSL 動態引擎 (SHA256 checksum cf86a3f17cd6c51969d4ce80c1e3ea6513b995611be7e2e72e5e5233c71d6add)
- JCE 提供者 (SHA256 checksum ae89e256eb89ec6b4fa0f001e7a4e1d8f1c08530423e81aa74d69a17b25d9a99)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum dfe6fe5d890c33b2f5d38f906ade113b06c8c05f3427a327744c454e7302f1a5)

在 ARM64 架構上下載適用於 RHEL 9 的 5.12.0 版軟體：

- PKCS #11 程式庫 (SHA256 checksum cad72a6ab2232b4c38b90d7c62147520b975d646773dd90d7be897fa0a537d2d)

- OpenSSL 動態引擎 (SHA256 checksum ad751f756530a2317c3c64380ea3a07865b13e1874fab0e61ac530b21487c7fb)
- JCE 提供者 (SHA256 checksum d204e69acfbb90996fb08ae3573607b65630b1124fb379e078c002d55ac07766)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum c0f412cc59bafd235e046cdc1a0c5d330f2d72f7d6434672e9522f86bc945090)

Ubuntu 20.04 LTS

下載版本 5.12.0 軟件的 Ubuntu 20.04 LTS 上的架構：

- PKCS #11 程式庫 (SHA256 checksum d37b1f872eb2b1ab34303d5b8b803daa925902b645c57c6e15a28bb6321e0f42)
- OpenSSL 動態引擎 (SHA256 checksum cdc6e737652556b57d26d8816b2bc9820128cb3919360660b6f7fe65f9d39e3f)
- JCE 提供者 (SHA256 checksum f567a08344414a4776e1c5a9715657476925ca32695c4c2dd84a4f3fc5dc1615)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum f2ee5ad01c5018fc3670f602228fd71087228cd3923bf5b9bc73e4d7084dac6c)

Ubuntu 22.04 LTS

下載版本 5.12.0 軟件，適用於在 x86_64 架構上進行 LTS：

- PKCS #11 程式庫 (SHA256 checksum 0e78928acd7a1662e4b07b15d5c3ccb88714ff89e47b991c8ab6e4c2229ee5aa)
- OpenSSL 動態引擎 (SHA256 checksum 4f3168745edc5592234891a7b1d82b179a4947e87c72fade1be3bad58b7ed1a3)
- JCE 提供者 (SHA256 checksum d4c3655cdc2b00d1ab5ceafac94dfbc5c5244ed20e10fdd9db9f4e741e013733)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum d00bbacb6f2e57bd92d832a2bd11cadede972f8e82cc402ec0684b9c6b23123c)

下載版本 5.12.0 軟件的 Ubuntu 22.04 LTS 在 ARM64 架構：

- PKCS #11 程式庫 (SHA256 checksum 0c1121535c523acb864215338292bab32acee438357878b5fc0b6d268713b86f)
- OpenSSL 動態引擎 (SHA256 checksum dc7a219302021570bc8c36674d2bd33165557bb2f9a0af8fdf114f1b85a70d84)
- JCE 提供者 (SHA256 checksum af3834a10081f1e4e7894275c8b9c7b7649b8de3b6f0aeb0781a3358183a9046)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)

- CloudHSM CLI (SHA256 checksum baa253ac62c2fbcc5712561e0fb0feb25461efc3ce68cf86d4c7bf0af0f14a34)

Windows Server 2016

下載版本 5.12.0 軟件視窗服務器 2016 年 x86_64 架構:

- PKCS #11 程式庫 (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- JCE 提供者 (SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

Windows Server 2019

在 x86_64 架構上下載適用於視窗服務器 2019 的 5.12.0 版軟件：

- PKCS #11 程式庫 (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- JCE 提供者 (SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

客戶端 SDK 5.12.0 為多個平台添加了 ARM 支持，並為所有 SDK 提高了性能。CloudHSM CLI 和 JCE 提供者已新增新功能。

平台支援

- 為所有開發套件增加了對 Amazon Linux 2023 在 ARM64 架構上的支持。
- 為所有開發套件增加了對 ARM64 架構上的 RHEL 9（9.2+）的支持。
- 為所有開發套件增加了對 ARM64 架構的 Ubuntu 22.04 LTS 的支持。

CloudHSM CLI

- 已新增下列指令：
  - 金鑰複製

- 增加了對連接到多個集群的支持。如需詳細資訊，請參閱 使用 CloudHSM CLI 連線到多個叢集。

JCE 提供者

- 增加了用KeyReferenceSpec於檢索密鑰使用KeyStoreWithAttributes.
- 增加了用getKeys於使用一次檢索多個密鑰KeyStoreWithAttributes。

效能改進

- 所有軟體開發套件的 AES CBC NoPadding 作業的效能改進。

先前的用戶端 SDK 版本

本節列出先前的用戶端 SDK 版本。

版本：

Amazon Linux 2

在 x86_64 架構上下載適用於 Amazon Linux 2 的 5.11.0 版本軟件：

- PKCS #11 程式庫 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL 動態引擎 (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE 提供者 (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

在 ARM64 架構上下載適用於 Amazon Linux 2 的 5.11.0 版本軟件：

- PKCS #11 程式庫 (SHA256 checksum 5ac16449ec149c9b5e7776865803245ab17d0f1ad56df80173840c5e8d257b19)
- OpenSSL 動態引擎 (SHA256 checksum 28c2eb7f3f60172b0186e5c25f71bb7341537058a71f288673936766048083c1)
- JCE 提供者 (SHA256 checksum 06c9d9d281c12b1d2bd9a7b601d6317e46cedf175706bbfa3e4dcaed6ba05448)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 218982bb17aa751969a7866b0a9ff27e7aa5007a07817627d9cc1f7d60a78160)

Amazon Linux 2023

在 x86_64 架構上下載適用於 Amazon 2023 的 5.11.0 版本軟件：

- PKCS #11 程式庫 (SHA256 checksum 55310ab333d18bcfabdc4b74115b040386b4508934bdff93e1d054c4c4a6f9ea)
- OpenSSL 動態引擎 (SHA256 checksum f3d4934dc872a9b5212a180b9814ca2af3eca01ee228a8725563f1770add0dce)
- JCE 提供者 (SHA256 checksum 757d3abb515aeb08f4b1c83970ee0979399efee00ee78c9a9dbec05f4ed9768d)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 22af8f0501ff9a45a9e0683a408a63771c2c06c66abf5478d310d6d32e013555)

CentOS 7 (7.8+)

在 x86_64 架構下載適用於 CentOS 7 的 5.11.0 版軟件：

- PKCS #11 程式庫 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL 動態引擎 (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE 提供者 (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 7 (7.8+)

在 x86_64 架構下載適用於 RHEL 7 的 5.11.0 版軟體：

- PKCS #11 程式庫 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL 動態引擎 (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE 提供者 (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 8 (8.3+)

在 64 架構上下載適用於 RHEL 8 的 5.11.0 版軟體：

- PKCS #11 程式庫 (SHA256 checksum b95b9f588656fb14fd08bb66ce0e0da807b96daa38348dec07a508c9bef7403a)
- OpenSSL 動態引擎 (SHA256 checksum 7bb437b91a52e863b2b00ff7f427ce22522026daf757be873ee031ec6ffffd88)
- JCE 提供者 (SHA256 checksum e0db887e05eb535314f4d99f21da12d87d35ebb8baf9726f4ce8f01d9df0ea01)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 8485b5a6d679767ca9b4f611718159a643cf3e85090a8e4d20fe53c3707e25c3)

RHEL 9 (9.2+)

