aws

用户指南

AWS CloudHSM

AWS CloudHSM: 用户指南

Table of Contents

什么是 AWS CloudHSM？

AWS CloudHSM 将 AWS 云的优势与硬件安全模块 (HSM) 的安全性相结合。硬件安全模块 (HSM) 是一种计算设备，可处理加密操作并提供加密密钥的安全存储。借助 AWS CloudHSM，您可以完全控制 AWS 云中的高可用性 HSM，这些密码具有低延迟访问权限，并拥有可自动执行 HSM 管理（包括备份、配置、配置和维护）的安全信任根。

AWS CloudHSM 为客户提供多种好处：

访问 FIPS 与非 FIPS 集群

AWS CloudHSM 提供两种模式的集群：FIPS 和非 FIPS。在 FIPS 模式下，只能使用联邦信息处理标准 (FIPS) 批准的密钥和算法。非 FIPS 模式提供支持的所有密钥和算法，无论 FIPS 是否获得批准。 AWS CloudHSM有关更多信息，请参阅 AWS CloudHSM 集群模式和 HSM 类型。

HSM 是通用型、单租户，已通过 FIPS 140-2 级三级验证，适用于处于 FIPS 模式的集群

AWS CloudHSM 使用通用的 HSM，与为应用程序预先确定算法和密钥长度的完全托管的 AWS 服务相比，灵活性更高。我们提供符合标准的、单租户的 HSM，并且已通过 FIPS 140-2 三级验证，适用于处于 FIPS 模式的集群。对于使用案例超出 FIPS 140-2 三级验证限制的客户， AWS CloudHSM 还提供非 FIPS 模式的集群。请参阅AWS CloudHSM 集群了解更多信息。

AWS 无法洞察 E2E 加密

由于您的数据平面已经 end-to-end (E2E) 加密且对 AWS 不可见，因此您可以控制自己的用户管理（在 IAM 角色之外）。获得这种控制权，也需要一定的代价，即与使用托管 Amazon Web Service 相比，您需要承担更多责任。

完全控制您的密钥、算法以及应用程序开发

AWS CloudHSM 让您可以完全控制所使用的算法和密钥。您可以生成、存储、导入、导出、管理和使用加密密钥（包括会话密匙、令牌密匙、对称密钥对和非对称密钥对）。此外， AWS CloudHSM SDK 让您可以完全控制应用程序开发、应用程序语言、线程以及应用程序的实际存在位置。

将您的加密工作负载迁移至云端

迁移使用公钥加密标准 #11 (PKCS #11)、Java 加密扩展 (JCE)、Cryptography API：下一代 (CNG) 或密钥存储提供商 (KSP) 的公钥基础设施的客户只需对应用程序进行较少的更改即可迁移到 AWS CloudHSM 。

要详细了解你可以做什么 AWS CloudHSM，请参阅以下主题。准备好开始使用时 AWS CloudHSM，请参阅开始使用。

> **Note**
>
> 如果您需要用于创建和控制加密密钥的托管服务，但不希望或不需要操作您自己的 HSM，请考虑使用 AWS Key Management Service。
> 如果您想要一种管理支付 HSM 和云端支付处理的密钥，建议您使用AWS 支付加密。

内容

AWS CloudHSM 用例

AWS CloudHSM 可以用来实现各种各样的目标。本主题中的内容概述了你可以做什么 AWS CloudHSM。

实现监管合规

需要符合企业安全标准的企业可以使用 AWS CloudHSM 来管理保护高度机密数据的私钥。提供的 HSM AWS CloudHSM 已通过 FIPS 140-2 3 级认证，符合 PCI DSS。此外， AWS CloudHSM 它符合 PCI PIN 标准并符合 PCI-3DS 标准。有关更多信息，请参阅 合规。

加密和解密数据

AWS CloudHSM 用于管理保护高度机密的数据、传输中的加密和静态加密的私钥。此外，还 AWS CloudHSM 提供与多个加密 SDK 的符合标准的集成。

使用私钥和公钥对文档进行签名与验证

加密过程中，使用私钥对文档签名，使收件人可通过公钥验证您（而不是其他人）是否确实发送了文档。用于 AWS CloudHSM 创建专为此目的设计的非对称公钥和私钥对。

通过 HMAC 和 CMAC 对消息进行身份验证

加密过程中，密码消息身份验证代码 (CMAC) 和 HMAC 散列消息认证码可用于身份验证，并确保通过不安全网络发送的消息的完整性。使用 AWS CloudHSM，您可以安全地创建和管理支持 HMAC 和 CMAC 的对称密钥。

利用 AWS CloudHSM 和的好处 AWS Key Management Service

客户可以在经过 FIPS 140-2 Level 3 认证的单租户环境中合并 AWS CloudHSM 和存储密钥材料，同时还可以获得密钥管理、扩展和云集成的优势。AWS KMS AWS KMS有关如何执行此操作的详细信息，AWS Key Management Service 开发者指南中的 AWS CloudHSM 密钥存储。

为网络服务器分流 SSL/TLS 处理

为通过互联网安全地发送数据，网络服务器使用公私密钥对和 SSL/TLS 公钥证书建立 HTTPS 会话。此过程涉及大量的 Web 服务器计算，但是您可以通过将部分计算负担转移到集群来减轻计算负担，同时提供额外的安全性。 AWS CloudHSM 有关使用设置 SSL/TLS 卸载的信息，请参阅。AWS CloudHSMSSL/TLS 分载

启用透明数据加密 (TDE)

透明数据加密（TDE）用于数据库文件加密。使用 TDE，数据库软件可以先对数据进行加密，然后再将其存储在磁盘上。通过将 TDE 主加密密钥存储在您 AWS CloudHSM的 HSM 中，可以提高安全性。有关使用设置 Oracle TDE 的信息 AWS CloudHSM，请参阅Oracle Database 加密。

管理证书颁发机构 (CA) 的私有密钥

证书颁发机构 (CA) 是受信任的实体，它颁发将公钥绑定至身份（个人或组织）的数字证书。要操作 CA，您必须通过保护签署由 CA 颁发的证书的私有密钥来维持信任。您可以将此类私钥存储在 AWS CloudHSM 集群中，然后使用 HSM 执行加密签名操作。

生成随机数

生成随机数来创建加密密钥是在线安全的核心。 AWS CloudHSM 可用于在您控制的 HSM 中安全生成随机数字，并且只有您自己可见。

如何 AWS CloudHSM 运作

本主题概述了在 HSM 中用于安全加密数据和执行加密操作的基本概念和架构。 AWS CloudHSM 在您自己的亚马逊虚拟私有云 (VPC) 中运行 Amazon Private Cloud。在使用之前 AWS CloudHSM，您需要先创建集群，向其中添加 HSM，创建用户和密钥，然后使用客户端 SDK 将 HSM 与应用程序集成。完成此操作后，您可以使用客户端 SDK 日志 AWS CloudTrail、审计日志和 Amazon CloudWatch 进行监控 AWS CloudHSM。

了解 AWS CloudHSM基本概念以及它们如何协同工作以帮助保护您的数据。

主题

AWS CloudHSM 集群

让各个 HSM 以同步、冗余和高度可用的方式协同工作可能很困难， AWS CloudHSM 但是通过在集群中提供硬件安全模块 (HSM)，可以为您带来繁重的工作。集群是 AWS CloudHSM 保持同步的单个 HSM 的集合。当您在集群中的一个 HSM 上执行任务或操作时，该集群中的其他 HSM 将自动保持最新。

AWS CloudHSM 提供两种模式的集群：FIPS 和非 FIPS。在 FIPS 模式下，只能使用联邦信息处理标准 (FIPS) 批准的密钥和算法。非 FIPS 模式提供支持的所有密钥和算法，无论 FIPS 是否获得批准。AWS CloudHSM AWS CloudHSM 还提供两种类型的 HSM：hs m1.medium 和 hsm2m.med ium。有关每种 HSM 类型和集群模式之间差异的详细信息，请参阅AWS CloudHSM 集群模式和 HSM 类型。

为实现可用性、持久性和可扩展性目标，您可以跨多个可用区设置集群中的 HSM 数量。您可以创建一个拥有 1 到 28 个 HSM 的集群（默认限制为每个AWS 区域每个 AWS 账户 6 个 HSM）。您可以将 HSM 放置在一个区域的不同可用 AWS 区中。向一个集群添加多个 HSM 可提供更高的性能。跨可用区分布集群可提供冗余和高可用性。

有关集群的更多信息，请参阅 管理 AWS CloudHSM 集群。

要创建集群，请参阅 开始使用。

HSM 用户

与大多数 AWS 服务和资源不同，您不使用 AWS Identity and Access Management (IAM) 用户或 IAM 策略来访问集群中的资源。相反，您可以直接在集群中的 HSM 上使用 HSM 用户。 AWS CloudHSM

HSM 用户与 IAM 用户不同。拥有正确证书的 IAM 用户可通过 AWS API 与资源交互来创建 HSM。由于 AWS 无法知悉 E2E 加密，因此您必须使用 HSM 用户凭证对 HSM 操作进行身份验证，原因是凭证直接在 HSM 上生效。HSM 通过您定义和管理的凭证，对每个 HSM 用户进行身份验证。每个 HSM 用户都有一个类型，用于确定该用户在 HSM 上可以执行的操作。每个 HSM 通过凭证（您通过 CloudHSM CLI 定义的）对 HSM 用户进行身份验证。

如果您使用的是 之前的 SDK 版本系列，则将使用 CloudHSM 管理实用程序 (CMU)。

HSM 密钥

AWS CloudHSM 用于在 AWS CloudHSM 集群的单租户 HSM 中安全生成、恢复和管理加密密钥。密钥可以是对称的或非对称，也可能是单个会话密钥（临时密钥）或长期令牌密钥（永久密钥），也可以从 AWS CloudHSM 导出和导入至 AWS CloudHSM。密钥还可用于完成常见的加密任务与功能：

- 使用对称和非对称加密算法执行加密数据签名与签名验证。
- 配合使用加密哈希函数来计算消息摘要和基于哈希的消息身份验证代码 (HMAC)。
- 包装并保护其他密钥。
- 以加密方式访问安全随机数据。

集群可以拥有的最大密钥数取决于集群中 HSM 的类型。例如，hsm2m.medium 存储的密钥比 hsm1（中等）还要多。有关比较，请参见AWS CloudHSM 配额。

此外， AWS CloudHSM 请遵循密钥使用和管理的一些基本原则：

许多密钥类型和算法可供您选择

> 为了允许您自定义自己的解决方案， AWS CloudHSM 提供了许多密钥类型和算法供您选择。算法支持一系列密钥大小。有关更多信息，请参阅每个 AWS CloudHSM 客户端 SDK 的属性和机制页面。

如何管理密钥

> AWS CloudHSM 密钥通过 SDK 和命令行工具进行管理。有关如何通过这些工具管理密钥的信息，请参阅 管理中的密钥 AWS CloudHSM 和 以下方面的最佳实践 AWS CloudHSM。

密钥所有者

在中 AWS CloudHSM，创建密钥的加密用户 (CU) 拥有该密钥。所有者可以使用 key share 和 key unshare 命令，与其他 CU 共享和取消共享密钥。有关更多信息，请参阅 使用 CloudHSM CLI 共享和取消共享密钥。

访问和使用可通过属性加密进行控制

AWS CloudHSM 允许您使用基于属性的加密，这是一种加密形式，允许您使用密钥属性来控制谁可以根据策略解密数据。

客户端软件开发工具包

使用时 AWS CloudHSM，您可以使用AWS CloudHSM 客户端软件开发套件 (SDK) 执行加密操作。AWS CloudHSM 客户端 SDK 包括：

- 公有密钥加密标准 #11 (PKCS #11)
- JCE 提供程序
- OpenSSL 动态引擎
- 加密 API：适用于 Microsoft Windows 的新一代 (CNG) 和密钥存储提供程序 (KSP)

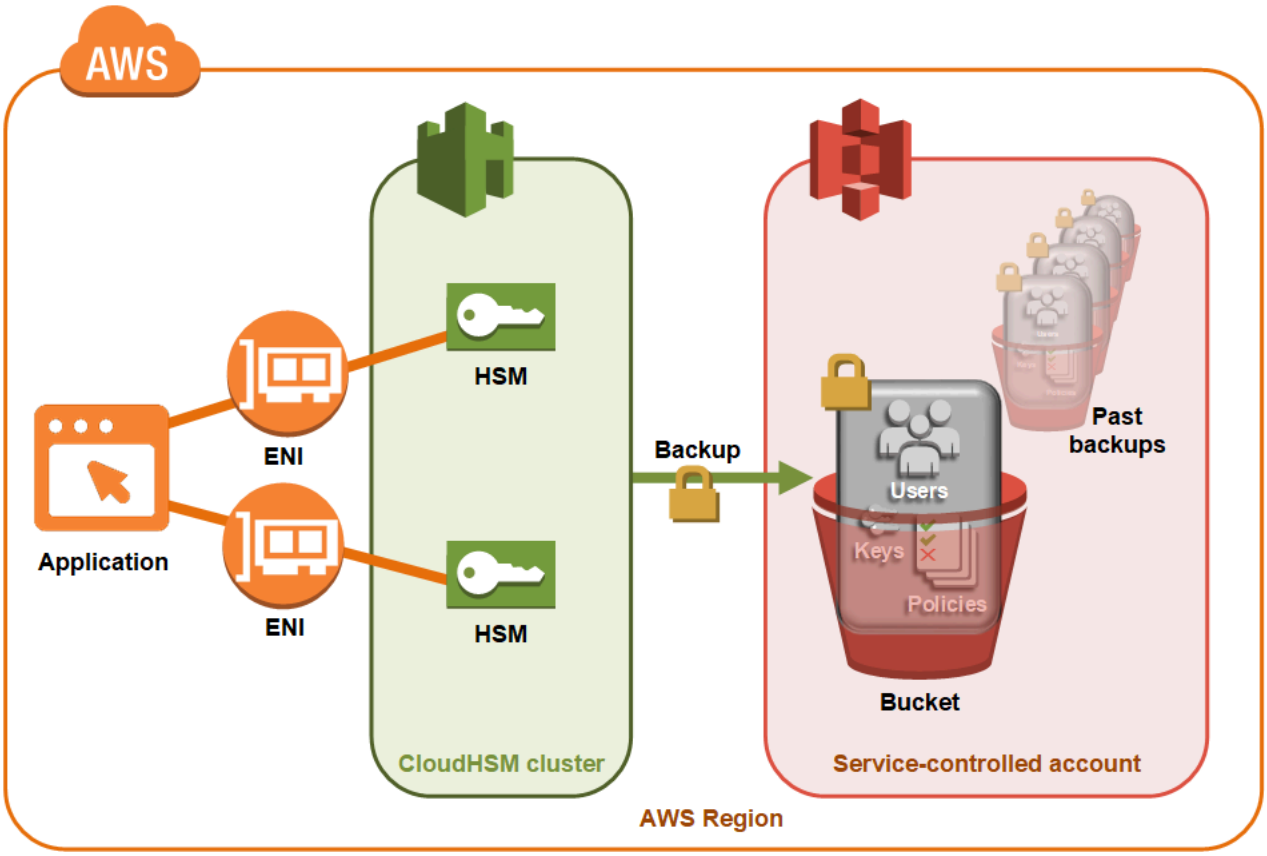
您可以在 AWS CloudHSM 集群中使用任何或全部这些 SDK。编写您的应用程序代码，以使用这些软件开发工具包在您的 HSM 中执行加密操作。要了解哪些平台和 HSM 类型支持每个 SDK，请参阅 客户端软件开发工具包 5 支持的平台

不仅要使用软件开发工具，还需要使用实用工具和命令行工具来配置应用程序的证书、策略和设置。有关更多信息，请参阅AWS CloudHSM 命令行工具。

有关安装和使用客户端软件开发工具包或客户端连接安全的更多信息，请参阅 客户端软件开发工具包 和 E nd-to-end 加密。

AWS CloudHSM 集群备份

AWS CloudHSM 定期备份集群中的用户、密钥和策略。备份安全、耐用，并可按预测时间表进行更新。下述为集群中各类备份的关系。



有关使用此备份的更多信息，请参阅 管理备份。

安全性

从 HSM AWS CloudHSM 进行备份时，HSM 会先对其所有数据进行加密，然后再将其发送到。AWS CloudHSM数据绝不会以明文形式离开 HSM。此外，无法通过解密备份， AWS 因为 AWS 无法访问用于解密备份的密钥。有关更多信息，请参阅 集群备份安全性。

持久性

AWS CloudHSM 将备份存储在服务控制的亚马逊简单存储服务 (Amazon S3) 存储桶中，该存储桶与您的集群位于同一区域。备份的持久性等级为 99.999999999%，与存储在 Amazon S3 中的任何对象都相同。

区域

有关支持的区域的信息 AWS CloudHSM，请参阅或AWS CloudHSM区域表中的AWS 一般参考区域和终端节点。

AWS CloudHSM 可能不适用于给定区域的所有可用区。但是，这不应影响性能，因为会 AWS CloudHSM 自动在集群中的所有 HSM 之间进行负载平衡。

与大多数 AWS 资源一样，集群和 HSM 也是区域资源。您不能在各区域中重复使用或扩展集群。您必须执行入门 AWS CloudHSM中列出的所有必需步骤才能在新区域中创建集群。

出于灾难恢复目的， AWS CloudHSM 允许您将 AWS CloudHSM 集群的备份从一个区域复制到另一个区域。有关更多信息，请参阅 AWS CloudHSM 集群备份。

定价

使用 AWS CloudHSM，您可以按小时付款，无需长期承诺或预付款。有关更多信息，请参阅 AWS 网站上的AWS CloudHSM 定价。

入门 AWS CloudHSM

以下主题可帮助您创建、初始化和激活集 AWS CloudHSM 群。在您完成这些过程后，您即可管理用户、管理集群并使用包含的软件库来执行加密操作。

内容

创建 IAM 管理组

作为最佳实践，请勿使用您的 AWS 账户根用户 与之互动 AWS，包括 AWS CloudHSM。而是使用 AWS Identity and Access Management (IAM) 创建 IAM 用户、IAM 角色或联合用户。按照一节创建 IAM 用户和管理员组中的步骤创建管理员群组并将AdministratorAccess策略附加到该群组。然后，创建新的管理员用户并将该用户添加到该组中。根据需要向组添加其他用户。您添加的每个用户都将从该组继承AdministratorAccess策略。

另一种最佳做法是创建一个仅具有运行所需权限的 AWS CloudHSM 管理员组 AWS CloudHSM。根据需要向此组添加单独用户。每个用户将继承已附加到该组的有限权限，而不是全部的 AWS 访问权限。以下客户管理的政策 AWS CloudHSM部分包含您应附加到 AWS CloudHSM 管理员组的策略。

AWS CloudHSM 为您的 AWS 账户定义服务关联角色。服务关联角色当前定义的权限允许您的账户记录 AWS CloudHSM 事件。该角色可以由您自动创建， AWS CloudHSM 也可以由您手动创建。您无法修改该角色，但可以将其删除。有关更多信息，请参阅 的服务相关角色 AWS CloudHSM。

创建 IAM 用户和管理员组

首先创建一个 IAM 用户及其管理员组。

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 账户

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，将接到一通电话，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为安全最佳实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS Management Console在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的以根用户身份登录。
2. 为您的根用户启用多重身份验证 (MFA)。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录网址。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

有关您可以附加 AWS CloudHSM 到 IAM 用户组的策略示例，请参阅的身份和访问管理 AWS CloudHSM。

创建虚拟私有云（VPC）

如果您还没有虚拟私有云（VPC），请按照本主题中的步骤创建一个虚拟私有云（VPC）。

> **Note**
>
> 按照这些步骤将创建公有子网和私有子网。

创建 VPC

1. 通过 https://console.aws.amazon.com/vpc/ 打开 Amazon VPC 控制台。
2. 在导航栏上，使用区域选择器选择当前支持的AWS 区域之一。 AWS CloudHSM
3. 选择创建 VPC 按钮。
4. 对于要创建的资源，选择 VPC 等。
5. 对于名称标签自动生成，键入一个可识别的名称（如 **CloudHSM**）。
6. 保留所有其他选项的默认值。
7. 选择创建 VPC。
8. 创建 VPC 后，选择查看 VPC 以查看您刚才所创建的 VPC。

创建集群

集群是单个 HSM 的集合。 AWS CloudHSM 同步每个群集中的 HSM，使其作为逻辑单元运行。 AWS CloudHSM 提供两种类型的 HSM：hs m1.medium 和 hsm2m.med ium。创建集群时，您可以选择两个集群中的哪一个将位于您的集群中。有关每种 HSM 类型和集群模式之间差异的详细信息，请参阅AWS CloudHSM 集群模式和 HSM 类型。

创建集群时， AWS CloudHSM 会代表您为该集群创建一个安全组。此安全组控制对集群中的 HSM 的网络访问。它仅允许来自安全组中的 Amazon Elastic Compute Cloud (Amazon EC2) 实例的入站连接。默认情况下，安全组不包含任何实例。稍后，您可以启动客户端实例和配置集群的安全组以允许与 HSM 的通信和连接。

> **Important**
>
> 创建集群时， AWS CloudHSM 会创建一个名 AWSServiceRoleForCloudHSM为的服务相关角色。如果 AWS CloudHSM 无法创建角色或角色尚不存在，则可能无法创建集群。有关更多信息，请参阅 解决集群创建失败问题。有关服务相关角色的更多信息，请参阅的服务相关角色 AWS CloudHSM。

可以通过 AWS CloudHSM 控制台、AWS Command Line Interface (AWS CLI) 或 AWS CloudHSM API 创建集群。

> **Note**
>
> 有关集群参数和 API 的详细信息，请参阅 AWS CLI 命令参考create-cluster中的。

创建集群 (控制台)

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 在导航栏上，使用区域选择器选择当前支持的AWS 区域之一。 AWS CloudHSM
3. 选择创建集群。
4. 在集群配置部分中，执行以下操作：

   a. 对于 VPC，选择您在 创建虚拟私有云（VPC） 中创建的 VPC。

   b. 对于可用区，在可用区旁边选择您创建的私有子网。

   > **Note**
   >
   > 即使给定可用区 AWS CloudHSM 不支持，性能也不应受到影响，因为集群中的所有 HSM AWS CloudHSM 会自动进行负载平衡。请参阅中的AWS CloudHSM 区域和终端节点 AWS 一般参考，以查看对可用区的支持 AWS CloudHSM。

   c. 对于 HSM 类型，请选择可以在您的集群中创建的 HSM 类型以及所需的集群模式。要查看每个区域所支持的 HSM 类型，请参阅 AWS CloudHSM 定价计算器。

   > **Important**
   >
   > 集群创建后，无法更改 HSM 类型和集群模式。有关哪种类型和模式适合您的用例的信息，请参阅AWS CloudHSM 集群模式和 HSM 类型。

   d. 对于集群源，请指定是要创建新集群还是要从现有备份中恢复集群。

   - 非 FIPS 模式下的集群备份只能用于恢复处于非 FIPS 模式的集群。
   - FIPS 模式下的集群备份只能用于还原处于 FIPS 模式的集群。

5. 选择 下一步。
6. 指定服务的备份保留期。

> **Note**
>
> 接受 90 天的默认保留期或键入一个介于 7 到 379 天之间的新值。该服务将自动删除此集群中早于您在此处所指定的值的备份。您以后可以更改此值。有关更多信息，请参阅 配置备份保留。

7. 选择下一步。
8. (可选) 键入标签键和一个可选标签值。要向集群添加多个标签，请选择 添加标签。
9. 选择审核。
10. 检查您的集群配置，然后选择 创建集群。

创建集群 (AWS CLI)

- 在命令提示符下，运行 create-cluster 命令。指定 HSM 实例类型、备份保留期和计划在其中创建 HSM 的子网 ID。使用您创建的私有子网的子网 ID。每个可用区仅指定一个子网。

```
$ 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，则默认为 FIPS 模式。要创建非 FIPS 集群，必须包含以下参数：--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
> ```

创建集群 (AWS CloudHSM API)

- 发送 CreateCluster 请求。指定 HSM 实例类型、备份保策略和计划在其中创建 HSM 的子网 ID。使用您创建的私有子网的子网 ID。每个可用区仅指定一个子网。

如果您尝试创建集群失败，可能与 AWS CloudHSM 服务相关角色的问题有关。有关解决故障的帮助，请参阅解决集群创建失败问题。

查看集群安全组

创建集群时，使用名称 AWS CloudHSM 创建安全组cloudhsm-cluster-*clusterID*-sg。此安全组包含一个预配置的 TCP 规则，允许集群安全组内通过端口 2223-2225 进行的入站和出站通信。此 SG 允许您的 EC2 实例使用您的 VPC 与集群中的 HSM 通信。

> **Warning**
>
> - 请勿删除或修改预配置的 TCP 规则，该规则填充在集群安全组中。此规则可以防止连接问题以及对 HSM 的未经授权的访问。
> - 集群安全组可以防止对 HSM 进行未经授权的访问。可访问安全组中的实例的任何人均可访问您的 HSM。大多数操作需要用户登录到 HSM。但可以将 HSM 清零而无需进行身份验证，这可能会销毁密钥材料、证书和其他数据。在发生这种情况时，在最新备份后创建或修

改的数据将会丢失且无法恢复。要防止未经授权的访问，请确保仅受信任的管理员能够修改或访问默认安全组中的实例。

在下一步骤中，您可以启动一个 Amazon EC2 实例并将它连接到 HSM，方式是向它附加集群安全组。

启动 Amazon EC2 客户端实例

要与 AWS CloudHSM 集群和 HSM 实例进行交互并进行管理，您必须能够与 HSM 的弹性网络接口进行通信。执行这一操作的最简单方式是使用与您的集群相同的 VPC 中的 EC2 实例。也可以使用以下 AWS 资源连接到您的集群：

- Amazon VPC 对等
- AWS Direct Connect
- VPN 连接

> **Note**
>
> 本指南提供了如何将 EC2 实例连接到 AWS CloudHSM 集群的简化示例。有关安全网络配置的最佳实践，请参阅安全访问您的集群。

本 AWS CloudHSM 文档通常假设您在创建集群的同一 VPC 和可用区 (AZ) 中使用 EC2 实例。

创建 EC2 实例

1. 通过以下网址打开 EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 选择启动实例。从下拉菜单中，选择启动实例。
3. 在名称字段中，输入您的 EC2 实例 名称。
4. 在应用程序和操作系统映像（亚马逊系统映像）部分，选择与 CloudHSM 支持的平台对应的亚马逊机器映像（AMI）。有关更多信息，请参阅 客户端软件开发工具包 5 支持的平台。
5. 在实例类型部分中，选择一个实例类型。
6. 在密钥对部分，使用现有密钥对或选择创建新密钥对并完成以下步骤：

    a. 对于密钥对名称，输入密钥对的名称。

b. 对于密钥对类型，选择一个密钥对。
c. 对于私有密钥文件格式，选择私有密钥文件格式。
d. 选择创建密钥对。
e. 下载并保存私有密钥文件。

> **Important**
>
> 这是您保存私有密钥文件的唯一机会。将文件下载并保存在安全的位置。在启动实例时必须提供密钥对的名称。此外，每次连接实例时都必须提供相应的私有密钥，并选择设置时创建的密钥对。

7. 在网络设置中，选择编辑。
8. 对于网络，选择您之前为集群创建的 VPC。
9. 对于子网，选择您为 VPC 创建的公有子网。
10. 对于自动分配公有 IP，选择启用。
11. 选择选择现有安全组。
12. 在通用安全组中，从下拉菜单中选择默认安全组。
13. 在配置存储中，使用下拉菜单选择存储配置。
14. 在摘要窗口中，选择 Launch instance（启动实例）。

> **Note**
>
> 完成此步骤将开始 EC2 实例创建过程。

有关创建 Linux Amazon EC2 客户端的更多信息，请参阅 Amazon EC2 Linux 实例入门。有关连接到正在运行的客户端的信息，请参见以下主题：

- 使用 SSH 连接到 Linux 实例
- 使用 PuTTY 从 Windows 连接到 Linux 实例

《Amazon EC2 用户指南》包含有关设置和使用 Amazon EC2 实例的详细说明。以下列表概述了适用于 Linux 和 Windows Amazon EC2 客户端的文档：

- 要创建 Linux Amazon EC2 客户端，请参阅 Amazon EC2 Linux 实例入门。

有关连接到正在运行的客户端的信息，请参见以下主题：

- 使用 SSH 连接到 Linux 实例
- 使用 PuTTY 从 Windows 连接到 Linux 实例

- 要创建 Windows Amazon EC2 客户端，请参阅 Amazon EC2 Windows 实例入门。有关连接到 Windows 客户端的更多信息，请参阅连接到 Windows 实例。

> **Note**
>
> 您的 EC2 实例可以运行本指南中包含的所有 AWS CLI 命令。如果未安装 AWS CLI ，您可以从 AWS Command Line Interface 中下载它。如果您使用的是 Windows，则可以下载并运行 64 位或 32 位 Windows 安装程序。如果您使用的是 Linux 或 macOS，则可以使用 pip 安装 CLI。

配置客户端 Amazon EC2 实例安全组

在启动 Amazon EC2 实例时，将它与默认 Amazon VPC 安全组关联。本主题介绍如何将集群安全组与 EC2 实例关联。此关联允许在您的 EC2 实例上运行的 AWS CloudHSM 客户端与您的 HSM 通信。要将您的 EC2 实例连接到您的 AWS CloudHSM 集群，您必须正确配置 VPC 默认安全组并将集群安全组与该实例关联。

修改默认安全组

您需要修改默认安全组以允许 SSH 或 RDP 连接，以便您可以下载和安装客户端软件，并与 HSM 进行交互。

修改默认安全组

1. 通过以下网址打开 EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 选择实例（正在运行），然后选中要安装 AWS CloudHSM 客户端的 EC2 实例旁边的复选框。
3. 在安全选项卡下，选择名为默认的安全组。
4. 在页面顶部，选择操作，然后选择编辑入站规则。
5. 选择添加规则。
6. 在类型中，执行下列操作之一：
    - 对于 Windows Server Amazon EC2 实例，选择 RDP。将自动填充端口范围 3389。

- 对于 Linux Amazon EC2 实例，选择 SSH。将自动填充端口范围 22。

7. 对于任一选项，请将源设置为我的 IP，以使您与您的 Amazon EC2 实例通信。

> **Important**
>
> 不要指定 0.0.0.0/0 作为端口范围，以避免允许任何人访问您的实例。

8. 选择保存。

将 Amazon EC2 实例连接到集 AWS CloudHSM 群

您必须将集群安全组附加到 EC2 实例，以便 EC2 实例能够与集群中的 HSM 进行通信。集群安全组包含一个预配置规则，允许通过端口 2223-2225 进行入站通信。

将 EC2 实例连接到集 AWS CloudHSM 群

1. 通过以下网址打开 EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 选择 “实例（正在运行）”，然后选中要安装 AWS CloudHSM 客户端的 EC2 实例对应的复选框。
3. 在页面顶部，依次选择操作、联网和更改安全组。
4. 选择组名与您的集群 ID 匹配的安全组，例如 `cloudhsm-cluster-`*`clusterID`*`-sg`。
5. 选择添加安全组。
6. 选择保存。

> **Note**
>
> 您最多可将五个安全组分配给 Amazon EC2 实例。如果您已达到最大限制，则必须修改 Amazon EC2 实例的默认安全组和集群安全组：
>
> 在默认安全组中，执行以下操作：
>
> - 添加入站规则以借助 TCP 协议通过端口 2223-2225 允许来自集群安全组的流量。
>
> 在集群安全组中，执行以下操作：
>
> - 添加入站规则以借助 TCP 协议通过端口 2223-2225 允许来自默认安全组的流量。

创建 HSM

创建集群后，您就可以创建 HSM 了。不过，集群必须先处于未初始化状态，然后您才能在集群中创建 HSM。要确定集群的状态，请在AWS CloudHSM 控制台中查看集群页面，使用运行describe-clusters命令或在 AWS CloudHSM API 中发送DescribeClusters请求。 AWS CLI 可以通过 AWS CloudHSM 控制台、AWS CLI 或 AWS CloudHSM API 创建 HSM。

创建 HSM (控制台)

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 选择要创建 HSM 的集群 ID 旁边的单选按钮。
3. 选择操作。从下拉菜单中，选择初始化。
4. 为正在创建的 HSM 选择可用区 (AZ)。
5. 选择创建。

创建 HSM (AWS CLI)

- 在命令提示符下，运行 create-hsm 命令。指定之前创建的集群的集群 ID，并为 HSM 指定可用区。以 `us-west-2a`、`us-west-2b` 等形式指定可用区。