在 x86_64 架構下載適用於 RHEL 9 的 5.11.0 版軟體：

- PKCS #11 程式庫 (SHA256 checksum 87b56a20accf67df53a203b7f115655b2acfaec4516682d4976d9475b10bec8e)
- OpenSSL 動態引擎 (SHA256 checksum 83a6b58572e985df937beede4b10e867b0ac6050ace8010dc8d535be365d2747)
- JCE 提供者 (SHA256 checksum ee95213d02d913250478d0793d6dd578e5c54d765e635c7468a49bdff4c2a6f3)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 7e168ed3bef8e9c5110645e9960680e9a57f7b94e16aec71422e3c67ebc58fb5)

Ubuntu 20.04 LTS

下載版本 5.11.0 軟件的 Ubuntu 20.04 LTS 上的架構：

- PKCS #11 程式庫 (SHA256 checksum abc3a339d1fe5850db65620804e9a910f8b4f913624ef9b7189f2f0df1825c01)
- OpenSSL 動態引擎 (SHA256 checksum 075fc3f9974d552f27ad67fa92c8abff31b756b9add875b8cd4957e6801583a4)
- JCE 提供者 (SHA256 checksum 5de45c519133a0dae8da3ac01809db7974be25c14c15eb773fc5c972c0178c13)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 83e0e4505a063792c19feb3d4cfd032b9089091916168d92b0f51a967a007734)

Ubuntu 22.04 LTS

下載版本 5.11.0 軟件，適用於在 x86_64 架構上進行 LTS：

- PKCS #11 程式庫 (SHA256 checksum b8f20be125c8530b2a7bd945956e9c04296fba5634af408b40be4e03bdbad72a)
- OpenSSL 動態引擎 (SHA256 checksum d728c156eb4ee5c67159e57d6b092785800baa5fb61c14d64f460a8b8f53a778)

- JCE 提供者 (SHA256 checksum 44e943b8cd1176ad666e249342687744a280c6222df58b5a9f084c932f628284)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 8ccf5389d459611be813e42d7f9d040090f94f3fe88f9d110bcfb25e9619e4a7)

Windows Server 2016

下載版本 5.11.0 軟件的視窗服務器 2016 年 x86_64 架構:

- PKCS #11 程式庫 (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- JCE 提供者 (SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

Windows Server 2019

在 x86_64 架構上下載適用於視窗服務器 2019 的 5.11.0 版軟件：

- PKCS #11 程式庫 (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- JCE 提供者 (SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

客戶端 SDK 5.11.0 添加了新功能，提高了穩定性，並包括所有 SDK 的錯誤修復。

平台支援

- 增加了對所有開發套件的 Amazon Linux 2023 和 RHEL 9（9.2 以上）的支持。
- 由於最近的生命週期結束，已移除對 Ubuntu 18.04 LTS 的支援。
- 由於最近的生命週期結束，刪除了對 Amazon Linux 的支持。

CloudHSM CLI

- 已新增下列指令：

- 加密符號
- 加密驗證
- 密鑰進口 PEM
- 密鑰展開
- 按鍵包裝

- 產生金鑰的檔案現在支援匯出公開金鑰。

OpenSSL 動態引擎

- AWS CloudHSM OpenSSL 動態引擎現在已經在安裝了 3.x 的 OpenSSL 程式庫版本的平台上受到支援。這包括 Amazon Linux 2023，瑞爾 9（9.2 +）和

JCE

- 增加了對 JDK 17 和 JDK 21 的支持。
- 增加了對用於 HMAC 操作的 AES 密鑰的支持。
- 添加了新的密鑰屬性ID。
- 引入了一個關鍵耗盡的新DataExceptionCause變體：DataExceptionCause.KEY_EXHAUSTED。

錯誤修正/效能改進：

- 將label屬性的最大長度從 126 個字元增加到 127 個字元。
- 修正了一個無法使用 RsaOaep 機制解包 EC 鍵的錯誤。
- 已解決 JCE 提供者中 GetKey 作業的已知問題。請參閱 問題：使用 GetKey 作業的用戶端 SDK 5 記憶體洩漏 以取得詳細資訊。
- 改進了每 FIPS 140-2 達到最大加密塊限制的三重 DES 密鑰的所有 SDK 的日誌記錄。
- 已新增 OpenSSL 動態引擎的已知問題。如需詳細資訊，請參閱 OpenSSL 動態引擎的已知問題。

版本：

Amazon Linux

下載適配 Amazon Linux (x86_64 架構) 的 5.10.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum d63adf3e96c19c2d894b2defcbadd916dbb0398993050b1358bd93a36aa5acab)
- OpenSSL 動態引擎 (SHA256 checksum 4daa3e591ffd5f7ce8ef3759c41deaa38867f5e5d21f15927aea83afb1678ac5)
- JCE 提供者 (SHA256 checksum 6c1ac94d3080f1c609d9dafbcb14480911beef3a488c4ed6f2b11b377da9b477)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum c12617fcd7990ba53e96f477979b410e3a5f17842ca7a912861b8b820809b5b5)

Amazon Linux 2

下載適配 Amazon Linux 2 (x86_64 架構) 的 5.10.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL 動態引擎 (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE 提供者 (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

下載適配 Amazon Linux 2 (ARM64 架構)的 5.10.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum 5d8dfd835f1ed5a7f5a4fcc8ecf81cfa29883aca7e2985de69b5db723ab663db)
- OpenSSL 動態引擎 (SHA256 checksum 91fb8efe2646bf0dbd9087554baa09554714e9d56e9bfd5c0dc3023a9f485574)
- JCE 提供者 (SHA256 checksum 99f6e55c37fdf00085a816d46835aeff54470797b3b71f4d28a70dc79c9caf44)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 4a88ba9b4cf0dd5573f3dd88ab9dc257e4c486069cb529c5d554979ee2dd83af)

CentOS 7 (7.8+)

下載適配 CentOS 7 (x86_64 架構) 的 5.10.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL 動態引擎 (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE 提供者 (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)

  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 7 (7.8+)

下載適配 RHEL 7 (x86_64 架構) 的 5.10.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL 動態引擎 (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE 提供者 (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 8 (8.3+)

下載適配 RHEL 8 (x86_64 架構) 的 5.10.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum 96afb7042a148ddc7a60ab6235b49e176d0460d1c2957bd76ca3d8406ac1cb03)
- OpenSSL 動態引擎 (SHA256 checksum 2caad2bffea8aef73c91ad422d09772ef830fe7f80a7be19020e6a107eadfbe8)
- JCE 提供者 (SHA256 checksum 3543551f08fbe3900821ea2d4ea148b4e86e2334bc94d7ffef6f3b831457cd71)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 812eccaadfc490f13bcd0b0a835ef58f3a3d4344ad7e0a237de476dd24509525)

Ubuntu 18.04 LTS

下載適配 Ubuntu 18.04 LTS (x86_64 架構) 的 5.10.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum be4c61766b8b46e1f6c14c3dcf90aaab9f38240fcd9c68b4009704276c5f6f4a)
- OpenSSL 動態引擎 (SHA256 checksum 64bd8af827b6dc3786e8ad28858cbc4ef6a0fd42164a0945f427eddcf5f02858)
- JCE 提供者 (SHA256 checksum 9fcbdf08e93641468588b608173f26f18781bbc029ed95b2e086da29a968cc00)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)

- CloudHSM CLI (SHA256 checksum 13808bddddb7eedeb2b8486d23a9976c7fa8d9220149a6b9400626bcaff3b513)

> **Note**
>
> 由於最近 Ubuntu 18.04 LTS 的生命週期結束，AWS CloudHSM 將不再能夠在下一個版本中支持此平台。

Ubuntu 20.04 LTS

下載適配 Ubuntu 20.04 LTS (x86_64 架構) 的 5.10.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum 99ae96504580ff85ed4958a582903a847f666bdaafafbe887a5a76db58f24500)
- OpenSSL 動態引擎 (SHA256 checksum 13e3f6fe086acf9617b163f66e3941f973daa583fb9322d16c396aa29fc3611d)
- JCE 提供者 (SHA256 checksum 44562cebd9af1aa965840cd9bcb237e518d24c715b3c8bca1405c9c1871835e2)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum ab71b4ec531c5e6d05c91539c7edc1c07e6c748052ebf6200f148cb6812538c5)

Ubuntu 22.04 LTS

下載適配 Ubuntu 22.04 LTS (x86_64 架構) 的 5.10.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum ee331a44fbe4936ec98a3ae55d58e67ed38e8bbff0a4f4ce8b1bd8239b75877b)
- 此平台尚不支援 OpenSSL 動態引擎。
- JCE 提供者 (SHA256 checksum 9e44d14dd33624f6fe36711633013e47e4a93f4d4635e08900546113ded56e3d)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 2df361546848cd3f8965b1007dca42a0c959eb10d9e3f4995e8e1c852406751d)

Windows Server 2016

下載適配 Windows Server 2016 (x86_64 架構) 的 5.10.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- JCE 提供者 (SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)

    - Javadocs 文檔 AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

Windows Server 2019

下載適配 Windows Server 2019 (x86_64 架構) 的 5.10.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- JCE 提供者 (SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)
    - Javadocs 文檔 AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

用戶端 SDK 5.10.0 提升了穩定性，並修復了所有 SDK 的錯誤。

CloudHSM CLI

- 新增了允許客戶使用 CloudHSM CLI 管理金鑰的新命令，包括：
    - 建立對稱金鑰和非對稱金鑰對
    - 共用和取消共用金鑰
    - 使用索引键屬性列出和篩選金鑰
    - 設定索引键屬性
    - 產生金鑰參考檔案
    - 刪除金鑰
- 已改善錯誤記錄。
- 新增了在互動式模式下對多行 unicode 命令的支持。

錯誤修正/效能改進：

- 提升了所有 SDK 在匯入、取消包裝、衍生和建立工作階段金鑰方面的性能。
- 修正了 JCE 提供者中無法在退出時移除暫存檔案的錯誤。
- 修正了因替換叢集中的 HSM 後造成特定情況下連線錯誤的錯誤。
- 修改了 JCE `getVersion` 輸出格式，以處理較大次要版本編號並包含修補程式編號。

平台支援

- 新增了 Ubuntu 22.04 對 JCE、PKCS #11 和 CloudHSM CLI 的支援 (該平台尚不支援 OpenSSL 動態引擎)。

5.9.0 版本

Amazon Linux

下載適配 Amazon Linux (x86_64 架構) 的 5.9.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum 4f368be41f006b751ac41b14e1435c27841f60bbde0f032ec02a359fea637dcf)
- OpenSSL 動態引擎 (SHA256 checksum 81af0d34683825cd6ff844ccacf9c8f4842a4ba76e3875a89121d09a286b4490)
- JCE 提供者 (SHA256 checksum e8e5bc09d8e0b3cb24f30ab420fe08902a19073012335ac94382ec55fcc45abd)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 17284144b45043204ce012fe8b62b1973f10068950abedbd9c2c6172ed0979c6)

Amazon Linux 2

下載適配 Amazon Linux 2 (x86_64 架構) 的 5.9.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL 動態引擎 (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE 提供者 (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

下載適配 Amazon Linux 2 (ARM64 架構) 的 5.9.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum 4337dca5a08c5194b1118fa197bb4a4f7988df4e1b961e6f2e367295ba99d61d)
- OpenSSL 動態引擎 (SHA256 checksum 4f08689934e877662a7ce64554fb04eb4b2c213b936018609ff187d100e34a85)
- JCE 提供者 (SHA256 checksum b337b80271a2d308949d5911971fe6ad35df4e34876a481fcac347f1d897fe39)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)

- CloudHSM CLI (SHA256 checksum a4d466e6b5f74dcd283ba32c9dd87441941d5e5a05936b7c2b4cc7ef85eb1071)

CentOS 7 (7.8+)

下載適配 CentOS 7 (x86_64 架構) 的 5.9.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL 動態引擎 (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE 提供者 (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 7 (7.8+)

下載適配 RHEL 7 (x86_64 架構) 的 5.9.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL 動態引擎 (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE 提供者 (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 8 (8.3+)

下載適配 RHEL 8 (x86_64 架構) 的 5.9.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum 081887f6ea1d9df9d1e409b2b5bde83e965c42229acbeb1f950c8fe478361edc)
- OpenSSL 動態引擎 (SHA256 checksum 6b0500a42fd57c39f076f14e5079f80145b6ebd2c441395761eb04600c07bda5)
- JCE 提供者 (SHA256 checksum 2bc7ac26b259af92a65fbd5a30d5eb2a92ce0e70efe41feb53bf82f168aa90bb)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 79ecbe9b4c5316ccf447d8c59b76b5ac2cc854bd79cd50c1f29197aa8cb080db)

Ubuntu 18.04 LTS

下載適配 Ubuntu 18.04 LTS (x86_64 架構) 的 5.9.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum bc6d2227edd7b5a83fed32741fbacbb1756d5df89ebb3435d96f0609a180db65)
- OpenSSL 動態引擎 (SHA256 checksum 2d6a26434fa6faf337f1dfb42de033220fa405a82d4540e279639a03b3ee6e9d)
- JCE 提供者 (SHA256 checksum e12aef122f490e9026452ce31c25625b1accb9a5866b3d470488f10f047f1873)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum f0bcabe594db3e8ff86cc0f65c2a10858d34452eb6b9fc33d7aac05c0f5f4f30)

Ubuntu 20.04 LTS

下載適配 Ubuntu 20.04 LTS (x86_64 架構) 的 5.9.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum 15dde8182f432de9e7d369b05e384e1f2d80dcca85db3b16ecc26cdef1a34bb9)
- OpenSSL 動態引擎 (SHA256 checksum c8ba94a999038af87d4905b7c1feb4cc87e20d1776a32ef6f6d11ee000b5a896)
- JCE 提供者 (SHA256 checksum de33cd3e8130a06d9da5207079533aac8276a1319ac435a3737b4f65bd8fb972)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum cfa31535ad9a99a5113496c06fbace38e9593491aca9bb031a18b51075973e68)

Windows Server 2016

下載適配 Windows Server 2016 (x86_64 架構) 的 5.9.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- JCE 提供者 (SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

Windows Server 2019

下載適配 Windows Server 2019 (x86_64 架構) 的 5.9.0 版本軟體：

- PKCS #11 程式庫 (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- JCE 提供者 (SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)
  - Javadocs 文檔 AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

用戶端 SDK 5.9.0 提升了穩定性，並修復了所有 SDK 的錯誤。已對所有 SDK 進行了優化，以便在確定 HSM 無法使用時可立即通知應用程式操作失敗。此版本改善了 JCE 的效能。

JCE 提供者

- 改善效能
- 修正工作階段集區耗盡的已知問題

版本 3.4.4 版本

若要在 Linux 平台上升級用戶端 SDK 3，您須使用可同時升級用戶端常駐程式和所有程式庫的批次命令。如需關於升級的詳細訊息，請參閱用戶端 SDK 3 升級。

> **Note**
>
> 用戶端 SDK 3 及其相關命令列工具 (金鑰管理公用程式和 CloudHSM 管理公用程式) 僅適用於 HSM 類型 hsm1.medium。如需詳細資訊，請參閱 AWS CloudHSM 叢集模式和 HSM 類型。