```
$ 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"
    }
}
```

创建 HSM (AWS CloudHSM API)

- 发送 CreateHsm 请求。指定之前创建的集群的集群 ID，并为 HSM 指定可用区。

在创建集群和 HSM 后，您可以选择验证 HSM 的身份或直接继续初始化集群。

验证集群的 HSM 的身份和真实性 (可选)

要初始化您的群集，请签署由群集的第一个 HSM 生成的证书签名请求 (CSR)。在执行此操作之前，您可能需要验证 HSM 的身份和真实性。

> **Note**
>
> 此过程是可选过程。但是，它仅在初始化群集之前有效。初始化群集后，您不能使用此过程来获取证书或验证 HSM。

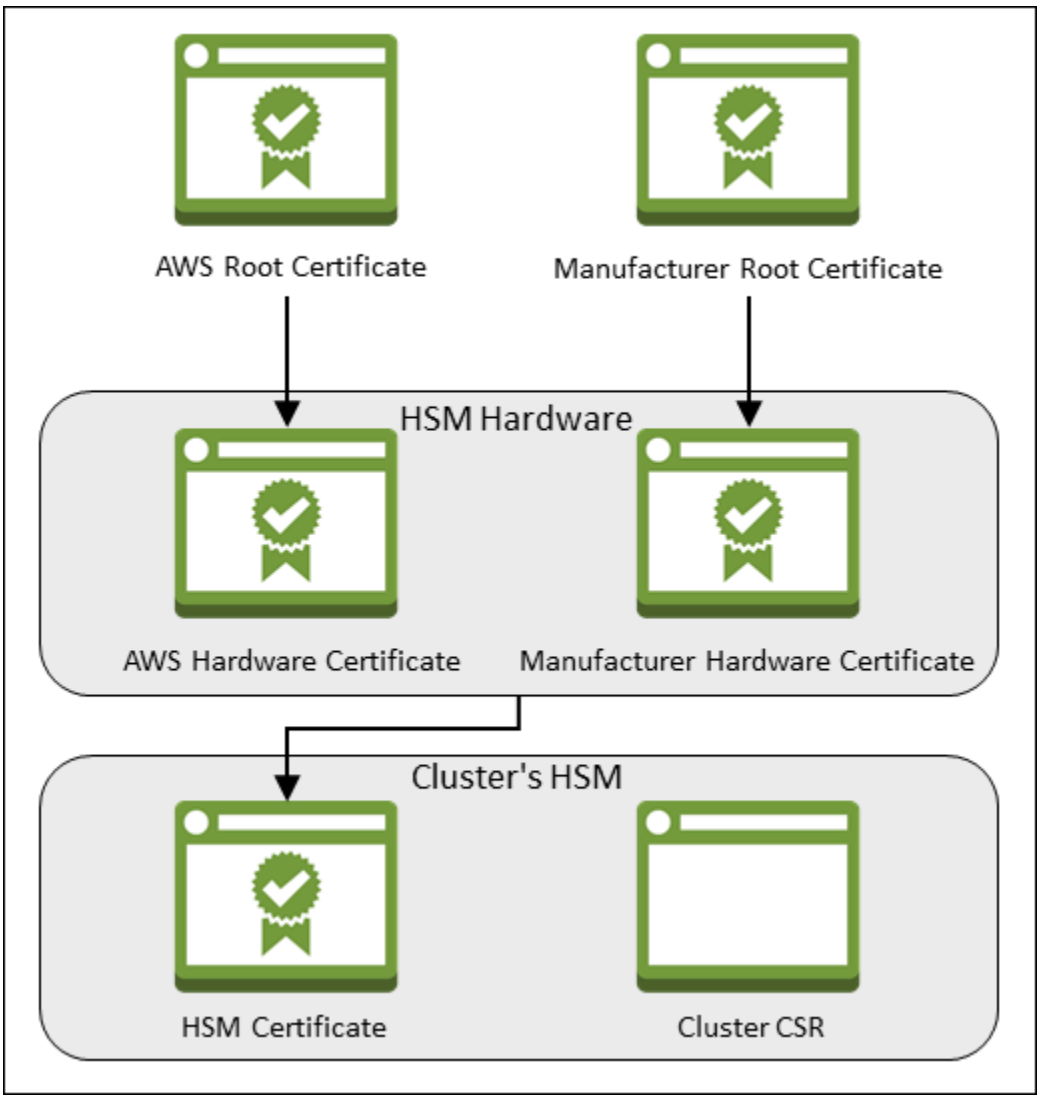
主题

概述

要验证群集的第一个 HSM 的身份，请完成以下步骤：

1. 获取证书和 CSR - 在此步骤中，您将从 HSM 获取三个证书和一个 CSR。您还将获得两个根证书，一个来自 HSM 硬件制造商 AWS CloudHSM ，一个来自 HSM 硬件制造商。
2. 验证证书链 — 在此步骤中，您将构造两个证书链，一个指向 AWS CloudHSM 根证书，另一个指向制造商根证书。然后，您使用这些证书链验证 HSM 证书，以确定该证书 AWS CloudHSM 和硬件制造商都证明 HSM 的身份和真实性。
3. 比较公有密钥 - 在此步骤中，您将提取并比较 HSM 证书和集群 CSR 中的公有密钥以确保二者相同。这将使您坚信 CSR 是由真实可信的 HSM 生成的。

下图显示了 CSR、证书及其相互关系。后续列表定义了每个证书。



AWS 根证书

AWS CloudHSM这是根证书。

制造商根证书

这是硬件制造商的根证书。

AWS 硬件证书

AWS CloudHSM 在将 HSM 硬件添加到队列时创建了此证书。此证书声明自己 AWS CloudHSM 拥有硬件。

制造商硬件证书

HSM 硬件制造商在制造 HSM 硬件时已创建此证书。此证书声明制造商创建了该硬件。

HSM 证书

HSM 证书在您创建群集中的第一个 HSM 时由经 FIPS 验证的硬件生成。此证书声明 HSM 硬件已创建 HSM。

群集 CSR

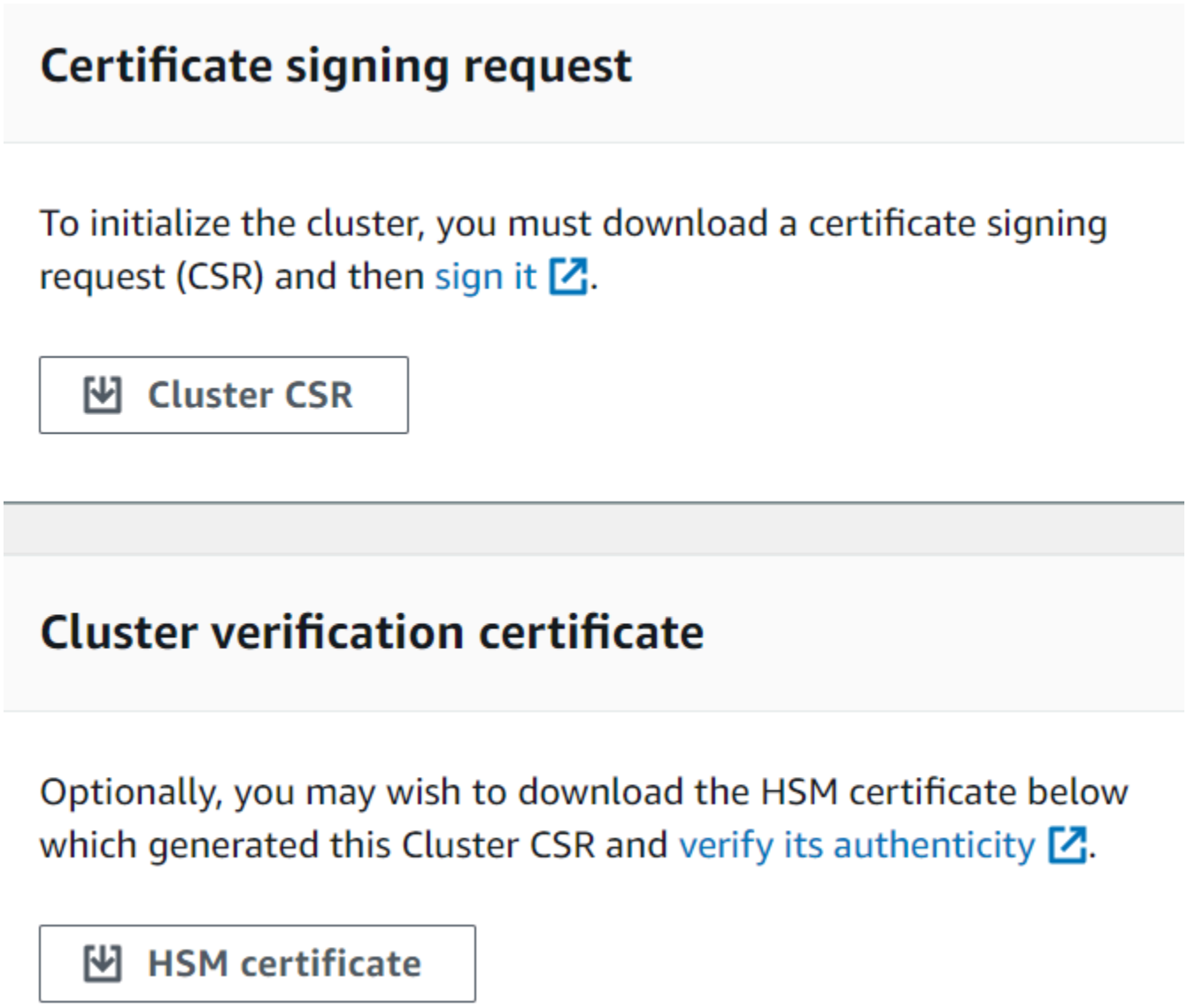
第一个 HSM 将创建群集 CSR。在对群集 CSR 进行签名时，您将声明群集。然后，您可以使用已签名 CSR 初始化群集。

从 HSM 获取证书

要验证 HSM 的身份和真实性，请先获取 1 个 CSR 和 5 个证书。您可以从 HSM 获得其中三个证书，您可以使用AWS CloudHSM 控制台、AWS Command Line Interface (AWS CLI) 或 AWS CloudHSM API 来完成。

获取 CSR 和 HSM 证书 (控制台)

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 选中要验证的 HSM 的集群 ID 旁边的单选按钮。
3. 选择操作。从下拉菜单中，选择初始化。
4. 如果您未完成旨在创建 HSM 的 上一步 操作，则请为正在创建的 HSM 选择可用区（AZ）。然后选择 创建。
5. 在证书和 CSR 准备就绪后，您将看到用于下载它们的链接。



6. 选择用于下载和保存 CSR 和证书的每个链接。要简化后续步骤，请将所有文件保存到同一目录中，并使用默认文件名。

获取 CSR 和 HSM 证书 (AWS CLI)

- 在命令提示符处，运行 describe-clusters 命令四次，每次提取 CSR 和不同的证书，然后将其保存到相应文件中。

  a. 发出以下命令以提取群集 CSR。将 *<cluster ID>* 替换为之前创建的群集的 ID。

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

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

b. 发出以下命令以提取 HSM 证书。将 *<cluster ID>* 替换为之前创建的群集的 ID。

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

c. 发出以下命令提取 AWS 硬件证书。将 *<cluster ID>* 替换为之前创建的群集的 ID。

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

d. 发出以下命令以提取制造商硬件证书。将 *<cluster ID>* 替换为之前创建的群集的 ID。

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

获取 CSR 和 HSM 证书 (AWS CloudHSM API)

- 发送 DescribeClusters 请求，然后从响应中提取并保存 CSR 和证书。

获取根证书

按照以下步骤获取 AWS CloudHSM 和制造商的根证书。将根证书文件保存到包含 CSR 和 HSM 证书文件的目录。

获取 AWS CloudHSM 和制造商根证书

1. 下载 AWS CloudHSM 根证书：AWS_CloudHSM_Root-G1.zip

2. 下载适合您的 HSM 类型的制造商根证书：

   - hsm1.medium 制造商根证书：liquid_security_certificate.zip
   - hsm2m.medium 制造商根证书：liquid_security_certificate.zip

   > **Note**
   >
   > 要从登录页面下载每份证书，请使用以下链接：
   >
   > - hsm1.medium 制造商根证书的登录页面
   > - hsm2m.medium 制造商根证书的登录页面
   >
   > 您可能需要右键单击 Download Certificate 链接，然后选择 Save Link As... 才能保存证书文件。

3. 在下载文件后，提取 (解压缩) 其内容。

验证证书链

在此步骤中，您将构造两个证书链，一个指向 AWS CloudHSM 根证书，另一个指向制造商根证书。然后使用 OpenSSL 通过每个证书链验证 HSM 证书。

要创建证书链，请打开 Linux shell。您需要 OpenSSL (在大多数 Linux shell 中可用)，并且需要已下载的根证书和 HSM 证书文件。但是，您不需要执行 AWS CLI 此步骤，也无需将外壳与您的 AWS 账户关联。

使用 AWS CloudHSM 根证书验证 HSM 证书

1. 导航到您将下载的根证书和 HSM 证书文件保存到的目录。以下命令假设所有证书都在当前目录中并使用默认文件名。

   使用以下命令按顺序创建包含 AWS 硬件证书和 AWS CloudHSM 根证书的证书链。将 *<cluster ID>* 替换为之前创建的群集的 ID。

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

2. 使用以下 OpenSSL 命令通过 AWS 证书链验证 HSM 证书。将 *<cluster ID>* 替换为之前创建的群集的 ID。

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

使用制造商根证书验证 HSM 证书

1. 使用以下命令创建一个证书链，其中按该顺序包含制造商硬件证书和制造商根证书。将 *<cluster ID>* 替换为之前创建的群集的 ID。

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

2. 使用以下 OpenSSL 命令通过制造商证书链验证 HSM 证书。将 *<cluster ID>* 替换为之前创建的群集的 ID。

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

提取和比较公有密钥

使用 OpenSSL 提取并比较 HSM 证书和群集 CSR 中的公有密钥以确保二者相同。

要比较公有密钥，请使用您的 Linux shell。你需要 OpenSSL，它在大多数 Linux 外壳中都可用，但此步骤 AWS CLI 不需要。外壳无需与您的 AWS 账户关联。

提取和比较公有密钥

1. 使用以下命令可从 HSM 证书中提取公有密钥。

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

2. 使用以下命令可从群集 CSR 中提取公有密钥。

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

3. 使用以下命令可比较公有密钥。如果公有密钥相同，以下命令不会生成任何输出。

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

在验证 HSM 的身份和真实性后，请继续初始化集群。

初始化集群

完成以下主题中的步骤以初始化您的 AWS CloudHSM 集群。

> **Note**
>
> 初始化集群之前，请审核流程，从而验证 HSM 的身份和真实性。此过程是可选的，并且仅在初始化集群之前有效。初始化集群后，您不能使用此过程来获取证书或验证 HSM。

主题

- 获取集群 CSR
- 签署 CSR
- 初始化集群

获取集群 CSR

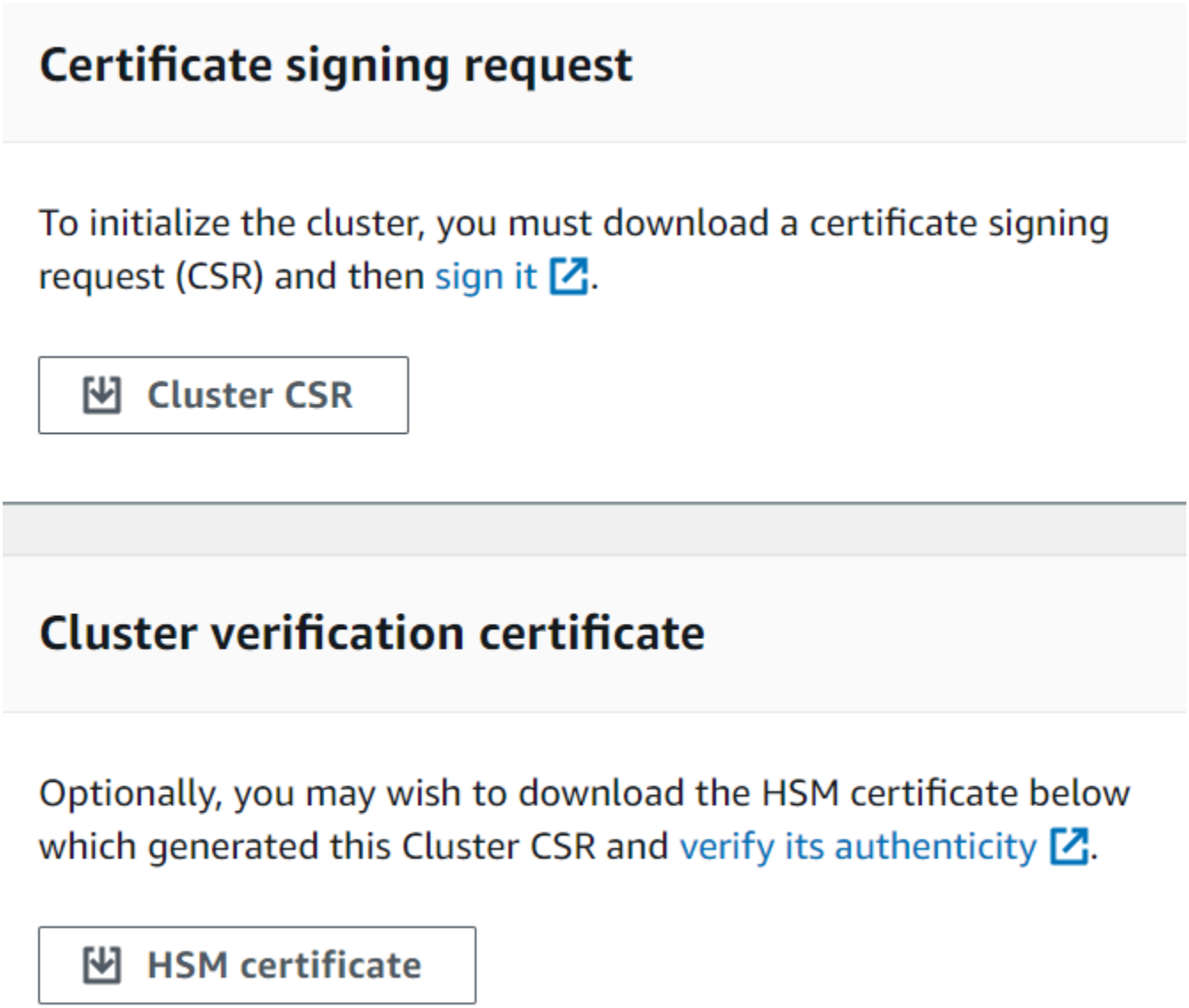
在初始化集群之前，您必须先下载并签署由集群的第一个 HSM 生成的证书签名请求 (CSR)。如果您按照步骤操作来验证您集群的 HSM 的身份，则您已有 CSR 并可以签署。否则，请立即使用AWS CloudHSM 控制台、AWS Command Line Interface (AWS CLI) 或 AWS CloudHSM API 获取 CSR。

> **Important**
>
> 要初始化集群，您的信任锚必须符合 RFC 5280 并满足以下要求：
>
> - 如果使用 X509v3 扩展，则必须存在 X509v3 基本约束扩展。
> - 信任锚必须是自签名证书。

- 扩展值不能相互冲突。

获取 CSR (控制台)

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 选中要验证的 HSM 的集群 ID 旁边的单选按钮。
3. 选择操作。从下拉菜单中，选择初始化。
4. 如果您未完成旨在创建 HSM 的 上一步 操作，则请为正在创建的 HSM 选择可用区（AZ）。然后选择创建。
5. 在 CSR 准备就绪时，您将看到用于下载它的链接。



6. 选择集群 CSR 以下载并保存 CSR。

获取 CSR (AWS CLI)

- 在命令提示符处，运行以下 describe-clusters 命令，该命令将提取 CSR 并将其保存到文件。将 *<cluster ID>* 替换为您之前创建的群集的 ID。

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

获取企业社会责任 (AWS CloudHSM API)

1. 发送 DescribeClusters 请求。
2. 从响应中提取并保存 CSR。

签署 CSR

目前，您必须创建自签名的签名证书，并使用它来签署您的集群的 CSR。您不需要执行 AWS CLI 此步骤，也无需将外壳与您的 AWS 账户关联。要签署 CSR，您必须执行以下操作：

1. 完成上一节（请参阅 获取集群 CSR）。
2. 创建私有密钥。
3. 使用私有密钥创建签名证书。
4. 签署您的集群 CSR。

创建私有密钥

> **Note**
>
> 对于生产集群，您将要创建的密钥应使用可信的随机掩码以安全的方式创建。建议您使用安全的异地和离线 HSM 或等效对象。安全地存储密钥。密钥用于建立集群身份以及您对集群所包含的 HSM 的唯一控制权。

在开发和测试期间，您可以使用任何方便的工具 (如 OpenSSL) 来创建和签署集群证书。以下示例为您展示了如何创建密钥。使用密钥创建自签名证书 (见下文) 之后，您应该以安全的方式将其存储起来。要登录您的 AWS CloudHSM 实例，证书必须存在，但私钥不存在。

使用以下命令创建私有密钥。初始化 AWS CloudHSM 集群时，必须使用 RSA 2048 证书或 RSA 4096 证书。

```
$ 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:
```

使用私有密钥创建自签名证书

您用来为生产集群创建私有密钥的可信硬件还应提供软件工具，以使用该密钥生成自签名证书。以下示例使用 OpenSSL 和您在上一步中创建的私有密钥创建签名证书。该证书的有效期为 10 年 (3652 天)。阅读屏幕上的说明，并按照提示操作。

```
$ 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 []:
```

完成后，此命令将创建一个名为 customerCA.crt 的证书文件。将此证书放在要连接到 AWS CloudHSM 集群的每台主机上。如果您为文件指定了不同的名称或将其存储在与主机根不同的路径，则应相应地编辑客户端配置文件。使用您刚创建的证书和私有密钥，在下一步中签署集群证书签名请求 (CSR)。

签署集群 CSR

您用来为生产集群创建私有密钥的可信硬件还应提供工具，以使用该密钥签署 CSR。以下示例使用 OpenSSL 来签署集群的 CSR。该示例使用您在上一步中创建的私有密钥和自签名证书。

```
$ 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:
```

完成后，此命令将创建一个名为 *<cluster ID>*_CustomerHsmCertificate.crt 的文件。在初始化集群时，将此文件用作签名证书。

初始化集群

使用已签名 HSM 证书和您的签名证书来初始化集群。您可以使用AWS CloudHSM 控制台AWS CLI、或 AWS CloudHSM API。

初始化群集（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 选中要验证的 HSM 的集群 ID 旁边的单选按钮。
3. 选择操作。从下拉菜单中，选择初始化。
4. 如果您未完成旨在创建 HSM 的 上一步 操作，则请为正在创建的 HSM 选择可用区（AZ）。然后选择 创建。
5. 在下载证书签名请求页面上，选择下一步。如果下一步不可用，请先选择一个 CSR 或证书链接。然后选择下一步。

6. 在签署证书签名请求（CSR）页面上，选择下一步。
7. 在上传证书页面上，执行以下操作：

   a. 在集群证书旁边，选择上传文件。然后找到并选中您之前签署的 HSM 证书。如果您完成了以上部分中的步骤，请选择名为 *<cluster ID>*_CustomerHsmCertificate.crt 的文件。

   b. 在颁发证书旁边，选择上传文件。然后，选择您的签名证书。如果您完成了以上部分中的步骤，请选择名为 customerCA.crt 的文件。

   c. 选择上传并初始化。

初始化群集 (AWS CLI)

- 在命令提示符下，运行 initialize-cluster 命令。提供以下项：
  - 您之前创建的集群的 ID。
  - 您之前签署的 HSM 证书。如果您完成了以上部分中的步骤，它保存在名为 *<cluster ID>*_CustomerHsmCertificate.crt 的文件中。
  - 您的签名证书。如果您完成了以上部分中的步骤，签名证书保存在名为 customerCA.crt 的文件中。

```
$ 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."
}
```

初始化集群 (AWS CloudHSM API)

- 使用以下内容发送 InitializeCluster 请求：
  - 您之前创建的集群的 ID。
  - 您之前签署的 HSM 证书。
  - 您的签名证书。

安装和配置 CloudHSM CLI

要与 AWS CloudHSM 集群中的 HSM 进行交互，你需要 CloudHSM CLI。

任务

- 安装 AWS CloudHSM 命令行工具

安装 AWS CloudHSM 命令行工具

连接到您的客户端实例并运行以下命令来下载和安装 AWS CloudHSM 命令行工具。有关更多信息，请参阅 启动 Amazon EC2 客户端实例。

使用以下命令下载和安装 CloudHSM CLI。

Amazon Linux 2

x86_64 架构上的 Amazon Linux 2：

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

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

ARM64 架构上的 Amazon Linux 2：

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

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

Amazon Linux 2023

基于 x86_64 架构的亚马逊 Linux 2023：

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

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

基于 ARM64 架构的亚马逊 Linux 2023：

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

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

CentOS 7 (7.8+)

x86_64 架构上的 CentOS 7：

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

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

RHEL 7 (7.8+)

x86_64 架构上的 RHEL 7：

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

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

RHEL 8 (8.3+)

x86_64 架构上的 RHEL 8：

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

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

RHEL 9 (9.2+)

x86_64 架构上的 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
```

ARM64 架构上的 RHEL 9：

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

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

Ubuntu 20.04 LTS

x86_64 架构上的 Ubuntu 20.04 LTS：

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

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

Ubuntu 22.04 LTS

x86_64 架构上的 Ubuntu 22.04 LTS：

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

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

ARM64 架构上的 Ubuntu 22.04 LTS：

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

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

Windows Server 2016

对于 x86_64 架构上的 Windows Server 2016，请 PowerShell 以管理员身份打开并运行以下命令：

```
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

对于 x86_64 架构上的 Windows Server 2019，请 PowerShell 以管理员身份打开并运行以下命令：

```
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
```

使用以下命令配置 CloudHSM CLI。

引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

引导适用于客户端软件开发工具包 5 的 Windows EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

激活集群

激活集群时，AWS CloudHSM 集群的状态会从已初始化变为活动。然后，您可以管理硬件安全模块 (HSM) 用户并使用 HSM。

Important

在激活集群之前，您必须首先将颁发证书复制到连接到集群的每个 EC2 实例上平台的默认位置（您在初始化集群时所创建的颁发证书）。

Linux

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

Windows

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

放置颁发证书后，安装 CloudHSM CLI 并在您的第一个 HSM 上运行 cluster activate 命令。您会注意到集群中第一个 HSM 上的管理员账户具有未激活的管理员角色。这是一个临时角色，仅存在于集群激活之前。激活集群后，未激活的管理员角色将变为管理员。

激活集群

1. 连接到之前已启动的客户端实例。有关更多信息，请参阅 启动 Amazon EC2 客户端实例。您可以启动 Linux 实例或 Windows Server。
2. 在交互模式下运行 CloudHSM CLI。

   Linux

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

   Windows

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

3. （可选）使用 user list 命令显示现有用户。

```
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 命令设置初始管理员密码。

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

我们建议您在密码工作表上记下新密码。切勿丢失该工作表。我们建议您打印一份密码工作表，用它来记录您的重要 HSM 密码，然后将其存储在安全位置。此外，建议您在安全的异地存储设施中存储此工作表的一个副本。

5. （可选）使用 user list 命令验证用户的类型是否已更改为管理员/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 命令停止 CloudHSM CLI 工具。

```
aws-cloudhsm > quit
```

有关使用 CloudHSM CLI 或 CMU 的更多信息，请参阅了解 HSM 用户和 使用 CMU 了解 HSM 用户管理。

使用新的证书和私有密钥重新配置 SSL（可选）

AWS CloudHSM 使用 SSL 证书建立与 HSM 的连接。安装客户端时包含默认的密钥和 SSL 证书。但是，您也可以创建和使用自己的密钥和 SSL 证书。请注意，您将需要在初始化*customerCA.crt*集群时创建的自签名证书 (???)。

从较高的层面上说，这是一个两步过程：

1. 首先，创建私有密钥，然后使用该密钥创建证书签名请求（CSR）。使用颁发证书（初始化集群时所创建的证书）签署 CSR。
2. 接下来，使用配置工具将密钥和证书复制到相应的目录中。

创建密钥、CSR，然后签署 CSR

客户端软件开发工具包 3 或客户端软件开发工具包 5 的步骤相同。

使用新的证书和私有密钥重新配置 SSL

1. 使用以下 OpenSSL 命令创建私有密钥：

```
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 命令创建证书签名请求 (CSR)。系统将询问您有关您的证书的一系列问题。

```
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. 使用您在初始化集群时创建的 *customerCA.crt* 证书签署 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
```

为启用自定义 SSL AWS CloudHSM

客户端软件开发工具包 3 或客户端软件开发工具包 5 的步骤不同。有关使用 configure 命令行工具的更多信息，请参阅 ???。

主题

- 适用于客户端软件开发工具包 3 的自定义 SSL
- 适用于客户端软件开发工具包 5 的自定义 SSL

适用于客户端软件开发工具包 3 的自定义 SSL

使用适用于客户端软件开发工具包 3 的配置工具启用自定义 SSL。有关适用于客户端软件开发工具包 3 的配置工具的更多信息，请参阅 ???。

在 Linux 上对客户端软件开发工具包 3 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份认证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用配置工具指定 `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. 将 customerCA.crt 证书添加到信任存储。创建此证书主题名的哈希。这会创建一个索引以允许按该名称查找此证书。

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

创建目录。

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

使用哈希名称创建一个包含此证书的文件。

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

适用于客户端软件开发工具包 5 的自定义 SSL

使用任何客户端软件开发工具包 5 配置工具启用自定义 SSL。有关适用于客户端软件开发工具包 5 的配置工具的更多信息，请参阅 ???。

PKCS #11 library

在 Linux 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份认证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用配置工具指定 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
```

在 Windows 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份验证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用 PowerShell 解释器，使用配置工具指定`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

在 Linux 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份认证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用配置工具指定 `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

在 Linux 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份认证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用配置工具指定 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
```

在 Windows 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份验证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用 PowerShell 解释器，使用配置工具指定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

在 Linux 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份认证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用配置工具指定 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
```

在 Windows 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份验证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用 PowerShell 解释器，使用配置工具指定`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
```

构建应用程序。

使用构建应用程序并使用密钥 AWS CloudHSM。

要开始在新集群中创建和使用密钥，必须首先使用 CloudHSM 管理实用程序（CMU）创建硬件安全模块（HSM）用户。有关更多信息，请参阅了解 HSM 用户管理任务、AWS CloudHSM 命令行界面（CLI）入门以及如何管理 HSM 用户。

> **Note**
>
> 如果使用客户端软件开发工具包 3，请使用 CloudHSM 管理实用程序（CMU）而不是 CloudHSM CLI。

利用 HSM 用户，您可以登录 HSM，并通过以下任一选项创建和使用密钥：

- 使用密钥管理实用程序，即命令行工具
- 使用 PKCS #11 库构建 C 应用程序
- 使用 JCE 提供程序构建 Java 应用程序

- 使用直接从命令行中使用 OpenSSL 动态引擎
- 使用 OpenSSL 动态引擎以及 NGINX 和 Apache Web 服务器进行 TLS 分载
- 使用 CNG 和 KSP 提供商与 AWS CloudHSM 微软 Windows 服务器证书颁发机构 (CA) 配合使用
- 使用 CNG 和 KSP 提供商与 AWS CloudHSM M ic rosoft 签名工具配合使用
- 使用 CNG 和 KSP 提供程序以及互联网信息服务器（IIS）Web 服务器 进行 TLS 分载

以下方面的最佳实践 AWS CloudHSM

执行本主题中的最佳做法以有效使用 AWS CloudHSM。

内容

集群管理

创建、访问和管理 AWS CloudHSM 集群时，请遵循本节中的最佳实践。

扩展集群以应对峰值流量

有几个因素影响您的集群可以处理的最大吞吐量，包括客户端实例大小、集群大小、网络拓扑以及您的用例所需的加密操作。

首先，有关常见集群大小和配置的性能估计，请参阅主题 AWS CloudHSM 性能。我们建议您根据峰值负载预期，对集群进行负载测试，以确定当前架构是否有弹性和规模得当。

构建集群以实现高可用性

增加冗余以进行维护： AWS 可能会替换您的 HSM 进行定期维护或检测到问题。一般而言，您的集群大小至少应有 +1 冗余。例如，如果您需要两个 HSM 才能让服务在峰值时段运行，那么理想的集群大小将为三。如果您遵循与可用性相关的最佳做法，则这些 HSM 替代品不应影响您的服务。但是，在更换后的 HSM 上运行的操作可能会失败，因此必须重试。

您的 HSM 分散到多个可用区：考虑在可用区中断期间，您的服务将如何运行。 AWS 建议您将您的 HSM 分布在尽可能多的可用区内。对于具有三个 HSM 的集群，您应该将 HSM 分布至三个可用区。根据您的系统，您可能需要额外冗余。

至少有三个 HSM，以确保新生成密钥的持久性

对于需要新生成密钥持久性的应用程序，我们建议跨一个区域中的所有可用区分布至少三个 HSM 实例。

安全访问您的集群

使用私有子网限制实例访问：在 VPC 的私有子网中启动 HSM 和客户端实例。这会限制从外部访问您的 HSM。

使用 VPC 终端节点访问 API： AWS CloudHSM 数据平面旨在无需访问互联网或 AWS API 即可运行。如果您的客户端实例需要访问 AWS CloudHSM API，则可以使用 VPC 终端节点访问 API，而无需在客户端实例上访问互联网。请参阅AWS CloudHSM 和 VPC 终端节点了解更多信息。

重新配置 SSL 以保护客户端-服务器通信： AWS CloudHSM 使用 TLS 与您的 HSM 建立连接。初始化集群后，您可替换用于建立外部 TLS 连接的默认 TLS 证书和密钥。有关更多信息，请参阅 使用 SSL/TLS 卸载来提高 Web 服务器的安全性 AWS CloudHSM。

根据您的需求扩展，从而降低成本

无需预付费用即可使用 AWS CloudHSM。终止 HSM 之前，您需要为启动的每个 HSM 支付每小时费用。如果您的服务不需要持续使用 AWS CloudHSM，则可以在不需要时将 HSM 缩小（删除）至零，从而降低成本。再次需要 HSM 时，您可从备份中恢复 HSM。例如，如果您的工作负载要求您每月签名一次代码（尤其是在该月的最后一天）则可以在此前纵向扩展集群，在工作完成后通过删除 HSM 来缩减集群，然后恢复集群以在下个月底再次执行签名操作。

AWS CloudHSM 自动对集群中的 HSM 进行定期备份。稍后添加新 HSM 时， AWS CloudHSM 会将最新的备份还原到新 HSM 上，这样您就可以从离开它的地方恢复使用。要计算您的 AWS CloudHSM 架构成本，请参阅AWS CloudHSM 定价。

相关的资源：

- 备份一般概述
- 备份保留政策
- 跨 AWS 区域复制备份

HSM 用户管理

请遵循本节中的最佳实践，以有效管理 AWS CloudHSM 集群中的用户。HSM 用户与 IAM 用户不同。如果 IAM 用户和实用程序拥有带适当许可的身份识别策略，则其可通过 AWS API 与资源交互，进而创建 HSM。创建 HSM 后，您必须使用 HSM 用户凭证对 HSM 上的操作进行身份验证。有关 HSM 用户的详细指南，请参阅 在中管理 HSM 用户 AWS CloudHSM。

保护您的 HSM 用户凭证

必须妥善保护您的 HSM 用户凭证，因为 HSM 用户是可以访问您的 HSM 并对其执行加密和管理操作的实体。 AWS CloudHSM 无法访问您的 HSM 用户凭证，如果您无法访问这些凭证，则无法为您提供帮助。

至少有两名管理员防止锁定

为避免被锁定在集群外，我们建议您至少配备两名管理员，以防一个管理员密码丢失。如果发生这种情况，您可使用其他管理员重置密码。

> **Note**
>
> 客户端软件开发工具包 5 中的 管理员与客户端软件开发工具包 3 中的加密员（CO）同义。

为所有用户管理操作启用仲裁

您可通过仲裁设置最小数量的管理员，该管理员必须在操作发生前批准用户管理操作。由于管理员拥有的权限，我们建议您为所有用户管理操作启用仲裁。如果某个管理员密码被泄露，这可能会限制可能造成的影响。有关更多信息，请参阅仲裁管理。

创建多个加密用户，每个用户都有有限权限

通过将加密用户的责任分开，没有用户可以完全控制整个系统。因此，我们建议您创建多个加密用户并限制每个用户权限。方法通常是赋予不同的加密用户明显不同的责任和执行操作（例如，一位加密用户负责生成密钥，并将其与稍后使用的其他加密用户分享）。

相关的资源：

- key share
- key unshare

HSM 密钥管理

在 AWS CloudHSM中管理密钥时，遵循本部分所述的最佳做法。

选择正确密钥类型

使用会话密钥时，您的每秒交易量 (TPS) 将仅限为此密钥所在位置的一项 HSM。集群中额外的 HSM 将不会增加该密钥请求的吞吐量。如果您对同一应用程序使用令牌密钥，则将在集群中所有可用的 HSM 之间，对您的请求进行负载平衡。有关更多信息，请参阅 中的密钥同步和耐久性设置 AWS CloudHSM。

管理密钥存储限制

HSM 对每次存储在 HSM 上的令牌和会话密钥的最大数量有限制。有关存储限制的信息，请参阅 AWS CloudHSM 配额。如果应用程序需要的密钥超限，可以使用以下一个或多个策略有效管理密钥：

使用信任包装将密钥存储在外部数据存储：您可通过信任密钥包装，恢复外部数据存储中包装的所有密钥，以克服密钥存储限制。当需要使用此密钥时，您可以将该密钥作为会话密钥解包至 HSM，使用该密钥进行所需的操作，然后丢弃会话密钥。原始密钥数据仍安全地存储在您的数据存储中，以在需要时随时使用。使用信任密钥执行此操作，以最大限度地保护您的安全。

跨集群分配密钥：克服密钥存储限制的另一种策略是将密钥存储至多个集群。按此方法，您可以保留存储在每个集群中的密钥的映射。使用此映射将您的客户端请求路由到带有所需密钥的集群。有关如何从同一个客户端应用程序连接至多个集群的信息，请参阅以下主题：

- 使用 JCE 提供程序连接到多个集群
- 连接至多个 PKCS#11 插槽

管理和保护密钥包装

可以通过 EXTRACTABLE 属性将密钥标记为可提取或不可提取。默认情况下，HSM 密钥标记为可提取。

可提取密钥是允许通过密钥包装从 HSM 中导入的密钥。包装的密钥是加密的，必须使用相同的封装密钥进行解包后才能使用。在任何情况下，都不能从 HSM 中导出不可提取密钥。无法将不可提取的密钥设置为可提取。据此，务必考虑您需要密钥为可提取与否，并据此设置相应的密钥属性。

如果您需要在应用程序中进行密钥包装，则应使用信任密钥包装限制您的 HSM 用户，使其只能包装/解包已被管理员明确标记为信任的密钥。有关更多信息，请参阅 管理中的密钥 AWS CloudHSM 中的信任密钥主题。

相关资源

- 包装和解包功能

- JCE 密码函数
- 受支持的 Java 密钥属性
- CloudHSM CLI 的密钥属性

应用程序集成

请遵循本节中的最佳实践，以优化您的应用程序与集群的 AWS CloudHSM 集成的方式。

客户端软件开发工具包 引导程序

将您的客户端软件开发工具包连接至您的集群前，必须对其进行引导。将 IP 地址引导至您的集群时，我们建议尽可能使用 `--cluster-id` 参数。此方法使用集群中的所有 HSM IP 地址填充您的配置，无需追踪每个单独的地址。这样可以在 HSM 进行维护或可用区中断时，为您的应用程序初始化增加额外弹性。有关更多详细信息，请参阅引导客户端软件开发工具包。

进行身份验证，以执行操作

在中 AWS CloudHSM，您必须先对集群进行身份验证，然后才能执行大多数操作，例如加密操作。

使用 CloudHSM CLI 进行身份验证：您可使用单命令模式 或 交互模式，通过 CloudHSM CLI 进行身份验证。使用 login 命令在交互模式下进行身份验证。若要在单命令模式下进行身份验证，必须设置环境变量 `CLOUDHSM_ROLE` 和 `CLOUDHSM_PIN`。有关执行此操作的详细信息，请参阅 单命令模式。 AWS CloudHSM 建议在应用程序不使用您的 HSM 凭证时安全存储。

使用 PKCS 进行身份验证 #11: 在 PKCS #11 中，在使用 C_ 打开会话后使用 C_Login API 登录。OpenSession每个插槽（集群）只需要执行一项 C_Login。成功登录后，您可以使用 C_ 打开其他会话，OpenSession 而无需执行其他登录操作。有关向 PKCS #11 进行身份验证的示例，请参阅 PKCS #11 库的代码示例。

使用 JCE 进行身份验证： AWS CloudHSM JCE 提供程序支持隐式登录和显式登录。选择哪种方法由使用案例而定。我们建议尽可能使用隐式登录，因为如果应用程序与集群断开连接并且需要重新进行身份验证，SDK 将自动处理身份验证。使用隐式登录后，当您使用不允许控制应用代码的集成项时，可为您的应用提供凭证。有关登录方法的更多信息，请参阅 向 JCE 提供程序提供凭证。

使用 OpenSSL 身份验证：您可利用 OpenSSL 动态引擎，通过环境变量提供凭证。当不适应时，AWS CloudHSM 建议您安全存储 HSM 凭证。如果可能，您应将环境配置为系统地检索和设置这些环境变量，无需手动输入。有关使用 OpenSSL 进行身份验证的详细信息，请参阅安装 OpenSSL 动态引擎。

有效管理应用程序中的密钥

使用密钥属性控制密钥可执行的操作：生成密钥时，使用密钥属性定义一组权限，允许或拒绝对该密钥进行特定类型的操作。我们建议在生成密钥时使用最少属性完成任务。例如，也不应使用 AES 加密密钥包装 HSM 以外的密钥。有关更多信息，请参阅以下客户端软件开发工具包属性页面：

- PKCS #11 密钥属性
- JCE 密钥属性

尽可能缓存密钥对象，以最大限度减少延迟：密钥查找操作将查询集群中的每个 HSM。此操作成本高昂，并且无法随集群内的 HSM 数量扩展。

- 您可通过 PKCS #11，使用 `C_FindObjects` API 查找密钥。
- 使用 JCE，您可以使用查找密钥。 KeyStore

为了获得最佳性能， AWS 建议您在应用程序启动期间仅使用一次键查找命令（如findKey和key list），并将返回的密钥对象缓存在应用程序内存中。如果您稍后需要此密钥对象，则应从缓存中检索该对象，而非为每个操作查询此对象，这将增加显著的性能开销。

使用多线程

AWS CloudHSM 支持多线程应用程序，但是对于多线程应用程序，需要注意一些事项。

使用 PKCS #11时，您应该只初始化 PKCS #11 库（调用`C_Initialize`）一次。应对每个线程分配自己的会话 (`C_OpenSession`)。不建议在多个线程中使用同一会话。

使用 JCE 时， AWS CloudHSM 提供程序只能初始化一次。不要跨线程共享 SPI 对象实例。例如，Cipher、Signature、Digest、Mac KeyFactory 或 KeyGenerator 对象只能在自己的线程上下文中使用。

处理节流错误

以下情况下，可能会出现 HSM 节流错误：

- 您的集群未正确扩展，因此无法管理峰值流量。
- 维护事件期间，您的集群大小没有 +1 冗余。
- 可用区域中断会导致集群中的可用 HSM 数量减少。

有关如何最好地处理这种情况的信息，请参阅 HSM 节流。

为确保您的集群大小足够且不会受到限制， AWS 建议您在您的环境中使用预期的峰值流量进行负载测试。

集群操作的集成重试次数

AWS 可能会出于操作或维护原因更换您的 HSM。为了使您的应用程序能够应对此类情况， AWS 建议您对路由到集群的所有操作实施客户端重试逻辑。对于因替换而导致的操作失败，其后续重试有望成功。

实施灾难恢复策略

为了应对事件，可能需要将流量从整个集群或区域转移。以下各节描述了实现此任务的多种策略。

使用 VPC 对等互连从其他账户或地区访问您的集群：您可以利用 VPC 对等互连从其他账户或地区访问您的 AWS CloudHSM 集群。有关更多信息，请参阅 VPC Peering Guide 中的什么是 VPC 对等连接？。建立对等连接并正确配置安全组后，就可以像往常一样与 HSM IP 地址通信。

从同一个应用程序连接到多个集群：客户端 SDK 5 中的 JCE 提供程序、PKCS #11 库和 CloudHSM CLI 支持从同一个应用程序连接到多个集群。例如，您可有两个活动集群，每个集群位于不同的区域，并且您的应用程序可以同时连接至这两个集群，并在两者之间进行负载平衡，这是正常操作的一部分。如果您的应用程序未使用客户端软件开发工具包 5 (最新 SDK)，则无法从同一个应用程序连接到多个集群。或者，您可以保持其他集群的正常运行，如果出现区域性中断，可以将流量转移之另一个集群，以最大限度地减少停机时间。详情请参见相应页面：

- 连接至多个 PKCS#11 插槽
- 使用 JCE 提供程序连接到多个集群
- 使用 CloudHSM CLI 连接到多个集群

恢复备份集群：您可通过现有集群备份，创建新集群。有关更多信息，请参阅 管理 AWS CloudHSM 备份。

监控

本节介绍可用于监控集群和应用的多种机制。有关监控的更多详细信息，请参阅 监控 AWS CloudHSM。

监控客户端日志

每个客户端软件开发工具包都会写入您可监控的日志。有关日志记录的信息，请参阅 使用客户端软件开发工具包日志。

在设计为临时性的平台（例如 Amazon ECS 和）上 AWS Lambda，从文件中收集客户端日志可能很困难。在这些情况下，最佳做法是配置客户端软件开发工具包日志以将日志写入控制台。大多数服务会自动收集此输出并将其发布到 Amazon CloudWatch 日志中，供您保存和查看。

如果您在 AWS CloudHSM 客户端 SDK 之上使用任何第三方集成，则应确保将该软件包配置为将其输出也记录到控制台。此软件包可能会捕获 AWS CloudHSM 客户端 SDK 的输出并将其写入自己的日志文件中。

有关如何在应用程序中配置日志记录选项的信息，请参阅 客户端软件开发工具包 5 配置工具。

监控审核日志

AWS CloudHSM 将审核日志发布到您的 Amazon CloudWatch 账户。审核日志来自 HSM，用于跟踪某些需要审核的操作。

您可以使用审核日志来跟踪在 HSM 上调用的、任何管理命令。例如，当您注意到正在执行意外的管理操作时，可触发警报。

有关更多信息，请参阅HSM 审核日志记录的工作原理。

监视器 AWS CloudTrail

AWS CloudHSM 与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在中执行的操作的记录 AWS CloudHSM。 AWS CloudTrail 将所有 API 调用捕获 AWS CloudHSM 为事件。捕获的调用包括来自 AWS CloudHSM 控制台的调用和对 AWS CloudHSM API 操作的代码调用。

您可以使用 AWS CloudTrail 来审核向 AWS CloudHSM 控制平面发出的任何 API 调用，以确保您的账户中没有发生任何不必要的活动。

有关详细信息，请参阅 使用 AWS CloudTrail 和 AWS CloudHSM。

监控 Amazon CloudWatch 指标

您可以使用 Amazon CloudWatch 指标来实时监控您的 AWS CloudHSM 集群。指标可按区域、集群 ID 或 HSM ID 和集群 ID 分组。

使用 Amazon CloudWatch 指标，您可以配置 Amazon CloudWatch 警报，提醒您注意可能出现的任何可能影响您服务的潜在问题。我们建议配置警报，以监控以下内容：

- 接近 HSM 上的密钥限制
- 接近 HSM 上的 HSM 会话计数限制
- 接近 HSM 上的 HSM 用户计数限制
- 用于识别同步问题的 HSM 用户数或密钥数差异
- 不健康的 HSM 可以向上扩展您的集群，直到 AWS CloudHSM 可以解决问题

有关更多详细信息，请参阅使用 Amazon CloudWatch 日志和 AWS CloudHSM 审核日志。

管理 AWS CloudHSM 集群

您可以通过控制AWS CloudHSM 台、AWS 软件开发工具包或命令行工具管理 AWS CloudHSM 集群。有关更多信息，请参阅以下主题。

要创建集群，请参阅 开始使用。

集群架构

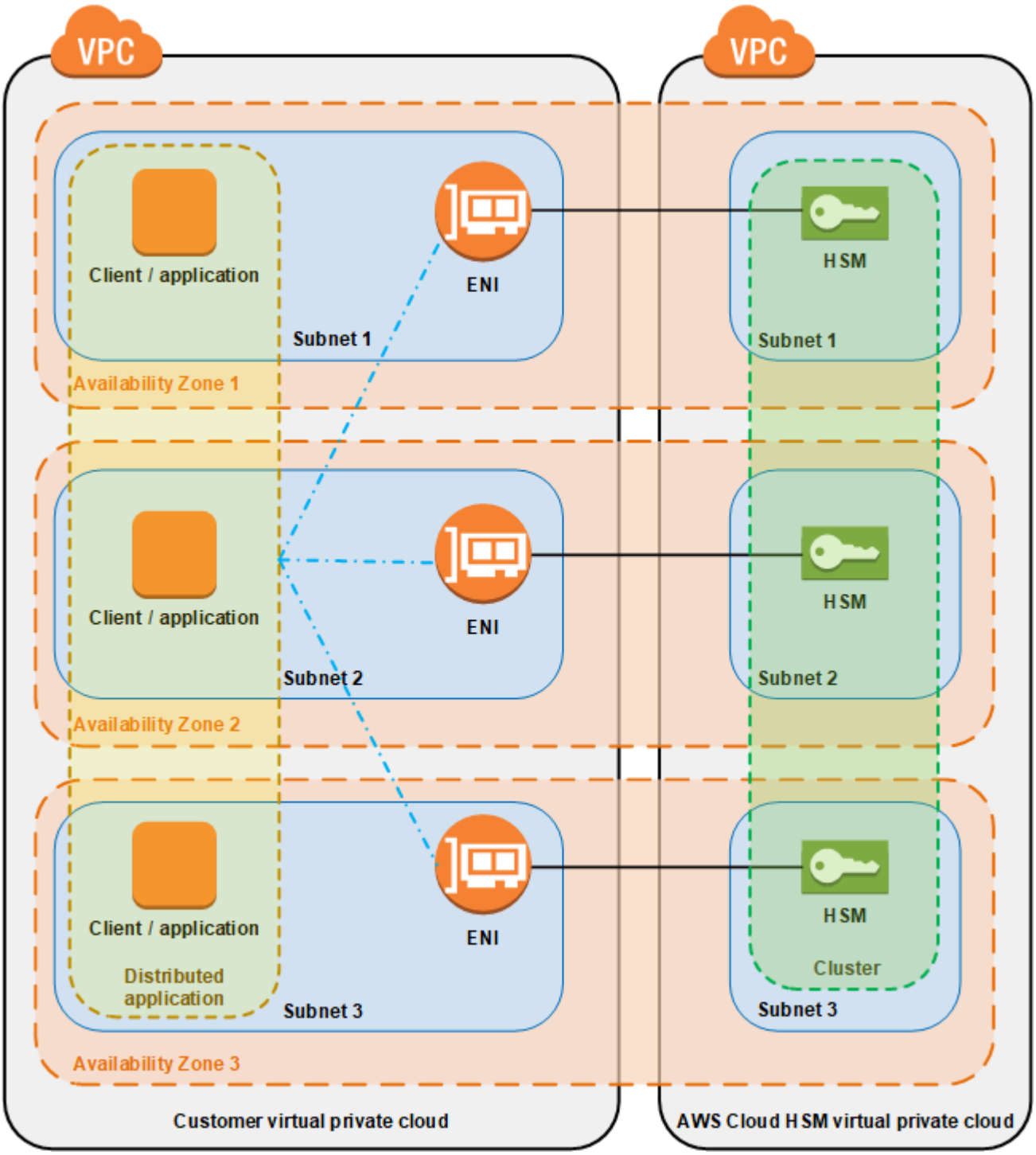
创建集群时，您可以在账户中指定亚马逊虚拟私有云 (VPC)，并在该 V AWS PC 中指定一个或多个子网。我们建议您在所选 AWS 区域的每个可用区 (AZ) 中创建一个子网。创建 VPC 时，您可创建私有子网。要了解更多信息，请参阅创建虚拟私有云（VPC）。

每当创建 HSM 时，您需要为该 HSM 指定群集和可用区。通过将 HSM 放入不同的可用区中，您可以实现冗余和高可用性，以防某个可用区不可用。

创建 HSM 时，在您 AWS 账户的指定子网中 AWS CloudHSM 放置一个弹性网络接口 (ENI)。该弹性网络接口是用于与 HSM 进行交互的接口。HSM 驻留在独立 VPC 中的一个 AWS 账户中，该账户归其 AWS CloudHSM所有。HSM 及其相应的网络接口位于同一可用区中。

要与集群中的 HSM 进行交互，您需要 AWS CloudHSM 客户端软件。通常情况下，您应该在与 HSM ENI 位于同一 VPC 中的 Amazon EC2 实例（称为客户端实例）上安装该客户端，如下图所示。但从技术上讲，此操作不是必需的；您可以在任何可以连接到 HSM ENI 的兼容计算机上安装该客户端。该客户端通过其 ENI 与您的群集中的各个 HSM 进行通信。

下图显示了一个包含三个 HSM 的 AWS CloudHSM 集群，每个 HSM 位于 VPC 的不同可用区。



集群同步

在集 AWS CloudHSM 群中， AWS CloudHSM 使各个 HSM 上的密钥保持同步。您无需执行任何操作即可使 HSM 上的密钥保持同步。要使每个 HSM 上的用户和策略保持同步，请在管理 HSM 用户之前更新 AWS CloudHSM 客户端配置文件。有关更多信息，请参阅 让 HSM 用户保持同步。

向集群添加新 HSM 时， AWS CloudHSM 会备份现有 HSM 上的所有密钥、用户和策略。然后，它将该备份还原到新 HSM 上。这将使两个 HSM 保持同步。

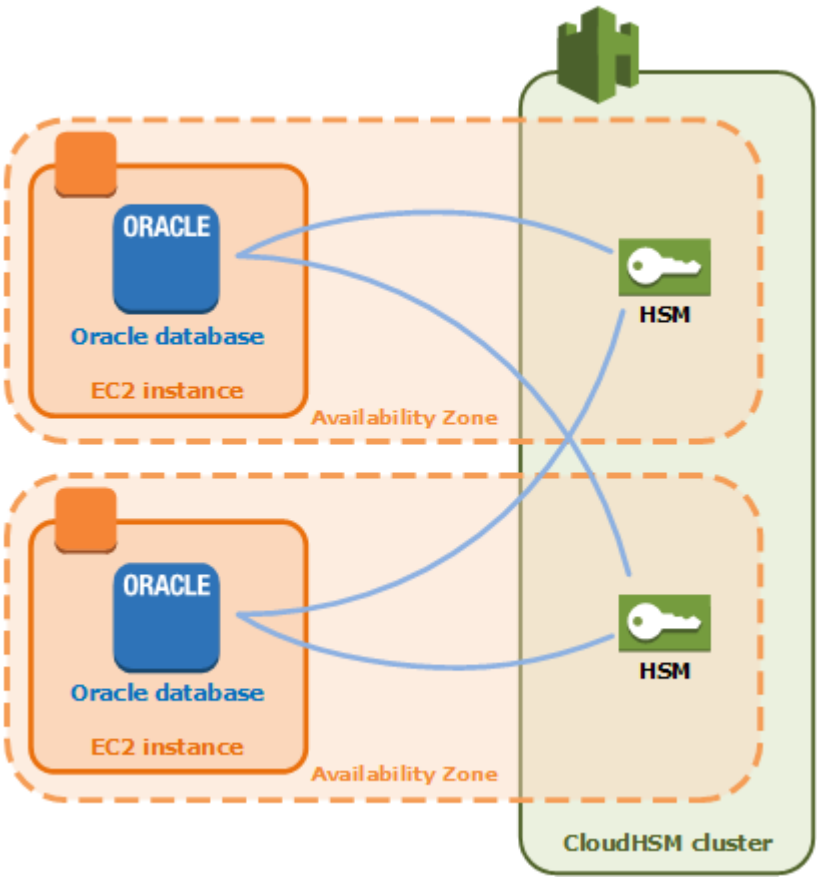
如果集群中的 HSM 无法同步，则 AWS CloudHSM 会自动对其进行重新同步。要启用此功能，请 AWS CloudHSM 使用设备用户的凭据。此用户存在于由提供的所有 HSM 上 AWS CloudHSM ，并且权限有限。该用户可以获取 HSM 上的对象的哈希，并且可以提取和插入遮蔽（加密）的对象。 AWS 无法查看或修改您的用户或密钥，并且无法使用这些密钥执行任何加密操作。

集群高可用性和负载均衡

当您创建包含多个 HSM 的 AWS CloudHSM 集群时，您会自动获得负载平衡。负载均衡意味着 AWS CloudHSM 客户端根据每个 HSM 的容量跨所有群集中的所有 HSM 分发加密操作以进行额外处理。

当您在不同的 AWS 可用区域创建 HSM 时，您将自动获得高可用性。高可用性意味着，您可以获得更高的可靠性，因为任何 HSM 均不会成为单点故障。我们建议您在每个集群中至少有两个 HSM，每个 HSM 位于一个 AWS 区域内的不同可用区。

例如，下图显示了分发到两个不同可用区的 Oracle Database 应用程序。数据库实例将其主密钥存储在集群中，每个可用区中都包含一个 HSM。 AWS CloudHSM 自动将两个 HSM 的密钥同步，以便它们可以立即访问和冗余。



AWS CloudHSM 集群模式和 HSM 类型

AWS CloudHSM 提供两种集群模式：FIPS 和非 FIPS。 AWS CloudHSM 还提供两种 HSM 类型：hsm1.medium 和 hsm2m.medi um。在决定哪种集群模式和 HSM 类型适合您的需求之前，请先查看此页面上的详细信息。

> **Note**
>
> 在 2024 年 6 月 10 日之前创建的所有集群都处于 FIPS 模式，HSM 类型为 hsm1.medium。

要查看集群的模式和 HSM 类型，请使用 desc ribe-clusters 命令。

集群模式

AWS CloudHSM 提供两种模式的集群：FIPS 和非 FIPS。在 FIPS 模式下，只能使用联邦信息处理标准 (FIPS) 批准的密钥和算法。非 FIPS 模式提供支持的所有密钥和算法，无论 FIPS 是否获得批准。AWS CloudHSM

下表列出了每种集群模式之间的主要区别：

区分功能	FIPS 模式	非 FIPS 模式
HSM 类型兼容性	适用于 hsm1.medium。	适用于 hsm2m.medium。
备份兼容性	只能用于在 FIPS 模式下备份还原集群。	只能用于在非 FIPS 模式下备份还原集群。
按键选择	支持 FIPS 批准[1]的 AWS CloudHSM 密钥。	支持既经 FIPS 批准又未经 FIPS 批准的 AWS CloudHSM 密钥。
算法	支持 FIPS 批准[1]的 AWS CloudHSM 算法。	支持既经 FIPS 批准又未经 FIPS 批准的 AWS CloudHSM 算法。
认证	兼容 FIPS 140-2、PCI PIN 和 PCI-3DS。	

[1] 有关详细信息，请参阅弃用通知。

在选择集群模式之前，请注意集群的模式（FIPS 或非 FIPS）在创建后无法更改，因此请确保根据需要选择正确的模式。

HSM 类型

除了集群模式外，还 AWS CloudHSM 提供两种 HSM 类型：hsm1.medium 和 hsm2m.medi um。每种 HSM 类型使用不同的硬件，且每个集群只能包含一种类型的 HSM。下表列出了两者的主要区别：

区分功能	hsm1.medium	hsm2m.medium
集群模式兼容性	适用于处于 FIPS 模式的集群。	目前适用于非 FIPS 模式下的集群。
备份兼容性	只能用于对 hsm1.m edium 群集进行备份恢复。	只能用于备份恢复 hsm2m.m edium 群集。

区分功能	hsm1.medium	hsm2m.medium
密钥容量	每个集群 3,300 个。	总共有 16,666 个密钥，其中非对称密钥每个集群最多有 3,333 个。
客户端软件开发工具包	支持所有客户端软件开发工具包。	支持除 CNG 和 KSP 提供商之外的所有客户端 SDK。
客户端软件开发工具包版本	与 SDK 版本 3.1.0 及更高版本兼容。	与客户端 SDK 版本 5.12.0 及更高版本兼容。
区域可用性	CloudHSM 在所有可用的地区都可用。	仅在数量有限的地区提供，其他受支持的区域即将推出。要查看此 HSM 类型可用的区域，请参阅定AWS CloudHSM价计算器。
性能	要查看每种 HSM 类型的性能，请参阅 AWS CloudHSM 性能。	
认证	兼容 FIPS 140-2、PCI DSS、PCI PIN、SOC2 和 PCI-3DS。	兼容 PCI DSS。

[1] 有关详细信息，请参阅弃用通知。

将客户端 SDK 连接到集 AWS CloudHSM 群

要使用客户端软件开发工具包 5 或客户端软件开发工具包 3 连接到集群，您必须首先做两件事情：

- 在 EC2 实例上使用颁发证书
- 将客户端软件开发工具包引导到集群

在每个 EC2 实例上使用颁发证书

在初始化集群时创建颁发证书。将颁发证书复制到连接到集群的每个 EC2 实例上平台的默认位置。

Linux

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

Windows

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

指定颁发证书的位置

使用客户端软件开发工具包 5，您可以使用配置工具指定颁发证书位置。

PKCS #11 library

在 Linux 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

在 Windows 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

OpenSSL Dynamic Engine

在 Linux 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

JCE provider

在 Linux 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

在 Windows 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

CloudHSM CLI

在 Linux 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

在 Windows 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

有关更多信息，请参阅配置工具。

有关初始化集群或创建和签名证书的更多信息，请参阅初始化集群。

引导客户端软件开发工具包

引导过程因您使用的客户端软件开发工具包的版本而异，但是您必须有集群中某个硬件安全模块（HSM）的 IP 地址。您可以使用连接到集群的任何 HSM 的 IP 地址。客户端软件开发工具包连接后，将检索任何其他 HSM 的 IP 地址，并执行负载平衡和客户端密钥同步操作。

要获取集群的 IP 地址

获取 HSM 的 IP 地址（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 要更改 Amazon Web Services Region，请使用页面右上角的区域选择器（Region selector）。
3. 要打开该集群的详细信息页面，请在集群表中选择集群 ID。
4. 要获取 IP 地址，请在 HSM 选项卡上，选择 ENI IP 地址下列出的 IP 地址之一。

获取 HSM 的 IP 地址 ()AWS CLI

- 使用 AWS CLI中的 describe-clusters 命令获取 HSM 的 IP 地址。在该命令的输出中，HSM 的 IP 地址为 `EniIp` 的值。