若要下載此軟體，請先選擇您偏好的作業系統標籤，然後選擇每個軟體套件的連結。

Amazon Linux

下載適配 Amazon Linux 的 3.4.4 版本軟體：

- AWS CloudHSM 客戶端 (SHA256 checksum 900de424d70f41e661aa636f256a6a79cc43bea6b0fe6eb95c2aaa63e5289505)
- PKCS #11 程式庫 (SHA256 checksum a3f93f084d59fee5d7c859292bc02cb7e7f15fb06e971171ebf9b52bbd229c30)
- OpenSSL 動態引擎 (SHA256 checksum 8db07b9843d49016b0b6fec46d39881d94e426fcaae1cee2747be14af9313bb0)
- JCE 提供者 (SHA256 checksum 360617c55bf4caa8e6e78ede079ca68cf9ef11473e7918154c22ba908a219843)

- AWS CloudHSM 管理工具 (SHA256 checksum c9961ffe38921131bd6f3702e10d73588e68b8ab10fbb241723e676f4fa8c4fa)

Amazon Linux 2

下載適配 Amazon Linux 2 的 3.4.4 版本軟體：

- AWS CloudHSM 客戶端 (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS #11 程式庫 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL 動態引擎 (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE 提供者 (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM 管理工具 (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 6

AWS CloudHSM 用戶端軟體套件版本 3.4.4 並不支援 CentOS 6。

使用 the section called “3.2.1 版本” 替代 CentOS 6 或選擇一個支援平台。

CentOS 7 (7.8+)

下載適配 CentOS 7 的 3.4.4 版本軟體：

- AWS CloudHSM 客戶端 (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS #11 程式庫 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL 動態引擎 (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE 提供者 (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM 管理工具 (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 8

下載適配 CentOS 8 的 3.4.4 版本軟體：

- AWS CloudHSM 客戶端 (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)
- PKCS #11 程式庫 (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)
- JCE 提供者 (SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM 管理工具 (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

> **Note**
>
> 因近期 CentOS 8 生命週期終止，下一個版本將不再支援此平台。

RHEL 6

AWS CloudHSM 不支援使用用戶端 SDK 版本 3.4.4 的 RedHat 企業版 Linux 6。

the section called “3.2.1 版本”適用於 RedHat 企業版 Linux 6，或選擇支援的平台。

RHEL 7 (7.8+)

下載適用於 RedHat 企業版本 3.4.4 軟件 7：

- AWS CloudHSM 客戶端 (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS #11 程式庫 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL 動態引擎 (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE 提供者 (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM 管理工具 (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

RHEL 8 (8.3+)

下載適用於 RedHat 企業版本 3.4.4 軟件 8：

- AWS CloudHSM 客戶端 (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)

- PKCS #11 程式庫 (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)
- JCE 提供者 (SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM 管理工具 (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

Ubuntu 16.04 LTS

下載適配 Ubuntu 16.04 LTS 的 3.4.4 版本軟體：

- AWS CloudHSM 客戶端 (SHA256 checksum 317c92c2e0b5d60afab1beb947f053d13ddaacb994cccc2c2b898e997ece29b9)
- PKCS #11 程式庫 (SHA256 checksum 91451c420c51488a022569fd32f052a3b988a2883ea4c2ac952acb61a2fea37c)
- OpenSSL 動態引擎 (SHA256 checksum 4098771ad0e38df9bf14d50520ca49b9395f819f0387e2bc3b0e61abb5888e66)
- JCE 提供者 (SHA256 checksum e136ff183271c2f9590a9fccb8261a7eb809506686b070e3854df1b8686c6641)
- AWS CloudHSM 管理工具 (SHA256 checksum cbf24a4032f393a913a9898b1b27036392104e8e05d911cab84049b2bcca2541)

> **Note**
>
> 因 Ubuntu 16.04 生命週期即將終止，下一個版本將不再支援此平台。

Ubuntu 18.04 LTS

下載適配 Ubuntu 18.04 LTS 的 3.4.4 版本軟體：

- AWS CloudHSM 客戶端 (SHA256 checksum cf57d5e0e95efbf032aac8887aebd59ac8cc80e97c69e7c39fdad40873374fe8)
- PKCS #11 程式庫 (SHA256 checksum 428f8bdad7925db5401112f707942ee8f3ca554f4ab53fa92237996e69144d2f)
- JCE 提供者 (SHA256 checksum 1ff17b8f7688e84f7f0bfc96383564dca598a1cab2f2c52c888d0361682f2b9e)
- AWS CloudHSM 管理工具 (SHA256 checksum afe253046146ed6177c520b681efc680dac1048c4a95b3d8ad0f305e79bbe93e)

Windows Server

AWS CloudHSM 支持 64 位版本的視窗服務器 2012, 視窗服務器 2012 R2, 視窗服務器 2016, 和視窗服務器 2019. 適用於 Windows 伺服器的 AWS CloudHSM 3.4.4 用戶端軟體包含所需的 CNG 和 KSP 提供者。如需詳細資訊，請參閱安裝和設定 AWS CloudHSM 用戶端 (Windows)。下載適配 Windows Server 的最新版本 (3.4.4)：

- AWS CloudHSM 視窗伺服器 (SHA256 checksum d51a7db588e9121d8f0b0351606bd986e1c4de6547f2c8235200dc8a5ffbe53e)
- AWS CloudHSM 管理工具 (SHA256 checksum 0c12d7da9086735cdf189535937a8e036163009c5018dcaf2ee9cdbb6bd4c06f)

3.4.4 版本新增了對 JCE 提供者的更新。

AWS CloudHSM 用戶端軟體

- 更新版本的一致性。

PKCS #11 程式庫

- 更新版本的一致性。

OpenSSL 動態引擎

- 更新版本的一致性。

JCE 提供者

- 將 log4j 更新至 2.17.1 版本。

Windows (CNG 和 KSP 提供者)

- 更新版本的一致性。

淘汰版本

5.8.0 及更早版本已棄用。建議在工作負載中不要使用已淘汰版本。後續不再對已淘汰版本提供相容更新，也不再託管已淘汰版本供下載。若您在使用已淘汰版本過程中，生產受到影響，則須升級版本進行軟體修復。

已停用的用戶端 SDK 5 版本

本節列出已淘汰的用戶端 SDK 5 版本。

5.8.0 版本

5.8.0 版本新增了如下功能：使用 CloudHSM CLI 進行規定人數驗證、使用 JSSE 進行 SSL/TLS 卸載、提供 PKCS #11 多插槽支援、提供 JCE 多叢集/多使用者支援、使用 JCE 擷取金鑰、提供支援 JCE 的 KeyFactory、重新設定非終端傳回碼。該版本也提升了穩定性並修正了錯誤。

PKCS #11 程式庫

- 新增了對多插槽配置的支援功能。

JCE 提供者

- 新增了基於組態的金鑰擷取功能。
- 新增了對多叢集和多使用者組態的支援功能。
- 新增了使用 JSSE 進行 SSL 和 TLS 卸載的支援功能。
- 增加了對 A NoPadding ES/CBC/ 的解包支持。
- 增加了新類型的關鍵工廠： SecretKeyFactory 和 KeyFactory。

CloudHSM CLI

- 新增了對規定人數驗證的支援功能。

5.7.0 版本

5.7.0 版本引入了 CloudHSM CLI 並包含一種全新加密式訊息驗證程式碼 (CMAC) 演算法。此版本在 Amazon Linux 2 上新增了 ARM 架構。JCE 提供者 Javadocs 現在可用於 AWS CloudHSM。

PKCS #11 程式庫

- 提升了穩定性並修正了錯誤。
- 目前 Amazon Linux 2 已支援 ARM 架构。
- 演算法
  - CKM_AES_CMAC (簽署及驗證)