```
$  aws cloudhsmv2 describe-clusters


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

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

有关引导程序的更多信息，请参阅配置工具。

引导客户端软件开发工具包 5

PKCS #11 library

引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

引导适用于客户端软件开发工具包 5 的 Windows EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

OpenSSL Dynamic Engine

引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

JCE provider

引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

引导适用于客户端软件开发工具包 5 的 Windows EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

CloudHSM CLI

引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

引导适用于客户端软件开发工具包 5 的 Windows EC2 实例

- 使用配置工具指定集群中 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 参数代替 -a <HSM_IP_ADDRESSES>。要查看使用 --cluster-id 的要求，请参阅 客户端软件开发工具包 5 配置工具。

引导客户端软件开发工具包 3

引导适用于客户端软件开发工具包 3 的 Linux EC2 实例

- configure用于指定集群中 HSM 的 IP 地址。

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

引导适用于客户端软件开发工具包 3 的 Windows EC2 实例

- configure用于指定集群中 HSM 的 IP 地址。

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

有关配置的更多信息，请参阅 ???。

在集群中添加或移除 HSM AWS CloudHSM

要向上或向下扩展 AWS CloudHSM 集群，请使用AWS CloudHSM 控制台、软件开发工具包或命令行工具添加或移除 H AWS SM。我们建议您对集群执行负载测试，以确定您的峰值负载预期，然后添加 HSM 以确保高可用性。

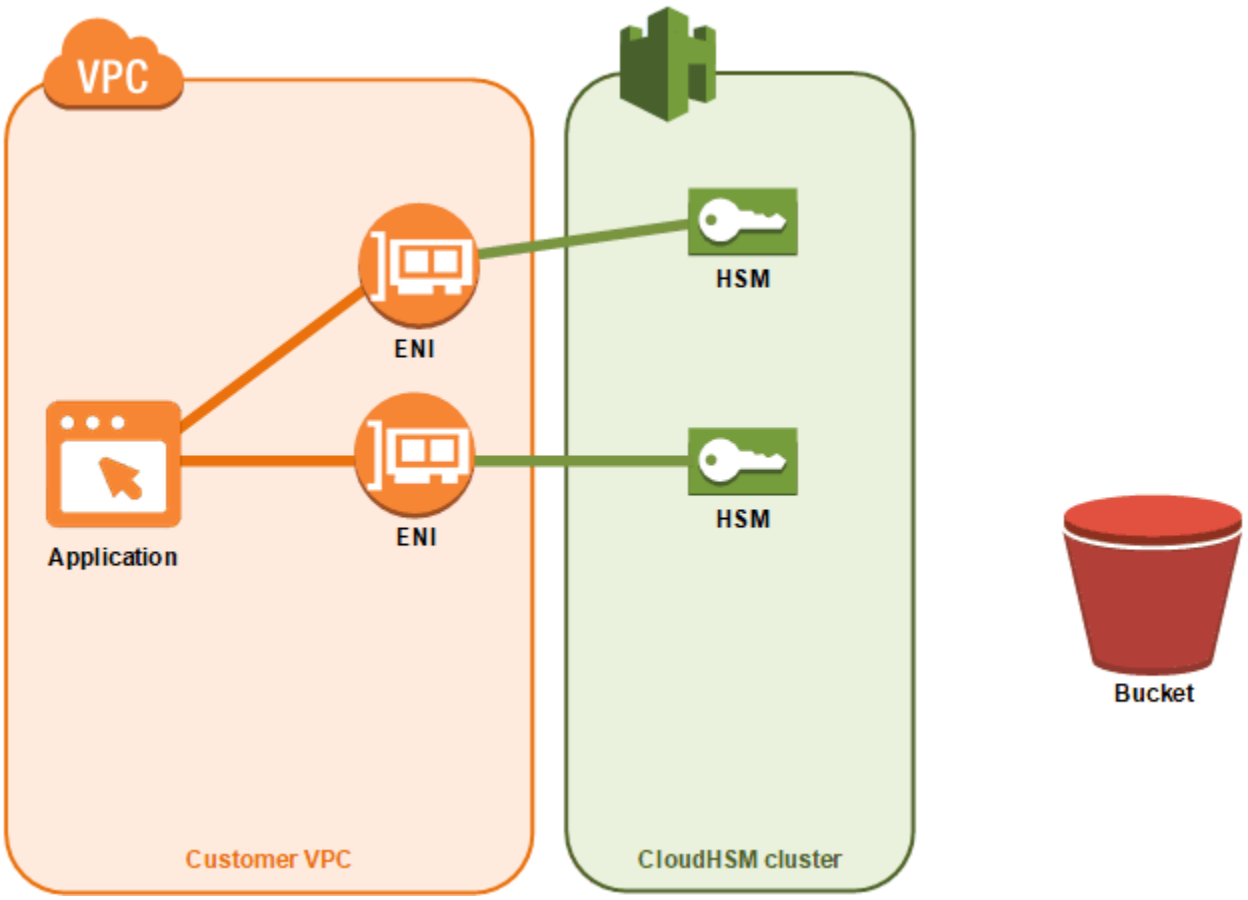
主题

- 添加 HSM

- 删除 HSM

添加 HSM

下图阐明了将 HSM 添加到集群时发生的事件。



1. 向集群添加一个新的 HSM。以下过程介绍如何通过 AWS CloudHSM 控制台、AWS Command Line Interface (AWS CLI) 和 AWS CloudHSM API 执行此操作。

   这是您执行的唯一操作。其余事件将自动发生。
2. AWS CloudHSM 创建集群中现有 HSM 的备份副本。有关更多信息，请参阅 备份。
3. AWS CloudHSM 将备份恢复到新的 HSM 上。这将确保该 HSM 与集群中的其他 HSM 同步。
4. 集群中的现有 HSM 会通知 AWS CloudHSM 客户端集群中有新的 HSM。
5. 此客户端将建立与新的 HSM 的连接。

添加 HSM (控制台)

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 为要添加的 HSM 选择集群。
3. 在 HSMs 选项卡上，选择 Create HSM。
4. 为正在创建的 HSM 选择可用区 (AZ)。然后选择 创建。

添加 HSM (AWS CLI)

- 在命令提示符处，发出 create-hsm 命令，并为要创建的 HSM 指定集群 ID 和可用区。如果您不知道首选集群的集群 ID，请发出 describe-clusters 命令。以 us-east-2a、us-east-2b 等形式指定可用区。

```
$ 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"
    }
}
```

添加 HSM (AWS CloudHSM API)

- 发送 CreateHsm 请求，并为要创建的 HSM 指定集群 ID 和可用区。

删除 HSM

您可以使用AWS CloudHSM 控制台AWS CLI、或 AWS CloudHSM API 删除 HSM。

删除 HSM (控制台)

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。

2. 选择包含要删除的 HSM 的集群。
3. 在 HSM 选项卡上，选择要删除的 HSM。然后选择删除 HSM。
4. 确认您要删除 HSM。然后选择删除。

删除 HSM (AWS CLI)

- 在命令提示符处，发出 delete-hsm 命令。传递包含要删除的 HSM 和下列 HSM 标识符之一的集群 ID：
  - HSM ID (`--hsm-id`)
  - HSM IP 地址 (`--eni-ip`)
  - HSM 的弹性网络接口 ID (`--eni-id`)

  如果您不知道这些标识符的值，请发出 describe-clusters命令。

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

移除 HSM (AWS CloudHSM API)

- 发送 DeleteHsm 请求，并为要删除的 HSM 指定集群 ID 和标识符。

删除集 AWS CloudHSM 群

您必须先从集群中删除所有 HSM，然后才能删除集群。有关更多信息，请参阅 删除 HSM。

删除所有 HSM 后，您可以使用AWS CloudHSM 控制台、AWS Command Line Interface (AWS CLI) 或 AWS CloudHSM API 删除集群。

删除一个集群 (控制台)

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 选择要删除的集群。然后选择 删除集群。
3. 确认您要删除该集群，然后选择 删除。

删除群集 (AWS CLI)

- 在命令提示符处，发出 delete-cluster 命令，以传递您要删除的群集的 ID。如果您不知道群集 ID，请发出 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"
    }
}
```

删除AWS CloudHSM 集群 (API)

- 发送 DeleteCluster 请求，以指定要删除的群集的 ID。

使用备份创建 AWS CloudHSM 集群

要从备份中恢复 AWS CloudHSM 集群，请按照本主题中的步骤操作。您的集群包含位于备份中的相同用户、密钥材料、证书、配置和策略。有关管理备份的更多信息，请参阅 管理备份。

通过备份创建集群（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 选择创建集群。
3. 在集群配置部分中，执行以下操作：
   a. 对于 VPC，为您要创建的集群选择 VPC。
   b. 对于可用区，为您要添加到集群的每个可用区选择私有子网。
4. 在集群源部分中，执行以下操作：
   a. 选择从现有备份恢复集群。
   b. 选择您要还原的备份。
5. 选择下一步：审核。
6. 检查您的集群配置，然后选择创建集群。
7. 指定服务的备份保留期。

   接受 90 天的默认保留期或键入一个介于 7 到 379 天之间的新值。该服务将自动删除此集群中早于您在此处所指定的值的备份。您以后可以更改此值。有关更多信息，请参阅 配置备份保留。
8. 选择下一步。
9. (可选) 键入标签键和一个可选标签值。要向集群添加多个标签，请选择 添加标签。
10. 选择审核。
11. 检查您的集群配置，然后选择 创建集群。

> **Tip**
>
> 要在此集群中创建包含与您恢复的备份中相同的用户、密钥材料、证书、配置和策略的 HSM，请向该集群添加 HSM。

使用备份创建集群 (AWS CLI)

要确定备份 ID，请发布 describe-backups 命令。

- 在命令提示符处，发出 create-cluster 命令。指定 HSM 实例类型、计划在其中创建 HSM 的子网的子网 ID 以及要还原的备份的备份 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"
    }
}
```

使用备份创建集群 (AWS CloudHSM API)

请参阅以下主题，了解如何使用 API 删除和还原备份。

- CreateCluster

管理 AWS CloudHSM 备份

AWS CloudHSM 至少每 24 小时对您的集群进行一次定期备份。每个备份均包含以下数据的加密副本：

- 用户（CO、CU 和 AU）
- 密钥材料和证书
- 硬件安全模块（HSM）配置和策略

您无法指示该服务制作备份，但您可以采取特定操作来强制该服务创建备份。该服务将在您执行以下任一操作时制作备份：

- 激活集群
- 向活动集群添加 HSM
- 从集群中删除 HSM。

AWS CloudHSM 根据您在创建集群时设置的备份保留策略删除备份。有关管理备份保留策略的信息，请参见 配置备份保留。

主题

使用备份

当您向之前包含一个或多个活动 HSM 的集群添加一个 HSM 时， 会将最新备份还原到新的 HSM 上。使用备份管理不经常使用的 HSM。当您不需要使用 HSM 时，可将其删除，这将触发备份。稍后，当您需要 HSM 时，在同一集群中创建一个新的 HSM，此操作将恢复您之前通过删除 HSM 操作而创建的备份。

移除过期的密钥或不活跃用户

您可能想从环境中移除不需要的加密材料，如过期的密钥或非活动用户。这是一个包括两个步骤的过程：首先，从 HSM 中删除这些材料。接下来，删除所有现有备份。按照此过程可确保您在通过备份初始化新集群时不会恢复已删除信息。有关更多信息，请参阅 the section called “删除和还原备份”。

考虑灾难恢复

从备份创建集群。您可能需要执行此操作为集群设置恢复点。指定一个包含恢复点所需的所有用户、密钥材料和证书的备份，然后使用该备份创建新集群。有关从备份创建集群的更多信息，请参阅 使用备份创建集群。

您还可以将集群的备份复制到不同的区域，在那里创建一个新的集群作为原始集群的克隆。您可能出于多种原因希望这样做，包括简化灾难恢复过程。有关将备份复制到区域的更多信息，请参阅 跨区域复制备份。

删除和还原备份

删除备份后，该服务将保留备份七天，在此期间您可以还原备份。七天期限过后，您将无法还原备份。有关管理备份的更多信息，请参阅 管理备份。

删除和还原备份（控制台）

删除备份（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 要更改 Amazon Web Services Region，请使用页面右上角的区域选择器（Region selector）。
3. 在导航窗格中，选择备份。
4. 选择要删除的备份。
5. 要删除所选备份，请选择操作，删除。

   此时显示删除备份对话框。
6. 选择删除。

   备份的状态更改为PENDING_DELETE。在您请求删除后，您可以还原删除天数在 7 天内的备份。

还原备份（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 要更改 Amazon Web Services Region，请使用页面右上角的区域选择器（Region selector）。
3. 在导航窗格中，选择 备份。
4. 选择处于 PENDING_DELETE 状态的要还原备份。
5. 要还原所选备份，请选择操作，还原。

删除和还原备份（AWS CLI）

使用 AWS CLI中的 describe-backups 命令检查备份的状态或查找其 ID。

删除备份 (AWS CLI)

- 在命令提示符处，运行 delete-backup 命令，以传递要删除的备份的 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"
    }
}
```

还原备份 (AWS CLI)

- 要还原备份，请发出 restore-backup 命令，从而传递处于 PENDING_DELETION 状态的备份的 ID。

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

```
        "BackupState": "READY",
        "BackupId": "backup-ro5c4er4aac"
    }
}
```

列出备份（AWS CLI）

- 要查看所有处于 PENDING_DELETION 状态的备份列表，请运行 describe-backups 命令并包含 states=PENDING_DELETION 作为筛选条件。

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

删除和恢复备份 (AWS CloudHSM API)

请参阅以下主题，了解如何使用 API 删除和还原备份。

- DeleteBackup
- RestoreBackup

配置 AWS CloudHSM 备份保留策略

除 2020 年 11 月 18 日之前创建的集群外，集群的默认备份保留策略均为 90 天。您可以将此时段设置为 7 到 379 天之间的任意数字。 AWS CloudHSM 不会删除集群的最后一个备份。有关管理备份的更多信息，请参阅 管理备份。

了解备份保留策略

AWS CloudHSM 根据您在创建集群时设置的备份保留策略清除备份。备份保留策略适用于集群。如果您将备份移至其他区域，则该备份将不再与集群关联，且没有备份保留策略。您必须手动删除所有与集群无关的备份。 AWS CloudHSM 不会删除集群的最后一个备份。

AWS CloudTrail 报告已标记为删除的备份。您可以还原服务清除的备份，就像还原手动删除的备份一样。为防止出现争用情况，在还原已被服务删除的备份之前，应更改集群的备份保留策略。如果要保持相同的保留策略并保留所选备份，则可以指定集群备份保留策略服务不包括备份。

现有集群豁免

AWS CloudHSM 已于 2020 年 11 月 18 日推出托管备份保留服务。2020 年 11 月 18 日之前创建的集群的备份保留策略为 90 天加上集群的使用期限。例如，如果您在 2019 年 11 月 18 日创建了一个集群，则该服务会为您的集群分配一年加 90 天（455 天）的备份保留策略。

> **Note**
>
> 您可以联系支持中心（https://aws.amazon.com/support）选择不使用托管备份保留服务。

配置备份保留（控制台）

要配置备份保留策略（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 要更改 Amazon Web Services Region，请使用页面右上角的区域选择器（Region selector）。
3. 单击处于活动状态的集群的集群 ID 以管理该集群的备份保留策略。
4. 要更改备份保留策略，请选择操作，更改备份保留期。

   此时显示更改备份保留期对话框。
5. 在 备份保留期（以天为单位）中，键入一个介于 7 至 379 天之间的值。
6. 选择 更改备份保留期。

要备份保留策略排除或包含备份（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。

2. 要查看您的备份，请在导航窗格中选择备份。
3. 单击处于 Ready 状态的备份的备份 ID 以排除或包含该备份。
4. 在备份详细信息页面上，执行以下操作之一。
   - 要排除日期为过期时间的备份，请选择操作，禁用过期。
   - 要包括未过期的备份，请选择操作，使用集群保留策略。

配置备份保留（AWS CLI）

使用 AWS CLI中的 describe-backups 命令检查备份的状态或查找其 ID。

要配置备份保留策略（AWS CLI）

- 在命令提示符处，发出 modify-cluster 命令。指定集群 ID 和备份保留策略。

```
$ 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"
   }
}
```

从备份保留策略中排除备份（AWS CLI）

- 在命令提示符处，发出 modify-backup-attributes 命令。指定备份 ID 并设置 no-never-expires（永不过期）标志以保留备份。

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

将备份纳入备份保留策略 (AWS CLI)

- 在命令提示符处，发出 modify-backup-attributes 命令。指定备份 ID 并设置该 no-never-expires 标志以将备份包含在备份保留策略中，这意味着该服务最终将删除备份。

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

配置备份保留期 (AWS CloudHSM API)

请参阅以下主题，了解如何使用 API 管理备份保留。

- ModifyCluster
- ModifyBackupAttributes

跨 AWS 区域复制备份

您可以出于多种原因跨区域复制备份，这些原因包括跨区域恢复能力、全球工作负载和灾难恢复。复制备份后，它们会以 CREATE_IN_PROGRESS 状态显示在目标区域。成功完成复制后，备份的状态更改为 READY。如果复制失败，备份状态将变为 DELETED。检查输入参数有无错误，并确保指定的源备份在重新运行操作之前不处于 DELETED 状态。有关备份和如何从备份创建集群的信息，请参阅 管理备份 或 使用备份创建集群。

请注意以下几点：

- 要将集群备份复制到目标区域，您的账户必须具有正确的 IAM policy 权限。为了将备份复制到其他区域，您的 IAM policy 必须允许访问备份所在的源区域。在跨区域复制后，您的 IAM policy 必须允许访问目标区域，以便与复制的备份进行交互，其中包括使用 CreateCluster 操作。有关更多信息，请参阅 创建 IAM 管理员。
- 原始群集和可能从目标区域中的备份生成的群集未关联。您必须独立管理这些群集。有关更多信息，请参阅 管理 集群。
- 无法在 AWS 受限区域和标准区域之间复制备份。可以在 AWS GovCloud （美国东部）和 AWS GovCloud （美国西部）区域之间复制备份。

将备份复制到不同的区域（控制台）

将备份复制到不同的区域（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 要更改 Amazon Web Services Region，请使用页面右上角的区域选择器（Region selector）。
3. 在导航窗格中，选择 备份。
4. 选择一个备份复制到其他区域。
5. 要复制选定的备份，请选择操作，将备份复制到其他区域。

将出现“将备份复制到其他区域”对话框。

6. 在目标区域中，从选择一个区域中选择一个区域。
7. (可选) 键入标签键和一个可选标签值。要向集群添加多个标签，请选择 添加标签。
8. 选择 复制备份。

将备份复制到不同的区域 (AWS CLI)

要确定备份 ID，请运行 describe-backups 命令。

将备份复制到不同的区域 (AWS CLI)

- 在命令提示符下，运行 copy-backup-to-region 命令。指定目标区域以及源备份的备份 ID。如果您指定备份 ID，则会复制关联的备份。

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

将备份复制到不同的区域 (AWS CloudHSM API)

请参阅以下主题，了解如何使用 API 将备份复制到不同的区域。

- CopyBackupToRegion

使用共享备份

CloudHSM AWS Resource Access Manager 与AWS RAM() 集成以实现资源共享。 AWS RAM 是一项使您能够与其他人或通过共享某些 CloudHSM 资源的服务。 AWS 账户 AWS Organizations使用 AWS RAM，您可以通过创建资源共享来共享您拥有的资源。资源共享指定要共享的资源以及与之共享资源的使用者。使用者可包括：

- 具体在其组织 AWS 账户 内部或外部 AWS Organizations
- 其组织内部的组织单位 AWS Organizations
- 整个组织都在 AWS Organizations

有关的更多信息 AWS RAM，请参阅《AWS RAM 用户指南》。

本主题说明如何共享您拥有的资源以及如何使用共享给您的资源。

内容

共享备份的先决条件

- 要共享备份，您必须将其归自己所有 AWS 账户。这意味着资源必须分配或预调配到您的账户。您无法共享已与您共享的备份。
- 要共享备份，备份必须处于 “就绪” 状态。
- 要与您的组织或中的组织单位共享备份 AWS Organizations，必须启用与共享 AWS Organizations。有关更多信息，请参阅AWS RAM 《用户指南》中的允许与 AWS Organizations 共享。

共享备份

当您与其他人共享备份时 AWS 账户，您可以使他们能够从包含存储在备份中的密钥和用户的备份中恢复集群。

要共享备份，必须将其添加到资源共享。资源共享是一项 AWS RAM 资源，可让您跨 AWS 账户共享资源。资源共享指定要共享的资源以及与之共享资源的使用者。当您使用 CloudHSM 控制台共享备份时，可以将其添加到现有资源共享中。要将备份添加到新的资源共享，必须先使用AWS RAM 控制台创建资源共享。

如果您是组织中的一员， AWS Organizations 并且启用了组织内部共享，则会自动授予组织中的消费者访问共享备份的权限。否则，消费者会收到加入资源共享的邀请，并在接受邀请后被授予访问共享备份的权限。

您可以使用 AWS RAM 控制台共享您拥有的备份，或者 AWS CLI。

使用 AWS RAM 控制台共享您拥有的备份

请参阅《AWS RAM 用户指南》中的创建资源共享。

共享您拥有的备份（AWS RAM 命令）

使用 create-resource-share 命令。

共享您拥有的备份（cloudHSM 命令）

> **Important**
>
> 虽然您可以使用 Cloud PutResourcePolicy HSM 操作共享备份，但我们建议改 AWS Resource Access Manager 用AWS RAM()。使用 AWS RAM 提供了多种好处，因为它可以为您创建策略，允许同时共享多个资源，并提高共享资源的可发现性。如果您使用 PutResourcePolicy 并希望使用者能够描述您与他们共享的备份，则必须使用 AWS RAM PromoteResourceShareCreatedFromPolicy API 操作将备份提升为标准 AWS RAM 资源共享。

使用 put-resource-policy 命令。

1. 创建一个名为的文件`policy.json`并将以下策略复制到其中。

```
{
  "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`使用备份 ARN 和标识符进行更新，以便与之共享。以下示例向根用户授予由 123456789012 标识的 AWS 账户的只读访问权限。

```
{
  "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**
>
> 您只能在账户 DescribeBackups 级别向授予权限。当您与其他客户共享备份时，任何在该账户中拥有 DescribeBackups 权限的委托人都可以描述该备份。

3. 运行 put-resource-policy 命令。

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

> **Note**
>
> 此时，使用者可以使用备份，但它不会显示在带有共享参数的 DescribeBackups 响应中。接下来的步骤将介绍如何提升 AWS RAM 资源共享，以便将备份包含在响应中。

4. 获取 AWS RAM 资源共享 ARN。

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

这将返回类似于以下内容的响应：

```
{
  "resources": [
    {
      "arn": "<project-arn>",
      "type": "<type>",
```