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。
- 目前 Amazon Linux 2 已支援 ARM 架构。

JCE 提供者

- 提升了穩定性並修正了錯誤。
- 演算法
  - AESCMAC

5.6.0 版本

5.6.0 版本包含 PKCS #11 程式庫和 JCE 提供者的新機制支援功能。5.6 版也支援 Ubuntu 20.04。

PKCS #11 程式庫

- 提升了穩定性並修正了錯誤。
- 機制
  - 適用於加密、解密、簽署及驗證模式的 CKM_RSA_X_509

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。

JCE 提供者

- 提升了穩定性並修正了錯誤。
- 加密方式
  - RSA/ECB/NoPadding, 用於加密和解密模式

支援的金鑰

- EC 曲線 secp224r1 和 secp521r1

平台支援

- 新增對 Ubuntu 20.04 的支援。

5.5.0 版本

5.5.0 版本新增了對 OpenJDK 11 的支援，整合了 Keytool 和 Jarsigner，同時也為 JCE 增加了其他機制。解決有關 KeyGenerator 類別錯誤地將索引鍵大小參數解譯為位元組數而非位元組數目的已知問題。

PKCS #11 程式庫

- 提升了穩定性並修正了錯誤。

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。

JCE 提供者

- 支援 Keytool 和 Jarsigner 公用程式
- 所有平台上均支援 OpenJDK 11
- 加密方式
  - AES/CBC/ 加密NoPadding 和解密模式
  - AES/ECB/PKCS5Padding 加密和解密模型
  - AES/CTR/ NoPadding 加密和解密模式
  - AES/GCM/包裝NoPadding 和解包模式
  - DESede/ECB/PKCS5Padding 加密和解密模型
  - 選擇/CBC/ NoPadding 加密和解密模式
  - 包裝/ECB/包裝和解NoPadding 包模式
  - AESWrap/ECB/PKCS5Padding 包裝和取消包裝模型
  - 包裝/ECB/包裝和解ZeroPadding 包模式
  - RSA/ECB/PKCS1Padding 包裝和取消包裝模型
  - RSA/ECB/OAEPPadding 包裝和取消包裝模型
  - RSA/ECB/OAEPWithSHA-1ANDMGF1Padding 包裝和取消包裝模型

- RSA/ECB/OAEPWithSHA-224ANDMGF1Padding 包裝和取消包裝模型
- RSA/ECB/OAEPWithSHA-256ANDMGF1Padding 包裝和取消包裝模型
- RSA/ECB/OAEPWithSHA-384ANDMGF1Padding 包裝和取消包裝模型
- RSA/ECB/OAEPWithSHA-512ANDMGF1Padding 包裝和取消包裝模型
- RSAAESWrap/ECB/OAEPPadding 包裝和取消包裝模型
- RSAAESWrap/ECB/OAEPWithSHA-1ANDMGF1Padding 包裝和取消包裝模型
- RSAAESWrap/ECB/OAEPWithSHA-224ANDMGF1Padding 包裝和取消包裝模型
- RSAAESWrap/ECB/OAEPWithSHA-256ANDMGF1Padding 包裝和取消包裝模型
- RSAAESWrap/ECB/OAEPWithSHA-384ANDMGF1Padding 包裝和取消包裝模型
- RSAAESWrap/ECB/OAEPWithSHA-512ANDMGF1Padding 包裝和取消包裝模型

- KeyFactory 和 SecretKeyFactory
  - RSA：2048 位元至 4096 位元 RSA 金鑰，以 256 位元為單位遞增
  - AES：128 位元、192 位元和 256 位元 AES 金鑰
  - NIST 曲線 secp256r1 (P-256)、secp384r1 (P-384) 和 secp256k1 的 EC 金鑰對
  - DESede (3DES)
  - GenericSecret
  - HMAC：支援 SHA1、SHA224、SHA256、SHA384 和 SHA512 雜湊
- **簽署/驗證**
  - RSASSA-PSS
  - SHA1withRSA/PSS
  - SHA224withRSA/PSS
  - SHA256withRSA/PSS
  - SHA384withRSA/PSS
  - SHA512withRSA/PSS
  - SHA1withRSAandMGF1
  - SHA224withRSAandMGF1
  - SHA256withRSAandMGF1
  - SHA384withRSAandMGF1
  - SHA512withRSAandMGF1

5.4.2 版本

5.4.2 版本提升了穩定性，並修復了所有 SDK 的錯誤。該版本也是 CentOS 8 平台的最後一個發行版本。如需詳細資訊，請參閱 CentOS 網站。

PKCS #11 程式庫

- 提升了穩定性並修正了錯誤。

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。

JCE 提供者

- 提升了穩定性並修正了錯誤。

5.4.1 版本

5.4.1 版本解決了 PKCS #11 程式庫的一個已知問題。該版本也是 CentOS 8 平台的最後一個發行版本。如需詳細資訊，請參閱 CentOS 網站。

PKCS #11 程式庫

- 提升了穩定性並修正了錯誤。

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。

JCE 提供者

- 提升了穩定性並修正了錯誤。

5.4.0 版本

5.4.0 版本為所有平台新增了對 JCE 提供者的初始支援。JCE 提供者與 OpenJDK 8 相容。

PKCS #11 程式庫

- 提升了穩定性並修正了錯誤。

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。

JCE 提供者

- 金鑰類型
  - RSA：2048 位元至 4096 位元 RSA 金鑰，以 256 位元為單位遞增。
  - AES：128 位元、192 位元和 256 位元 AES 金鑰。
  - NIST 曲線 secp256r1 (P-256)、secp384r1 (P-384) 和 secp256k1 的 ECC 金鑰對。
  - DESede (3DES)
  - HMAC：支援 SHA1、SHA224、SHA256、SHA384 和 SHA512 雜湊。
- 加密方式 (僅限加密和解密)
  - AES/GCM/ NoPadding
  - AES/ECB/ NoPadding
  - AES/CBC/PKCS5Padding
  - Desee/ECB/ NoPadding
  - DESede/CBC/PKCS5Padding
  - A/ 控制/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
- 摘要
  - SHA-1

  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- **簽署/驗證**
  - NONEwithRSA
  - SHA1withRSA
  - SHA224withRSA
  - SHA256withRSA
  - SHA384withRSA
  - SHA512withRSA
  - NONEwithECDSA
  - SHA1withECDSA
  - SHA224withECDSA
  - SHA256withECDSA
  - SHA384withECDSA
  - SHA512withECDSA
- **與 Java 的集成 KeyStore**

5.3.0 版本

PKCS #11 程式庫

- **提升了穩定性並修正了錯誤。**

OpenSSL 動態引擎

- **新增了曲線 P-256、P-384 和 secp256k1 的 ECDSA 簽署/驗證支援。**
- **添加對平台的支持：Amazon Linux，Amazon Linux 2，琴 7.8+，RHEL 7（7.8+）。**
- **新增了對 OpenSSL 1.0.2 版本的支援。**
- **提升了穩定性並修正了錯誤。**

JCE 提供者

- **金鑰類型**
  - RSA：2048 位元至 4096 位元 RSA 金鑰，以 256 位元為單位遞增。
  - AES：128 位元、192 位元和 256 位元 AES 金鑰。
  - NIST 曲線 secp256r1 (P-256)、secp384r1 (P-384) 和 secp256k1 的 EC 金鑰對。
  - DESede (3DES)
  - HMAC：支援 SHA1、SHA224、SHA256、SHA384 和 SHA512 雜湊。
- 加密方式 (僅限加密和解密)
  - AES/GCM/ NoPadding
  - AES/ECB/ NoPadding
  - AES/CBC/PKCS5Padding
  - Desee/ECB/ NoPadding
  - DESede/CBC/PKCS5Padding
  - A/ 控制/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
- **摘要**
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- **簽署/驗證**
  - NONEwithRSA