```
      "resourceShareArn": "<resource-share-arn>",
      "creationTime": "<creation-time>",
      "lastUpdatedTime": "<last-update-time>"
    }
  ]
}
```

从响应中复制 *< resource-share-arn >* 值，以便在后续步骤中使用。

5. 运行 AWS RAM promote-resource-share-created-from-policy 命令。

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

6. 要验证资源共享是否已升级，可以运行 AWS RAM get-resource-shares命令。

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

提升策略后，响应中`featureSet`列出的内容为`STANDARD`。这也意味着可以用策略中的新账户来描述备份。

取消共享备份

取消共享资源后，使用者可能不再使用该资源来恢复集群。消费者仍然可以访问他们从共享备份中恢复的任何集群。

要取消共享您拥有的共享备份，必须将其从资源共享中移除。您可以使用 AWS RAM 控制台或执行此操作 AWS CLI。

使用控制台取消共享您拥有的共享备份 AWS RAM

请参阅《AWS RAM 用户指南》中的更新资源共享。

取消共享您拥有的共享备份（AWS RAM 命令）

使用 disassociate-resource-share 命令。

取消共享您拥有的共享备份（CloudHSM 命令）

使用 delete-resource-policy 命令。

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

识别共享备份

消费者可以使用 CloudHSM 控制台识别与他们共享的备份，然后。 AWS CLI

使用 CloudHSM 控制台识别与您共享的备份

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 要更改 AWS 区域，请使用页面右上角的区域选择器。
3. 在导航窗格中，选择备份。
4. 在表中，选择 “共享备份” 选项卡。

要识别与您共享的备份，请使用 AWS CLI

使用带`--shared`参数的 describe-bac kups 命令返回与您共享的备份。

共享备份的权限

所有者的权限

Backup 所有者可以描述和管理共享备份，也可以使用它来恢复集群。

使用者的权限

Backup 使用者无法修改共享备份，但他们可以对其进行描述并使用它来恢复集群。

计费和计量

共享备份不收取额外费用。

为资源添加标签 AWS CloudHSM

标签是您分配给 AWS 资源的标签。您可以将标签分配给您的 AWS CloudHSM 群集。每个标签都包含您定义的一个标签密钥和一个标签值。例如，标签密钥可能是 Cost Center，标签值可能是 12345。每个群集的标签键必须是唯一的。

您可以出于各种目的使用标签。一种常见用途是对 AWS 成本进行分类和跟踪。您可以设置代表业务类别（例如成本中心、应用程序名称或所有者）的标签，以便整理多种服务的成本。向 AWS 资源添加标签时，AWS 会生成一份成本分配报告，其中包含按标签汇总的使用量和成本。您可以使用此报告按项目或应用程序查看 AWS CloudHSM 成本，而不必将所有 AWS CloudHSM 成本作为单个行项目来查看。

有关对成本分配使用标签的更多信息，请参阅 AWS Billing 用户指南中的使用成本分配标签。

您可使用 AWS CloudHSM 控制台或者 AWS 开发工具包或命令行工具之一来添加、更新、列出和删除标签。

主题

添加或更新标签

您可从 AWS CloudHSM 控制台、AWS Command Line Interface (AWS CLI) 或 AWS CloudHSM API 添加或更新标签。

添加或更新标签 (控制台)

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 选择要为其添加标签的群集。
3. 选择标签。
4. 要添加标签，请执行以下操作：
    a. 选择编辑标签，然后选择添加标签。
    b. 对于键，键入标签的键。

c. （可选）对于值，键入标签的值。

d. 选择保存。

5. 要更新标签，请执行以下操作：

a. 选择编辑标签。

> **Note**
>
> 如果更新现有标签的标签值，控制台将删除现有标签并创建新标签。

b. 键入新标签值。

c. 选择保存。

添加或更新标签 (AWS CLI)

1. 在命令提示符处，发出 tag-resource 命令，并指定标签和您要为其添加标签的群集的 ID。如果您不知道群集 ID，请发出 describe-clusters 命令。

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

2. 要更新标签，请使用相同的命令，但指定现有标签键。当您为现有标签指定新的标签值时，该标签将由新值覆盖。

添加或更新标签 (AWS CloudHSM API)

- 发送 TagResource 请求。指定标签和要为其添加标签的群集的 ID。

列出标签

您可以通过AWS CloudHSM 控制台AWS CLI、或 AWS CloudHSM API 列出集群的标签。

列出标签 (控制台)

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 选择要列出其标签的群集。
3. 选择标签。

列出标签 (AWS CLI)

- 在命令提示符处，发出 list-tags 命令，并指定要列出其标签的群集的 ID。如果您不知道群集 ID，请发出 describe-clusters 命令。

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

列出标签 (AWS CloudHSM API)

- 发送 ListTags 请求，并指定要列出其标签的群集的 ID。

删除标签

您可以使用 AWS CloudHSM 控制台、AWS CLI 或 AWS CloudHSM API 从群集中删除标签。

删除标签 (控制台)

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 选择要删除其标签的群集。
3. 选择标签。
4. 选择编辑标签，然后为要删除的标签选择移除标签。
5. 选择保存。

删除标签 (AWS CLI)

- 在命令提示符处，发出 untag-resource 命令，并指定要删除的标签的标签键以及要删除其标签的群集的 ID。使用删除标签时，请仅指定标签键，而不是标签值。 AWS CLI

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

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

移除标签 (AWS CloudHSM API)

- 在 AWS CloudHSM API 中发送UntagResource请求，指定集群的 ID 和要删除的标签。

管理 HSM 用户和密钥 AWS CloudHSM

必须先在 AWS CloudHSM 集群中的 HSM 上创建用户和密钥，然后才能使用集群进行加密处理。有关管理 AWS CloudHSM中的 HSM 用户和密钥的更多信息，请参阅以下主题。您还可以了解如何使用 quorum 身份验证 (也称为“N 中的 M 访问控制”)。

主题

- 在中管理 HSM 用户 AWS CloudHSM
- 管理中的密钥 AWS CloudHSM
- 管理克隆的集群

在中管理 HSM 用户 AWS CloudHSM

在中 AWS CloudHSM，您必须使用 C loudHSM CLI 或 CloudHSM 管理实用程序 (CMU) 命令行工具在您的 H SM 上创建和管理用户。CloudHSM CLI 可与最新的 SDK 版本系列配合使用，而 CMU 则可与以前的 SDK 版本系列配合使用。

主题

- 使用 CloudHSM CLI 管理 HSM 用户
- 使用 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility) 管理 HSM 用户

使用 CloudHSM CLI 管理 HSM 用户

使用 CloudHSM CLI 命令行工具通过最新的开发工具包在您的 HSM 上创建和管理用户。

主题

- 了解 HSM 用户
- HSM 用户权限表
- 使用 CloudHSM CLI 管理用户
- 使用 CloudHSM CLI 管理 MFA
- 使用 CloudHSM CLI 管理仲裁身份验证（M of N 访问控制）

了解 HSM 用户

您在 HSM 上执行的大多数操作都需要 HSM 用户 的凭证。HSM 对每个 HSM 用户进行身份验证，每个 HSM 用户都有一种类型，其可确定您可以在 HSM 上以该用户身份执行的操作。

> **Note**
>
> HSM 用户与 IAM 用户不同。拥有正确证书的 IAM 用户可通过 AWS API 与资源交互来创建 HSM。创建 HSM 后，您必须使用 HSM 用户凭证对 HSM 上的操作进行身份验证。

用户类型

- 未激活的管理员
- Admin
- 加密用户 (CU)
- 设备用户 (AU)

未激活的管理员

在 CloudHSM CLI 中，未激活的管理员是一个临时用户，仅存在于从未激活过的集群 AWS CloudHSM 中的第一个 HSM 上。要激活集群，请在 CloudHSM CLI 中运行 cluster activate 命令。运行此命令后，系统会提示未激活的管理员更改密码。更改密码后，未激活的管理员将成为管理员。

Admin

在 CloudHSM CLI 中，管理员可以执行用户管理操作。例如，它们可以创建和删除用户以及更改用户密码。有关管理员的更多信息，请参阅 HSM 用户权限表。

加密用户 (CU)

加密用户 (CU) 可以执行以下密钥管理和加密操作。

- 密钥管理 – 创建、删除、共享、导入和导出加密密钥。
- 加密操作 – 使用加密密钥来执行加密、解密、签名、验证及更多操作。

有关更多信息，请参阅 HSM 用户权限表。

设备用户 (AU)

设备用户 (AU) 可以在集群的 HSM 上执行克隆和同步操作。 AWS CloudHSM 使用 AU 同步 AWS CloudHSM 集群中的 HSM。AU 存在于由提供的所有 HSM 上 AWS CloudHSM，并且权限有限。有关更多信息，请参阅 HSM 用户权限表。

AWS 无法对您的 HSM 执行任何操作。 AWS 无法查看或修改您的用户或密钥，也无法使用这些密钥执行任何加密操作。

HSM 用户权限表

下表列出了按可以执行该操作的 HSM 用户或会话类型排序的 HSM 操作。

	Admin	加密用户 (CU)	设备用户 (AU)	未经身份验证的会话
获取基本集群信息[1]	 是	 是	 是	 是
更改自己的密码	 是	 是	 是	不适用
更改任意用户的密码	 是	 不支持	 不支持	 否
添加、删除用户	 是	 不支持	 不支持	 否

	Admin	加密用户 (CU)	设备用户 (AU)	未经身份验证的会话
获取同步状态[2]	是	是	是	不支持
提取、插入遮蔽对象[3]	是	是	是	不支持
密钥管理功能[4]	否	是	不支持	否
加密、解密	否	是	不支持	否
签署、验证	否	是	不支持	否
生成摘要和 HMAC	否	是	不支持	否

- [1] 基本集群信息包括集群中 HSM 的数量和每个 HSM 的 IP 地址、型号、序列号、设备 ID、固件 ID 等。

- [2] 用户可以获取与 HSM 上的密钥对应的一组摘要（哈希值）。应用程序可以将这些摘要集进行比较来了解集群中 HSM 的同步状态。
- [3] 遮蔽对象是在离开 HSM 之前进行加密的密钥。它们无法在 HSM 外部加密。只有在将其插入 HSM 之后，并且该 HSM 所在集群与提取它们时的 HSM 的集群相同，才会解密它们。应用程序可以提取和插入遮蔽对象，以用于同步集群中的 HSM。
- [4] 密钥管理功能包括创建、删除、包装、解开包装和修改密钥的属性。

使用 CloudHSM CLI 管理用户

本主题提供有关使用 CloudHSM CLI 管理硬件安全模块 (HSM) 用户的 step-by-step 说明。有关 CloudHSM CLI 或 HSM 用户的更多信息，请参阅CloudHSM CLI和使用 CloudHSM CLI。

Sections

使用 CloudHSM CLI 了解 HSM 用户管理

要管理 HSM 用户，您必须使用管理员的用户名和密码登录 HSM。只有管理员才能管理用户。HSM 包含名为 admin 的默认管理员。在激活集群时，您可以设置 admin 的密码。

要使用 CloudHSM CLI，您必须使用配置工具更新本地配置。有关使用 CloudHSM CLI 运行配置工具的说明，请参阅 开始使用 CloudHSM 命令行界面（CLI）。-a 参数要求您在集群中添加 HSM 的 IP 地址。如果您有多个 HSM，则可以使用任一 IP 地址。这可确保 CloudHSM CLI 可以将您所做的任何变更传播到整个集群中。请记住，CloudHSM CLI 使用其本地文件来跟踪集群信息。如果自上次在特定主机上使用 CloudHSM CLI 以来该集群已更改，则必须将此类更改添加到存储在该主机上的本地配置文件中。在使用 CloudHSM CLI 时，切勿移除 HSM。

获取 HSM 的 IP 地址（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 要更改 Amazon Web Services Region，请使用页面右上角的区域选择器（Region selector）。
3. 要打开该集群的详细信息页面，请在集群表中选择集群 ID。
4. 要获取 IP 地址，请在 HSM 选项卡上，选择 ENI IP 地址下列出的 IP 地址之一。

获取 HSM 的 IP 地址 ()AWS CLI

- 使用 AWS CLI中的 describe-clusters 命令获取 HSM 的 IP 地址。在该命令的输出中，HSM 的 IP 地址为 `EniIp` 的值。

```
$  aws cloudhsmv2 describe-clusters


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

下载 CloudHSM CLI

最新版本的 CloudHSM CLI 可用于客户端开发工具包 5 的 HSM 用户管理任务。要下载和安装 CloudHSM CLI，请按照安装和配置 CloudHSM CLI 中的说明操作。

如何使用 CloudHSM CLI 管理 HSM 用户

本节包括使用 CloudHSM CLI 管理 HSM 用户的基本命令。

> **Note**
>
> 请注意：CloudHSM CLI 用户命令在 CloudHSM CLI 用户命令参考中列出

主题

- 创建管理员
- 创建加密用户
- 要列出集群中所有的 HSM 用户

- 要更改 HSM 用户密码
- 删除 HSM 用户

创建管理员

按照以下步骤创建管理员。

1. 使用以下命令启动 CloudHSM CLI 交互模式。

   Linux

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

   Windows

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

2. 使用 login 命令，并以管理员身份登录该集群。

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

3. 系统将会提示您输入密码。输入密码后，输出将显示该命令已成功。

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

4. 输入以下命令来创建管理员：

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

5. 输入新用户的密码。
6. 再次输入密码以确认您输入的密码正确无误。

创建加密用户

按照以下步骤创建用户。

1. 使用以下命令启动 CloudHSM CLI 交互模式。

   Linux

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

   Windows

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

2. 使用 login 命令，并以管理员身份登录该集群。

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

3. 系统将会提示您输入密码。输入密码后，输出将显示该命令已成功。

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

4. 输入以下命令以创建加密用户：

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

5. 输入新加密用户的密码。
6. 再次输入密码以确认您输入的密码正确无误。

要列出集群中所有的 HSM 用户

使用 user list 命令列出该集群中的所有用户。无需登录即可运行 user list。所有用户类型均可列出用户。

按照以下步骤列出该集群中的所有用户

1. 使用以下命令启动 CloudHSM CLI 交互模式。

   Linux

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

   Windows

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

2. 输入以下命令列出该集群中的所有用户：

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

有关 user list 的更多信息，请参阅 user list。

要更改 HSM 用户密码

使用 user change-password 命令更改密码。

用户类型和密码区分大小写，但是用户名不区分大小写。

管理员、加密用户（CU）和应用程序用户（AU）只能更改自己的密码。要更改其他用户的密码，您必须以管理员身份登录。您无法更改当前已登录用户的密码。

更改您自己的密码

1. 使用以下命令启动 CloudHSM CLI 交互模式。

   Linux

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

   Windows

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

2. 使用 login 命令并以待更改密码的用户身份登录。

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

3. 输入用户的密码。

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

4. 输入 user change-password 命令。

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

5. 输入新密码。
6. 再次输入新密码。

更改其他用户的密码

1. 使用以下命令启动 CloudHSM CLI 交互模式。

   Linux

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

   Windows

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

2. 使用login命令并以管理员身份登录。

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

3. 输入管理员密码。

```
Enter password:
{
```

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

4. 输入user change-password命令以及待更改密码的用户名称。

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

5. 输入新密码。
6. 再次输入新密码。

有关 user change-password 的更多信息，请参阅user change-password。

删除 HSM 用户

使用 user delete 删除用户。您必须以管理员身份登录才能删除其他用户。

> **Tip**
> 您无法删除拥有密钥的加密用户 (CU)。

删除用户

1. 使用以下命令启动 CloudHSM CLI 交互模式。

   Linux

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

   Windows

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

2. 使用 login 命令，并以管理员身份登录该集群。

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

3. 系统将会提示您输入密码。输入密码后，输出将显示该命令已成功。

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

4. 使用 user delete 命令删除用户。

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

有关 user delete 的更多信息，请参阅 deleteUser。

使用 CloudHSM CLI 管理 MFA

为增强安全性，您可以配置多重身份验证（MFA）以帮助保护集群。有关更多信息，请参阅以下主题。

主题

- 了解 HSM 用户的 MFA
- 为 HSM 用户使用 MFA

了解 HSM 用户的 MFA

当您使用启用 MFA 的 HSM 用户账户登录集群时，您需要向 CloudHSM CLI 提供您的密码（第一个重身份，即您所知道的），CloudHSM CLI 会为您提供令牌并提示您对令牌进行签名。

要提供第二重身份（即您所拥有的），您可以使用已经创建并与 HSM 用户关联的密钥对中的私有密钥签署令牌。要访问该集群，您需要向 CloudHSM CLI 提供已签署的令牌。

有关对用户设置 MFA 的更多信息，请参阅 为 CloudHSM CLI 设置 MFA

仲裁身份验证和 MFA

该集群使用相同的密钥进行仲裁身份验证和 MFA。这表明启用了 MFA 的用户实际上已经注册了 MofN 或仲裁访问权限控制。要成功对同一 HSM 用户使用 MFA 和仲裁身份验证，请考虑以下几点：

- 如果现在对用户使用仲裁身份验证，则应使用为仲裁用户创建的同一密钥对，为该用户启用 MFA。
- 如果为非仲裁身份验证用户的非 MFA 用户增加了 MFA 要求，则可以将该用户注册为采用 MFA 身份验证的仲裁（MofN）用户。
- 如果删除了 MFA 要求或更改了同时也是仲裁身份验证用户的 MFA 用户的密码，则将同时删除该用户的仲裁（MofN）用户注册。
- 如果删除了 MFA 要求或更改了同时也是仲裁身份验证用户的 MFA 用户的密码，但您仍希望该用户参与仲裁身份验证，则必须将该用户重新注册为仲裁（MofN）用户。

有关仲裁身份验证的更多信息，请参阅 管理仲裁（M of N）。

为 HSM 用户使用 MFA

本主题提供了使用 CloudHSM CLI 管理多重身份验证 (MFA) 的信息和说明。有关 CloudHSM CLI 的更多信息，请参阅 CloudHSM 命令行界面（CLI）。

主题

- MFA 密钥对要求
- 为 CloudHSM CLI 设置 MFA
- 创建启用 MFA 的用户
- 登录启用 MFA 的用户
- 为启用 MFA 的用户轮换密钥
- 注册 MFA 公有密钥后，注销管理员用户的 MFA 公有密钥
- 令牌文件参考

有关使用 HSM 用户的更多信息，请参阅 CloudHSM 命令行界面（CLI）

MFA 密钥对要求

要对 HSM 用户启用 MFA，您可以创建新的密钥对或使用满足以下要求的现有密钥：

- 密钥类型：非对称密钥
- 密钥用法：签名和验证
- 密钥规范：RSA_2048
- 签名算法包括： sha256WithRSAEncryption

> **Note**
>
> 如果您正在使用仲裁身份验证或计划使用仲裁人数身份验证，请参阅 仲裁身份验证和 MFA

您可以使用 CloudHSM CLI 和密钥对来创建启用 MFA 的新管理员用户。

为 CloudHSM CLI 设置 MFA

按照以下步骤设置适用于 CloudHSM CLI 的 MFA。

1. 要使用令牌签名策略设置 MFA，您必须首先生成 2048 位的 RSA 私有密钥以及关联的公有密钥。

```
$ 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 登录您的用户账户。

```
$ 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. 接下来，执行命令以更改您的 MFA 策略。您必须提供参数 `--token`。此参数指定将写入未签名令牌的文件。

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

4. 现在，您有一个包含未签名令牌的待签名文件：unsigned-tokens.json。此文件中的令牌数取决于集群中的 HSM 数。每个令牌均代表一个 HSM。此文件采用 JSON 格式，包含能证明您拥有私有密钥的待签名令牌。

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

5. 下一步是使用步骤 1 中创建的私有密钥签署此类令牌。将签名放回文件。首先，您必须提取和解码 base64 编码的令牌。

```
$ 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. 现在，您有了二进制令牌，可以使用步骤 1 中创建的 RSA 私有密钥对其进行签名。

```
$ 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. 现在，您有了令牌的二进制签名。您必须使用 base64 对其编码，然后把其放回令牌文件中。

```
$ 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. 最后，您可以将 base64 值复制并粘贴回令牌文件中：

```
{
  "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. 现在，您的令牌文件已包含所有必需签名，您可以继续操作。输入包含签名令牌的文件名称，然后按 Enter 键。最后，输入公有密钥的路径。

```
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"
  }
}
```

现在，您已经对用户设置了 MFA。

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

创建启用 MFA 的用户

按照以下步骤创建启用 MFA 的用户。

1. 使用 CloudHSM CLI 以管理员身份登录 HSM。
2. 使用 user create 命令创建所选用户。然后按照 为 CloudHSM CLI 设置 MFA 中的步骤为用户设置 MFA。

登录启用 MFA 的用户

按照以下步骤登录启用 MFA 的用户。

1. 使用 CloudHSM CLI 中的 login mfa-token-sign 命令通过 MFA 为启用 MFA 的用户启动登录过程。

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

2. 输入您的密码。然后，系统将提示您输入令牌文件路径，该文件包含未签名/已签名令牌对，其中已签名令牌是使用私有密钥生成的令牌。

```
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. 当系统提示您输入已签名令牌文件路径时，您可以在单独终端中检查未签名令牌文件。使用待签名的未签名令牌识别文件：unsigned-tokens.json。此文件中的令牌数取决于集群中的 HSM 数。每个令牌均代表一个 HSM。此文件采用 JSON 格式，包含能证明您拥有私有密钥的待签名令牌。

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

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

4. 使用步骤 2 中创建的私有密钥签署未签名令牌。首先，您必须提取和解码 base64 编码的令牌。

```
$ 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. 现在，您有了二进制代币。使用您之前在 MFA 设置的第 1 步中创建的 RSA 私有密钥对其进行签名。

```
$ 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. 现在，您有了令牌的二进制签名。使用 base64 对其进行编码，然后将其放回令牌文件。

```
$ 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. 最后，将 base64 值复制并粘贴回令牌文件中：

```
{
  "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. 现在，您的令牌文件已包含所有必需签名，您可以继续操作。输入包含签名令牌的文件名称，然后按 Enter 键。现在，您应该已成功登录。

```
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>"
  }
}
```

为启用 MFA 的用户轮换密钥

按照以下步骤为启动 MFA 的用户轮换密钥。

<result>

您已使用私有密钥对生成的 JSON 格式的令牌文件签名，并注册了新的 MFA 公有密钥。
</result>

1. 使用 CloudHSM CLI 以任何管理员或启用 MFA 的特定用户身份登录 HSM（有关详细信息，请参阅登录启用 MFA 的用户）。
2. 接下来，执行命令以更改您的 MFA 策略。您必须提供参数 `--token`。此参数指定将写入未签名令牌的文件。

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

3. 使用待签名的未签名令牌识别文件：`unsigned-tokens.json`。此文件中的令牌数取决于集群中的 HSM 数。每个令牌均代表一个 HSM。此文件采用 JSON 格式，包含能证明您拥有私有密钥的待签名令牌。这将是您希望用于轮换当前注册的公有密钥的新 RSA 公有密钥/私有密钥对中的新私有密钥。

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

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

4. 使用您之前在设置期间创建的私有密钥对这些令牌签名。首先，必须提取和解码 base64 编码的令牌。

```
$ 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. 现在，您有了二进制代币。使用您之前在设置期间创建的 RSA 私有密钥对其签名。

```
$ 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. 现在，您有了令牌的二进制签名。使用 base64 对其进行编码，然后将其放回令牌文件。

```
$ 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. 最后，将 base64 值复制并粘贴回令牌文件中：

```
{
  "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. 现在，您的令牌文件已包含所有必需签名，您可以继续操作。输入包含签名令牌的文件名称，然后按 Enter 键。最后，输入新公有密钥的路径。现在，您将看到以下内容作为 user list 输出的一部分。

```
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"
  }
}
```

现在，已经对用户设置了 MFA。

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

注册 MFA 公有密钥后，注销管理员用户的 MFA 公有密钥

注册 MFA 公有密钥后，按照以下步骤注销管理员用户的 MFA 公有密钥。

1. 使用 CloudHSM CLI 以启用 MFA 的管理员身份登录 HSM。
2. 使用 user change-mfa token-sign 命令删除用户的 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"
  }
}
```

令牌文件参考

注册 MFA 公有密钥或尝试使用 MFA 登录时生成的令牌文件包含以下内容：

- 令牌：以 JSON 对象文本形式进行 base64 编码的未签名/签名令牌对。
- 未签名： base64 编码和 SHA256 哈希令牌。
- 已签名：未签名令牌的 base64 编码签名令牌（签名），使用 RSA 2048 位私有密钥。

```
{
  "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 管理仲裁身份验证（M of N 访问控制）

AWS CloudHSM 集群中的 HSM 支持法定身份验证，也称为 M of N 访问控制。对于仲裁身份验证，HSM 上的单个用户不能在 HSM 上执行仲裁控制型操作。必须有最少数量 (至少 2 个) 的 HSM 用户合作才能执行这些操作。对于仲裁身份验证，要求多个 HSM 用户批准，这样可以增加一层额外的保护。

仲裁身份验证可以控制以下操作：

- 管理员 HSM 用户管理 – 创建和删除 HSM 用户，以及更改其他 HSM 用户的密码。有关更多信息，请参阅 为管理员使用仲裁身份验证。

下列主题提供了有关 AWS CloudHSM中的仲裁身份验证的更多信息。

主题

- 使用令牌签名策略的仲裁身份验证概述
- 有关仲裁身份验证的其他详细信息
- 支持仲裁身份验证的服务名称和类型
- 使用面向管理者的仲裁身份验证：首次设置
- 为管理员使用仲裁身份验证
- 更改管理员的仲裁最小值

使用令牌签名策略的仲裁身份验证概述

下列步骤概括了仲裁身份验证过程。有关特定步骤和工具，请参阅为管理员使用仲裁身份验证。

1. 每个 HSM 用户都将创建用于签名的非对称密钥。用户在 HSM 外部执行此操作，并注意适当地保护密钥。
2. 每个 HSM 用户都将登录 HSM 并向 HSM 注册其签名密钥的公有部分 (公有密钥)。
3. 当 HSM 用户要执行仲裁控制型操作时，同一用户将登录 HSM 并获取仲裁令牌。
4. HSM 用户将仲裁令牌提供给一个或多个其他 HSM 用户并请求其批准。

5. 其他 HSM 用户通过使用其密钥对仲裁令牌进行加密签名来进行批准。上述操作是在 HSM 外部进行的。
6. 当 HSM 用户获得所需数量的批准时，同一用户将登录 HSM 并使用 --approval 参数运行仲裁控制操作，提供包含所有必要批准（签名）的已签名仲裁令牌文件。
7. HSM 将使用每个签署人的已注册公有密钥来验证签名。如果签名有效，则 HSM 将批准该令牌并执行仲裁控制操作。

有关仲裁身份验证的其他详细信息

注意以下有关在 AWS CloudHSM中使用仲裁身份验证的更多信息。

- HSM 用户可为自己的仲裁令牌签名 — 也就是说，请求用户可提供仲裁身份验证所需的批准之一。
- 为仲裁控制型操作选择最小数量的仲裁审批者。您可以选择的最小数字是二（2），您可以选择的最大数字是八（8）。
- HSM 可容纳多达 1024 个仲裁令牌。如果 HSM 在您尝试创建新的令牌时已有 1024 个令牌，HSM 将清除过期令牌之一。默认情况下，令牌将在创建 10 分钟后过期。
- 如果启用了 MFA，则集群将使用同一密钥进行仲裁身份验证和多重身份验证（MFA）。有关使用仲裁身份验证和双因素身份验证的更多信息，请参阅使用 CloudHSM CLI 管理 MFA。
- 针对每项服务，每个 HSM 一次只能包含一个令牌。

支持仲裁身份验证的服务名称和类型

管理员服务：仲裁身份验证用于管理员特权服务，例如创建用户、删除用户、更改用户密码、设置仲裁值以及停用仲裁和 MFA 功能。

每个服务类型都进一步细分为限定服务名称，其中包含一组特定的、可执行的仲裁支持服务操作。

服务名称	服务类型	服务操作
用户	Admin	• user create • user delete • user change-password • user change-mfa
仲裁	Admin	• 法定代币符号 set-quorum-value

使用面向管理者的仲裁身份验证：首次设置

下列主题描述要配置硬件安全模块必须完成的步骤，以便管理员可以使用仲裁身份验证。当您首次为管理员配置仲裁身份验证时，只需执行下列步骤一次。完成这些步骤后，请参阅 为管理员使用仲裁身份验证。

主题

- 先决条件
- 创建并注册签名密钥
- 在 HSM 上设置仲裁最小值

先决条件

要理解此示例，应熟悉 CloudHSM CLI。在此示例中， AWS CloudHSM 集群有两个 HSM，每个 HSM 都有相同的管理员，如命令的以下输出所user list示。有关创建用户的更多信息，请参见 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"
      }
    ]
  }
}
```

创建并注册签名密钥

要使用仲裁身份验证，每个管理员都必须完成以下所有操作：

主题

- 创建 RSA 密钥对
- 创建注册令牌并签名
- 通过 HSM 注册公钥

创建 RSA 密钥对

创建和保护密钥对的方式有多种。以下示例说明如何使用 OpenSSL 执行该操作。

Example - 使用 OpenSSL 创建私有密钥

以下示例演示如何使用 OpenSSL 创建受密码保护的 2048 位 RSA 密钥。要使用此示例，请将 *<admin.key>* 替换为您要存储密钥的文件的名称。

```
$ 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:
```

接下来，使用您刚刚创建的私钥生成公有密钥。

Example - 使用 OpenSSL 创建公有密钥

以下示例演示如何使用 OpenSSL，根据您刚刚创建的私有密钥创建公钥。

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

创建注册令牌并签名

创建一个令牌，并使用上一步生成的私有密钥签名。

Example – 创建注册令牌

1. 使用以下命令启动 CloudHSM CLI。

   Linux

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

   Windows

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

2. 通过运行quorum token-sign generate命令创建注册令牌：

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

3. quorum token-sign generate命令在指定的文件路径上生成注册令牌。检查令牌文件：

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

令牌路径包含以下内容：

- approval_data：base64 编码的随机数据令牌，其原始数据最大不超过 245 字节。
- 未签名：approval_data 的 base64 编码和 SHA-256 哈希令牌。
- 签名：未签名令牌的 base64 编码签名令牌（签名），使用之前由 OpenSSL 生成的 RSA 2048 位私钥。

您通过私钥对未签名令牌进行签名，以证明您有权访问私钥。您需要在注册令牌文件中完全填充签名和公钥，才能将管理员注册为集群的法定用户。 AWS CloudHSM

Example – 对未签名注册令牌签名

1. 解码 base64 编码的未签名令牌，并将其放入二进制文件：

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

2. 使用 OpenSSL 和私钥对当前未签名的二进制注册令牌签名，并创建二进制注册文件：

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

3. 将二进制签名编码为 base64：

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

4. 将 base64 编码的签名复制并粘贴至令牌文件：

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

通过 HSM 注册公钥

创建密钥后，管理员必须向 AWS CloudHSM 集群注册公钥。

向 HSM 注册公有密钥

1. 使用以下命令启动 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，以管理员身份登录。

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

3. 使用 user change-quorum token-sign register 命令注册公有密钥。有关更多信息，请参阅以下示例或使用 help user change-quorum token-sign register 命令。

Example — 向集 AWS CloudHSM 群注册公钥

以下示例说明如何使用 CloudHSM CLI 中的 user change-quorum token-sign register 命令向 HSM 注册管理员的公有密钥。要使用此命令，管理员必须登录 HSM。将这些值替换为您自己的值：

```
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：公钥 PEM 文件的文件路径
> 必需：是
> /path/tokenfile：用户私钥签名令牌的文件路径
> 必需：是

在所有管理员注册其公钥后，user list 命令的输出将显示在仲裁字段中，表明启用仲裁策略，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"
      }
    ]
  }
}
```

在 HSM 上设置仲裁最小值

要使用仲裁身份验证，管理员必须登录 HSM，然后设置仲裁最小值。这是执行管理员用户管理操作所需的最少数量的 CO 审批。HSM 上的任何管理员 (包括尚未注册签名密钥的管理员) 都可设置仲裁最小值。您可以随时更改仲裁最小值；有关更多信息，请参阅 更改最小值。

在 HSM 上设置仲裁最小值

1. 使用以下命令启动 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，以管理员身份登录。

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

3. 使用 法定代币符号 set-quorum-value 命令设置仲裁最小值。有关更多信息，请参阅以下示例或使用 help quorum token-sign set-quorum-value 命令。

Example - 在 HSM 上设置仲裁最小值

此示例使用仲裁最小值二 (2)。您可以选择介于二 (2) 到八 (8) 范围之内的任何值 (最多为 HSM 上的管理员总数量)。在此示例中，HSM 包含四 (4) 位管理员，因此可能的最大值为四 (4)。

要使用以下示例命令，请将最终数量 (*<2>*) 替换为首选仲裁最小值。

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

在此示例中，服务标识您将为其设置仲裁最小值的 HSM 服务。该 法定代币符号 list-quorum-values 命令列出了服务中包含的 HSM 服务类型、名称和描述。

管理员服务：仲裁身份验证用于管理员特权服务，例如创建用户、删除用户、更改用户密码、设置仲裁值以及停用仲裁和 MFA 功能。

每个服务类型都进一步细分为限定服务名称，其中包含一组特定的、可执行的仲裁支持服务操作。

服务名称	服务类型	服务操作
用户	Admin	• user create • user delete • user change-password • user change-mfa
仲裁	Admin	• 法定代币符号 set-quorum-value

使用 quorum token-sign list-quorum-values 命令获取服务的仲裁最小值。

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

来自前面的 quorum token-sign list-quorum-values 命令的输出显示，HSM 用户管理操作的仲裁最小值（负责用户管理操作）现在为二 (2)。完成这些步骤后，请参阅 使用仲裁 (M 或 N)。

为管理员使用仲裁身份验证

HSM 上的管理员可以为集群中的以下操作配置法定身份验证： AWS CloudHSM

- user create
- user delete
- user change-password
- user change-mfa

AWS CloudHSM 集群配置为法定身份验证后，管理员无法自行执行 HSM 用户管理操作。以下示例显示管理员尝试在 HSM 上创建新用户时的输出。该命令因出错而失败，指出需要进行仲裁身份验证。

```
aws-cloudhsm > user create --username user1 --role crypto-user
```

```
Enter password:
Confirm password:
{
  "error_code": 1,
  "data": "Quorum approval is required for this operation"
}
```

要执行 HSM 用户管理操作，管理员必须完成以下任务：

主题

- 获取仲裁令牌
- 获得批准管理员的签名
- 在 AWS CloudHSM 集群上批准令牌并执行用户管理操作

获取仲裁令牌

首先，管理员必须使用 CloudHSM CLI 申请仲裁令牌。

获取仲裁令牌

1. 使用以下命令启动 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，并以管理员身份登录集群。

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

3. 使用 quorum token-sign generate 命令获取仲裁令牌。有关更多信息，请参阅以下示例或使用 help quorum token-sign generate 命令。

Example – 生成仲裁令牌

此示例将为用户名为 admin 的管理员获取一个仲裁令牌，并将该令牌保存到一个名为 admin.token 的文件中。要使用示例命令，可将这些值替换为您自己的值：

- *<###>* – 获取令牌的管理员姓名。这必须是已登录 HSM 并且正在运行此命令的管理员。
- *<admin.token>* – 用于存储仲裁令牌的文件的名称。

在以下命令中，user 标识您可为之使用所生成之令牌的服务名称。在此示例中，令牌用于 HSM 用户管理操作 (user 服务)。

```
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"
  }
}
```

该 quorum token-sign generate 命令在指定的文件路径上生成用户服务仲裁令牌。可查看以下令牌文件：

```
$cat </path/admin.token>
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW/
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAABAAMAAAAUA
+b23gAAAAAAAAAA",
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": []
}
```

令牌路径包含以下内容：

- approval_data：由 HSM 生成的 base64 编码原始数据令牌。
- 令牌：approval_data 的 base64 编码和 SHA-256 哈希令牌
- 签名：base64 编码签名令牌阵列，其中每个批准者签名均采用 JSON 对象常量形式：

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

每个签名均由批准者使用相应的 RSA 2048 位私钥创建，其公钥通过 HSM 注册。

通过运行 quorum token-sign list 命令，确认 CloudHSM 集群中包含已生成用户服务仲裁令牌。

```
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` 时间表示生成的令牌在过期之前获批的超时时间（以秒为单位）。

获得批准管理员的签名

具有仲裁令牌的管理员必须获得其他管理员批准的令牌。为了提供其批准，其他管理员使用其签名密钥以加密方式对令牌进行签名。他们在 HSM 外部执行此操作。

可通过多种不同方式对令牌进行签名。以下示例说明如何使用 OpenSSL 执行该操作。要使用其他签名工具，请确保该工具使用管理员的私有密钥 (签名密钥) 对令牌的 SHA-256 摘要进行签名。

Example - 获得批准管理员的签名

在此示例中，具有令牌 (`admin`) 的管理员至少需要两 (2) 次批准。以下示例命令说明两 (2) 个管理员如何使用 OpenSSL 以加密方式对令牌进行签名。

1. 解码 base64 编码的未签名令牌，并将其放入二进制文件：

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

2. 使用 OpenSSL 和相应的批准者 (`admin3`) 私钥为当前未签名的二进制仲裁令牌签名，以获取用户服务和创建二进制签名文件：

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

3. 将二进制签名编码为 base64：

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

4. 最后，按此前批准者签名指定的 JSON 对象常量形式，将 base64 编码签名复制并粘贴至令牌文件。


```
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAAABAAMAA
+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=="
    }
  ]
}
```

在 AWS CloudHSM 集群上批准令牌并执行用户管理操作

管理员获得必要的批准/签名（如上一节所述）后，管理员可以向 AWS CloudHSM 集群提供该令牌，并执行以下用户管理操作之一：

- 创建
- 删除
- 更改密码
- user change-mfa

有关使用这些命令的更多信息，请参阅使用 CloudHSM CLI。

在交易过程中，令牌将在 AWS CloudHSM 集群内获得批准并执行请求的用户管理操作。用户管理操作的成功取决于已批准的仲裁令牌和用户管理操作是否有效。

管理员只能将令牌用于一项操作。在该操作成功后，该令牌不再有效。若要执行其他 HSM 用户管理操作，管理员必须重复上述进程。那就是，管理员必须生成新的仲裁令牌、获取来自批准者的新签名，并按照所请求的用户管理操作批准和在 HSM 上使用新令牌。

> **Note**
>
> 仅当您当前登录会话打开时，仲裁令牌才有效。如果您登出 CloudHSM CLI 或者断开网络连接，则令牌将不再有效。同样，授权令牌只能用于 CloudHSM CLI 中。它无法用于在其他应用程序中进行身份验证。

Example 以管理员身份创建新用户

在以下示例命令中，已登录管理员将在 HSM 上创建一个新用户：

```
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"
  }
}
```

然后，管理员输入 user list 命令以确认创建新用户：

```
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"
      }
    ]
  }
}
```

如果管理员尝试执行其他 HSM 用户管理操作，该操作将失败并出现仲裁身份验证错误：

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

如下所示，该 quorum token-sign list 命令显示管理员未批准令牌。若要执行其他 HSM 用户管理操作，管理员必须生成新的仲裁令牌，从批准者处获取新签名，然后使用 --approver 参数执行所需的用户管理操作，以提供待批准并在用户管理操作中使用的仲裁令牌。

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

更改管理员的仲裁最小值

在设置仲裁最小值以便管理员能够使用仲裁身份验证后，您可能希望更改仲裁最小值。仅当审批者的数量大于或等于当前仲裁最小值时，HSM 才允许您更改 quorum 最小值。例如，如果仲裁最小值为二 (2)，则必须至少有两 (2) 个管理员批准更改仲裁最小值。

> **Note**
>
> 用户服务的仲裁值必须小于仲裁服务的仲裁值。有关服务名称（例如仲裁服务和用户服务）的信息，请参阅支持仲裁身份验证的服务名称和类型。

要获取更改仲裁最小值的仲裁批准，您需要一个使用 quorum token-sign set-quorum-value 命令用于 quorum service 的仲裁令牌。若要使用 quorum token-sign set-quorum-value 命令为 quorum service 生成仲裁令牌，仲裁服务值必须大于一 (1)。这意味着，您可能需要先更改用户服务的仲裁最小值，然后才能更改仲裁服务的仲裁最小值。

若要更改管理员的最小仲裁值

1. 使用以下命令启动 CloudHSM CLI 交互模式。

   Linux

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

   Windows

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

2. 使用 login 命令，并以管理员身份登录该集群。

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

3. 使用 quorum token-sign list-quorum-values 命令获取所有服务名称的仲裁最小值。有关更多信息，请参阅下面的示例。
4. 如果仲裁服务的仲裁最小值小于用户服务的相应值，请使用 quorum token-sign set-quorum-value 命令更改仲裁服务的值。将仲裁服务的值更改为一 (1)，等于或大于用户服务的值。有关更多信息，请参阅以下示例。
5. 生成一个仲裁令牌，并注意指定仲裁服务作为您可以对其使用令牌的服务。
6. 获得其他管理员的批准 (签名)。
7. 在 AWS CloudHSM 集群上批准令牌并执行用户管理操作。 。
8. 使用 quorum token-sign set-quorum-value 命令获取用户服务的仲裁最小值。

Example – 获取仲裁最小值并更改仲裁服务的相应值

以下示例命令显示了用户服务的仲裁最小值当前为二 (2)。

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

要更改仲裁服务的仲裁最小值，请使用 quorum token-sign set-quorum-value 命令，并设置一个等于或大于用户服务的相应值的值。以下示例将仲裁服务的仲裁最小值设置为二 (2)，与为用户服务设置的值相同。

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

以下命令显示了用户服务和仲裁服务的仲裁最小值现在为二 (2)。

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

使用 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility) 管理 HSM 用户

在中 AWS CloudHSM，您必须使用 C loudHSM CLI 或 CloudHSM 管理实用程序 (CMU) 命令行工具在您的 H SM 上创建和管理用户。CloudHSM CLI 可与最新的 SDK 配合使用，而 CMU 则可与之前的 SDK 配合使用。

主题

- 了解 HSM 用户

- HSM 用户权限表
- 使用 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility) 管理用户
- 使用 CloudHSM 管理实用程序（CMU）管理加密员（CO）的双重身份验证（2FA）
- 使用 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility) 管理仲裁身份验证（M of N 访问控制）

了解 HSM 用户

您在 HSM 上执行的大多数操作都需要 HSM 用户 的凭证。HSM 对每个 HSM 用户进行身份验证，每个 HSM 用户都有一种类型，其可确定您可以在 HSM 上以该用户身份执行的操作。

> **Note**
>
> HSM 用户与 IAM 用户不同。拥有正确证书的 IAM 用户可通过 AWS API 与资源交互来创建 HSM。创建 HSM 后，您必须使用 HSM 用户凭证对 HSM 上的操作进行身份验证。

用户类型

- 准加密员 (PRECO)
- 加密员 (CO, Crypto officer)
- 加密用户 (CU)
- 设备用户 (AU)

准加密员 (PRECO)

在云管理实用程序 (CMU, cloud management utility) 和密钥管理实用程序 (KMU, key management utility) 中，PRECO 都是临时用户，仅存在于 AWS CloudHSM 集群中的第一个 HSM 上。新集群中的第一个 HSM 包含一个 PRECO 用户，表示该集群从未被激活。要激活集群，请执行 cloudhsm-cli 并运行 cluster activate 命令。登录该 HSM 并更改 PRECO 的密码。更改密码后，该用户就会变成加密员 (CO, crypto officer)。

加密员 (CO, Crypto officer)

在云管理实用程序 (CMU, cloud management utility) 和密钥管理实用程序 (KMU, key management utility) 中，加密员（CO）都可以执行用户管理操作。例如，它们可以创建和删除用户以及更改用户密码。有关 CO 用户的更多信息，请参阅 HSM 用户权限表。当您激活新集群时，此用户将从准加密员（PRECO, Precrypto Officer）更改为加密员 (CO, crypto officer)。-->

加密用户 (CU)

加密用户 (CU) 可以执行以下密钥管理和加密操作。

- 密钥管理 – 创建、删除、共享、导入和导出加密密钥。
- 加密操作 – 使用加密密钥来执行加密、解密、签名、验证及更多操作。

有关更多信息，请参阅 HSM 用户权限表。

设备用户 (AU)

设备用户 (AU) 可以在集群的 HSM 上执行克隆和同步操作。 AWS CloudHSM 使用 AU 同步 AWS CloudHSM 集群中的 HSM。AU 存在于由提供的所有 HSM 上 AWS CloudHSM，并且权限有限。有关更多信息，请参阅 HSM 用户权限表。

AWS 无法对您的 HSM 执行任何操作。 AWS 无法查看或修改您的用户或密钥，也无法使用这些密钥执行任何加密操作。

HSM 用户权限表

下表列出了按可以执行该操作的 HSM 用户或会话类型排序的 HSM 操作。

	加密员 (CO, Crypto officer)	加密用户 (CU)	设备用户 (AU)	未经身份验证的会话
获取基本集群信息[1]	是	是	是	是
更改自己的密码	是	是	是	不适用
更改任意用户的密码	是	不支持	不支持	否

	加密员 (CO, Crypto officer)	加密用户 (CU)	设备用户 (AU)	未经身份验证的会话
添加、删除用户	 是	 不支持	 不支持	 否
获取同步状态[2]	 是	 是	 是	 不支持
提取、插入遮蔽对象[3]	 是	 是	 是	 不支持
密钥管理功能[4]	 否	 是	 不支持	 否
加密、解密	 否	 是	 不支持	 否
签署、验证	 否	 是	 不支持	 否
生成摘要和 HMAC	 否	 是	 不支持	 否

- [1] 基本集群信息包括集群中 HSM 的数量和每个 HSM 的 IP 地址、型号、序列号、设备 ID、固件 ID 等。
- [2] 用户可以获取与 HSM 上的密钥对应的一组摘要（哈希值）。应用程序可以将这些摘要集进行比较来了解集群中 HSM 的同步状态。
- [3] 遮蔽对象是在离开 HSM 之前进行加密的密钥。它们无法在 HSM 外部加密。只有在将其插入 HSM 之后，并且该 HSM 所在集群与提取它们时的 HSM 的集群相同，才会解密它们。应用程序可以提取和插入遮蔽对象，以用于同步集群中的 HSM。
- [4] 密钥管理功能包括创建、删除、包装、解开包装和修改密钥的属性。

使用 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility) 管理用户

本主题提供有关使用 CloudHSM 管理实用程序 (CMU)（客户端 SDK 附带的命令行工具）管理硬件安全模块 (HSM) 用户的 step-by-step 说明。有关 CMU 或 HSM 用户的更多信息，请参阅 CloudHSM 管理实用程序 和 了解 HSM 用户。

Sections

- 使用 CMU 了解 HSM 用户管理
- 下载 CloudHSM 管理实用程序
- 如何使用 CMU 管理 HSM 用户

使用 CMU 了解 HSM 用户管理

要管理 HSM 用户，您必须使用加密员 (CO, cryptographic officer) 的用户名和密码登录到 HSM。只有 CO 才能管理用户。HSM 包含名为 admin 的默认 CO。在激活集群时，您可以设置 admin 的密码。

要使用 CMU，必须使用配置工具更新本地配置。CMU 创建自己的集群连接，但此连接不具备集群感知功能。为了跟踪集群信息，CMU 会维护一个本地配置文件。这意味着，每次使用 CMU 时，都应首先通过运行带有 `--cmu` 参数的配置命令行工具来更新配置文件。如果您正在使用客户端软件开发工具包 3.2.1 或更早版本，则必须使用与 `--cmu` 不同的参数。有关更多信息，请参阅 the section called “通过客户端软件开发工具包 3.2.1 及更早版本使用 CMU”。

`--cmu` 参数要求您在集群中添加 HSM 的 IP 地址。如果您有多个 HSM，则可以使用任一 IP 地址。这可确保 CMU 可以将您所做的任何变更传播到整个集群中。请记住，CMU 使用其本地文件来跟踪集群信息。如果自上次在特定主机上使用 CMU 以来该集群已更改，则必须将此类更改添加到存储在该主机上的本地配置文件中。使用 CMU 时，切勿添加或删除 HSM。

获取 HSM 的 IP 地址（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 要更改 Amazon Web Services Region，请使用页面右上角的区域选择器（Region selector）。
3. 要打开该集群的详细信息页面，请在集群表中选择集群 ID。
4. 要获取 IP 地址，请在 HSM 选项卡上，选择 ENI IP 地址下列出的 IP 地址之一。

获取 HSM 的 IP 地址 ()AWS CLI

- 使用 AWS CLI中的 describe-clusters 命令获取 HSM 的 IP 地址。在该命令的输出中，HSM 的 IP 地址为 `EniIp` 的值。