- SHA1withRSA
- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA
- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

- 與 Java 的集成 KeyStore

5.2.1 版本

PKCS #11 程式庫

- 提升了穩定性並修正了錯誤。

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。

5.2.0 版本

5.2.0 版本新增了將其他金鑰類型和機制添加至 PKCS #11 程式庫的支援。

PKCS #11 程式庫

金鑰類型

- ECDSA：P-224、P-256、P-384、P-521 和 secp256k1 曲線
- 三重 DES (3DES)

機制

- CKM_EC_KEY_PAIR_GEN
- CKM_DES3_KEY_GEN
- CKM_DES3_CBC
- CKM_DES3_CBC_PAD
- CKM_DES3_ECB
- CKM_ECDSA
- CKM_ECDSA_SHA1
- CKM_ECDSA_SHA224
- CKM_ECDSA_SHA256
- CKM_ECDSA_SHA384
- CKM_ECDSA_SHA512
- 用於加密/解密的 CKM_RSA_PKCS

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。

5.1.0 版本

5.1.0 版本新增了將其他機制添加至 PKCS #11 程式庫的支援。

PKCS #11 程式庫

機制

- 用於包裝/取消包裝的 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
- CKM_AES_ECB

- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_SP800_108_COUNTER_KDF
- CKM_GENERIC_SECRET_KEY_GEN
- CKM_SHA_1_HMAC
- CKM_SHA224_HMAC
- CKM_SHA256_HMAC
- CKM_SHA384_HMAC
- CKM_SHA512_HMAC
- 僅用於包裝/取消包裝的 CKM_RSA_PKCS_OAEP
- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD

API 操作

- C_ CreateObject
- C_ DeriveKey
- C_ WrapKey
- C_ UnWrapKey

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。

5.0.1 版本

5.0.1 版本新增了對 OpenSSL 動態引擎的初始支援。

PKCS #11 程式庫

- 提升了穩定性並修正了錯誤。

OpenSSL 動態引擎

- OpenSSL 動態引擎的初始版本。
- 此版本為金鑰類型和 OpenSSL API 提供了介紹性支援：
  - 產生 2048 位元、3072 位元和 4096 位元金鑰的 RSA 金鑰
  - OpenSSL API：
    - 使用帶有 SHA1/224/256/384/512 和 RSA PSS 進行 RSA 簽署
    - RSA 金鑰產生

  如需詳細資訊，請參閱 OpenSSL 動態引擎。
- 支援平台：CentOS 8.3+、Red Hat Enterprise Linux (RHEL) 8.3+ 和 Ubuntu 18.04 LTS
  - 版本要求：OpenSSL 1.1.1

  如需詳細資訊，請參閱支援平台。
- CentOS 8.3+、Red Hat Enterprise Linux (RHEL) 8.3、Ubuntu 18.04 LTS、包括適用於特定加密套件的 NGINX 1.19 支持 SSL/TLS 卸載。

  如需詳細資訊，請參閱 Linux 上使用 SSL/TLS 卸載。

5.0.0 版本

5.0.0 版本 為初始版本。

PKCS #11 程式庫

- 此版本為初始版本。

用戶端 SDK 5.0.0 版本對 PKCS #11 程式庫的支援情況說明

本節詳細介紹了用戶端 SDK 5.0.0 版本在金鑰類型、機制、API 操作和屬性方面的支援情況。

金鑰類型：

- AES：128 位元、192 位元和 256 位元 AES 金鑰。
- RSA：2048 位元至 4096 位元 RSA 金鑰，以 256 位元為單位遞增

機制：

- CKM_AES_GCM
- CKM_AES_KEY_GEN
- CKM_CLOUDHSM_AES_GCM
- CKM_RSA_PKCS
- CKM_RSA_X9_31_KEY_PAIR_GEN
- CKM_SHA1
- CKM_SHA1_RSA_PKCS
- CKM_SHA224
- CKM_SHA224_RSA_PKCS
- CKM_SHA256
- CKM_SHA256_RSA_PKCS
- CKM_SHA384
- CKM_SHA384_RSA_PKCS
- CKM_SHA512
- CKM_SHA512_RSA_PKCS

API 操作：

- 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

屬性：

- GenerateKeyPair
  - 所有 RSA 金鑰屬性
- GenerateKey
  - 所有 AES 金鑰屬性
- GetAttributeValue
  - 所有 RSA 金鑰屬性
  - 所有 AES 金鑰屬性

範例：

- 產生金鑰 (AES、RSA、EC)
- 列出金鑰屬性
- 使用 AES GCM 加密和解密資料
- 使用 RSA 簽署和驗證資料

已停用的用戶端 SDK 3 版本

本節列出已淘汰的用戶端 SDK 3 版本。

3.4.3 版本

3.4.3 版本新增了對 JCE 提供者的更新。

AWS CloudHSM 用戶端軟體

- 更新版本的一致性。

PKCS #11 程式庫

- 更新版本的一致性。

OpenSSL 動態引擎

- 更新版本的一致性。

JCE 提供者

- 將 log4j 更新至 2.17.0 版本。

Windows (CNG 和 KSP 提供者)

- 更新版本的一致性。

3.4.2 版本

3.4.2 版本新增了對 JCE 提供者的更新。

AWS CloudHSM 用戶端軟體

- 更新版本的一致性。

PKCS #11 程式庫

- 更新版本的一致性。

OpenSSL 動態引擎

- 更新版本的一致性。

JCE 提供者

- 將 log4j 更新至 2.16.0 版本。

Windows (CNG 和 KSP 提供者)

- 更新版本的一致性。

3.4.1 版本

3.4.1 版本新增了對 JCE 提供者的更新。

AWS CloudHSM 用戶端軟體

- 更新版本的一致性。

PKCS #11 程式庫

- 更新版本的一致性。

OpenSSL 動態引擎

- 更新版本的一致性。

JCE 提供者

- 將 log4j 更新至 2.15.0 版本。

Windows (CNG 和 KSP 提供者)

- 更新版本的一致性。

3.4.0 版本

3.4.0 版本新增了對所有元件的更新。

AWS CloudHSM 用戶端軟體

- 提升了穩定性並修正了錯誤。

PKCS #11 程式庫

- 提升了穩定性並修正了錯誤。

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。

JCE 提供者

- 提升了穩定性並修正了錯誤。

Windows (CNG 和 KSP 提供者)

- 提升了穩定性並修正了錯誤。

3.3.2 版本

3.3.2 版本解決了用戶端信息腳本的問題。

AWS CloudHSM 用戶端軟體

- 更新版本的一致性。

PKCS #11 程式庫

- 更新版本的一致性。

OpenSSL 動態引擎

- 更新版本的一致性。

JCE 提供者

- 更新版本的一致性。

Windows (CNG 和 KSP 提供者)

- 更新版本的一致性。

3.3.1 版本

3.3.1 版本新增了對所有元件的更新。

AWS CloudHSM 用戶端軟體

- 提升了穩定性並修正了錯誤。

PKCS #11 程式庫

- 提升了穩定性並修正了錯誤。

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。

JCE 提供者

- 提升了穩定性並修正了錯誤。

Windows (CNG 和 KSP 提供者)

- 提升了穩定性並修正了錯誤。

3.3.0 版本

3.3.0 版本新增了雙重要素驗證 (2FA) 和其他效能改進。

AWS CloudHSM 用戶端軟體

- 為加密管理員 (CO) 添加了 2FA 驗證。如需詳細資訊，請參閱加密管理員雙重要素驗證管理。
- 已移除對 RedHat 企業版 Linux 6 及 CentOS 6 的平台支援。如需詳細資訊，請參閱 Linux 支援。
- 新增了 CMU 的獨立版本，可與用戶端 SDK 5 或用戶端 SDK 3 搭配使用。該版本與用戶端常駐程式 3.3.0 版本隨附的 CMU 版本相同。現在您可下載 CMU，無需下載用戶端常駐程式。

PKCS #11 程式庫

- 提升了穩定性並修正了錯誤。
- 已移除對 RedHat 企業版 Linux 6 及 CentOS 6 的平台支援。如需詳細資訊，請參閱 Linux 支援。

OpenSSL 動態引擎

- 更新版本的一致性。
- 已移除對 RedHat 企業版 Linux 6 及 CentOS 6 的平台支援。如需詳細資訊，請參閱 Linux 支援。

JCE 提供者

- 提升了穩定性並修正了錯誤。
- 已移除對 RedHat 企業版 Linux 6 及 CentOS 6 的平台支援。如需詳細資訊，請參閱 Linux 支援。

Windows (CNG 和 KSP 提供者)

- 更新版本的一致性。

3.2.1 版本

3.2.1 版增加了 PKCS #11 程式庫的 AWS CloudHSM 實作與 PKCS #11 標準、新平台和其他改進之間的合規性分析。

AWS CloudHSM 用戶端軟體

- 新增對 CentOS 8、RHEL 8 和 Ubuntu 18.04 LTS 的平台支援。如需詳細資訊，請參閱 ???。

PKCS #11 程式庫

- 用戶端 SDK 3.2.1 的 PKCS #11 程式庫合規性報告
- 新增對 CentOS 8、RHEL 8 和 Ubuntu 18.04 LTS 的平台支援。如需詳細資訊，請參閱 ???。

OpenSSL 動態引擎

- 不支援 CentOS 8、RHEL 8 和 Ubuntu 18.04 LTS。如需詳細資訊，請參閱 ???。

JCE 提供者

- 新增對 CentOS 8、RHEL 8 和 Ubuntu 18.04 LTS 的平台支援。如需詳細資訊，請參閱 ???。

Windows (CNG 和 KSP 提供者)

- 提升了穩定性並修正了錯誤。

3.2.0 版本

3.2.0 版本新增了對遮罩密碼和其他效能改進的支援。

AWS CloudHSM 用戶端軟體

- 新增了在使用命令行工具時隱藏密碼的支援。如需詳細資訊，請參閱 loginHSM 和 logoutHSM (cloudhsm_mgmt_util) 和 loginHSM 和 logoutHSM (key_mgmt_util)。

PKCS #11 程式庫

- 對於先前某些未受支援的 PKCS #11 機制，新增了對軟體中大型雜湊資料的支援。如需詳細資訊，請參閱受支援的機制。

OpenSSL 動態引擎

- 提升了穩定性並修正了錯誤。

JCE 提供者

- 更新版本的一致性。

Windows (CNG 和 KSP 提供者)

- 提升了穩定性並修正了錯誤。

3.1.2 版本

3.1.2 版本新增了對 JCE 提供者的更新。

AWS CloudHSM 用戶端軟體

- 更新版本的一致性。

PKCS #11 程式庫

- 更新版本的一致性。

OpenSSL 動態引擎

- 更新版本的一致性。

JCE 提供者

- 將 log4j 更新至 2.13.3 版本

Windows (CNG 和 KSP 提供者)

- 更新版本的一致性。

3.1.1 版本

AWS CloudHSM 用戶端軟體

- 更新版本的一致性。

PKCS #11 程式庫

- 更新版本的一致性。

OpenSSL 動態引擎

- 更新版本的一致性。

JCE 提供者

- 錯誤修正與效能改進。

Windows (CNG、KSP)

- 更新版本的一致性。

3.1.0 版本

3.1.0 版本新增符合標準的 AES 金鑰包裝。

AWS CloudHSM 用戶端軟體

- 升級新要求：您的用戶端版本須跟您正在使用的任何軟體程式庫版本相符。若要升級，您必須使用可同時升級用戶端和所有程式庫的批次命令。如需詳細資訊，請參閱用戶端 SDK 3 升級。
- Key_mgmt_util (KMU) 包含下列更新：
  - 新增了兩種新的 AES 金鑰包裝方法，即：符合標準的零填補 AES 金鑰包裝和無填補 AES 金鑰包裝。如需詳細資訊，請參閱 wrapKey 和 unwrapKey。

- 使用 AES_KEY_WRAP_PAD_PKCS5 包裝金鑰時，無法使用自訂 IV 功能。如需詳細資訊，請參閱AES 金鑰包裝。

PKCS #11 程式庫

- 新增了兩種新的 AES 金鑰包裝方法，即：符合標準的零填補 AES 金鑰包裝和無填補 AES 金鑰包裝。如需詳細資訊，請參閱AES 金鑰包裝。
- 您可以設定 RSA-PSS 簽章的 salt 長度。若要瞭解如何使用此功能，請參閱上的可設定 RSA-PSS 簽章的 Salt 長度。 GitHub

OpenSSL 動態引擎

- 重大變更：使用 SHA1 的 TLS 1.0 和 1.2 加密套件不適配 OpenSSL 引擎 3.1.0。這個問題很快就會解決。
- 如果您想要在 RHEL 6 或 CentOS 6 上安裝 OpenSSL 動態引擎程式庫，請參閱有關這些作業系統上安裝之預設 OpenSSL 版本的已知問題。
- 已提升穩定性並修正錯誤

JCE 提供者

- 重大變更：為解決 Java Cryptography Extension (JCE) 相容性問題，AES 包裝和取消包裝現在會正確使用 AesWrap 演算法，而不是 AES 演算法。這代表 `Cipher.WRAP_MODE` 和 `Cipher.UNWRAP_MODE` 不再適配 AES/ECB 和 AES/CBC 機制。

  若要升級至用戶端 3.1.0 版本，您須更新程式碼。若您已有包裝金鑰，則須特別注意您用來取消包裝的機制以及 IV 預設值的變更方式。若您使用用戶端 3.0.0 版本或以下版本的包裝金鑰，則在 3.1.1 中，您須使用 AESWrap/ECB/PKCS5Padding 來取消包裝現有金鑰。如需詳細資訊，請參閱AES 金鑰包裝。
- 您可以從 JCE 提供者中列出具有相同標籤的多個金鑰。若要瞭解如何重複執行所有可用的金鑰，請參閱在上尋找所有金鑰 GitHub。
- 您可在金鑰建立期間為屬性設定更多限制值，包括為公有金鑰和私有金鑰指定不同的標籤。如需詳細資訊，請參閱支援的 Java 屬性。

Windows (CNG、KSP)

- 提升了穩定性並修正了錯誤。

E nd-of-life 版本

AWS CloudHSM 宣布不再與該服務兼容的版本的生命週期結束。為了維護您的應用程式安全，我們保留主動拒絕 end-of-life 發佈連線的權利。

- 目前沒有 end-of-life 發行版本的用戶端 SDK。

文件歷史紀錄

此主題說明 AWS CloudHSM 使用者指南的重大更新。

主題

最近更新

下表說明此文件自 2018 年 4 月後的重大變更。除了這裡所列的主要變更外，我們也會經常更新文件以改進說明內容和範例，並且反映您傳送給我們的意見回饋。如要接收重大變更的通知，請使用右上角的連結來訂閱 RSS 摘要。