```
$  aws cloudhsmv2 describe-clusters


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

通过客户端软件开发工具包 3.2.1 及更早版本使用 CMU

在 Client SDK 3.3.0 AWS CloudHSM 中，增加了对`--cmu`参数的支持，这简化了更新 CMU 配置文件的过程。如果您正在使用来自客户端软件开发工具包 3.2.1 或更早版本的 CMU，则必须继续使用 `-a` 和 `-m` 参数来更新配置文件。有关这些参数的更多信息，请参阅配置工具。

下载 CloudHSM 管理实用程序

无论您正在使用客户端软件开发工具包 5 还是客户端软件开发工具包 3，最新版本的 CMU 都可用于 HSM 用户管理任务。

下载并安装 CMU。

下载并安装 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. 下载 CloudHSM 管理实用程序。
2. 使用 Windows 管理权限运行 CMU 安装程序 (AWSCloudHSMManagementUtil-latest.msi)。

Windows Server 2012 R2

1. 下载 CloudHSM 管理实用程序。
2. 使用 Windows 管理权限运行 CMU 安装程序 (AWSCloudHSMManagementUtil-latest.msi)。

Windows Server 2016

1. 下载 CloudHSM 管理实用程序。
2. 使用 Windows 管理权限运行 CMU 安装程序 (AWSCloudHSMManagementUtil-latest.msi)。

如何使用 CMU 管理 HSM 用户

本节包括使用 CMU 管理 HSM 用户的基本命令。

创建 HSM 用户

createUser 用于在 HSM 上创建新用户。必须以 CO 身份登录才能创建用户。

创建一个新的 CO 用户

1. 使用配置工具更新 CMU 配置。

   Linux

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

   Windows

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

2. 启动 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 用户身份登录 HSM。

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

确保 CMU 列出的连接数与集群中的 HSM 数量相匹配。如果没有，请注销并重新开始。

4. 使用 createUser 创建名为 **example_officer** 的 CO 用户，密码为 **password1**。

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

CMU 会提示您关于创建用户操作的信息。

```
**************************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. 键入 **y**。

创建一个新的 CU 用户

1. 使用配置工具更新 CMU 配置。

   Linux

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

   Windows

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

2. 启动 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 用户身份登录 HSM。

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

确保 CMU 列出的连接数与集群中的 HSM 数量相匹配。如果没有，请注销并重新开始。

4. 使用 createUser 创建名为 **example_user** 的 CU 用户，密码为 **password1**。

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

CMU 会提示您关于创建用户操作的信息。

```
************************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. 键入 **y**。

有关 createUser 的更多信息，请参阅 createUser。

列出集群上的所有 HSM 用户

使用 listUsers 命令列出该集群中的所有用户。您无需登录即可运行 listUsers，且所有用户类型都可以列出用户。

列出集群上的所有用户

1. 使用配置工具更新 CMU 配置。

Linux

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

Windows

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

2. 启动 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. 使用 listUsers 列出集群上的所有用户。

```
aws-cloudhsm>listUsers
```

CMU 会列出集群上的所有用户。

```
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
```

有关 listUsers 的更多信息，请参阅 listUsers。

要更改 HSM 用户密码

要更改用户密码，请使用 changePswd。

用户类型和密码区分大小写，但是用户名不区分大小写。

CO、加密用户（CU）和应用程序用户（AU）能更改自己的密码。要更改其他用户的密码，您必须以 CO 身份登录。您无法更改当前已登录用户的密码。

更改您自己的密码

1. 使用配置工具更新 CMU 配置。

   Linux

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

   Windows

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

2. 启动 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. 登录到 HSM。

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

   确保 CMU 列出的连接数与集群中的 HSM 数量相匹配。如果没有，请注销并重新开始。

4. 使用 changePswd 更改您自己的密码。

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

   CMU 会提示您更改密码的操作。

   ```
   ************************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. 键入 **y**。

   CMU 会提示您更改密码的操作。

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

更改其他用户的密码

1. 使用配置工具更新 CMU 配置。

   Linux

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

   Windows

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

2. 启动 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 用户身份登录 HSM。

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

   确保 CMU 列出的连接数与集群中的 HSM 数量相匹配。如果没有，请注销并重新开始。

4. 使用 changePswd 更改其他用户的密码。

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

   CMU 会提示您更改密码的操作。

   ```
   
   **********************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. 键入 **y**。

   CMU 会提示您更改密码的操作。

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

有关 changePswd 的更多信息，请参阅 changePswd。

删除 HSM 用户

使用 deleteUser 删除用户。您必须以 CO 身份登录才能删除其他用户。

> **Tip**
> 您无法删除拥有密钥的加密用户 (CU)。

删除用户

1. 使用配置工具更新 CMU 配置。

   Linux

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

   Windows

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

2. 启动 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 用户身份登录 HSM。

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

确保 CMU 列出的连接数与集群中的 HSM 数量相匹配。如果没有，请注销并重新开始。

4. 使用 deleteUser 删除用户。

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

CMU 会删除该用户。

```
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)
```

有关 deleteUser 的更多信息，请参阅 deleteUser。

使用 CloudHSM 管理实用程序（CMU）管理加密员（CO）的双重身份验证（2FA）

为了提高安全性，可以配置双重身份验证（2FA）来帮助保护集群。您只能为加密员（CO）启用 2FA。

> **Note**
>
> 您无法为加密用户（CU）或应用程序启用 2FA。双重身份验证（2FA）仅适用于 CO 用户。

主题

- 了解 HSM 用户的 2FA
- 为 HSM 用户使用 2FA

了解 HSM 用户的 2FA

使用启用 2FA 的硬件服务模（HSM）账户登录集群时，您需要向 cloudhsm_mgmt_util（CMU）提供您的密码（第一重身份，即您所知道的），CMU 将为您提供令牌并提示您签署令牌。要提供第二重身份（即您所拥有的），您可以使用已经创建并与 HSM 用户关联的密钥对中的私有密钥签署令牌。要访问集群，您需要向 CMU 提供已签署的令牌。

仲裁身份验证和双因素身份验证

集群使用相同的密钥进行仲裁身份验证和 2FA。这表明启用了 2FA 的用户已经进行了有效的 M-of-N-access-control (MofN) 注册。要成功使用同一 HSM 用户的 2FA 和仲裁身份验证，请考虑以下事项：

- 如果现在对用户使用仲裁身份验证，则应使用为仲裁用户创建的同一密钥对，为该用户启用 2FA。
- 如果为非仲裁身份验证用户的非 2FA 用户增加了 2FA 要求，则可以将该用户注册为采用 2FA 身份验证的 MofN 用户。
- 如果删除了 2FA 要求或更改了同时也是仲裁身份验证用户的 2FA 用户的密码，则将同时删除该仲裁用户的 MofN 用户注册。
- 如果删除了 2FA 要求或更改了同时也是仲裁身份验证用户的 2FA 用户的密码，但仍希望该用户参与仲裁身份验证，则必须将该用户重新注册为 MofN 用户。

有关仲裁身份验证的更多信息，请参阅 使用 CMU 管理仲裁身份认证。

为 HSM 用户使用 2FA

本节将介绍如何对 HSM 用户进行 2FA，包括创建 2FA HSM 用户、轮换密钥以及以启动 2FA 的用户身份登录 HSM。有关使用 HSM 用户的更多信息，请参阅 ???、???、???、??? 和 ???。

创建 2FA 用户

要对 HSM 用户启用 2FA，请使用满足以下要求的密钥。

2FA 密钥对要求

您可以创建新的密钥对或使用满足以下要求的现有密钥。

- 密钥类型：非对称密钥
- 密钥用法：签名和验证
- 密钥规范：RSA_2048
- 签名算法包含：

- sha256WithRSAEncryption

> **Note**
>
> 如果您正在使用仲裁身份验证或计划使用仲裁身份验证，请参阅 the section called “仲裁身份验证和双因素身份验证”。

使用 CMU 和密钥对创建启用了 2FA 的新 CO 用户。

创建启用 2FA 的 CO 用户

1. 在一个终端中执行以下步骤：

   a. 访问您的 HSM 并登录 CloudHSM 管理实用程序：

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

   b. 以 CO 身份登录，使用以下命令创建启用 2FA 的新用户 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. 使上述终端保持该状态。请勿按回车键或输入任何文件名。

2. 在另一个终端中执行以下步骤：

a. 访问您的 HSM 并登录 CloudHSM 管理实用程序：

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

b. 使用以下命令生成公有密钥/私有密钥对：

```
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. 运行以下命令安装 json 查询功能，以从 authdata 文件中提取摘要：

```
sudo yum install jq
```

d. 要提取摘要值，请先在 authdata 文件中找到以下数据：

```
{
  "Version":"1.0",
  "PublicKey":"",
  "Data":[
    {
      "HsmId": <"HSM ID">,
      "Digest": <"DIGEST">,
      "Signature":""
    }
  ]
}
```

> **Note**
>
> 所得摘要采用 base64 编码，但如要签署摘要，则需先解码文件再进行签名。以下命令将解码摘要并将解码后的内容存储在 ‘digest1.bin’ 中
>
> ```
> cat authdata | jq '.Data[0].Digest' | cut -c2- | rev | cut -c2- | rev |
>  base64 -d > digest1.bin
> ```

e. 转换公有密钥内容，添加 "\n" 并删除空格，如下所示：

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

> **Important**
>
> 上述命令显示了如何在 BEGIN PUBLIC KEY----- 后立即添加 "\n"、如何删除 "\n" 与公有密钥的第一个字符之间的空格，如何在 -----END PUBLIC KEY 前添加 "\n"，以及如何删除 "\n" 与公有密钥结尾处之间的空格。

这是 authdata 文件中接受的公有密钥的 PEM 格式。

f. 将公有密钥 pem 格式的内容粘贴到 authdata 文件中的公有密钥部分。

```
vi authdata
```

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

g. 使用以下命令签署令牌文件：

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

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

> **Note**
>
> 如上述命令所示，使用 openssl pkeyutl 而不是 openssl dgst 进行签名。

h. 在 Authdata 文件的“签名”字段中添加已签名的摘要。

```
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. 返回第一个终端并按 **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 HSM 用户的 2FA

通过更改密码来更改 2FA 用户的密码、启用或禁用 2FA 或轮换 2FA 密钥。每次启用 2FA 时，均须为 2FA 登录提供公有密钥。

更改密码执行以下任何场景：

- 更改 2FA 用户的密码
- 更改非 2FA 用户的密码
- 对非 2FA 用户添加 2FA
- 删除 2FA 用户的 2FA
- 轮换 2FA 用户的密钥

您还可以合并任务。例如，您可以删除用户的 2FA 并同时更改密码，或轮换 2FA 密钥并更改用户密码。

更改启用 2FA 的 CO 用户的密码或轮换其密钥

1. 通过 CMU 以启用 2FA 的 CO 身份登录 HSM。
2. 使用 changePswd 更改启用 2FA 的 CO 用户的密码或轮换其密钥。使用 `-2fa` 参数并在文件系统中添加系统写入 `authdata` 文件的位置。此文件包含集群中每个 HSM 的摘要。

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

   CMU 将提示您使用私有密钥签署 `authdata` 文件中的摘要，并使用公有密钥返回签名。
3. 使用私有密钥签署 `authdata` 文件中的摘要，将签名和公有密钥添加到 JSON 格式 `authdata` 的文件中，然后向 CMU 提供 `authdata` 文件的位置。有关更多信息，请参阅 the section called “配置参考”。

> **Note**
>
> 集群使用相同的密钥进行仲裁身份验证和双因素身份验证。如果您正在使用仲裁身份验证或计划使用仲裁身份验证，请参阅 the section called “仲裁身份验证和双因素身份验证”。

要禁用启用 2FA 的 CO 用户的 2FA

1. 通过 CMU 以启用 2FA 的 CO 身份登录 HSM。
2. 使用 changePswd 移除启用 2FA 的 CO 用户的 2FA。

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

CMU 将提示您确认更改密码操作。

> **Note**
>
> 如果删除了 2FA 要求或更改了同时也是仲裁身份验证用户的 2FA 用户的密码，则将同时删除该仲裁用户的 MofN 用户注册。有关仲裁用户和 2FA 的更多信息，请参阅 the section called “仲裁身份验证和双因素身份验证”。

3. 键入 **y**。

   CMU 将确认更改密码操作。

配置参考

以下是 CMU 生成的请求和您的响应在 `authdata` 文件中的 2FA 属性的示例。

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

数据

顶级节点。包含集群中每个 HSM 的从属节点。出现在所有 2FA 命令的请求和响应中。

摘要

这是您必须签名才能提供第二重身份验证。在所有 2FA 命令的请求中生成的 CMU。

HsmId

您的 HSM 的 ID。出现在所有 2FA 命令的请求和响应中。

PublicKey

作为 PEM 格式的字符串插入的您所生成的密钥对的公有密钥部分。您输入此内容以响应 createUser 和 changePswd。

签名

Base 64 编码的签名摘要。您输入此内容以响应所有 2FA 命令。

版本

JSON 格式的身份验证数据文件的版本。出现在所有 2FA 命令的请求和响应中。

使用 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility) 管理仲裁身份验证（M of N 访问控制）

AWS CloudHSM 集群中的 HSM 支持法定身份验证，也称为 M of N 访问控制。对于仲裁身份验证，HSM 上的单个用户不能在 HSM 上执行仲裁控制型操作。必须有最少数量 (至少 2 个) 的 HSM 用户合作才能执行这些操作。对于仲裁身份验证，要求多个 HSM 用户批准，这样可以增加一层额外的保护。

仲裁身份验证可以控制以下操作：

- 由加密员（CO）进行的 HSM 用户管理：创建和删除 HSM 用户，以及更改其他 HSM 用户的密码。有关更多信息，请参阅 使用面向加密员（CO）的 Quorum 身份验证。

下列主题提供了有关 AWS CloudHSM中的仲裁身份验证的更多信息。

主题

- 仲裁身份验证概述
- 有关仲裁身份验证的其他详细信息
- 使用面向加密员（CO）的仲裁身份验证：首次设置
- 使用面向加密员（CO）的 Quorum 身份验证
- 为加密员（CO）更改仲裁最小值

仲裁身份验证概述

下列步骤概括了仲裁身份验证过程。有关特定步骤和工具，请参阅使用面向加密员（CO）的 Quorum 身份验证。

1. 每个 HSM 用户都将创建用于签名的非对称密钥。他们在 HSM 外部执行此操作，并注意适当保护密钥。
2. 每个 HSM 用户都将登录 HSM 并向 HSM 注册其签名密钥的公有部分 (公有密钥)。
3. 当 HSM 用户要执行仲裁控制型操作时，该用户将登录 HSM 并获取仲裁令牌。
4. HSM 用户将仲裁令牌提供给一个或多个其他 HSM 用户并请求其批准。
5. 其他 HSM 用户通过使用其密钥对仲裁令牌进行加密签名来进行批准。上述操作是在 HSM 外部进行的。
6. 当 HSM 用户获得所需数量的批准后，同一用户将登录 HSM 并向 HSM 提供仲裁令牌和批准 (签名)。
7. HSM 将使用每个签署人的已注册公有密钥来验证签名。如果签名有效，HSM 将批准此令牌。
8. HSM 用户现在可以执行仲裁控制型操作。

有关仲裁身份验证的其他详细信息

注意以下有关在 AWS CloudHSM中使用仲裁身份验证的更多信息。

- HSM 用户可为自己的仲裁令牌签名，也就是说，请求用户可提供仲裁身份验证所需的批准之一。
- 为仲裁控制型操作选择最小数量的仲裁审批者。您可以选择的最小数字是二（2），您可以选择的最大数字是八（8）。
- HSM 可容纳多达 1024 个仲裁令牌。如果 HSM 在您尝试创建新的令牌时已有 1024 个令牌，HSM 将清除过期令牌之一。默认情况下，令牌将在创建 10 分钟后过期。
- 集群使用相同的密钥进行仲裁身份验证和双重身份验证 (2FA, two-factor authentication)。有关使用仲裁身份验证和双因素身份验证的更多信息，请参阅仲裁身份验证和双因素身份验证。

使用面向加密员（CO）的仲裁身份验证：首次设置

下列主题描述要配置硬件安全模块（HSM, hardware security module）必须完成的步骤，以便加密员（CO） 可以使用仲裁身份验证。当您首次为 CO 配置仲裁身份验证时，只需执行下列步骤一次。完成这些步骤后，请参阅 使用面向加密员（CO）的 Quorum 身份验证。

主题

- 先决条件
- 创建并注册签名密钥
- 在 HSM 上设置仲裁最小值

先决条件

要理解此示例，应熟悉 cloudhsm_mgmt_util (CMU) 命令行工具。在此示例中， AWS CloudHSM 集群有两个 HSM，每个 HSM 都有相同的 CoS，如listUsers命令的以下输出所示。有关创建用户的更多信息，请参见 管理 HSM 用户。

```
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
```

创建并注册签名密钥

要使用仲裁身份验证，每个 CO 都必须完成以下所有步骤：

主题

- 创建 RSA 密钥对
- 创建注册令牌并签名
- 通过 HSM 注册公钥

创建 RSA 密钥对

创建和保护密钥对的方式有多种。以下示例说明如何使用 OpenSSL 执行该操作。

Example - 使用 OpenSSL 创建私有密钥

以下示例演示如何使用 OpenSSL 创建受密码保护的 2048 位 RSA 密钥。要使用此示例，请将 *officer1.key* 替换为您要存储密钥的文件的名称。

```
$ 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:
```

接下来，使用您刚刚创建的私钥生成公有密钥。

Example - 使用 OpenSSL 创建公有密钥

以下示例演示如何使用 OpenSSL，根据您刚刚创建的私有密钥创建公钥。

```
$ openssl rsa -in officer1.key -outform PEM -pubout -out officer1.pub
Enter pass phrase for officer1.key:
writing RSA key
```

创建注册令牌并签名

创建一个令牌，并使用上一步生成的私有密钥签名。

Example - 创建令牌

注册令牌只是一个包含任何随机数据的文件，其大小不超过 245 字节的最大值。您通过私有密钥对令牌进行签名，以证明您有权访问私钥。以下命令使用“回显”将字符串重定向到文件。

```
$ echo "token to be signed" > officer1.token
```

对令牌进行签名并将其保存到签名文件中。您需要已签名令牌、未签名令牌和公有密钥才能在 HSM 中将 CO 注册为 MofN 用户。

Example ：签署令牌

使用 OpenSSL 和私钥对注册令牌签名，并创建签名文件：

```
$ openssl dgst -sha256 \
    -sign officer1.key \
    -out officer1.token.sig officer1.token
```

通过 HSM 注册公钥

在创建密钥之后，CO 必须向 HSM 注册密钥的公有部分 (公有密钥)。

向 HSM 注册公有密钥

1. 使用以下命令启动 cloudhsm_mgmt_util 命令行工具。

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

2. 使用 loginHSM 命令以 CO 身份登录 HSM。有关更多信息，请参阅 ???。
3. 使用 registerQuorumPubKey 命令注册公有密钥。有关更多信息，请参阅以下示例或使用 help registerQuorumPubKey 命令。

Example ：向 HSM 注册公有密钥

以下示例说明如何使用 cloudhsm_mgmt_util 命令行工具中的 registerQuorumPubKey 命令向 HSM 注册 CO 的公有密钥。要使用此命令，CO 必须登录 HSM。将这些值替换为您自己的值：

```
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>

包含未签名注册令牌的文件路径。可以有最大文件大小为 245 字节的任何随机数据。

必需：是

<officer1.token.sig>

包含注册令牌 SHA256_PKCS 机制签名哈希的文件路径。

必需：是

<officer1.pub>

包含非对称 RSA-2048 密钥对公有密钥的文件路径。使用私有密钥对注册令牌进行签名。

必需：是

在所有 CO 注册自己的公有密钥之后，来自 listUsers 命令的输出将在 MofnPubKey 列中显示它，如以下示例中所示。

```
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
```

在 HSM 上设置仲裁最小值

要使用 CO 仲裁身份验证，CO 必须登录 HSM，然后设置仲裁最小值 (也称为 m 值)。这是执行 HSM 用户管理操作所需的最少数量的 CO 审批。HSM 上的任何 CO (包括尚未注册签名密钥的 CO) 都可设置仲裁最小值。您可以随时更改仲裁最小值；有关更多信息，请参阅更改最小值。

在 HSM 上设置仲裁最小值

1. 使用以下命令启动 cloudhsm_mgmt_util 命令行工具。

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

2. 使用 loginHSM 命令以 CO 身份登录 HSM。有关更多信息，请参阅 ???。
3. 使用 setMValue 命令设置仲裁最小值。有关更多信息，请参阅以下示例或使用 help setMValue 命令。

Example - 在 HSM 上设置仲裁最小值

此示例使用仲裁最小值 2。您可以选择介于二 (2) 到八 (8) 范围之内的任何值 (最多为 HSM 上的 CO 总数量)。在此示例中，HSM 有 6 个 CO，因此最大可取值为 6。

要使用以下示例命令，请将最终数量 (*2*) 替换为首选仲裁最小值。

```
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
```

在前面的示例中，第一个数字 (3) 标识您将为其设置仲裁最小值的 HSM 服务。

下表列出了 HSM 服务标识符及其名称、描述和服务所包含命令。

服务标识符	服务名称	服务描述	HSM 命令
3	USER_MGMT	HSM 用户管理	• createUser • deleteUser • changePswd (仅适用于更改其他 HSM 用户的密码)

服务标识符	服务名称	服务描述	HSM 命令
4	MISC_CO	其他 CO 服务	• setMValue

要获取服务的仲裁最小值，请使用 getMValue 命令，如以下示例所示。

```
aws-cloudhsm>getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

来自前面的 getMValue 命令的输出显示，HSM 用户管理操作（服务 3）的仲裁最小值现在为 2。

完成这些步骤后，请参阅 使用面向加密员（CO）的 Quorum 身份验证。

使用面向加密员（CO）的 Quorum 身份验证

HSM 上的加密管理者 (CO) 可为 HSM 上的以下操作配置 quorum 身份验证：

- 创建 HSM 用户
- 删除 HSM 用户
- 更改其他 HSM 用户的密码

在针对 quorum 身份验证配置 HSM 后，CO 无法自行执行 HSM 用户管理操作。以下示例显示 CO 尝试在 HSM 上创建新用户时的输出。该命令失败，并出现 RET_MXN_AUTH_FAILED 错误，这指示 quorum 身份验证失败。

```
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
```

要执行 HSM 用户管理操作，CO 必须完成以下任务：

1. 获取 quorum 令牌。
2. 获得其他 CO 的批准 (签名)。
3. 批准 HSM 上的令牌。
4. 执行 HSM 用户管理操作。

如果您尚未为 CO 的 quorum 身份验证配置 HSM，请立即执行此操作。有关更多信息，请参阅 首次设置。

获取仲裁令牌

首先，CO 必须使用 cloudhsm_mgmt_util 命令行工具请求仲裁令牌。

获取仲裁令牌

1. 使用以下命令启动 cloudhsm_mgmt_util 命令行工具。

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

2. 使用 loginHSM 命令以 CO 身份登录 HSM。有关更多信息，请参阅 ???。
3. 使用 getToken 命令获取仲裁令牌。有关更多信息，请参阅以下示例或使用 help getToken 命令。

Example ：获取仲裁令牌

此示例将为用户名为 officer1 的 CO 获取一个 quorum 令牌，并将该令牌保存到一个名为 `officer1.token` 的文件中。要使用示例命令，可将这些值替换为您自己的值：

- *officer1*：要获取令牌的 CO 的姓名。这必须是已登录 HSM 并且正在运行此命令的 CO。
- *officer1.token*：用于存储仲裁令牌的文件的名称。

在以下命令中，3 标识可将获取的令牌用于的服务。在此示例中，令牌用于 HSM 用户管理操作 (服务 3)。有关更多信息，请参阅 在 HSM 上设置仲裁最小值。

```
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
```

获得批准 CO 的签名

具有 quorum 令牌的 CO 必须获得其他 CO 批准的令牌。为了提供其批准，其他 CO 使用其签名密钥以加密方式对令牌进行签名。他们在 HSM 外部执行此操作。

可通过多种不同方式对令牌进行签名。以下示例说明如何使用 OpenSSL 执行该操作。要使用其他签名工具，请确保该工具使用 CO 的私有密钥 (签名密钥) 对令牌的 SHA-256 摘要进行签名。

Example ：获得批准 CO 的签名

在此示例中，具有令牌的 CO (officer1) 至少需要两次批准。以下示例命令说明两个 CO 如何使用 OpenSSL 以加密方式对令牌进行签名。

在第一条命令中，officer1 对自己的令牌进行签名。要使用以下示例命令，请将这些值替换为您自己的值：

- *officer1.key* 和 *officer2.key*：包含 CO 的签名密钥的文件的名称。
- *officer1.token.sig1* 和 *officer1.token.sig2*：要用于存储签名的文件的名称。确保将每个签名保存在不同的文件中。
- *officer1.token*：包含 CO 正在签名的令牌的文件的名称。

```
$ 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 将对同一令牌进行签名。

```
$ openssl dgst -sha256 -sign officer2.key -out officer1.token.sig2 officer1.token
Enter pass phrase for officer2.key:
```

批准 HSM 上的已签名令牌

在一个 CO 获得来自其他 CO 的最小数量的批准 (签名) 后，该 CO 必须批准 HSM 上的已签名令牌。

批准 HSM 上的已签名令牌

1. 创建一个令牌批准文件。有关更多信息，请参阅以下示例。
2. 使用以下命令启动 cloudhsm_mgmt_util 命令行工具。

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

3. 使用 loginHSM 命令以 CO 身份登录 HSM。有关更多信息，请参阅 ???。
4. 使用 approveToken 命令批准已签名令牌，并传递令牌批准文件。有关更多信息，请参阅以下示例。

Example - 创建一个令牌批准文件并批准 HSM 上的已签名令牌

令牌批准文件是一个采用了 HSM 要求的特定格式的文本文件。该文件包含有关令牌、其批准者和批准者签名的信息。下面显示了示例令牌批准文件。

```
# 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;
```

创建令牌批准文件后，CO 使用 cloudhsm_mgmt_util 命令行工具登录 HSM。随后，CO 使用 approveToken 命令批准令牌，如以下示例中所示。将 *approval.txt* 替换为令牌批准文件的名称。

```
aws-cloudhsm>approveToken approval.txt
approveToken success on server 0(10.0.2.14)
approveToken success on server 1(10.0.1.4)
```

如果此命令成功，则表示 HSM 已批准 quorum 令牌。要查看令牌的状态，请使用 listTokens 命令，如以下示例中所示。此命令的输出表明，令牌拥有所需数量的批准。

令牌有效时间指示保证令牌在 HSM 上保留的时间长度。甚至在令牌有效时间结束 (零秒) 后，您仍可以使用令牌。

```
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
```

将令牌用于用户管理操作

在 CO 拥有具有所需数量的批准的令牌后 (如上一部分中所示)，CO 可以执行下列 HSM 用户管理操作之一：

- 使用 createUser 命令创建 HSM 用户
- 使用 deleteUser 命令删除 HSM 用户
- 使用 changePswd 命令更改其他 HSM 用户的密码

有关使用这些命令的更多信息，请参阅管理 HSM 用户。

CO 只能将令牌用于一项操作。在该操作成功后，该令牌不再有效。要执行其他 HSM 用户管理操作，CO 必须获取新的 quorum 令牌、获取来自批准者的新签名，并批准 HSM 上的新令牌。

> **Note**
>
> 仅当您当前登录会话打开时，MofN 令牌才有效。如果您注销 cloudhsm_mgmt_util 或网络连接断开，则该令牌将不再有效。同样，授权令牌只能在 cloudhsm_mgmt_util 中使用，不能用于在其他应用程序中进行身份验证。

在以下示例命令中，CO 将在 HSM 上创建一个新用户。

```
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
```

在上一条命令成功后，后续 listUsers 命令将显示新用户。

```
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 尝试执行其他 HSM 用户管理操作，该操作将失败并出现 quorum 身份验证错误，如以下示例中所示。

```
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 命令指明 CO 没有已批准令牌，如以下示例中所示。要执行其他 HSM 用户管理操作，CO 必须获取新的 quorum 令牌、获取来自批准者的新签名，并批准 HSM 上的新令牌。

```
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
```

为加密员（CO）更改仲裁最小值

在设置 quorum 最小值以便加密管理者 (CO) 能够使用 quorum 身份验证后，您可能希望更改 quorum 最小值。仅当审批者的数量大于或等于当前仲裁最小值时，HSM 才允许您更改 quorum 最小值。例如，如果 quorum 最小值为 2，则必须至少有两个 CO 批准更改 quorum 最小值。

要获取更改仲裁最小值的仲裁批准，您需要一个用于 setMValue 命令 (服务 4) 的仲裁令牌。要获取用于 setMValue 命令（服务 4）的仲裁令牌，服务 4 的仲裁最小值必须大于 1。这意味着，您可能需要先更改服务 4 的 quorum 最小值，然后才能更改 CO (服务 3) 的 quorum 最小值。

下表列出了 HSM 服务标识符及其名称、描述和服务所包含命令。

服务标识符	服务名称	服务描述	HSM 命令
3	USER_MGMT	HSM 用户管理	• createUser • deleteUser • changePswd（仅适用于更改其他 HSM 用户的密码）
4	MISC_CO	其他 CO 服务	• setMValue

为加密员更改仲裁最小值

1. 使用以下命令启动 cloudhsm_mgmt_util 命令行工具。

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

2. 使用 loginHSM 命令以 CO 身份登录 HSM。有关更多信息，请参阅 ???。
3. 使用 getMValue 命令获取服务 3 的仲裁最小值。有关更多信息，请参阅以下示例。
4. 使用 getMValue 命令获取服务 4 的仲裁最小值。有关更多信息，请参阅以下示例。
5. 如果服务 4 的仲裁最小值小于服务 3 的相应值，请使用 setMValue 命令更改服务 4 的值。将服务 4 的值更改为大于或等于服务 3 的值。有关更多信息，请参阅以下示例。
6. 获取一个 quorum 令牌，并注意指定服务 4 作为您可以对其使用令牌的服务。
7. 获得其他 CO 的批准 (签名)。

8. 批准 HSM 上的令牌。
9. 使用 setMValue 命令更改服务 3（由 CO 执行的用户管理操作）的仲裁最小值。

Example – 获取仲裁最小值并更改服务 4 的相应值

以下示例命令显示了服务 3 的 quorum 最小值当前为 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]
```

以下示例命令显示了服务 4 的 quorum 最小值当前为 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]
```

要更改服务 4 的仲裁最小值，请使用 setMValue 命令，并设置一个大于或等于服务 3 的相应值的值。以下示例将服务 4 的 quorum 最小值设置为二 (2)，与为服务 3 设置的值相同。

```
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
```

以下命令显示了服务 3 和服务 4 的 quorum 最小值现在为 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]
```

管理中的密钥 AWS CloudHSM

在中 AWS CloudHSM，使用以下任一方法来管理集群中 HSM 上的密钥：

- PKCS #11 库
- JCE 提供程序
- CNG 和 KSP 提供程序
- CloudHSM CLI

要管理密钥，必须使用加密用户（CU）的用户名和密码登录 HSM。只有 CU 可以创建密钥。创建密钥的 CU 拥有并管理该密钥。

主题

中的密钥同步和耐久性设置 AWS CloudHSM

本主题介绍中的密钥同步设置 AWS CloudHSM、客户在集群上使用密钥时面临的常见问题以及使密钥更耐用的策略。

主题

概念

令牌密钥

您在密钥生成、导入或解包操作期间创建的永久密钥。 AWS CloudHSM 跨集群同步令牌密钥。

会话密钥

仅存在于集群中一个硬件安全模块 (HSM) 上的临时密钥。 AWS CloudHSM 不会跨集群同步会话密钥。

客户端密钥同步

一个客户端进程，用于克隆您在密钥生成、导入或解包操作期间创建的令牌密钥。您可以运行有两个及以上 HSM 的集群提高令牌密钥的耐久性。

服务器端密钥同步

定期将密钥克隆到集群中的每个 HSM。无需管理。

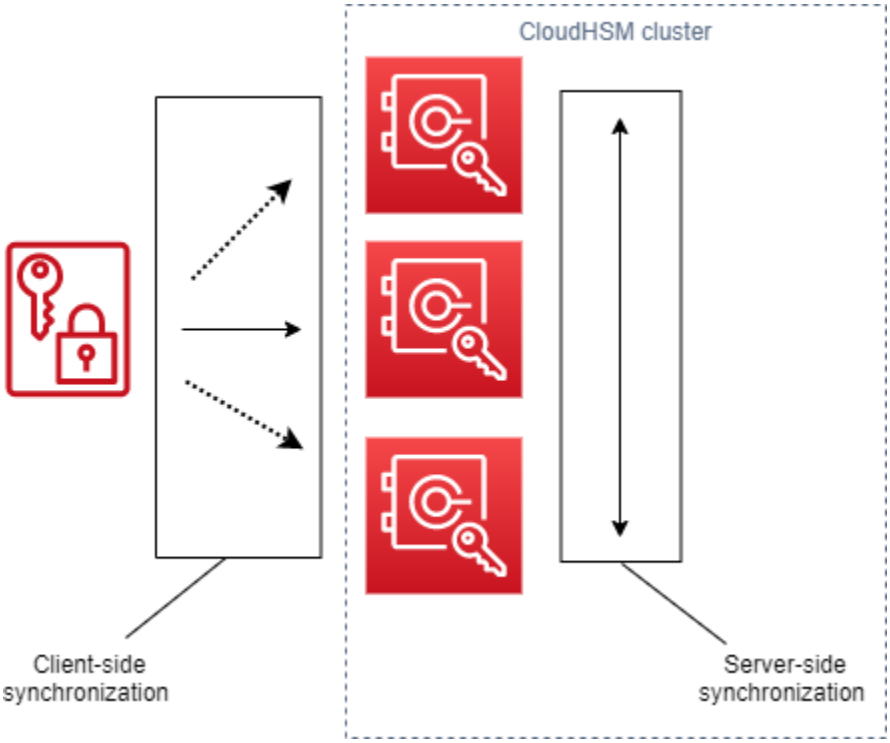
客户端密钥持久性设置

您在客户端上配置的会影响密钥持久性的设置。这些设置在客户端软件开发工具包 5 和客户端软件开发工具包 3 中的运作方式有所不同。

- 在客户端软件开发工具包 5 中，使用此设置运行单个 HSM 集群。
- 在客户端软件开发工具包 3 中，使用此设置指定成功创建密钥操作所需的 HSM 数量。

了解密钥同步

AWS CloudHSM 使用密钥同步在集群中的所有 HSM 上克隆令牌密钥。在密钥生成、导入或解包操作期间，您可以将令牌密钥创建为永久密钥。为了在集群中分配这些密钥，CloudHSM 提供了客户端和服务器端密钥同步功能。



密钥同步（服务器端和客户端）的目标是在创建新密钥后尽快在集群中分发新密钥。这很重要，因为您为使用新密钥而进行的后续调用可以路由到集群中任何可用的 HSM。如果您拨打的呼叫路由到没有密钥的 HSM，则呼叫将失败。您可以通过指定应用程序在密钥创建操作后重试后续调用来缓解这些类型故障。同步所需的时间可能会有所不同，具体取决于集群的工作负载和其他无形因素。使用 CloudWatch 指标来确定您的应用程序在这种情况下应采用的时机。有关更多信息，请参阅CloudWatch 指标。

云环境中密钥同步面临的挑战在于密钥的耐久性。您在单个 HSM 上创建密钥，并且通常会立即开始使用这些密钥。如果您创建密钥的 HSM 在密钥被克隆到集群中的另一个 HSM 之前出现故障，则您将丢失密钥并无法访问由密钥加密的任何内容。为了降低这种风险，我们提供客户端同步功能。客户端同步为客户端进程，用于克隆您在密钥生成、导入或解包操作期间创建的密钥。在创建密钥时对其进行克隆可以使密钥更持久。当然，您无法使用单个 HSM 克隆集群中的密钥。为了使密钥更持久，我们还建议您将集群配置为使用两个及以上的 HSM。通过客户端同步和具有两个 HSM 的集群，您可以应对云环境中密钥持久性的挑战。

使用客户端密钥持久性设置

密钥同步总体是一个自动过程，但您可以管理客户端密钥的耐久性设置。客户端软件开发工具包 5 和客户端软件开发工具包 3 中的客户端密钥持久性设置的运作方式不同。

- 在客户端软件开发工具包 5 中，我们引入了密钥可用性仲裁的概念，它要求您运行有两个及以上 HSM 的集群。您可以使用客户端密钥持久性设置，选择不要求使用两个 HSM。有关仲裁的更多信息，请参阅 the section called “客户端软件开发工具包 5 概念”。
- 在客户端软件开发工具包 3 中，您可以使用客户端密钥持久性设置来指定要使整个操作视为成功必须成功创建密钥的 HSM 数量。

客户端软件开发工具包 5 客户端密钥持久性设置

在客户端软件开发工具包 5 中，密钥同步是全自动的过程。对于密钥可用性仲裁，新创建的密钥必须存在于集群中的两个 HSM 上，然后您的应用程序才能使用该密钥。要使用密钥可用性仲裁，您的集群必须有两个及以上的 HSM。

如果您的集群配置不符合密钥持久性要求，则任何创建或使用令牌密钥的尝试都将失败，并在日志中显示以下错误消息：

```
Key <key handle> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

您可以使用客户端配置设置选择退出密钥可用性仲裁。例如，您可能希望选择不使用单个 HSM 运行集群。

客户端软件开发工具包 5 概念

密钥可用性仲裁

AWS CloudHSM 指定集群中必须存在密钥的 HSM 数量，然后您的应用程序才能使用密钥。需要有两个及以上 HSM 的集群。

管理客户端密钥持久性设置

要管理客户端密钥持久性设置，必须使用客户端软件开发工具包 5 的配置工具。

PKCS #11 library

在 Linux 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

在 Windows 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-
availability-check
```

OpenSSL Dynamic Engine

在 Linux 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

JCE provider

在 Linux 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

在 Windows 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-
availability-check
```

CloudHSM CLI

在 Linux 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

在 Windows 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-
availability-check
```

客户端软件开发工具包 3 客户端密钥持久性设置

在客户端软件开发工具包 3 中，密钥同步总体是一个自动过程，但您可以使用客户端密钥持久性设置来提高密钥的耐久性。您必须指定成功创建密钥的 HSM 数量，整个操作才算成功。无论您选择什么设置，客户端同步功能始终会尽力将密钥克隆到集群中的每个 HSM。您的设置会在您指定数量的 HSM 上强制创建密钥。如果您指定了一个数量，但系统无法将该密钥复制到该数量的 HSM 上，则系统会自动清理所有不需要的密钥材料，且您可以重试。

> **Important**
>
> 如果您未设置客户端密钥持久性设置（或者使用默认值 1），则密钥很容易丢失。如果您当前的 HSM 在服务器端服务将该密钥克隆到另一个 HSM 前出现故障，则密钥材料会丢失。

为了最大限度地提高密钥的耐久性，请考虑指定两个及以上 HSM 进行客户端同步。请记住，无论您指定多少个 HSM，集群上的工作负载都将保持不变。客户端同步总是尽最大努力将密钥克隆到集群中的每个 HSM。

建议

- 最小值：每个集群两台 HSM
- 最大值：比您集群中的 HSM 总数少一个

如果客户端同步失败，则客户端服务会清理可能已创建且现在不需要的所有不需要的密钥。这种清理是尽力而为的应对措施，可能并不总是奏效。如果清理失败，则可能需要您删除不需要的密钥材料。有关更多信息，请参阅密钥同步故障。

为客户端密钥持久性设置配置文件

要指定客户端密钥持久性设置，必须编辑 `cloudhsm_client.cfg`。

编辑客户端配置文件

1. 打开 `cloudhsm_client.cfg`。

   Linux：

   ```
   /opt/cloudhsm/etc/cloudhsm_client.cfg
   ```

   Windows:

   ```
   C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
   ```

2. 在文件`client`节点中，添加`create_object_minimum_nodes`并指定一个值，表示 AWS CloudHSM 必须成功创建密钥才能成功创建密钥的 HSM 的最小数量。

   ```
   "create_object_minimum_nodes" : 2
   ```

   > **Note**
   >
   > key_mgmt_util (KMU) 命令行工具还有额外的客户端密钥持久性设置。有关更多信息，请参阅 the section called “KMU 和客户端同步”。

配置参考

以下是客户端同步属性，如以下 `cloudhsm_client.cfg` 的摘录所示：

```
{
    "client": {
        "create_object_minimum_nodes" : 2,
        ...
    },

    ...
}
```

create_object_minimum_nodes

> 指定密钥生成、密钥导入或密钥解包操作视为成功所需的最少 HSM 数量。如果已设置，默认为“1”。这意味着，对于每个密钥创建操作，客户端服务都会尝试在集群中的每个 HSM 上创建密钥，但要返回“成功”，只需在集群中的一个 HSM 上创建单个密钥即可。

KMU 和客户端同步

如果使用 key_mgmt_util (KMU) 命令行工具创建密钥，则要使用可选的命令行参数 (`-min_srv`) 来限制要克隆密钥的 HSM 数量。如果您在配置文件中指定命令行参数和值，则使用两个值 AWS CloudHSM 中较大的一个。

有关更多信息，请参阅以下主题：

- GendSA KeyPair
- GenecC KeyPair
- GenrSA KeyPair
- genSymKey
- importPrivateKey
- importPubKey
- imSymKey
- insertMaskedObject
- unWrapKey

跨克隆集群同步密钥

客户端和服务器端同步仅用于同步同一集群内的密钥。如果将集群的备份复制到另一区域，则可以使用 cloudhsm_mgmt_util (CMU) 的 syncKey 命令在集群之间同步密钥。您可以使用克隆的集群来实现跨区域冗余或简化灾难恢复过程。有关更多信息，请参阅 syncKey。

AES 密钥封装进去 AWS CloudHSM

本主题介绍了 AES 密钥封装的选项 AWS CloudHSM。AES 密钥包装使用 AES 密钥（包装密钥）来包装任何类型的其他密钥（目标密钥）。您可以使用密钥包装来保护存储的密钥或通过不安全的网络传输密钥。

主题

支持的算法

AWS CloudHSM 为 AES 密钥包装提供了三种选项，每个选项都基于目标密钥在封装之前的填充方式。在调用密钥包装时，将根据您使用的算法自动完成填充。下表列出了支持的算法和相关的详细信息，以帮助您为应用程序选择适当的包装机制。

AES 密钥包装算法	规范	支持的目标密钥类型	填充方案	AWS CloudHSM 客户机可用性
零填充的 AES 密钥包装	RFC 5649 和 SP 800 - 38F	全部	如有必要，可在密钥位后添加零来阻止对齐	SDK 3.1 及更高版本
无填充的 AES 密钥包装	RFC 3394 和 SP 800 - 38F	块对齐的密钥，例如 AES 和 3DES	无	SDK 3.1 及更高版本
带 PKCS #5 填充的 AES 密钥包装	无	全部	根据 PKCS #5 填充方案添加至少 8 个字节来阻止对齐	全部

要了解如何在应用程序中使用上表中的 AES 密钥包装算法，请参阅 在 AWS CloudHSM中使用 AES 密钥包装。

了解 AES 密钥包装中的初始化向量

在包装之前，CloudHSM 会将初始化向量 (IV) 附加到目标密钥来确保数据完整。每种密钥包装算法对允许的 IV 类型都施加了特定的限制。要设置 IV AWS CloudHSM，有两个选项：

- 隐式：将 IV 设置为 NULL，CloudHSM 使用该算法的默认值进行包装和解开包装操作（推荐）
- 显式：通过将默认的 IV 值传递给密钥包装函数来设置 IV

> **Important**
>
> 您必须了解应用程序中使用的 IV。要对密钥解开包装，您必须提供用于包装密钥的同一 IV。如果您使用隐式 IV 来包装，请使用隐式 IV 来解开包装。对于隐式 IV，CloudHSM 将使用默认值来解开包装。

下表介绍了包装算法指定的 IV 所允许的值。

AES 密钥包装算法	隐式 IV	显式 IV
零填充的 AES 密钥包装	必需 默认值：（按规格在内部计算的 IV）	不允许
无填充的 AES 密钥包装	允许（推荐） 默认值：0xA6A6A6A6A6A6A6A6	已允许 仅接受此值：0xA6A6A6A6A6A6A6A6
带 PKCS #5 填充的 AES 密钥包装	允许（推荐） 默认值：0xA6A6A6A6A6A6A6A6	已允许 仅接受此值：0xA6A6A6A6A6A6A6A6

使用 AES 密钥包装 AWS CloudHSM

对密钥进行包装和解开包装，如下所示：

- 在 PCKS #11 库中，为 C_WrapKey 和 C_UnWrapKey 函数选择适当的机制，如下表所示。
- 在 JCE 提供程序中，选择适当的算法、模式和填充组合，实施密码方法 Cipher.WRAP_MODE 和 Cipher.UNWRAP_MODE，如下表所示。
- 在 C loudHSM CLI 中，从密钥解包支持的算法和算法列表钥匙包装中选择适当的算法，如下表所示。
- 在 key_mgmt_util (KMU) 中，将命令 wrapKey 和 unWrapKey 与适当的 m 值结合使用，如下表所示。

AES 密钥包装算法	PKCS #11 机制	Java 方法	CloudHSM CLI 子命令	密钥管理实用程序 (KMU) 参数
零填充的 AES 密钥包装	• CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD （供应商定义的机制）	AESWrap/ECB/ZeroPadding	aes-zero-pad	m = 6
无填充的 AES 密钥包装	• CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD （供应商定义的机制）	AESWrap/ECB/NoPadding	aes-no-pad	m = 5
带 PKCS #5 填充的 AES 密钥包装	• CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD （供应商定义的机制）	AESWrap/ECB/PKCS5Padding	aes-pkcs5-pad	m = 4

在中使用可信密钥 AWS CloudHSM

AWS CloudHSM 支持可信密钥封装，以保护数据密钥免受内部威胁。本主题介绍如何创建可信密钥以确保数据安全。

主题

了解可信密钥

可信密钥是用于包装其他密钥的密钥，管理员和加密员（CO, cryptographic officer）使用属性 CKA_TRUSTED 特意将其标识为可信。此外，管理员和加密员（CO, cryptographic officer）使用 CKA_UNWRAP_TEMPLATE 和相关属性来指定数据密钥在被可信密钥解包后可以执行的操作。由可信密钥解包的数据密钥还必须包含这些属性，解包操作才能成功，这有助于确保解包的数据密钥仅允许用于您想要的用途。

使用该 CKA_WRAP_WITH_TRUSTED 属性来标识要用可信密钥包装的所有数据密钥。这样做可以限制数据密钥，这样一来应用程序只能使用可信密钥来解包它们。一旦在数据密钥上设置了该属性，该属性就会变成只读，无法更改。有了这些属性后，应用程序只能使用您信任的密钥来解包您的数据密钥，而解包总是会产生带有限制这些密钥使用方式的属性的数据密钥。

可信密钥属性

以下属性允许您将密钥标记为可信，指定数据密钥只能使用可信密钥进行包装和解包以及控制数据密钥解包后可以执行的操作：

- CKA_TRUSTED：将此属性（除 CKA_UNWRAP_TEMPLATE 外）应用于将包装数据密钥的密钥，以指定已完成必要调查的管理员或加密员（CO, crypto officer）并信任此密钥。只有管理员或 CO 可以设置 CKA_TRUSTED。加密用户（CU）拥有密钥，但只有 CO 可以设置其 CKA_TRUSTED 属性。
- CKA_WRAP_WITH_TRUSTED：将此属性应用于可导出数据密钥，以指定只能用标记为 CKA_TRUSTED 的密钥来包装此密钥。将 CKA_WRAP_WITH_TRUSTED 设置为 true 后，该属性将变为只读，并且您无法更改或移除该属性。
- CKA_UNWRAP_TEMPLATE：将此属性应用于包装密钥（除 CKA_TRUSTED 外），以指定服务必须自动将哪些属性名称和值应用于服务解包的数据密钥。应用程序提交密钥供解包时，也可以提供自己的

解包模板。如果您指定了解包模板并且应用程序提供了自己的解包模板，则 HSM 会使用这两个模板将属性名称和值应用于密钥。但是，如果包装密钥的 CKA_UNWRAP_TEMPLATE 中的值与应用程序在解包请求时提供的属性相冲突，则解包请求会失败。

了解有关属性的更多信息，请参阅以下主题：

- PKCS #11 密钥属性
- JCE 密钥属性
- CloudHSM CLI 密钥属性

如何使用可信密钥来包装数据密钥

要使用可信密钥包装数据密钥，必须完成三个基本步骤：

1. 对于计划使用可信密钥包装的数据密钥，请将其 CKA_WRAP_WITH_TRUSTED 属性设置为“true”。
2. 对于计划用其包装数据密钥的可信密钥，请将其 CKA_TRUSTED 属性设置为“true”。
3. 使用可信密钥包装数据密钥。

步骤 1：将数据密钥的 **CKA_WRAP_WITH_TRUSTED** 设置为“true”

对于要包装的数据密钥，请选择以下选项之一，将密钥的 CKA_WRAP_WITH_TRUSTED 属性设置为“true”。这样做可以限制数据密钥，因此应用程序只能使用可信密钥来解包它数据密钥。

选项 1：如果生成新密钥，则设置 **CKA_WRAP_WITH_TRUSTED** 为“true”

使用 PKCS #11、JCE 或 CloudHSM CLI 生成密钥。了解更多详细信息，请参阅以下内容。

PKCS #11

要使用 PKCS #11 生成密钥，您需要将密钥的 CKA_WRAP_WITH_TRUSTED 属性设置为“true”。如以下示例所示，通过在密钥的 CK_ATTRIBUTE template 中包含此属性，然后将该属性设置为“true”来执行此操作：

```
CK_BYTE_PTR label = "test_key";
CK_ATTRIBUTE template[] = {
        {CKA_WRAP_WITH_TRUSTED, &true_val,         sizeof(CK_BBOOL)},
        {CKA_LABEL,             label,             strlen(label)},
        ...
```

```
};
```

有关更多信息，请参阅我们演示使用 PKCS #11 生成密钥的公开示例。

JCE

要使用 JCE 生成密钥，您需要将密钥的 WRAP_WITH_TRUSTED 属性设置为“true”。如以下示例所示，通过在密钥的 KeyAttributesMap 中包含此属性，然后将该属性设置为“true”来执行此操作：

```
final String label = "test_key";
final KeyAttributesMap keySpec = new KeyAttributesMap();
keySpec.put(KeyAttribute.WRAP_WITH_TRUSTED, true);
keySpec.put(KeyAttribute.LABEL, label);
...
```

有关更多信息，请参阅我们演示使用 JCE 生成密钥的公开示例。

CloudHSM CLI

要使用 CloudHSM CLI 生成密钥，您需要将密钥的 wrap-with-trusted 属性设置为“true”。方法是在密钥生成命令的相应参数中加入 wrap-with-trusted=true：

- 对于对称密钥，请在 attributes 参数中添加 wrap-with-trusted。
- 对于公有密钥，请在 public-attributes 参数中添加 wrap-with-trusted。
- 对于私有密钥，请在 private-attributes 参数中添加 wrap-with-trusted。

有关密钥对生成的更多信息，请参阅 钥匙 generate-asymmetric-pair 。

有关生成对称密钥的更多信息，请参阅 key generate-symmetric 。

选项 2：如果使用现有密钥，请使用 CloudHSM CLI 将它的 **CKA_WRAP_WITH_TRUSTED** 设置为“true”

要将现有密钥的 CKA_WRAP_WITH_TRUSTED 属性设置为“true”，请执行以下步骤：

1. 使用 login 命令以加密用户（CU）身份登录。
2. 使用 key set-attribute 命令将密钥的 wrap-with-trusted 属性设置为“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"
  }
}
```

步骤 2：将可信密钥的 **CKA_TRUSTED** 设置为“true”

要使密钥成为可信密钥，必须将其 CKA_TRUSTED 属性设置为“true”。您可以使用 CloudHSM CLI 或 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility) 来执行此操作。

- 如果使用 CloudHSM CLI 来设置密钥的 CKA_TRUSTED 属性，请参阅 如何用 CloudHSM CLI 将密钥标记为可信。
- 如果使用 CMU 来设置密钥的 CKA_TRUSTED 属性，请参阅 如何使用 CMU 将密钥标记为可信。

第 3 步。使用可信密钥包装数据密钥

要对步骤 1 中引用的数据密钥与您在步骤 2 中设置的可信密钥进行包装，请参阅以下链接以获取代码示例。每个都演示了如何包装密钥。

- AWS CloudHSM PKCS #11 示例
- AWS CloudHSM JCE 示例

如何使用可信密钥解包数据密钥

要解包数据密钥，您需要一个已将 CKA_UNWRAP 设置为“true”的可信密钥。要成为这样的密钥，还必须满足以下标准：

- 密钥的 CKA_TRUSTED 属性必须设置为“true”。
- 密钥必须使用 CKA_UNWRAP_TEMPLATE 和相关属性来指定数据密钥解包后可以执行的操作。例如，如果您希望未解包的密钥不可导出，则可以将 CKA_EXPORTABLE = FALSE 设置为 CKA_UNWRAP_TEMPLATE 的一部分。

> **Note**
>
> CKA_UNWRAP_TEMPLATE 仅在 PKCS #11 中可用。

当应用程序提交要解包的密钥时，该应用程序还可提供自己的解包模板。如果您指定了解包模板并且应用程序提供了自己的解包模板，则 HSM 会使用这两个模板将属性名称和值应用于密钥。但是，如果在解包请求期间，可信密钥的 CKA_UNWRAP_TEMPLATE 中的值与应用程序提供的属性冲突，则解包请求将失败。

要查看使用可信密钥解包数据密钥的示例，请参阅此 PKCS #11 示例。

使用 CloudHSM CLI 管理密钥

如果使用最新的 SDK 版本系列，请使用 C loudHSM CLI 来管理集群中的密钥 AWS CloudHSM 。了解更多详细信息，请参阅以下主题。

- 使用可信密钥描述了如何使用 CloudHSM CLI 创建可信密钥来保护数据。
- 生成密钥包括有关创建密钥的说明，包括创建对称密钥、RSA 密钥和 EC 密钥。
- 删除密钥介绍了密钥所有者如何删除密钥。
- 共享和取消共享密钥详细说明了密钥所有者如何共享和取消共享密钥。
- 筛选密钥提供了有关使用筛选器查找密钥的指南。

使用 CloudHSM CLI 生成密钥

在生成密钥之前，必须启动 CloudHSM CLI 并以加密用户（CU）身份登录。要生成 HSM 上的密钥，请使用与要生成的密钥的类型对应的命令。

主题

- 生成对称密钥
- 生成非对称密钥
- 相关 主题

生成对称密钥

使用 key generate-symmetric 中列出的命令生成对称密钥。要查看所有可用的选项，请使用 help key generate-symmetric 命令。

生成 AES 密钥

使用 key generate-symmetric aes 命令生成 AES 密钥。要查看所有可用的选项，请使用 help key generate-symmetric aes 命令。

Example

以下示例生成一个 32 字节的 AES 密钥。

```
aws-cloudhsm > key generate-symmetric aes \
    --label aes-example \
    --key-length-bytes 32
```

参数

<LABEL>

为 AES 密钥指定用户定义的标签。

必需：是

<KEY-LENGTH-BYTES>

指定以字节为单位的密钥长度。

有效值：

- 16、24 和 32

必需：是

<KEY_ATTRIBUTES>

指定一个空格分隔的密钥属性列表，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如 token=true）的形式为生成的 AES 密钥进行设置

有关支持的 AWS CloudHSM 密钥属性的列表，请参阅CloudHSM CLI 的密钥属性。

必需：否

<SESSION>

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

要将会话密钥更改为永久（令牌）密钥，请使用 key set-attribute。

默认情况下，生成密钥时，它们是永久密钥/令牌密钥。使用 <SESSION> 更改此设置，确保使用此参数生成的密钥是会话/临时密钥

必需：否

生成通用机密密钥

使用 key generate-symmetric generic-secret 命令生成通用机密密钥。要查看所有可用的选项，请使用 help key generate-symmetric generic-secret 命令。

Example

以下示例生成一个 32 字节的通用机密密钥。

```
aws-cloudhsm > key generate-symmetric generic-secret \
    --label generic-secret-example \
    --key-length-bytes 32
```

参数

<LABEL>

为通用机密密钥指定用户定义的标签。

必需：是

<KEY-LENGTH-BYTES>

指定以字节为单位的密钥长度。

有效值：

- 1 至 800

必需：是

<KEY_ATTRIBUTES>

指定要为生成的通用机密密钥设置的以空格分隔的密钥属性列表，其形式为 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如 `token=true`）

有关支持的 AWS CloudHSM 密钥属性的列表，请参阅CloudHSM CLI 的密钥属性。

必需：否

<SESSION>

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

要将会话密钥更改为永久（令牌）密钥，请使用 key set-attribute。

默认情况下，生成密钥时，它们是永久密钥/令牌密钥。使用 <SESSION> 更改此设置，确保使用此参数生成的密钥是会话/临时密钥

必需：否

生成非对称密钥

使用 钥匙 generate-asymmetric-pair 中列出的命令生成对称密钥对。

生成 RSA 密钥

使用 key generate-asymmetric-pair rsa 命令生成 RSA 密钥对。要查看所有可用的选项，请使用 help key generate-asymmetric-pair rsa 命令。

Example

以下示例生成 RSA 2048 位密钥对。

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
    --public-exponent 65537 \
    --modulus-size-bits 2048 \
    --public-label rsa-public-example \
    --private-label rsa-private-example
```

参数

<PUBLIC_LABEL>

为公有密钥指定用户定义的标签。

必需：是

<PRIVATE_LABEL>

为私有密钥指定用户定义的标签。

必需：是

<MODULUS_SIZE_BITS>

指定模数的长度 (以位为单位)。最小值为 2048。

必需：是

<PUBLIC_EXPONENT>

指定公有指数。此值必须为大于或等于 65537 的奇数。

必需：是

<PUBLIC_KEY_ATTRIBUTES>

指定一个以空格分隔的密钥属性列表，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如 `token=true`）的形式为生成的 RSA 公有密钥进行设置。

有关支持的 AWS CloudHSM 密钥属性的列表，请参阅CloudHSM CLI 的密钥属性。

必需：否

<SESSION>

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

要将会话密钥更改为永久（令牌）密钥，请使用 key set-attribute。

默认情况下，生成密钥时，它们是永久密钥/令牌密钥。使用 <SESSION> 更改此设置，确保使用此参数生成的密钥是会话/临时密钥

必需：否

生成 EC (椭圆曲线加密) 密钥对

使用 key generate-asymmetric-pair ec 命令生成 EC 密钥对。要查看所有可用选项（包括受支持的椭圆曲线的列表），请使用 help key generate-asymmetric-pair ec 命令。

Example

以下示例使用 Secp384r1 椭圆曲线生成一个 EC 密钥对。

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp384r1 \
    --public-label ec-public-example \
    --private-label ec-private-example
```

参数

<PUBLIC_LABEL>

为公有密钥指定用户定义的标签。客户端 SDK 5.11 及更高版本允许的最大大小`label`为 127 个字符。客户端 SDK 5.10 及更低版本的字符数限制为 126 个。

必需：是

<PRIVATE_LABEL>

为私有密钥指定用户定义的标签。客户端 SDK 5.11 及更高版本允许的最大大小`label`为 127 个字符。客户端 SDK 5.10 及更低版本的字符数限制为 126 个。

必需：是

<CURVE>

指定椭圆曲线的标识符。

有效值：

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1

必需：是

<PUBLIC_KEY_ATTRIBUTES>

以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如 `token=true`）的形式指定要为生成的 EC 公有密钥设置的以空格分隔的密钥属性列表。

有关支持的 AWS CloudHSM 密钥属性的列表，请参阅CloudHSM CLI 的密钥属性。

必需：否

<PRIVATE_KEY_ATTRIBUTES>

以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如 `token=true`）的形式指定要为生成的 EC 私有密钥设置的以空格分隔的密钥属性列表。

有关支持的 AWS CloudHSM 密钥属性的列表，请参阅CloudHSM CLI 的密钥属性。

必需：否

<SESSION>

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

要将会话密钥更改为永久（令牌）密钥，请使用 key set-attribute。

默认情况下，生成的密钥是永久（令牌）密钥。传入 <SESSION> 会改变这一点，确保使用此参数生成的密钥是会话（临时）密钥。

必需：否

相关 主题

- CloudHSM CLI 的密钥属性
- 钥匙 generate-asymmetric-pair
- key generate-symmetric

使用 CloudHSM CLI 删除密钥

使用本主题的示例通过 CloudHSM CLI 删除密钥。只有密钥所有者才能删除密钥。

主题

- 示例：删除密钥
- 相关 主题

示例：删除密钥

1. 运行 key list 命令以确定要删除的密钥：

```
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. 识别密钥后，运行带有密钥唯一 `label` 属性的 key delete 删除密钥：

```
aws-cloudhsm > key delete --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

3. 运行带有密钥唯一 `label` 属性的 key list 命令并确认密钥已被删除。如下例所示，HSM 集群中没有带有标签 `my_key_to_delete` 的密钥：

```
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
  }
}
```

相关 主题

- CloudHSM CLI 的密钥属性
- key delete

使用 CloudHSM CLI 共享和取消共享密钥

使用本主题中的命令在 CloudHSM CLI 中共享和取消共享密钥。在中 AWS CloudHSM，创建密钥的加密用户 (CU) 拥有该密钥。所有者可以使用 key share 和 key unshare 命令，与其他 CU 共享和取消共享密钥。密钥的共享用户可在加密操作中使用密钥，但无法导出或删除密钥，也无法与其他用户共享密钥。

要共享密钥，您必须以拥有该密钥的加密用户（CU）的身份登录到 HSM。

主题

- 示例：共享和取消共享密钥
- 相关 主题

示例：共享和取消共享密钥

Example

以下示例演示了如何与加密用户（CU）alice 共享密钥和与其取消共享。除 key share 和 key unshare 命令外，共享和取消共享命令还需要使用 CloudHSM CLI 密钥筛选器的特定密钥以及要与之共享或取消共享密钥的用户的特定用户名。

1. 首先，使用筛选器运行 key list 命令以返回特定的密钥，然后查看密钥已与谁共享。

```
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. 查看 shared-users 输出以确定当前与谁共享密钥。
3. 要与加密用户（CU）alice 共享此密钥，请输入以下命令：

```
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"
  }
}
```

请注意，除了 key share 命令外，此命令还使用密钥的唯一标签和将与之共享密钥的用户的姓名。

4. 再次运行 key list 命令以确认已与 alice 共享该密钥：

```
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. 要取消与 alice 共享同一密钥，请运行以下 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"
  }
}
```

请注意，除了 key unshare 命令外，此命令还使用密钥的唯一标签和将与之共享密钥的用户的姓名。

6. 再次运行 key list 命令并确认已取消与加密用户 alice 共享该密钥：

```
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
  }
}
```


相关 主题

- CloudHSM CLI 的密钥属性
- key share
- key unshare
- 使用 CloudHSM CLI 筛选密钥

使用 CloudHSM CLI 筛选密钥

使用以下密钥命令利用 CloudHSM CLI 的标准化密钥过滤机制。

- key list
- key delete

- key share
- key unshare
- key set-attribute

要使用 CloudHSM CLI 选择和/或筛选密钥，密钥命令使用基于 CloudHSM CLI 的密钥属性 的标准化过滤机制。可以在键盘命令中使用一个或多个属性来指定一个或多个按键，这些 AWS CloudHSM 属性可以识别一个或多个按键。密钥过滤机制仅对当前登录的用户拥有和共享的密钥以及 AWS CloudHSM 集群中的所有公钥起作用。

主题

要求

要筛选密钥，您必须是以加密用户 (CU,crypto user) 身份登录的用户。

筛选以查找单个密钥

请注意，在以下示例中，用作筛选器的每个属性都必须以 `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` 形式写入。例如，如果要按标签属性进行筛选，则可以写入 `attr.label=my_label`。

Example 使用单个属性来查找单个密钥

此示例演示如何仅使用单个标识属性筛选到单个唯一密钥。

```
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 使用多个属性来查找单个密钥

以下示例演示如何使用多个密钥属性查找单个密钥。

```
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 筛选以查找一组密钥

以下示例演示如何通过筛选查找一组私有 rsa 密钥。

```
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
  }
}
```

过滤错误

某些密钥操作一次只能对一个密钥执行。对于这些操作，如果筛选标准未充分细化且多个密钥与标准匹配，CloudHSM CLI 将提供错误。下面显示了一个这样的示例，其中包含“密钥删除”。

Example 匹配太多密钥时出现过滤错误

```
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."
}
```

相关 主题

- CloudHSM CLI 的密钥属性

如何用 CloudHSM CLI 将密钥标记为可信

本节内容提供有关使用 CloudHSM CLI 将密钥标记为可信密钥的说明。

1. 使用 CloudHSM CLI login 命令，以加密用户（CU）身份登录。
2. 使用 key list 命令标识要标记为可信密钥的密钥引用。下面的示例将列出带有标签 `key_to_be_trusted` 的密钥。

```
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 注销 命令注销加密用户（CU）。
4. 使用 login 命令，以管理员身份登录。
5. 使用带有您在步骤 2 中确定的密钥引用的 key set-attribute 命令，将密钥的可信值设置为“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 管理密钥

如果使用最新的 SDK 版本系列，请使用 C loudHSM CLI 来管理集群中的密钥 AWS CloudHSM 。

如果使用之前的 SDK 版本系列，则可以使用 key_mgmt_util 命令行工具管理 AWS CloudHSM 集群中 HSM 上的密钥。在管理密钥之前，必须启动 AWS CloudHSM 客户端，启动 key_mgmt_util，然后登录 HSM。有关更多信息，请参阅《key_mgmt_util 入门》。

- 使用可信密钥描述了如何使用 PKCS #11 库属性和 CMU 来创建可信密钥来保护数据。
- 生成密钥包含有关生成密钥的说明，包括对称密钥、RSA 密钥和 EC 密钥。
- 导入密钥提供了有关密钥所有者导入密钥方式的详细信息。
- 导出密钥提供了有关密钥所有者导出密钥方式的详细信息。
- 删除密钥详细介绍了密钥所有者删除密钥的方式。
- 共享和取消共享密钥详细说明了密钥所有者如何共享和取消共享密钥。

生成密钥

要生成 HSM 上的密钥，请使用与要生成的密钥的类型对应的命令。

主题

- 生成对称密钥
- 生成 RSA 密钥对
- 生成 ECC (椭圆曲线加密) 密钥对

生成对称密钥

使用genSymKey命令生成 AES 和其他类型的对称密钥。要查看所有可用的选项，请使用 genSymKey -h 命令。

以下示例创建 256 位 AES 密钥。

```
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
```

生成 RSA 密钥对

要生成 RSA 密钥对，请使用 genR KeyPair SA 命令。要查看所有可用的选项，请使用 genRSAKeyPair -h 命令。

以下示例生成 RSA 2048 位密钥对。

```
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
```

生成 ECC (椭圆曲线加密) 密钥对

要生成 ECC 密钥对，请使用 Genec KeyPair C 命令。要查看所有可用选项（包括受支持的椭圆曲线的列表），请使用 genECCKeyPair -h 命令。

以下示例将使用 NIST FIPS 刊物 186-4 中定义的 P-384 椭圆曲线生成一个 ECC 密钥对。

```
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
```

导入密钥

要将机密密钥，即对称密钥和非对称私有密钥导入 HSM 中，您必须先在 HSM 上创建一个包装密钥。您可以在没有包装密钥的情况下直接导入公有密钥。

主题

- 导入私有密钥
- 导入公有密钥

导入私有密钥

完成以下步骤以导入机密密钥。在导入私有密钥之前，请将它保存到一个文件中。将对称密钥另存为原始字节，并以 PEM 格式保存非对称私有密钥。

此示例演示如何将明文机密密钥从文件导入到 HSM 中。要将加密密钥从文件导入 HSM，请使用unWrapKey命令。

导入私有密钥

1. 使用genSymKey命令创建包装密钥。以下命令创建一个 128 位 AES 包装密钥，此密钥仅对当前会话有效。您可以使用会话密钥或永久密钥作为包装密钥。

```
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. 根据您所导入的私有密钥的类型，使用下列命令之一。

   - 要导入对称密钥，请使用imSymKey命令。以下命令使用上一步中创建的包装密钥，从名为 aes256.key 的文件中导入 AES 密钥。要查看所有可用的选项，请使用 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
     ```

   - 要导入非对称私钥，请使用importPrivateKey命令。以下命令使用上一步中创建的包装密钥，从名为 rsa2048.key 的文件中导入私有密钥。要查看所有可用的选项，请使用 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
```

导入公有密钥

使用importPubKey命令导入公钥。要查看所有可用的选项，请使用 importPubKey -h 命令。

以下示例从名为 `rsa2048.pub` 的文件中导入 RSA 公有密钥。

```
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
```

导出密钥

要将机密密钥，即对称密钥和非对称私有密钥从 HSM 中导出，您必须先创建一个包装密钥。您可以在没有包装密钥的情况下直接导出公有密钥。

只有密钥所有者才能导出密钥。密钥的共享用户可在加密操作中使用密钥，但无法导出密钥。运行此示例时，请确保导出您创建的密钥。

> **Important**
>
> 该exSymKey命令将密钥的纯文本（未加密）副本写入文件。此导出过程需要包装密钥，但此文件中的密钥不 是已包装的密钥。要导出密钥的已包装 (已加密) 副本，请使用 wrapKey 命令。

主题

- 导出私有密钥
- 导出公有密钥

导出私有密钥

完成以下步骤以导出机密密钥。

导出私有密钥

1. 使用genSymKey命令创建包装密钥。以下命令创建一个 128 位 AES 包装密钥，此密钥仅对当前会话有效。

```
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. 根据您所导出的私有密钥的类型，使用下列命令之一。

   - 要导出对称密钥，请使用exSymKey命令。以下命令将 AES 密钥导出到名为 aes256.key.exp 的文件。要查看所有可用的选项，请使用 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**
>
> 此命令的输出显示，“已包装的对称密钥”已写入到输出文件中。但是，输出文件包含明文 (未包装) 密钥。要将已包装 (已加密) 密钥导出到文件，请使用 wrapKey 命令。

   - 要导出私有密钥，请使用 exportPrivateKey 命令。以下命令将私有密钥导出到名为 rsa2048.key.exp 的文件。要查看所有可用的选项，请使用 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
```

导出公有密钥

使用 exportPubKey 命令导出公有密钥。要查看所有可用的选项，请使用 exportPubKey -h 命令。

以下示例将 RSA 公有密钥导出到名为 `rsa2048.pub.exp` 的文件。

```
Command: exportPubKey -k 524294 -out rsa2048.pub.exp
PEM formatted public key is written to rsa2048.pub.key

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

删除密钥

使用 deleteKey 命令删除密钥，如以下示例所示。只有密钥所有者才能删除密钥。

```
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
```

共享和取消共享密钥

在中 AWS CloudHSM，创建密钥的 CU 拥有该密钥。所有者管理密钥，可导出和删除密钥，并且可在加密操作中使用密钥。所有者还可以与其他 CU 用户共享密钥。密钥的共享用户可在加密操作中使用密钥，但无法导出或删除密钥，也无法与其他用户共享密钥。

您可以在创建密钥时与其他 CU 用户共享密钥，例如使用genSymKey或 GenRSA KeyPair 命令的-u参数。要与其他 HSM 用户共享现有密钥，请使用 cloudhsm_mgmt_util 命令行工具。这与此部分中所述的使用 key_mgmt_util 命令行工具的大多数任务不同。

在共享密钥之前，必须启动 cloudhsm_mgmt_util，启用 end-to-end加密并登录 HSM。要共享密钥，请以拥有该密钥的加密用户 (CU) 的身份登录到 HSM。只有密钥所有者能够共享密钥。

使用 shareKey 命令共享或取消共享密钥，并指定密钥的句柄以及用户的 ID。要与多个用户共享或取消共享，请指定用户 ID 的逗号分隔列表。要共享密钥，请使用 1 作为命令的最后一个参数，如以下示例所示。要取消共享，请使用 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)
```

下面说明 shareKey 命令的语法。

```
aws-cloudhsm>shareKey <key handle> <user ID> <Boolean: 1 for share, 0 for unshare>
```

如何使用 CMU 将密钥标记为可信

本节中的内容将提供有关使用 CMU 将密钥标记为可信密钥的说明。

1. 使用 loginHSM 命令，以加密员（CO）的身份登录。
2. 使用 OBJ_ATTR_TRUSTED（值 134）设置为 true（1）的 setAttribute 命令。

   ```
   setAttribute <Key Handle> 134 1
   ```

管理克隆的集群

使用 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility) 同步远程区域中的集群，前提为该区域的集群最初是从另一个区域集群的备份中创建的。假设您将一个集群复制到另一个区域（目标），之后您想同步原始集群（源）的更改。在这样的场景中，您可以使用 CMU 来同步集群。具体方法是创建一个新的 CMU 配置文件，在新文件中指定两个集群的硬件安全模块 (HSM, hardware security modules)，然后使用 CMU 通过该文件连接到集群。

跨克隆集群使用 CMU

1. 创建当前配置文件的副本，然后将副本的名称更改为其他名称。

   例如，使用以下文件位置查找并创建当前配置文件的副本，然后将副本的名称从 `cloudhsm_mgmt_config.cfg` 更改为 `syncConfig.cfg`。

   - Linux：`/opt/cloudhsm/etc/cloudhsm_mgmt_config.cfg`
   - Windows：`C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_config.cfg`

2. 在重命名的副本中，添加目标 HSM（需要同步的国外区域的 HSM）的弹性网络接口（ENI）IP。我们建议您在源 HSM 下方添加目标 HSM。

```
{
    ...
    "servers": [
        {
            ...
            "hostname": "<ENI Source IP>",
            ...
        },
        {
            ...
            "hostname": "<ENI Destination IP>",
            ...
        }
    ]
}
```

   有关如何获取 IP 地址的更多信息，请参阅 the section called “获取 HSM 的 IP 地址”。

3. 使用新的配置文件初始化 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. 检查返回的状态消息，以确保 CMU 连接到所有所需的 HSM，并确定返回的 ENI IP 中哪个 IP 与每个集群相对应。使用 syncUser 和 syncKey 手动同步用户和密钥。有关更多信息，请参阅 syncUser 和 syncKey。

获取 HSM 的 IP 地址

使用此节获取 HSM 的 IP 地址。

获取 HSM 的 IP 地址（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 要更改 Amazon Web Services Region，请使用页面右上角的区域选择器（Region selector）。
3. 要打开该集群的详细信息页面，请在集群表中选择集群 ID。
4. 要获取 IP 地址，请在 HSM 选项卡上，选择 ENI IP 地址下列出的 IP 地址之一。

获取 HSM 的 IP 地址 ()AWS CLI

- 使用 AWS CLI中的 describe-clusters 命令获取 HSM 的 IP 地址。在该命令的输出中，HSM 的 IP 地址为 `EniIp` 的值。

```
$  aws cloudhsmv2 describe-clusters


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

相关 主题

- syncUser
- syncKey
- 跨区域复制备份

AWS CloudHSM 命令行工具

本主题介绍可用于管理和使用 AWS CloudHSM的命令行工具。

主题

了解命令行工具

除了用于管理 AWS 资源的 AWS Command Line Interface (AWS CLI) 之外，还 AWS CloudHSM 提供了用于在 HSM 上创建和管理 HSM 用户和密钥的命令行工具。在里面， AWS CloudHSM 你可以使用熟悉的 CLI 来管理集群，使用 CloudHSM 命令行工具来管理你的 HSM。

以下是各类命令行工具：

管理集群和 HSM

cloudhsmv2 命令在 AWS CLI，模块里有 H PowerShell SM2 cmdlet AWSPowerShell

- 这些工具获取、创建、删除和标记 AWS CloudHSM 集群和 HSM：
- 要在 CLI 中使用 CloudHSMv2 命令中的命令，您需要进行安装和配置。 AWS CLI
- 该模块中的 HS PowerShell M2 cmdlet 在 Windows AWSPowerShell 模块和跨平台核心 PowerShell模块中可用。 PowerShell