如需新版本的詳細資訊，請參閱 AWS CloudHSM 用戶端 SDK 的下載

變更	描述	日期
新的 HSM 類型和叢集模式	推出新的 HSM 類型 (hsm2m.媒體) 和新的叢集模式 (非 FIP)。	2024年6月10日
新增版本	已發行的 AWS CloudHSM 用戶端版本 5.12.0。	2024年3月20日
新增版本	已發行的 AWS CloudHSM 用戶端版本 5.11.0。	2024年1月17日
新增版本	已發行的 AWS CloudHSM 用戶端版本 5.10.0。	2023 年 7 月 28 日
新增版本	發布 AWS CloudHSM 客戶端版本 5.9.0。	2023 年 5 月 23 日
新增版本	發行的 AWS CloudHSM 用戶端版本 5.8.0。	2023 年 3 月 16 日

新增版本	發布 AWS CloudHSM 客戶端版本 5.7.0。	2022 年 11 月 16 日
新增版本	發布 AWS CloudHSM 客戶端版本 5.6.0。	2022 年 9 月 1 日
新增版本	發布 AWS CloudHSM 客戶端版本 5.5.0。	2022 年 5 月 13 日
新增版本	發布 AWS CloudHSM 客戶端版本 5.4.2。	2022 年 3 月 18 日
新增版本	發布 AWS CloudHSM 客戶端版本 5.4.1。	2022 年 2 月 10 日
新增版本	發布了適用於視窗平台的 AWS CloudHSM JCE 提供程序 5.4.0 版本。	2022 年 2 月 1 日
新增版本	發布了 AWS CloudHSM 客戶端版本 5.4.0，該版本為所有 Linux 平台添加了對 JCE 提供程序的初始支持。	2022 年 1 月 28 日
新增版本	發布 AWS CloudHSM 客戶端版本 5.3.0。	2022 年 1 月 3 日
新增版本	發布 AWS CloudHSM 客戶端版本 3.4.4。	2022 年 1 月 3 日
新增版本	發行 AWS CloudHSM 客戶端版本 3.4.3。	2021 年 12 月 20 日
新增版本	發布 AWS CloudHSM 客戶端版本 3.4.2。	2021 年 12 月 15 日
新增版本	發布 AWS CloudHSM 客戶端版本 3.4.1。	2021 年 12 月 10 日

新增版本	發布 AWS CloudHSM 客戶端版本 5.2.1。	2021 年 10 月 4 日
新增版本	發布 AWS CloudHSM 客戶端版本 3.4.0。	2021 年 8 月 25 日
新增版本	發布 AWS CloudHSM 客戶端版本 5.2.0。	2021 年 8 月 3 日
新增版本	發布 AWS CloudHSM 客戶端版本 3.3.2。	2021 年 7 月 2 日
新增版本	發布 AWS CloudHSM 客戶端版本 5.1.0。	2021 年 6 月 1 日
新增版本	發布 AWS CloudHSM 客戶端版本 3.3.1。	2021 年 4 月 26 日
新增版本	發布 AWS CloudHSM 客戶端版本 5.0.1。	2021 年 4 月 8 日
新增版本	發布的 AWS CloudHSM 客戶端版本 5.0.0。	2021 年 3 月 12 日
已新增的內容	新增了介面 VPC 端點，這是一項 AWS 功能，可讓您在 VPC 之間建立私有連線， AWS CloudHSM 而無需透過網際網路、NAT 裝置、VPN 連線或連線存取 AWS Direct Connect 。	2021 年 2 月 10 日
新增版本	發布 AWS CloudHSM 客戶端 3.3.0 版本。	2021 年 2 月 3 日
已新增的內容	新增管理備份保留功能，這項功能可自動刪除舊備份。	2020 年 11 月 18 日

已新增的內容	已新增符合性報告，AWS CloudHSM 用於分析 PKCS #11 標準的 PKCS #11 程式庫的用戶端 SDK 3.2.1 實作。	2020 年 10 月 29 日
新增版本	發布 AWS CloudHSM 客戶端版本 3.2.1。	2020 年 10 月 8 日
已新增的內容	已新增說明 AWS CloudHSM中金鑰同步處理設定的文件。	2020 年 9 月 1 日
新增版本	發布 AWS CloudHSM 客戶端版本 3.2.0。	2020 年 8 月 31 日
新增版本	發布 AWS CloudHSM 客戶端 3.1.2 版本。	2020 年 7 月 30 日
新增版本	發布 AWS CloudHSM 客戶端 3.1.1 版本。	2020 年 6 月 3 日
新增版本	發布 AWS CloudHSM 客戶端版本 3.1.0。	2020 年 5 月 21 日
新增版本	發布 AWS CloudHSM 客戶端 3.0.1 版本。	2020 年 4 月 20 日
新增版本	發布了 AWS CloudHSM 用於視窗服務器平台的客戶端 3.0.0 版本。	2019 年 10 月 30 日
新增版本	發布的 AWS CloudHSM 客戶端版本 3.0.0 適用於所有平台，除了視窗。	2019 年 10 月 22 日
新增版本	發布 AWS CloudHSM 客戶端 2.0.4 版本。	2019 年 8 月 26 日
新增版本	發行的 AWS CloudHSM 用戶端版本 2.0.3。	2019 年 5 月 13 日

新增版本	發布 AWS CloudHSM 客戶端 2.0.1 版本。	2019 年 3 月 21 日
新增版本	發布的 AWS CloudHSM 客戶端版本 2.0.0。	2019 年 2 月 6 日
已新增的區域支援	新增 AWS CloudHSM 對歐盟 (斯德哥爾摩) 和 AWS GovCloud (美國東部) 區域的支援。	2018 年 12 月 19 日
新增版本	發布 AWS CloudHSM 客戶端版本 1.1.2 視窗。	2018 年 11 月 20 日
更新已知問題	將新內容新增至《故障診斷》指南。	2018 年 11 月 8 日
新增版本	發行適用於 Linux 平台的 AWS CloudHSM 用戶端版本 1.1.2。	2018 年 11 月 8 日
已新增的區域支援	增加了 AWS CloudHSM 對歐盟（巴黎）和亞太區域（首爾）區域的支持。	2018 年 10 月 24 日
已新增的內容	添加了刪除和還原 AWS CloudHSM 備份的功能。	2018 年 9 月 10 日
已新增的內容	添加了自動審計日誌交付到 Amazon CloudWatch 日誌。	2018 年 8 月 13 日
已新增的內容	新增跨區域複製 AWS CloudHSM 叢集備份的功能。	2018 年 7 月 30 日
已新增的區域支援	增加了 AWS CloudHSM 對歐盟（倫敦）區域的支持。	2018 年 6 月 13 日

已新增的內容	增加了 AWS CloudHSM 客戶端和庫支持 Amazon Linux 2，紅帽企業 Linux 6，紅帽企業 Linux（RHEL）7，CentOS 6，CentOS 7 和 Ubuntu 16.04 LTS。	2018 年 5 月 10 日
新增版本	添加了一個視窗 AWS CloudHSM 客戶端。	2018 年 4 月 30 日

舊版更新

下表說明 2018 年 AWS CloudHSM 之前版本的重要變更。

變更	描述	日期
新內容	新增加密管理員 (CO) 的規定人數身分驗證 (M/N 存取控制)。如需詳細資訊，請參閱使用 CloudHSM 管理公用程式 (CMU) 管理規定人數身分驗證 (M/N 個存取控制)。	2017 年 11 月 9 日
更新	新增有關使用 key_mgmt_util 命令列工具的文件。如需詳細資訊，請參閱 key_mgmt_util 命令參考。	2017 年 11 月 9 日
新內容	新增 Oracle 透明資料加密。如需詳細資訊，請參閱 Oracle 資料庫加密。	2017 年 10 月 25 日
新內容	新增 SSL 卸載。如需詳細資訊，請參閱 SSL/TLS 卸載。	2017 年 10 月 12 日
新指南	此版本介紹 AWS CloudHSM	2017 年 8 月 14 日

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。