如何管理 HSM 用户

CloudHSM CLI

- 使用 CloudHSM CLI 创建用户、删除用户、列出用户、更改用户密码和更新用户多重身份验证 (MFA, multifactor authentication)。它不包含在 AWS CloudHSM 客户端软件中。有关安装此工具的指导，请参阅安装和配置 CloudHSM CLI。

帮助程序工具

有两个工具可以帮助您使用 AWS CloudHSM 工具和软件库：

- 配置工具将更新您的 CloudHSM 客户端配置文件。这 AWS CloudHSM 允许同步集群中的 HSM。

  AWS CloudHSM 提供两个主要版本，客户端 SDK 5 是最新版本。与客户端软件开发工具包 3（之前的系列）相比，它具有多种优势。
- pkpspeed 测量独立于软件库的 HSM 硬件的性能。

适用于以前的 SDK 的工具

使用密钥管理工具 (KMU, key management tool) 创建、删除、导入和导出对称密钥和非对称密钥对：

- key_mgmt_util。此工具包含在 AWS CloudHSM 客户端软件中。

使用 CloudHSM 管理工具 (CMU, CloudHSM management tool) 创建和删除 HSM 用户，包括完成用户管理任务的仲裁身份验证

- cloudhsm_mgmt_util。此工具包含在 AWS CloudHSM 客户端软件中。

Configure 工具

AWS CloudHSM 自动同步集群中所有硬件安全模块 (HSM) 之间的数据。configure 工具将更新同步机制所使用的配置文件中的 HSM 数据。在您使用命令行工具之前，尤其是集群中的 HSM 发生更改时，使用 configure 可刷新 HSM 数据。

AWS CloudHSM 包括两个主要的客户端 SDK 版本：

- 客户端软件开发工具包 5：这是我们最新的默认客户端软件开发工具包。有关它提供的益处和优势的信息，请见 客户端软件开发工具包 5 的优点。
- 客户端软件开发工具包 3 ：这是我们的旧版客户端软件开发工具包。它包括一整套用于平台和语言应用程序的兼容性组件以及管理工具。

有关从客户端 SDK 3 迁移到客户端 SDK 5 的说明，请参阅从客户端软件开发工具包 3 迁移到客户端软件开发工具包 5。

主题

客户端软件开发工具包 5 配置工具

使用客户端软件开发工具包 5 配置工具更新客户端配置文件。

客户端软件开发工具包 5 中的每个组件都包含一个配置工具，配置工具的文件名中包含该组件的指示符。例如，客户端软件开发工具包 5 的 PKCS #11 库包括一个在 Linux 或 `configure-pkcs11.exe` Windows 上名为 `configure-pkcs11` 的配置工具。

语法

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]
```

高级配置

有关特定于客户端软件开发工具包 5 配置工具的高级配置列表，请参阅客户端软件开发工具包 5 配置工具的高级配置。

> **Important**
>
> 对配置进行任何更改之后，均需重启应用程序才能使更改生效。

示例

以下示例演示如何使用客户端软件开发工具包 5 的配置工具。

引导客户端软件开发工具包 5

Example

本示例使用 -a 参数更新客户端软件开发工具包 5 的 HSM 数据。要使用 -a 参数，您必须拥有集群中其中一个 HSM 的 IP 地址。

PKCS #11 library

引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

引导适用于客户端软件开发工具包 5 的 Windows EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

OpenSSL Dynamic Engine

引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

JCE provider

引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

引导适用于客户端软件开发工具包 5 的 Windows EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

CloudHSM CLI

引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

引导适用于客户端软件开发工具包 5 的 Windows EC2 实例

- 使用配置工具指定集群中 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` 参数代替 `-a <HSM_IP_ADDRESSES>`。要查看使用 `--cluster-id` 的要求，请参阅 客户端软件开发工具包 5 配置工具。

有关 `-a` 参数的更多信息，请参阅 the section called “参数”。

为客户端软件开发工具包 5 指定集群、区域和端点

Example

此示例使用 `cluster-id` 参数通过 `DescribeClusters` 调用来引导客户端软件开发工具包 5。

PKCS #11 library

使用 **`cluster-id`** 引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用集群 ID `cluster-1234567` 指定集群中 HSM 的 IP 地址。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id cluster-1234567
```

使用 **`cluster-id`** 引导适用于客户端软件开发工具包 5 的 Windows EC2 实例

- 使用集群 ID `cluster-1234567` 指定集群中 HSM 的 IP 地址。

```
"C:\Program Files\Amazon\CloudHSM\configure-pkcs11.exe" --cluster-
id cluster-1234567
```

OpenSSL Dynamic Engine

使用 **cluster-id** 引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用集群 ID `cluster-1234567` 指定集群中 HSM 的 IP 地址。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id cluster-1234567
```

JCE provider

使用 **cluster-id** 引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用集群 ID `cluster-1234567` 指定集群中 HSM 的 IP 地址。

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id cluster-1234567
```

使用 **cluster-id** 引导适用于客户端软件开发工具包 5 的 Windows EC2 实例

- 使用集群 ID `cluster-1234567` 指定集群中 HSM 的 IP 地址。

```
"C:\Program Files\Amazon\CloudHSM\configure-jce.exe" --cluster-
id cluster-1234567
```

CloudHSM CLI

使用 **cluster-id** 引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用集群 ID `cluster-1234567` 指定集群中 HSM 的 IP 地址。

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id cluster-1234567
```

使用 **cluster-id** 引导适用于客户端软件开发工具包 5 的 Windows EC2 实例

- 使用集群 ID `cluster-1234567` 指定集群中 HSM 的 IP 地址。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --cluster-
id cluster-1234567
```

您可以将 `--region` 和 `--endpoint` 参数与 `cluster-id` 参数结合使用，以指定系统如何进行 `DescribeClusters` 调用。例如，如果集群的区域与 AWS CLI 默认配置区域不同，则应使用 `--region` 参数使用该区域。此外，您还可以指定用于调用的 AWS CloudHSM API 终端节点，这可能是各种网络设置所必需的，例如使用不使用默认 DNS 主机名的 VPC 接口终端节点。 AWS CloudHSM PKCS #11 library

使用自定义端点和区域启动 Linux EC2 实例

- 使用配置工具使用自定义区域和终端节点在集群中指定 HSM 的 IP 地址。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id cluster-1234567 --
region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

启动带有端点和区域的 Windows EC2 实例

- 使用配置工具使用自定义区域和终端节点在集群中指定 HSM 的 IP 地址。

```
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

使用自定义端点和区域启动 Linux EC2 实例

- 使用配置工具使用自定义区域和终端节点在集群中指定 HSM 的 IP 地址。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

JCE provider

使用自定义端点和区域启动 Linux EC2 实例

- 使用配置工具使用自定义区域和终端节点在集群中指定 HSM 的 IP 地址。

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

启动带有端点和区域的 Windows EC2 实例

- 使用配置工具使用自定义区域和终端节点在集群中指定 HSM 的 IP 地址。

```
"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

使用自定义端点和区域启动 Linux EC2 实例

- 使用配置工具使用自定义区域和终端节点在集群中指定 HSM 的 IP 地址。

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id cluster-1234567 --region us-
east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

启动带有端点和区域的 Windows EC2 实例

- 使用配置工具使用自定义区域和终端节点在集群中指定 HSM 的 IP 地址。

```
"C:\Program Files\Amazon\CloudHSM\configure-cli.exe" --cluster-
id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-
east-1.amazonaws.com
```

有关 `--cluster-id`、`--region` 和 `--endpoint` 参数的更多信息，请参阅 the section called “参数”。

更新 TLS 客户端与服务器之间双向认证的客户端证书和密钥

Example

此示例说明如何使用`server-client-cert-file`和`--server-client-key-file`参数通过为其指定自定义密钥和 SSL 证书来重新配置 SSL AWS CloudHSM

PKCS #11 library

在 Linux 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份认证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用配置工具指定 `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
```

在 Windows 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份验证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用 PowerShell 解释器，使用配置工具指定`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

在 Linux 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份认证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用配置工具指定 `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

在 Linux 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份认证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用配置工具指定 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
```

在 Windows 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份验证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用 PowerShell 解释器，使用配置工具指定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

在 Linux 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份认证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用配置工具指定 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
```

在 Windows 上对客户端软件开发工具包 5 使用自定义证书和密钥进行 TLS 客户端与服务器之间的双向身份验证

1. 将您的密钥和证书复制到相应目录。

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

2. 使用 PowerShell 解释器，使用配置工具指定`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
```

有关 `server-client-cert-file` 和 `--server-client-key-file` 参数的更多信息，请参阅 the section called “参数”。

禁用客户端密钥持久性设置

Example

此示例使用 `--disable-key-availability-check` 参数禁用客户端密钥持久性设置。要使用单个 HSM 运行集群，必须禁用客户端密钥持久性设置。

PKCS #11 library

在 Linux 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

在 Windows 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-
availability-check
```

OpenSSL Dynamic Engine

在 Linux 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

JCE provider

在 Linux 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

在 Windows 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-
availability-check
```

CloudHSM CLI

在 Linux 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

在 Windows 上禁用客户端软件开发工具包 5 的客户端密钥持久性

- 使用配置工具禁用客户端密钥持久性设置。

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-
availability-check
```

有关 `--disable-key-availability-check` 参数的更多信息，请参阅 the section called “参数”。

管理日志记录选项

Example

客户端软件开发工具包 5 使用 `log-file`、`log-level`、`log-rotation` 和 `log-type` 参数来管理日志记录。

> **Note**
>
> 要为无服务器环境（例如 AWS Fargate 或 AWS Lambda）配置您的软件开发工具包，我们建议您将日志类型配置为 AWS CloudHSM 。`term`客户机日志将输出到为该环境配置的 Lo CloudWatch gs 日志组，`stderr`并在该组中捕获。

PKCS #11 library

默认日志记录位置

- 如果您没有为文件指定位置，则系统会将日志写入以下默认位置：

Linux

```
/opt/cloudhsm/run/cloudhsm-pkcs11.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
```

配置日志记录级别并将其他日志记录选项设置为默认值

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-level info
  ```

配置文件日志记录选项

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type file --log-file <file name
   with path> --log-rotation daily --log-level info
  ```

配置终端日志记录选项

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type term --log-level info
  ```

OpenSSL Dynamic Engine

默认日志记录位置

- 如果您没有为文件指定位置，则系统会将日志写入以下默认位置：

  Linux

  ```
  stderr
  ```
  ```

配置日志记录级别并将其他日志记录选项设置为默认值

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-level info
  ```

配置文件日志记录选项

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type <file name> --log-file file --log-rotation daily --log-level info
  ```

配置终端日志记录选项

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type term --log-level info
  ```

JCE provider

默认日志记录位置

- 如果您没有为文件指定位置，则系统会将日志写入以下默认位置：

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-jce.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
  ```

配置日志记录级别并将其他日志记录选项设置为默认值

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-level info
  ```

配置文件日志记录选项

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type file --log-file <file name> --log-rotation daily --log-level info
  ```

配置终端日志记录选项

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type term --log-level info
  ```

CloudHSM CLI

默认日志记录位置

- 如果您没有为文件指定位置，则系统会将日志写入以下默认位置：

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-cli.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-cli.log
  ```

配置日志记录级别并将其他日志记录选项设置为默认值

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-level info
  ```

配置文件日志记录选项

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type file --log-file <file name> --log-rotation daily --log-level info
  ```

配置终端日志记录选项

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type term --log-level info
  ```

有关 log-file、log-level、log-rotation 和 log-type 参数的更多信息，请参阅 the section called “参数”。

为客户端软件开发工具包 5 放置颁发证书

Example

此示例使用 --hsm-ca-cert 参数更新客户端软件开发工具包 5 的颁发证书的位置。

PKCS #11 library

在 Linux 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

在 Windows 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

OpenSSL Dynamic Engine

在 Linux 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。
  ```

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

JCE provider

在 Linux 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

在 Windows 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

CloudHSM CLI

在 Linux 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

在 Windows 上放置适用于客户端软件开发工具包 5 的颁发证书

- 使用配置工具指定签发证书的位置。

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

有关 `--hsm-ca-cert` 参数的更多信息，请参阅 the section called “参数”。

参数

-a *<ENI IP address>*

将指定的 IP 地址添加到客户端软件开发工具包 5 配置文件。输入集群中 HSM 的任何 ENI IP 地址。有关如何使用该选项的更多信息，请参阅引导客户端软件开发工具包 5。

必需：是

--hsm-ca-cert <customerCA certificate file path>

存储用于将 EC2 客户端实例连接到集群的证书颁发机构 (CA, certificate authority) 证书的目录路径。这是您在初始化集群时创建的文件。默认情况下，系统会在以下位置查找此文件：

Linux

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

Windows

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

有关初始化集群或放置证书的更多信息，请参阅 ??? 和 ???。

必需：否

--cluster-id *<cluster ID>*

进行 `DescribeClusters` 调用以查找集群中与集群 ID 关联的所有 HSM 弹性网络接口（ENI）IP 地址。系统将 ENI IP 地址添加到 AWS CloudHSM 配置文件中。

> **Note**
>
> 如果您在无法访问公共互联网的 VPC 中使用来自 EC2 实例的`--cluster-id`参数，则必须创建一个接口 VPC 终端节点进行连接 AWS CloudHSM。有关 VPC 端点的更多信息，请参阅 ???。

必需：否

--endpoint ***<endpoint>***

指定用于进行`DescribeClusters`呼叫的 AWS CloudHSM API 端点。设置此选项时，您必须与 `--cluster-id` 结合。

必需：否

--region ***<region>***

指定集群所在区域。设置此选项时，您必须与 `--cluster-id` 结合。

如果您不提供 `--region` 参数，则系统会通过尝试读取 `AWS_DEFAULT_REGION` 或 `AWS_REGION` 环境变量选择区域。如果未设置这些变量，则系统会在 AWS Config 文件中检查与您的配置文件关联的区域（通常 `~/.aws/config`），除非您在 `AWS_CONFIG_FILE` 环境变量中指定了其他文件。如果以上均未设置，则系统默认为 `us-east-1` 区域。

必需：否

--server-client-cert-file <client certificate file path>

用于 TLS 客户端服务器相互身份验证的客户端认证路径。

只有当您不希望使用客户端软件开发工具包 5 中包含的默认密钥和 SSL/TLS 证书时，才使用此选项。设置此选项时，您必须与 `--server-client-key-file` 结合。

必需：否

--server-client-key-file <client key file path>

用于 TLS 客户端-服务器相互身份验证的客户端密钥的路径。

只有当您不希望使用客户端软件开发工具包 5 中包含的默认密钥和 SSL/TLS 证书时，才使用此选项。设置此选项时，您必须与 `--server-client-cert-file` 结合。

必需：否

--log-level ***<error | warn | info | debug | trace>***

指定系统应写入日志文件的最低日志记录级别。每个级别都包括之前的级别，以“error”为最低等级，并追踪最高等级。这意味着，如果您指定“error”，系统只会将错误写入日志。如果指定“trace”，则系统会将错误、警告、提示 (info) 消息和调试消息写入日志。有关更多信息，请参阅客户端软件开发工具包 5 日志记录。

必需：否

--log-rotation ***<daily | weekly>***

指定系统轮换日志的频率。有关更多信息，请参阅客户端软件开发工具包 5 日志记录。

必需：否

--log-file ***<file name with path>***

指定系统将写入日志文件的位置。有关更多信息，请参阅客户端软件开发工具包 5 日志记录。

必需：否

--log-type ***<term | file>***

指定系统是将日志写入文件还是终端。有关更多信息，请参阅客户端软件开发工具包 5 日志记录。

必需：否

-h | --help

显示“帮助”。

必需：否

-v | --version

显示版本。

必需：否

--disable-key-availability-check

标记以禁用密钥可用性仲裁。使用此标志表示 AWS CloudHSM 应禁用密钥可用性法定人数，并且您可以使用集群中仅存在于一个 HSM 上的密钥。有关使用此标记设置密钥可用性仲裁的更多信息，请参阅 ???。

必需：否

--enable-key-availability-check

标记以启用密钥可用性仲裁。使用此标志表示 AWS CloudHSM 应使用密钥可用性法定人数，并且在这些密钥存在于集群中的两个 HSM 上之前不允许您使用密钥。有关使用此标记设置密钥可用性仲裁的更多信息，请参阅 ???。

默认情况下启用。

必需：否

--disable-validate-key-at-init

通过指定您可以跳过初始化调用来验证密钥的权限以便后续调用，从而提高性能。请谨慎使用。

背景：PKCS #11 库中的某些机制支持多部分操作，在这些操作中，初始化调用会验证您是否可以在后续调用中使用该密钥。这需要对 HSM 进行验证调用，这会延长整体操作的延迟。此选项使您可以禁用后续调用，并有可能提高性能。

必需：否

--enable-validate-key-at-init

指定您应该使用初始化调用来验证密钥的权限，以便后续调用。这是默认选项。`enable-validate-key-at-init` 用于在您使用 `disable-validate-key-at-init` 暂停这些初始化调用后恢复这些调用。

必需：否

相关 主题

- DescribeClusters API 操作
- describe-clusters AWS CLI
- Get-HSM2Cluster PowerShell cmdlet
- 引导客户端软件开发工具包 5
- AWS CloudHSM VPC 端点
- 管理客户端软件开发工具包 5 密钥持久性设置
- 客户端软件开发工具包 5 日志记录

客户端软件开发工具包 5 配置工具的高级配置

客户端软件开发工具包 5 配置工具包括高级配置，这些配置不是大多数客户使用的常规功能。高级配置提供了其他功能。

- PKCS #11 高级配置
  - 连接至多个 PKCS#11 插槽
  - PKCS #11 重试命令
- JCE 的高级配置
  - 使用 JCE 提供程序连接到多个集群
  - JCE 的重试命令
  - 使用 JCE 提取密钥
- OpenSSL 的高级配置
  - OpenSSL 重试命令
- AWS CloudHSM 命令行界面 (CLI) 的高级配置
  - 使用 CloudHSM CLI 连接到多个集群

客户端软件开发工具包 3 配置工具

使用客户端软件开发工具包 3 配置工具启动客户端进程守护程序并配置 CloudHSM 管理实用程序。

语法

```
configure -h | --help
          -a <ENI IP address>
          -m [-i <daemon_id>]
          --ssl --pkey <private key file> --cert <certificate file>
          --cmu <ENI IP address>
```

示例

以下示例演示如何使用 configure 工具。

Example : 更新 AWS CloudHSM 客户端的 HSM 数据和 key_mgmt_util

此示例使用-a参数更新 AWS CloudHSM 客户端和 key configure _mgmt_util 的 HSM 数据。要使用 -a 参数，您必须拥有集群中其中一个 HSM 的 IP 地址。使用控制台或 AWS CLI 获取 IP 地址。

获取 HSM 的 IP 地址（控制台）

1. 打开 AWS CloudHSM 控制台，网址为 https://console.aws.amazon.com/cloudhsm/home。
2. 要更改 Amazon Web Services Region，请使用页面右上角的区域选择器（Region selector）。
3. 要打开该集群的详细信息页面，请在集群表中选择集群 ID。
4. 要获取 IP 地址，请在 HSM 选项卡上，选择 ENI IP 地址下列出的 IP 地址之一。

获取 HSM 的 IP 地址 ()AWS CLI

- 使用 AWS CLI中的 describe-clusters 命令获取 HSM 的 IP 地址。在该命令的输出中，HSM 的 IP 地址为 `EniIp` 的值。

```
$  aws cloudhsmv2 describe-clusters


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

更新 HSM 数据

1. 在更新-a参数之前，请停止 AWS CloudHSM 客户端。这将防止在 configure 编辑客户端的配置文件时可能出现的冲突。如果客户端已停止，则此命令不会产生影响，因此您可以在脚本中使用它。

   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

- 对于 Windows 客户端 1.1.2 以上版本：

  ```
  C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
  ```

- 对于 Windows 客户端 1.1.1 及更低版本：

  在启动 AWS CloudHSM 客户端的命令窗口中使用 C trl + C。

2. 此步骤使用 configure 的 `-a` 参数将 `10.0.0.9` ENI IP 地址添加到配置文件中。

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. 接下来，重新启动 AWS CloudHSM 客户端。当 客户端启动时，它将使用其配置文件中的 ENI IP 地址查询集群。然后，它会将集群中所有 HSM 的 ENI IP 地址写入到 `cluster.info` 文件中。

   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

   - 对于 Windows 客户端 1.1.2 以上版本：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 对于 Windows 客户端 1.1.1 及更低版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

命令完成后，AWS CloudHSM 客户端和 key_mgmt_util 使用的 HSM 数据将完整且准确。

Example : 从客户端软件开发工具包 3.2.1 及更早版本更新 CMU 的 HSM 数据

此示例使用 `-m` configure 命令将更新的 HSM 数据从 `cluster.info` 文件复制到 cloudhsm_mgmt_util 使用的 `cloudhsm_mgmt_util.cfg` 文件。将其与客户端软件开发工具包 3.2.1 及更早版本附带的 CMU 配合使用。

- 在运行之前`-m`，请停止 AWS CloudHSM 客户端，运行`-a`命令，然后重新启动 AWS CloudHSM 客户端，如前面的示例所示。这将确保从 `cluster.info` 文件复制到 `cloudhsm_mgmt_util.cfg` 文件的数据是完整且准确的。

  Linux

  ```
  $ sudo /opt/cloudhsm/bin/configure -m
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\bin\ configure.exe -m
  ```

Example ：从客户端软件开发工具包 3.3.0 及更高版本更新 CMU 的 HSM 数据

此示例使用 configure 命令的 `--cmu` 参数更新 CMU 的 HSM 数据。将其与客户端软件开发工具包 3.3.0 及更高版本附带的 CMU 配合使用。有关使用 CMU 的更多信息，请参阅使用 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility) 管理用户和将 CMU 与客户端软件开发工具包 3.2.1 及更早版本一起使用。

- 使用 `--cmu` 参数传递集群中 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>
```

参数

-h | --help

显示命令语法。

必需：是

-a *<ENI IP address>*

将指定的 HSM 弹性网络接口 (ENI) IP 地址添加到 AWS CloudHSM 配置文件。输入集群中任意 HSM 的 ENI IP 地址。您选择哪个地址并不重要。

要获取集群中 HSM 的 ENI IP 地址，请使用DescribeClusters操作、desc ribe-clusters AWS CLI 命令或 cmdlet。Get-HSM2Cluster PowerShell

> **Note**
>
> 在运行 -aconfigure命令之前，请停止 AWS CloudHSM 客户端。然后，-a命令完成后，重新启动 AWS CloudHSM 客户端。有关详细信息，请参阅示例。

此参数编辑以下配置文件：

- `/opt/cloudhsm/etc/cloudhsm_client.cfg`: 由 AWS CloudHSM 客户端和 key_mgmt_util 使用。
- `/opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg`：由 cloudhsm_mgmt_util 使用。

当 AWS CloudHSM 客户端启动时，它使用其配置文件中的 ENI IP 地址来查询集群，并使用集群中所有 HSM 的正确 ENI IP 地址更新`cluster.info`文件 (`/opt/cloudhsm/daemon/1/cluster.info`)。

必需：是

-m

更新 CMU 使用的配置文件中的 HSM ENI IP 地址。

> **Note**
>
> 该 -m 参数适用于与客户端软件开发工具包 3.2.1 及更早版本中的 CMU 一起使用。对于来自客户端软件开发工具包 3.3.0 及更高版本的 CMU，请参阅 --cmu 参数，它简化了 CMU 更新 HSM 数据的过程。

更新-a参数configure然后启动客户端时， AWS CloudHSM 客户端守护程序会查询集群，并使用集群中所有 HSM 的正确 HSM IP 地址更新cluster.info文件。运行 -m configure 命令，通过将 HSM IP 地址从 cluster.info 复制到 cloudhsm_mgmt_util 使用的 cloudhsm_mgmt_util.cfg 配置文件，完成更新。

在运行-aconfigure命令之前，请务必运行命令并重新启动 AWS CloudHSM 客户端。-m这将确保从 cluster.info 复制到 cloudhsm_mgmt_util.cfg 的数据是完整且准确的。

必需：是

-i

指定备用客户端守护程序。默认值表示 AWS CloudHSM 客户端。

默认值：1

必需：否

--ssl

使用指定的私有密钥和证书替换集群的 SSL 密钥与证书。使用此参数时，--pkey 和 --cert 参数为必需项。

必需：否

--pkey

指定新的私有密钥。输入包含私有密钥的文件对应的路径和文件名。

在指定 --ssl 时，必需：是。否则，不应使用此项。

--cert

指定新证书。输入包含证书的文件对应的路径和文件名。此证书应一直串联到 `customerCA.crt` 证书，后者是用于初始化集群的自签名证书。有关更多信息，请参见初始化集群。

在指定 --ssl 时，必需：是。否则，不应使用此项。

--cmu ***<ENI IP address>***

将 `-a` 和 `-m` 参数合并为一个参数。将指定的 HSM 弹性网络接口 (ENI) IP 地址添加到 AWS CloudHSM 配置文件中，然后更新 CMU 配置文件。输入集群中任何 HSM 的 IP 地址。对于客户端软件开发工具包 3.2.1 及更早版本，请参阅将 CMU 与客户端软件开发工具包 3.2.1 及更早版本一起使用。

必需：是

相关 主题

- 设置 key_mgmt_util

CloudHSM 命令行界面（CLI）

CloudHSM CLI 可帮助管理员管理用户和加密用户管理其集群中的密钥。它包括可用于创建、删除和列出用户、更改用户密码、更新用户多因素身份验证 (MFA) 的工具。它还包括生成、删除、导入和导出密钥、获取和设置属性、查找密钥以及执行加密操作的命令。

有关定义的 CloudHSM CLI 用户列表，请参阅 使用 CloudHSM CLI 管理 HSM 用户 。有关 CloudHSM CLI 的已定义关键属性列表，请参阅。CloudHSM CLI 的密钥属性有关如何使用 CloudHSM CLI 管理密钥的信息，请参阅。使用 CloudHSM CLI 管理密钥

要快速开始使用，请参阅 开始使用 CloudHSM 命令行界面（CLI）。有关 CloudHSM CLI 命令的详细信息以及使用这些命令的示例，请参阅 CloudHSM CLI 命令的引用。

主题

- CloudHSM 命令行界面（CLI）支持的平台
- 开始使用 CloudHSM 命令行界面（CLI）
- 交互模式和单命令模式
- CloudHSM CLI 的密钥属性

- 从客户端 SDK 3 CMU 和 KMU 迁移到客户端 SDK 5 CloudHSM CLI
- CLI 的高级配置
- CloudHSM CLI 命令的引用

CloudHSM 命令行界面（CLI）支持的平台

Linux 支持

支持的平台	架构：x86_64	ARM 架构
Amazon Linux 2	是	是
Amazon Linux 2023	是	是
CentOS 7 (7.8+)	是	不支持
红帽企业 Linux 7 (7.8+)	是	不支持
红帽企业 Linux 8 (8.3+)	是	不支持
红帽企业 Linux 9 (9.2+)	是	是
Ubuntu 20.04 LTS	是	不支持
Ubuntu 22.04 LTS	是	是

注意：SDK 5.4.2 是最后一个提供 CentOS 8 平台支持的版本。有关更多信息，请参阅 CentOS 网站。

Windows 支持

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

开始使用 CloudHSM 命令行界面（CLI）

CloudHSM 命令行界面 (CLI) 允许您管理集群中的用户。 AWS CloudHSM 使用本主题开始执行基础 HSM 用户管理任务，例如创建用户、列出用户以及将 CloudHSM CLI 连接至集群。

安装 CloudHSM CLI

使用以下命令下载和安装 CloudHSM CLI。

Amazon Linux 2

x86_64 架构上的 Amazon Linux 2：

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

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

ARM64 架构上的 Amazon Linux 2：

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

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

Amazon Linux 2023

基于 x86_64 架构的亚马逊 Linux 2023：

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

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

基于 ARM64 架构的亚马逊 Linux 2023：

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

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

CentOS 7 (7.8+)

x86_64 架构上的 CentOS 7：

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

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

RHEL 7 (7.8+)

x86_64 架构上的 RHEL 7：

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

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

RHEL 8 (8.3+)

x86_64 架构上的 RHEL 8：

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

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

RHEL 9 (9.2+)

x86_64 架构上的 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
```

ARM64 架构上的 RHEL 9：

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

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

Ubuntu 20.04 LTS

x86_64 架构上的 Ubuntu 20.04 LTS：

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

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

Ubuntu 22.04 LTS

x86_64 架构上的 Ubuntu 22.04 LTS：

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

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

ARM64 架构上的 Ubuntu 22.04 LTS：

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

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

Windows Server 2016

对于 x86_64 架构上的 Windows Server 2016，请 PowerShell 以管理员身份打开并运行以下命令：

```
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

对于 x86_64 架构上的 Windows Server 2019，请 PowerShell 以管理员身份打开并运行以下命令：

```
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
```

使用以下命令配置 CloudHSM CLI。

引导适用于客户端软件开发工具包 5 的 Linux EC2 实例

- 使用配置工具指定集群中 HSM 的 IP 地址。

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

引导适用于客户端软件开发工具包 5 的 Windows EC2 实例

- 使用配置工具指定集群中 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. 使用以下命令启动 CloudHSM CLI。

   Linux

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

   Windows

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

2. 使用 login 命令登录到集群。所有用户都可以使用此命令。

   以下示例日志中的命令将登录 admin，即默认管理员账户。您在激活集群时设置该用户的密码。

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

   系统将会提示您输入密码。您输入密码，输出显示命令已成功。

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

3. 运行 user list 命令列出集群上的所有用户。

```
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. 使用 user create 创建名为 **example_user** 的 CU 用户。

   您可以创建 CU，因为在上一步中，您以管理员用户身份登录。只有管理员用户才能执行用户管理任务，例如创建和删除用户以及更改其他用户的密码。

```
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. 使用 user list 列出集群上的所有用户。

```
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命令注销集 AWS CloudHSM 群。

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

7. 使用 quit 命令来停止 CLI。

```
aws-cloudhsm > quit
```

交互模式和单命令模式

在 CloudHSM CLI 中，您可以通过两种不同的方式运行命令：单命令模式和交互模式。交互模式专为用户设计，而单命令模式专为脚本而设计。

> **Note**
>
> 所有命令均可在交互模式和单命令模式下运行。

交互模式

使用以下命令启动 CloudHSM CLI 交互模式

Linux

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

Windows

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

在交互模式下使用 CLI 时，您可以使用 login 命令登录用户账户。

要列出所有 CloudHSM CLI 命令，请运行以下命令：

```
aws-cloudhsm > help
```

要获取 CloudHSM CLI 命令的语法，请运行以下命令：

```
aws-cloudhsm >  help <command-name>
```

例如，要获取 HSM 上用户的列表，请输入 user list。

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

要结束 CloudHSM CLI 会话，请运行以下命令：

```
aws-cloudhsm > quit
```

单命令模式

如果您使用单命令模式运行 CloudHSM CLI，则需要设置两个环境变量来提供凭证：CLOUDHSM_PIN 和 CLOUDHSM_ROLE：

```
$ export CLOUDHSM_ROLE=admin
```

```
$ export CLOUDHSM_PIN=admin_username:admin_password
```

完成此操作后，您可以使用存储在环境中的凭证执行命令。

```
$ 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 的密钥属性

本主题介绍了如何使用 CloudHSM CLI 来设置密钥属性。CloudHSM CLI 中的密钥属性可以定义密钥的类型、密钥的功能或密钥添加标签的方式。有些属性定义了独特的特征（例如密钥的类型）。其他属性可以设置为 true 或 false，更改它们会激活或停用密钥的部分功能。

有关如何使用密钥属性的示例，请参阅父命令 key 下面列出的命令。

支持的属性

作为最佳实践，仅为应受限的属性设置值。如果您未指定值，CloudHSM CLI 将使用下表中指定的默认值。

下表列出了密钥属性、可能的值、默认值和相关注释。值列中的空单元格表示没有给该属性分配特定的默认值。

CloudHSM CLI 属性	值	可使用 key set-attribute 进行修改	可在创建密钥时进行设置
always-sensitive	如果 sensitive 一直设置为 True 且从未改变，则该值为 True。	否	否
check-value	密钥的检查值。有关更多信息，请参阅其他详细信息。	否	否
class	可能的值：secret-key 、public-key 和 private-key 。	否	是
curve	用于生成 EC 密钥对的椭圆曲线。 有效值：secp224r1 、secp256r1 、prime256v	否	可使用 RSA 进行设置，无法使用 EC 进行设置

CloudHSM CLI 属性	值	可使用 key set-attribute 进行修改	可在创建密钥时进行设置
	1 、secp384r1 、secp256k1 和 secp521r1		
decrypt	默认：False	支持	是
derive	默认：False	支持	是
destroyable	默认：True	支持	是
ec-point	对于 EC 密钥，十六进制格式的 ANSI X9.62 ecPoint 值 Q 的 DER 编码。 对于其他密钥类型，此属性不存在。	否	否
encrypt	默认：False	支持	是
extractable	默认：True	否	是
id	默认值：空	否	是
key-length-bytes	生成 AES 密钥所必需的。 有效值：16、24、和 32 字节。	否	否
key-type	可能的值：aes、rsa 和 ec	否	是
label	默认值：空	支持	是

CloudHSM CLI 属性	值	可使用 key set-attribute 进行修改	可在创建密钥时进行设置
local	默认：对于在 HSM 中生成的密钥为 True，对于导入到 HSM 的密钥为 False。	否	否
modifiable	默认：True	否	否
modulus	用于生成 RSA 密钥对的模数。对于其他密钥类型，此属性不存在。	否	否
modulus-size-bits	生成 RSA 密钥对所必需的。 最小值为 2048。	否	可使用 RSA 进行设置，无法使用 EC 进行设置
never-extractable	如果从未将“可提取”设置为 False，则该值为 True。 如果曾将“可提取”设置为 True，则该值为 False。	否	否
private	默认：True	否	是
public-exponent	生成 RSA 密钥对所必需的。 有效值：此值必须为大于或等于 65537 的奇数。	否	可使用 RSA 进行设置，无法使用 EC 进行设置

CloudHSM CLI 属性	值	可使用 key set-attribute 进行修改	可在创建密钥时进行设置
sensitive	默认值： • 对于 AES 密钥以及 EC 和 RSA 私钥，该值为 True。 • 对于 EC 和 RSA 公有密钥，该值为 False。	否	可以用私钥设置，不能用公有密钥设置。
sign	默认值： • 对于 AES 密钥，该值为 True。 • 对于 RSA 和 EC 密钥，该值为 False。	支持	是
token	默认：False	否	是
trusted	默认：False	是	不支持
unwrap	默认：False	支持	是
unwrap-template	值应使用应用于使用此包装密钥解开包装的任何密钥的属性模板。	是	不支持
verify	默认值： • 对于 AES 密钥，该值为 True。 • 对于 RSA 和 EC 密钥，该值为 False。	支持	是

CloudHSM CLI 属性	值	可使用 key set-attribute 进行修改	可在创建密钥时进行设置
wrap	默认：False	支持	是
wrap-template	值应使用属性模板来匹配使用此包装密钥包装的密钥。	是	不支持
wrap-with-trusted	默认：False	支持	是

其他详细信息

校验值

密钥校验值 (KCV) 是 HSM 导入或生成密钥时所产生密钥的 3 字节哈希值或校验和。您也可以在 HSM 之外计算检查值，例如在导出密钥之后。然后，您可对比检查值，以确认密钥的标识和完整性。要获取密钥的校验值，请使用带有 verbose 标记的key list。

AWS CloudHSM 使用以下标准方法生成校验值：

- 对称密钥：密钥加密零块结果的前 3 个字节。
- 非对称密钥对：公钥 SHA-1 哈希值的前 3 个字节。
- HMAC 密钥：目前不支持 HMAC 密钥 KCV。

相关 主题

- key
- CloudHSM CLI 命令的引用

从客户端 SDK 3 CMU 和 KMU 迁移到客户端 SDK 5 CloudHSM CLI

使用本主题将使用客户端 SDK 3 命令行工具、CloudHSM 管理实用程序 (CMU) 和密钥管理实用程序 (KMU) 的工作流迁移为改用客户端 SDK 5 命令行工具 CloudHSM CLI。

在中 AWS CloudHSM，客户应用程序使用 AWS CloudHSM 客户端软件开发套件 (SDK) 执行加密操作。客户端 SDK 5 是主要 SDK，它继续添加新功能和平台支持。本主题提供了命令行工具从客户端 SDK 3 迁移到客户端 SDK 5 的特定详细信息。

客户端 SDK 3 包括两个独立的命令行工具：用于管理用户的 CMU 和用于管理密钥和使用密钥执行操作的 KMU。客户端 SDK 5 将 CMU 和 KMU（客户端 SDK 3 中提供的工具）的功能整合到一个工具中，即. CloudHSM 命令行界面（CLI）用户管理操作可以在子命令用户和仲裁下找到。密钥管理操作可以在 k ey 子命令下找到，加密操作可以在 crypt o 子命令下找到。有关命令CloudHSM CLI 命令的引用的完整列表，请参阅。

> **Note**
>
> 如果您在 Client SDK 3 中依赖syncKey跨集群同步syncUser功能，请继续使用 CMU。客户端 SDK 5 中的 CloudHSM CLI 目前不支持此功能。

有关迁移到客户端 SDK 5 的说明，请参阅从客户端软件开发工具包 3 迁移到客户端软件开发工具包 5。有关迁移的好处，请参阅客户端软件开发工具包 5 的优点。

CLI 的高级配置

AWS CloudHSM 命令行界面 (CLI) 包括以下高级配置，这不是大多数客户使用的常规配置的一部分。这些配置还提供了其他功能。

- 连接到多个集群

使用 CloudHSM CLI 连接到多个集群

使用客户端 SDK 5，您可以将 CloudHSM CLI 配置为允许从单个 CLI 实例连接到多个 CloudHSM 集群。

按照本主题中的说明使用 CloudHSM CLI 使用多集群功能连接多个集群。

主题

- 多集群先决条件
- 配置 CloudHSM CLI 以实现多集群功能
- 配置-cli 添加集群

- 配置-cli 移除群集
- 使用多个集群

多集群先决条件

- 您要连接的两个或更多 AWS CloudHSM 集群及其集群证书。
- 已正确配置安全组的 EC2 实例，可连接至上述所有集群。有关如何设置集群和客户端实例的更多信息，请参阅入门 AWS CloudHSM。
- 要设置多集群功能，您必须已经下载并安装了 CloudHSM CLI。如果您尚未执行此操作，请参阅 ??? 中的说明。
- 您将无法访问配置为的集群，`./configure-cli[.exe] -a`因为它不会与关联`cluster-id`。您可以按照本指南中的说明`config-cli add-cluster`进行重新配置。

配置 CloudHSM CLI 以实现多集群功能

要配置 CloudHSM CLI 以实现多集群功能，请按照以下步骤操作：

1. 确定要连接的集群。
2. 使用 configu `add-cluster` re-cli 子命令将这些集群添加到您的 CloudHSM CL I 配置中，如下所述。
3. 重新启动任何 CloudHSM CLI 进程以使新配置生效。

配置-cli 添加集群

连接到多个集群时，使用`configure-cli add-cluster`命令将集群添加到您的配置中。

语法

```
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]
```

示例

使用 **cluster-id** 参数添加集群

Example

使用 configure-cli add-cluster 和 cluster-id 参数，将集群添加至 (ID 为 cluster-1234567) 您的配置。

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**
>
> 如果使用 cluster-id 参数和 configure-cli add-cluster 无法添加集群，请参见以下示例，以获取也需要通过 --region 和 --endpoint 参数识别待添加集群的、更长版本的命令。例如，如果集群的区域与配置默认 AWS CLI 配置区域不同，则应使用 --region 参数使用正确区域。此外，您还可以指定用于调用的 AWS CloudHSM API 终端节点，这可能是各种网络设置所必需的，例如使用不使用默认 DNS 主机名的 VPC 接口终端节点。 AWS CloudHSM

使用 **cluster-id**、**endpoint** 和 **region** 参数添加集群

Example

使用 configure-cli add-cluster、cluster-id、endpoint 和 region 参数，将集群 (ID 为 cluster-1234567) 添加至您的配置。

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
```

有关 `--cluster-id`、`--region` 和 `--endpoint` 参数的更多信息，请参阅 the section called “参数”。

参数

--cluster-id ***<Cluster ID>***

进行 `DescribeClusters` 调用以查找集群中与集群 ID 关联的所有 HSM 弹性网络接口（ENI）IP 地址。系统将 ENI IP 地址添加到 AWS CloudHSM 配置文件中。

> **Note**
>
> 如果您在无法访问公共互联网的 VPC 中使用来自 EC2 实例的`--cluster-id`参数，则必须创建要连接的接口 VPC 终端节点 AWS CloudHSM。有关 VPC 端点的更多信息，请参阅 ???。

必需：是

--endpoint ***<Endpoint>***

指定用于进行`DescribeClusters`呼叫的 AWS CloudHSM API 端点。设置此选项时，您必须与 `--cluster-id` 结合。

必需：否

--hsm-ca-cert <HsmCA Certificate Filepath>

指定 HSM CA 证书的文件路径。

必需：否

--region ***<Region>***

指定集群所在区域。设置此选项时，您必须与 `--cluster-id` 结合。

如果您不提供 `--region` 参数，则系统会通过尝试读取 `AWS_DEFAULT_REGION` 或 `AWS_REGION` 环境变量选择区域。如果未设置这些变量，则系统会在 AWS Config 文件中检查与您的配置文件关联的区域（通常 `~/.aws/config`），除非您在 `AWS_CONFIG_FILE` 环境变量中指定了其他文件。如果以上均未设置，则系统默认为 `us-east-1` 区域。

必需：否

--server-client-cert-file <Client Certificate Filepath>

用于 TLS 客户端服务器相互身份验证的客户端认证路径。

只有当您不希望使用客户端软件开发工具包 5 中包含的默认密钥和 SSL/TLS 证书时，才使用此选项。设置此选项时，您必须与 `--server-client-key-file` 结合。

必需：否

--server-client-key-file <Client Key Filepath>

用于 TLS 客户端-服务器相互身份验证的客户端密钥的路径。

只有当您不希望使用客户端软件开发工具包 5 中包含的默认密钥和 SSL/TLS 证书时，才使用此选项。设置此选项时，您必须与 `--server-client-cert-file` 结合。

必需：否

配置-cli 移除群集

使用 CloudHSM CLI 连接到多个集群时，`configure-cli remove-cluster`请使用命令从配置中删除一个集群。

语法

```
configure-cli remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

示例

使用 `cluster-id` 参数移除集群

Example

使用 `configure-cli remove-cluster` 和 `cluster-id` 参数，从您的配置中移除集群 (ID 为 `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
```

有关 `--cluster-id` 参数的更多信息，请参阅 the section called “参数”。

参数

--cluster-id ***<Cluster ID>***

要从配置中删除的集群的 ID。

必需：是

使用多个集群

使用 CloudHSM CLI 配置多个集群后，`cloudhsm-cli`使用命令与它们进行交互。

示例

使用交互模式**`cluster-id`**时设置默认值

Example

使用和`cluster-id`参数从您的配置中设置默认集群（ID 为`cluster-1234567`）。???

Linux

```
$ cloudhsm-cli interactive --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\cloudhsm-cli.exe interactive --cluster-
id cluster-1234567
```

运行单个命令**cluster-id**时设置

Example

使用cluster-id参数设置要???从中获取的集群（ID 为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 命令的引用

CloudHSM CLI 可帮助管理员管理其集群中的用户。 AWS CloudHSM CloudHSM CLI 可以在两种模式下运行：交互模式和单命令模式。要快速开始使用，请参阅 开始使用 CloudHSM 命令行界面（CLI）。

要运行大多数 CloudHSM CLI 命令，必须启动 CloudHSM CLI 并登录 HSM。如果要添加或删除 HSM，请更新 CloudHSM CLI 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

以下主题介绍 CloudHSM CLI 中的命令：

命令	描述	用户类型
集群激活	激活一个 CloudHSM 集群并确认该集群是新集群。必须先完成此操作，然后才能执行任何其他操作。	未激活的管理员
cluster hsm-info	列出集群中的 HSM。	全部[1]，包括未经身份验证的用户。不需要登录名。
加密标志 ecdsa	使用 EC 私钥和 ECDSA 签名机制生成签名。	加密用户 (CU)

命令	描述	用户类型
加密标志 rsa-pkcs	使用 RSA 私钥和 RSA-PKCS 签名机制生成签名。	CU
加密标志 rsa-pkcs-pss	使用 RSA 私钥和 RSA-PKCS-PSS 签名机制生成签名。	CU
加密验证 ecdsa	确认文件已通过给定的公钥在 HSM 中签名。验证签名是使用 ECDSA 签名机制生成的。将签名文件与源文件进行比较，并根据给定的 ecdsa 公钥和签名机制确定两者在密码学上是否相关。	CU
加密验证 rsa-pkcs	确认文件已通过给定的公钥在 HSM 中签名。验证签名是使用 RSA-PKCS 签名机制生成的。将签名文件与源文件进行比较，并根据给定的 rsa 公钥和签名机制确定两者在密码学上是否相关。	CU
加密验证 rsa-pkcs-pss	确认文件已通过给定的公钥在 HSM 中签名。验证签名是使用 RSA-PKCS-PSS 签名机制生成的。将签名文件与源文件进行比较，并根据给定的 rsa 公钥和签名机制确定两者在密码学上是否相关。	CU
key delete	从您的 AWS CloudHSM 集群中删除密钥。	CU
key generate-file	在您的 AWS CloudHSM 集群中生成密钥文件。	CU

命令	描述	用户类型
密钥 generate-asymmetric-pair rsa	在您的 AWS CloudHSM 集群中生成非对称 RSA 密钥对。	CU
密钥 generate-asymmetric-pair ec	在集群中生成非对称椭圆曲线 (EC) 密钥对。 AWS CloudHSM	CU
key generate-symmetric aes	在 AWS CloudHSM 集群中生成对称 AES 密钥。	CU
key generate-symmetric generic-secret	在您的 AWS CloudHSM 集群中生成对称通用密钥。	CU
密钥导入 pem	将 PEM 格式的密钥导入 HSM。您可以使用它来导入在 HSM 外生成的公有密钥。	CU
key list	查找 AWS CloudHSM 集群中当前用户的所有密钥。	CU
密钥复制	将密钥从源集群复制到克隆的目标集群。	CU
key set-attribute	设置集 AWS CloudHSM 群中密钥的属性。	CU 可以运行此命令，管理员可以设置可信属性。
key share	与 AWS CloudHSM 集群中的其他 CU 共享密钥。	CU
key unshare	取消与 AWS CloudHSM 集群中的其他 CU 共享密钥。	CU
key unwrap aes-gcm	使用 AES 封装密钥和 AES-GCM 解包机制将有效载荷密钥解封到集群中。	CU

命令	描述	用户类型
密钥解包 aes-no-pad	使用 AES 封装密钥和 AES-NO-PAD 解包机制将有效载荷密钥解封到集群中。	CU
key unwrap aes-pkcs5-pad	使用 AES 封装密钥和 AES-PKCS5-PAD 解包机制解开有效载荷密钥。	CU
密钥解包 aes-zero-pad	使用 AES 封装密钥和 AES-ZERO-PAD 解包机制将有效载荷密钥解包到集群中。	CU
密钥解包 cloudhsm-aes-gcm	使用 AES 封装密钥和 CLOUDHSM-AES-GCM 解包机制将有效载荷密钥解封到集群中。	CU
密钥解包 rsa-aes	使用 RSA 私钥和 RSA-AES 解包机制解封有效载荷密钥。	CU
key unwrap rsa-oaep	使用 RSA 私钥和 RSA-OAEP 解包机制解封有效载荷密钥。	CU
key unwrap rsa-pkcs	使用 RSA 私钥和 RSA-PKCS 解包机制解封有效载荷密钥。	CU
钥匙包装 aes-gcm	使用 HSM 上的 AES 密钥和 AES-GCM 包装机制封装有效载荷密钥。	CU
钥匙包装 aes-no-pad	使用 HSM 上的 AES 密钥和 AES-NO-PAD 包装机制封装有效载荷密钥。	CU
钥匙包裹 aes-pkcs5-pad	使用 HSM 上的 AES 密钥和 AES-PKCS5-PAD 包装机制封装有效载荷密钥。	CU

命令	描述	用户类型
钥匙包装 aes-zero-pad	使用 HSM 上的 AES 密钥和 AES-ZERO-PAD 包装机制封装有效载荷密钥。	CU
钥匙包装 cloudhsm-aes-gcm	使用 HSM 上的 AES 密钥和 CLOUDHSM-AES-GCM 包装机制封装有效载荷密钥。	CU
密钥包装 rsa-aes	使用 HSM 上的 RSA 公钥和 RSA-AES 封装机制封装有效载荷密钥。	CU
密钥包装 rsa-oaep	使用 HSM 上的 RSA 公钥和 RSA-OAEP 包装机制封装有效载荷密钥。	CU

命令	描述	用户类型

该key wrap rsa-pkcs命令使用 HSM 上的 RSA 公钥和包装机制封装有效载荷密钥。RSA-PKCS有效载荷密钥的extractable 属性必须设置为true。
 	
只有密钥的所有者，即创建密钥的加密用户 (CU)，才能封装密钥。共享密钥的用户可以在加密操作中使用该密钥。
 	
要使用该key wrap rsa-pkcs命令，您必须先在 AWS CloudHSM 集群中拥有 RSA 密钥。您可以使用钥匙 generate-asymmetric-pair 命令和设置为的wrap属性生成 RSA 密钥对。true
 	
用户类型
 	
以下类型的用户均可运行此命令。
 	
• 加密用户（CU）
 	
要求
 	
• 要运行此命令，必须以 CU 身份登录。
 	
语法
 	
`aws-cloudhsm > help key wrap rsa-pkcs`
 	
`Usage: key wrap rsa-pkcs [OPTIONS] --payload`
	使用 HSM 上的 RSA 公钥和 RSA-PKCS 包装机制封装有效载荷密钥。	CU

命令	描述	用户类型
login	登录您的集 AWS CloudHSM 群。	管理员、加密用户 (CU) 和应用程序用户 (AU)
logout	注销您的 AWS CloudHSM 集群。	管理员、加密用户 (CU) 和应用程序用户 (AU, appliance user)
quorum token-sign delete	删除仲裁授权服务的一个或多个令牌。	Admin
quorum token-sign generate	为仲裁授权服务生成令牌。	Admin
quorum token-sign list	列出您的 CloudHSM 集群中存在的所有令牌签名仲裁令牌。	全部[1]，包括未经身份验证的用户。不需要登录名。
法定代币符号 list-quorum-values	列出您的 CloudHSM 集群中设置的仲裁值。	全部[1]，包括未经身份验证的用户。不需要登录名。
quorum token-sign list-timeouts	获取所有令牌类型的令牌超时期限（以秒为单位）。	管理员和加密用户
法定代币符号 set-quorum-value	为仲裁授权服务设置新的仲裁值。	Admin
quorum token-sign set-timeout	为每种令牌类型设置令牌超时期限（以秒为单位）。	Admin
user change-mfa	更改用户的多重身份验证 (MFA, multi-factor authentication) 策略。	管理员，CU
user change-password	更改用户在 HSM 上的密码。所有用户都可更改自己的密码。管理员可以更改任何人的密码。	管理员，CU
user create	在您的 AWS CloudHSM 集群中创建用户。	Admin

命令	描述	用户类型
user delete	删除集 AWS CloudHSM 群中的用户。	Admin
user list	列出 AWS CloudHSM 集群中的用户。	全部[1]，包括未经身份验证的用户。不需要登录名。
user change-quorum token-sign register	为用户注册仲裁令牌签名仲裁策略。	Admin

注释

- [1] 所有用户包括所有列出的角色和未登录的用户。

cluster

cluster 是一组命令的父类别，当这些命令与父类别结合使用时，会创建特定于用户的命令。当前，用户类别由以下命令组成：

- 集群激活
- cluster hsm-info

集群激活

要激活新集群，请在 CloudHSM CLI 中运行 cluster activate 命令。使用该集群执行加密操作之前，必须运行该命令。

用户类型

以下类型的用户均可运行此命令。

- 未激活的管理员

语法

此命令没有输出参数。

```
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')
```

示例

此命令通过为您的管理员用户设置初始密码来激活您的集群。

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

相关 主题

- user create
- user delete
- user change-password

cluster hsm-info

请在 CloudHSM CLI 中使用 cluster hsm-info 命令以列出您的集群中的 HSM。您不必登录 CloudHSM CLI 即可运行此命令。

> **Note**
>
> 如果您添加或删除 HSM，请更新 AWS CloudHSM 客户端和命令行工具使用的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下类型的用户均可运行此命令。

- 所有用户。您不必登录即可运行此命令。

语法

```
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
```

示例

此命令列出了您的 AWS CloudHSM 集群中存在的 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]"
      }
    ]
  }
}
```

输出具有以下属性：

- 供应商：HSM 的供应商名称。
- 模型：HSM 的型号。

- 序列号 ：设备的序列号。情况可能因替换而变化。
- H ardware-version-major：主要硬件版本。
- H ardware-version-minor：次要硬件版本。
- F irmware-version-major：主要固件版本。
- F irmware-version-minor：次要固件版本。
- F irmware-build-number：固件版本号。
- Firmware-id：固件 ID，包括主要版本、次要版本和生成版本。
- FIPS 状态：集群及其中的 HSM 的 FIPS 模式。如果处于 FIPS 模式，则输出为 “2 [采用单因素身份验证的 FIPS 模式]”。如果在非 FIPS 模式下，则输出为 “0 [使用单因素身份验证的非 FIPS 模式]”。

相关 主题

- 集群激活

crypto

crypto是一组命令的父类别，当这些命令与父类别结合使用时，会创建专用于加密操作的命令。目前，此类别由以下命令组成：

- 加密标志
  - 加密标志 ecdsa
  - 加密标志 rsa-pkcs
  - 加密标志 rsa-pkcs-pss
- 加密验证
  - 加密验证 ecdsa
  - 加密验证 rsa-pkcs
  - 加密验证 rsa-pkcs-pss

加密标志

crypto sign是一组命令的父类别，当这些命令与父类别结合使用时，使用 AWS CloudHSM 集群中选定的私钥生成签名。 crypto sign有以下子命令：

- 加密标志 ecdsa

- 加密标志 rsa-pkcs
- 加密标志 rsa-pkcs-pss

要使用crypto sign，您的 HSM 中必须有私钥。您可以使用以下命令生成私钥：

- key generate-asymmetric-pair ec
- 密钥 generate-asymmetric-pair rsa

加密标志 ecdsa

该crypto sign ecdsa命令使用 EC 私钥和 ECDSA 签名机制生成签名。

要使用该crypto sign ecdsa命令，您的 AWS CloudHSM 集群中必须首先有一个 EC 私钥。您可以使用`sign`属性设置为的key generate-asymmetric-pair ec命令生成 EC 私钥`true`。

> **Note**
>
> 可以使用加密验证子命令验证签名。 AWS CloudHSM

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明crypto sign ecdsa如何使用 ECDSA 签名机制和SHA256哈希函数生成签名。此命令在 HSM 中使用私钥。

Example 示例：为 base 64 编码的数据生成签名

```
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 示例：为数据文件生成签名

```
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=="
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<DATA>

要签名的 Base64 编码数据。

必填项：是（除非通过数据路径提供）

<DATA_PATH>

指定要签名的数据的位置。

必填项：是（除非通过数据路径提供）

<HASH_FUNCTION>

指定哈希函数。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必需：是

<KEY_FILTER>

键引用（例如 key-reference=0xabc）或以 attr.key_attribute_name=key_attribute_Value 的形式以空格分隔的键属性列表，用于选择匹配的密钥。

有关支持的 CloudHSM CLI 密钥属性的列表，请参阅 CloudHSM CLI 的关键属性。

必需：是

相关 主题

- 加密标志
- 加密验证

加密标志 rsa-pkcs

该crypto sign rsa-pkcs命令使用 RSA 私钥和 RSA-PKCS 签名机制生成签名。

要使用该crypto sign rsa-pkcs命令，您的 AWS CloudHSM 集群中必须首先有一个 RSA 私钥。您可以使用`sign`属性设置为的密钥 generate-asymmetric-pair rsa命令生成 RSA 私钥。`true`

> **Note**
>
> 可以使用加密验证子命令验证签名。 AWS CloudHSM

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明crypto sign rsa-pkcs如何使用 RSA-PKCS 签名机制和哈希函数生成签名。SHA256此命令在 HSM 中使用私钥。

Example 示例：为 base 64 编码的数据生成签名

```
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 示例：为数据文件生成签名

```
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=="
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<DATA>

要签名的 Base64 编码数据。

必填项：是（除非通过数据路径提供）

<DATA_PATH>

指定要签名的数据的位置。

必填项：是（除非通过数据提供）

<HASH_FUNCTION>

指定哈希函数。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必需：是

<KEY_FILTER>

按键引用（例如`key-reference=0xabc`）或以空格分隔的按键属性列表，`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`以选择匹配的密钥。

有关支持的 CloudHSM CLI 密钥属性的列表，请参阅 CloudHSM CLI 的关键属性。

必需：是

相关 主题

- 加密标志
- 加密验证

加密标志 rsa-pkcs-pss

该crypto sign rsa-pkcs-pss命令使用 RSA 私钥和签名机制生成`RSA-PKCS-PSS`签名。

要使用该crypto sign rsa-pkcs-pss命令，您的 AWS CloudHSM 集群中必须首先有一个 RSA 私钥。您可以使用`sign`属性设置为的密钥 generate-asymmetric-pair rsa命令生成 RSA 私钥。`true`

> **Note**
>
> 可以使用加密验证子命令验证签名。 AWS CloudHSM

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明crypto sign rsa-pkcs-pss如何使用签名机制和SHA256哈希函数生成RSA-PKCS-PSS签名。此命令在 HSM 中使用私钥。

Example 示例：为 base 64 编码的数据生成签名

```
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 示例：为数据文件生成签名

```
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=="
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<DATA>

要签名的 Base64 编码数据。

必填项：是（除非通过数据路径提供）

<DATA_PATH>

指定要签名的数据的位置。

必填项：是（除非通过数据提供）

<HASH_FUNCTION>

指定哈希函数。

有效值：

- sha1
- sha224
- sha256

- sha384
- sha512

必需：是

<KEY_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以选择匹配的密钥。

有关支持的 CloudHSM CLI 密钥属性的列表，请参阅 CloudHSM CLI 的关键属性。

必需：是

<MGF>

指定蒙版生成函数。

> **Note**
>
> 掩码生成函数哈希函数必须与签名机制哈希函数相匹配。

有效值：

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

必需：是

<SALT_LENGTH>

指定盐的长度。

必需：是

相关 主题

- 加密标志
- 加密验证

相关 主题

- 加密验证

加密验证

crypto verify是一组命令的父类别，当这些命令与父类别结合使用时，可以确认文件是否已由给定密钥签名。 crypto verify有以下子命令：

- 加密验证 ecdsa
- 加密验证 rsa-pkcs
- 加密验证 rsa-pkcs-pss

该crypto verify命令将签名文件与源文件进行比较，并根据给定的公钥和签名机制分析它们在密码学上是否相关。

> **Note**
>
> 可以通过加密标志操作登录 AWS CloudHSM 文件。

加密验证 ecdsa

该crypto verify ecdsa命令用于完成以下操作：

- 确认文件已通过给定的公钥在 HSM 中签名。
- 验证签名是使用 ECDSA 签名机制生成的。
- 将签名文件与源文件进行比较，并根据给定的 ecdsa 公钥和签名机制确定两者在密码学上是否相关。

要使用该crypto verify ecdsa命令，您的 AWS CloudHSM 集群中必须首先有一个 EC 公钥。您可以使用`verify`属性设置为的密钥导入 pem命令导入 EC 公钥`true`。

> **Note**
>
> 您可以使用子命令在 CloudHSM CLI 加密标志 中生成签名。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明如何使用crypto verify ecdsa来验证使用 ECDSA 签名机制和SHA256哈希函数生成的签名。此命令使用 HSM 中的公钥。

Example 示例：使用 Base64 编码的数据验证采用 Base64 编码的签名

```
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 示例：使用数据文件验证签名文件

```
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 示例：证明虚假的签名关系

此命令验证位于/home/data的数据是否由带有标签的公钥签名，该公钥ecdsa-public使用 ECDSA 签名机制生成位于中的签名。/home/signature由于给定的参数不构成真正的签名关系，因此该命令会返回错误消息。

```
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"
```

```
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<DATA>

要签名的 Base64 编码数据。

必填项：是（除非通过数据路径提供）

<DATA_PATH>

指定要签名的数据的位置。

必填项：是（除非通过数据路径提供）

<HASH_FUNCTION>

指定哈希函数。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必需：是

<KEY_FILTER>

按键引用（例如`key-reference=0xabc`）或以空格分隔的按键属性列表，`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`以选择匹配的密钥。

有关支持的 CloudHSM CLI 密钥属性的列表，请参阅 CloudHSM CLI 的关键属性。

必需：是

<SIGNATURE>

Base64 编码的签名。

必填：是（除非通过签名路径提供）

<SIGNATURE_PATH>

指定签名的位置。

必填：是（除非通过签名路径提供）

相关 主题

- 加密标志
- 加密验证

加密验证 rsa-pkcs

该crypto verify rsa-pkcs命令用于完成以下操作：

- 确认文件已通过给定的公钥在 HSM 中签名。
- 验证签名是否使用RSA-PKCS签名机制生成。
- 将签名文件与源文件进行比较，并根据给定的 rsa 公钥和签名机制确定两者在密码学上是否相关。

要使用该crypto verify rsa-pkcs命令，您的 AWS CloudHSM 集群中必须首先有一个 RSA 公钥。

> **Note**
>
> 您可以使用 CloudHSM CLI 加密标志 和子命令生成签名。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明如何使用crypto verify rsa-pkcs来验证使用 RSA-PKCS 签名机制和哈希函数生成的签名。SHA256此命令在 HSM 中使用公钥。

Example 示例：使用 Base64 编码的数据验证采用 Base64 编码的签名

```
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 示例：使用数据文件验证签名文件

```
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 示例：证明虚假的签名关系

此命令验证无效数据是否由带有标签的公钥签名，rsa-public使用 RSAPKCS 签名机制生成位于中的签名。/home/signature由于给定的参数不构成真正的签名关系，因此该命令会返回错误消息。

```
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"
```

```
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<DATA>

要签名的 Base64 编码数据。

必填项：是（除非通过数据路径提供）

<DATA_PATH>

指定要签名的数据的位置。

必填项：是（除非通过数据路径提供）

<HASH_FUNCTION>

指定哈希函数。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必需：是

<KEY_FILTER>

按键引用（例如`key-reference=0xabc`）或以空格分隔的按键属性列表，`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`以选择匹配的密钥。

有关支持的 CloudHSM CLI 密钥属性的列表，请参阅 CloudHSM CLI 的关键属性。

必需：是

<SIGNATURE>

Base64 编码的签名。

必填项：是（除非通过签名路径提供）

<SIGNATURE_PATH>

指定签名的位置。

必填项：是（除非通过签名路径提供）

相关 主题

- 加密标志
- 加密验证

加密验证 rsa-pkcs-pss

该crypto sign rsa-pkcs-pss命令用于完成以下操作。

- 确认文件已通过给定的公钥在 HSM 中签名。
- 验证签名是使用 RSA-PKCS-PSS 签名机制生成的。
- 将签名文件与源文件进行比较，并根据给定的 rsa 公钥和签名机制确定两者在密码学上是否相关。

要使用该crypto verify rsa-pkcs-pss命令，您的 AWS CloudHSM 集群中必须首先有一个 RSA 公钥。您可以使用 key import pem 命令（在此处添加 UNWRAP LINK）导入 RSA 公钥，`verify`属性设置为。`true`

> **Note**
>
> 您可以使用 CloudHSM CLI 加密标志 和子命令生成签名。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明如何使用crypto verify rsa-pkcs-pss来验证使用 RSA-PKCS-PSS 签名机制和哈希函数生成的签名。SHA256此命令在 HSM 中使用公钥。

Example 示例：使用 Base64 编码的数据验证采用 Base64 编码的签名

```
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 示例：使用数据文件验证签名文件

```
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 示例：证明虚假的签名关系

此命令验证无效数据是否由带有标签的公钥签名，rsa-public使用 RSAPKCSPSS 签名机制生成位于中的签名。/home/signature由于给定的参数不构成真正的签名关系，因此该命令会返回错误消息。

```
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"
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<DATA>

要签名的 Base64 编码数据。

必填项：是（除非通过数据路径提供）

<DATA_PATH>

指定要签名的数据的位置。

必填项：是（除非通过数据路径提供）

<HASH_FUNCTION>

指定哈希函数。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必需：是

<KEY_FILTER>

按键引用（例如`key-reference=0xabc`）或以空格分隔的按键属性列表，`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`以选择匹配的密钥。

有关支持的 CloudHSM CLI 密钥属性的列表，请参阅 CloudHSM CLI 的关键属性。

必需：是

<MFG>

指定蒙版生成函数。

> **Note**
>
> 掩码生成函数哈希函数必须与签名机制哈希函数相匹配。

有效值：

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

必需：是

<SIGNATURE>

以 Base64 编码的签名。

必填项：是（除非通过签名路径提供）

<SIGNATURE_PATH>

指定签名的位置。

必填项：是（除非通过签名路径提供）

相关 主题

- 加密标志

- 加密验证

key

key 是一组命令的父类别，当其与父类别一起使用时，将创建特定于密钥的命令。目前，此类别由以下命令组成：

- key delete
- key generate-file
- 钥匙 generate-asymmetric-pair
  - 密钥 generate-asymmetric-pair rsa
  - key generate-asymmetric-pair ec
- key generate-symmetric
  - key generate-symmetric aes
  - key generate-symmetric generic-secret
- 密钥导入 pem
- key list
- 密钥复制
- key set-attribute
- key share
- key unshare
- 密钥解包
- 钥匙包装

key delete

使用 CloudHSM CLI 中的key delete命令从集群中删除密钥 AWS CloudHSM 。您一次只能删除一个密钥。删除某密钥对中的一个密钥不会影响该密钥对中的另一个密钥。

只有创建密钥并因此拥有密钥的 CU 才能删除该密钥。共享密钥但不拥有密钥的用户可以在加密操作中使用密钥，但无法将其删除。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

```
aws-cloudhsm > key delete --filter attr.label="ec-test-public-key"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如`key-reference=0xabc`）或以空格分隔的按键属性列表，`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`以选择要删除的匹配密钥。

有关支持的 CloudHSM CLI key 密钥属性列表，请参阅 CloudHSM CLI 的密钥属性。

必需：是

相关 主题

- key list
- key generate-file
- key unshare
- CloudHSM CLI 的密钥属性
- 使用 CloudHSM CLI 筛选密钥

key generate-file

该key generate-file命令从 HSM 导出非对称密钥。如果目标是私钥，则对私钥的引用将以虚假的 PEM 格式导出。如果目标是公钥，则公钥字节将以 PEM 格式导出。

虚假的 PEM 文件不包含实际的私钥材料，而是引用 HSM 中的私钥，可用于将 SSL/TLS 从您的 Web 服务器卸载到。 AWS CloudHSM有关更多信息，请参阅 SSL/TLS 分载。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

要运行此命令，必须以 CU 身份登录。

语法

```
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')
```

示例

此示例说明key generate-file如何使用在 AWS CloudHSM 集群中生成密钥文件。

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"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如`key-reference=0xabc`）或以空格分隔的按键属性列表，`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`以选择要删除的匹配密钥。

有关支持的 CloudHSM CLI key 密钥属性列表，请参阅 CloudHSM CLI 的密钥属性

必需：否

<ENCODING>

指定密钥文件的编码格式

必需：是

<PATH>

指定将写入密钥文件的文件路径

必需：是

相关 主题

- CloudHSM CLI 的密钥属性
- 使用 CloudHSM CLI 筛选密钥
- 钥匙 generate-asymmetric-pair
- key generate-symmetric

钥匙 generate-asymmetric-pair

key generate-asymmetric-pair 是一组命令的父类别，当这些命令与父类别结合使用时，会创建一个生成非对称密钥对的命令。目前，此类别由以下命令组成：

- key generate-asymmetric-pair ec

- 密钥 generate-asymmetric-pair rsa

key generate-asymmetric-pair ec

使用 CloudHSM CLI 中的key asymmetric-pair ec命令在集群中生成非对称椭圆曲线 (EC) 密钥对。AWS CloudHSM

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例演示如何使用 key generate-asymmetric-pair ec 命令创建 EC 密钥对。

Example 示例：创建 EC 密钥对

```
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 示例：使用可选属性创建 EC 密钥对

```
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"
      }
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<CURVE>

指定椭圆曲线的标识符。

- prime256v1
- secp256r1
- secp224r1

- secp384r1
- secp256k1
- secp521r1

必需：是

<PUBLIC_KEY_ATTRIBUTES>

指定一个以空格分隔的密钥属性列表，以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如 token=true）的形式为生成的 EC 公有密钥进行设置

有关支持的密钥属性的列表，请参阅 CloudHSM CLI 的密钥属性。

必需：否

<PUBLIC_LABEL>

为公有密钥指定用户定义的标签。客户端 SDK 5.11 及更高版本允许的最大大小label为 127 个字符。客户端 SDK 5.10 及更低版本的字符数限制为 126 个。

必需：是

<PRIVATE_KEY_ATTRIBUTES>

以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如 token=true）的形式指定要为生成的 EC 私有密钥设置的以空格分隔的密钥属性列表

有关支持的密钥属性的列表，请参阅 CloudHSM CLI 的密钥属性。

必需：否

<PRIVATE_LABEL>

为私有密钥指定用户定义的标签。客户端 SDK 5.11 及更高版本允许的最大大小label为 127 个字符。客户端 SDK 5.10 及更低版本的字符数限制为 126 个。

必需：是

<SESSION>

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。

如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

默认情况下，生成的密钥是永久（令牌）密钥。传入 <SESSION> 会改变这一点，确保使用此参数生成的密钥是会话（临时）密钥。

必需：否

相关 主题

- CloudHSM CLI 的密钥属性
- 使用 CloudHSM CLI 筛选密钥

密钥 generate-asymmetric-pair rsa

使用key generate-asymmetric-pair rsa命令在您的 AWS CloudHSM 集群中生成非对称 RSA key pair。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例演示如何使用 `key generate-asymmetric-pair rsa` 创建 RSA 密钥对。

Example 示例：创建 RSA 密钥对

```
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 示例：使用可选属性创建 RSA 密钥对

```
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
      }
```

```
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<MODULUS_SIZE_BITS>

指定模数的长度 (以位为单位)。最小值为 2048。

必需：是

<PRIVATE_KEY_ATTRIBUTES>

以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如 `token=true`）的形式指定要为生成的 RSA 私有密钥设置的以空格分隔的密钥属性列表

有关支持的密钥属性的列表，请参阅 CloudHSM CLI 的密钥属性。

必需：否

<PRIVATE_LABEL>

为私有密钥指定用户定义的标签。客户端 SDK 5.11 及更高版本允许的最大大小`label`为 127 个字符。客户端 SDK 5.10 及更低版本的字符数限制为 126 个。

必需：是

<PUBLIC_EXPONENT>

指定公有指数。此值必须为大于或等于 65537 的奇数。

必需：是

<PUBLIC_KEY_ATTRIBUTES>

以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如 `token=true`）的形式指定要为生成的 RSA 公有密钥设置的以空格分隔的密钥属性列表。

有关支持的密钥属性的列表，请参阅 CloudHSM CLI 的密钥属性。

必需：否

<PUBLIC_LABEL>

为公有密钥指定用户定义的标签。客户端 SDK 5.11 及更高版本允许的最大大小`label`为 127 个字符。客户端 SDK 5.10 及更低版本的字符数限制为 126 个。

必需：是

<SESSION>

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。

如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

默认情况下，生成的密钥是永久（令牌）密钥。传入 <SESSION> 会改变这一点，确保使用此参数生成的密钥是会话（临时）密钥。

必需：否

相关 主题

- CloudHSM CLI 的密钥属性
- 使用 CloudHSM CLI 筛选密钥

key generate-symmetric

key generate-symmetric 是一组命令的父类别，当这些命令与父类别结合使用时，会创建一个生成对称密钥的命令。目前，此类别由以下命令组成：

- key generate-symmetric aes
- key generate-symmetric generic-secret

key generate-symmetric aes

该key generate-symmetric aes命令会在您的 AWS CloudHSM 集群中生成对称 AES 密钥。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

以下示例显示了如何使用 key generate-symmetric aes 命令创建 AES 密钥。

Example 示例：创建 AES 密钥

```
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 示例：使用可选属性创建 AES 密钥对

```
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
      }
```

```
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<KEY_ATTRIBUTES>

以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如 `token=true`）的形式指定要为生成的 AES 密钥设置的按空格分隔的密钥属性列表。

有关支持的密钥属性的列表，请参阅 CloudHSM CLI 的密钥属性。

必需：否

<KEY-LENGTH-BYTES>

指定以字节为单位的密钥长度。

有效值：

- 16、24 和 32

必需：是

<LABEL>

为 AES 密钥指定用户定义的标签。客户端 SDK 5.11 及更高版本允许的最大大小`label`为 127 个字符。客户端 SDK 5.10 及更低版本的字符数限制为 126 个。

必需：是

<SESSION>

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。

如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

默认情况下，生成的密钥是永久（令牌）密钥。传入 <SESSION> 会改变这一点，确保使用此参数生成的密钥是会话（临时）密钥。

必需：否

相关 主题

- CloudHSM CLI 的密钥属性
- 使用 CloudHSM CLI 筛选密钥

key generate-symmetric generic-secret

key generate-asymmetric-pair 命令将在您的 AWS CloudHSM 集群中生成对称通用密钥。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

以下示例显示了如何使用 key generate-symmetric generic-secret 命令创建通用机密密钥。

Example 示例：创建通用密钥

```
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 示例：使用可选属性创建通用机密密钥

```
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
      }
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<KEY_ATTRIBUTES>

以 KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如 token=true）的形式指定要为生成的 AES 密钥设置的按空格分隔的密钥属性列表。

有关支持的密钥属性的列表，请参阅 CloudHSM CLI 的密钥属性。

必需：否

<KEY-LENGTH-BYTES>

指定以字节为单位的密钥长度。

有效值：

- 1 至 800

必需：是

<LABEL>

为通用密钥指定用户定义的标签。客户端 SDK 5.11 及更高版本允许的最大大小`label`为 127 个字符。客户端 SDK 5.10 及更低版本的字符数限制为 126 个。

必需：是

<SESSION>

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。

如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

默认情况下，生成的密钥是永久（令牌）密钥。传入 <SESSION> 会改变这一点，确保使用此参数生成的密钥是会话（临时）密钥。

必需：否

相关 主题

- CloudHSM CLI 的密钥属性
- 使用 CloudHSM CLI 筛选密钥

密钥导入 pem

中的key import pem命令将 PEM 格式的密钥 AWS CloudHSM 导入 HSM。您可以使用它来导入在 HSM 外生成的公有密钥。

> **Note**
>
> 使用key generate-file命令从公钥创建标准 PEM 文件或使用私钥创建引用 PEM 文件。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

以下示例说明如何使用key import pem命令从 PEM 格式的文件中导入 RSA 公钥。

Example 示例：导入 RSA 公钥

```
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 示例：导入带有可选属性的 RSA 公钥

```
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"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<PATH>

指定密钥文件所在的文件路径。

必需：是

<LABEL>

为导入的密钥指定用户定义的标签。label 的最大大小为 126 个字符。

必需：是

<KEY_TYPE_CLASS>

封装密钥的密钥类型和类别。

可能的值：

- ec-public
- rsa-public

必需：是

<IMPORT_KEY_ATTRIBUTES>

以KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE（例如token=true）的形式指定要为导入的密钥设置的按空格分隔的按键属性列表。有关支持的密钥属性的列表，请参阅 CloudHSM CLI 的密钥属性。

必需：否

相关 主题

- 加密标志
- 加密验证

key list

该key list命令会查找 AWS CloudHSM 集群中当前用户的所有密钥。输出包含用户拥有和共享的密钥，以及 CloudHSM 集群中的所有公有密钥。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

语法

```
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
```

示例

以下示例显示运行 key list 命令的不同方式。

Example 示例：Find all keys - default

此命令列出了 AWS CloudHSM 集群中已登录用户的密钥。

> **Note**
>
> 默认情况下，仅显示当前登录用户的 10 个按键，并且输出中仅显示 key-reference 和 label。使用适当的分页选项将更多或更少的密钥显示为输出。

```
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 示例：Find all keys - verbose

输出包含用户拥有和共享的密钥，以及 HSM 中的所有公有密钥。

> **Note**
>
> 注意：默认情况下，仅显示当前登录用户的 10 个密钥。使用适当的分页选项将更多或更少的密钥显示为输出。

```
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 示例：Paginated return

以下示例显示密钥的分页子集，其中仅显示两个密钥。然后，该示例提供了一个后续调用以显示接下来的两个密钥。

```
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"
  }
}
```

要显示接下来的 2 个密钥，可以进行后续调用：

```
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"
  }
}
```


有关演示 CloudHSM CLI 中密钥过滤机制的工作原理的更多示例，请参阅 使用 CloudHSM CLI 筛选密钥。

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如`key-reference=0xabc`）或以空格分隔的按键属性列表`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`，以选择要列出的匹配密钥。

有关支持的 CloudHSM CLI key 密钥属性列表，请参阅 CloudHSM CLI 的密钥属性

必需：否

<MAX_ITEMS>

命令的输出中要返回的项目总数。如果可用的总项目数超过指定的值，则会在命令的输出中提供 next-token。要恢复分页，请在后续命令的 starting-token 参数中提供 next-token 值。

必需：否

<STARTING_TOKEN>

指定从何处开始分页的令牌。这是先前截断的响应中的 next-token。

必需：否

<VERBOSE>

如果包含，则打印每个匹配密钥的所有属性和密钥信息。默认情况下，每个匹配的密钥仅显示其键引用和标签属性。

必需：否

相关 主题

- key delete
- key generate-file
- key unshare

- CloudHSM CLI 的密钥属性
- 使用 CloudHSM CLI 筛选密钥

密钥复制

该key replicate命令将密钥从源集 AWS CloudHSM 群复制到目标集 AWS CloudHSM 群。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

> **Note**
>
> 加密用户必须拥有密钥才能使用此命令。

要求

- 源集群和目标集群必须是克隆。这意味着一个是从另一个的备份中创建的，或者它们都是从公共备份中创建的。请参阅使用备份创建集群了解更多信息。
- 密钥的所有者必须存在于目标集群上。此外，如果与任何用户共享密钥，则这些用户也必须存在于目标集群中。
- 要运行此命令，您必须以 CU 身份登录到源集群和目标集群。
  - 在单命令模式下，该命令将使用 CLOUDHSM_PIN 和 CLOUDHSM_ROLE 环境变量在源集群上进行身份验证。请参阅单命令模式了解更多信息。要为目标集群提供凭证，您需要另外设置两个环境变量：DESTINATION_CLOUDHSM_PIN 和 DESTINATION_CLOUDHSM_ROLE：

```
$ export DESTINATION_CLOUDHSM_ROLE=crypto-user
```

```
$ export DESTINATION_CLOUDHSM_PIN=username:password
```

  - 在交互模式下，用户需要明确登录到源集群和目标集群。

语法

```
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
```

示例

Example 示例：复制密钥

此命令将密钥从源集群复制到克隆的目标集群。

```
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"
  }
}
```

参数

<FILTER>

密钥引用（例如key-reference=0xabc）或以空格分隔的键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于在源集群上选择匹配的密钥。

有关支持的 CloudHSM CLI key 密钥属性列表，请参阅 CloudHSM CLI 的密钥属性

必需：是

<SOURCE_CLUSTER_ID>

源集群 ID。

必需：是

<DESTINATION_CLUSTER_ID>

目标集群 ID。

必需：是

相关 主题

- 使用 CloudHSM CLI 连接到多个集群

key set-attribute

使用key set-attribute命令设置集 AWS CloudHSM 群中密钥的属性。只有创建密钥、并因此拥有密钥的 CU 才能更改密钥属性。

有关可在 CloudHSM CLI 中使用的密钥属性列表，请参阅 CloudHSM CLI 的密钥属性。

用户类型

以下类型的用户均可运行此命令。

- 加密用户 (CU) 可运行此命令。
- 管理员可以设置信任属性。

要求

要运行此命令，必须以 CU 身份登录。若要设置信任属性，您必须以管理员身份登录。

语法

```
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
```

示例：设置密钥属性

以下示例将介绍了如何使用 key set-attribute 命令设置标签。

Example

1. 使用带有 my_key 标签的密钥，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 命令确认 encrypt 属性已更改：

```
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
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<KEY_ATTRIBUTE>

指定了键属性名称。

必需：是

<FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以选择要删除的匹配密钥。

有关支持的 CloudHSM CLI key 密钥属性列表，请参阅 CloudHSM CLI 的密钥属性

必需：否

<KEY_ATTRIBUTE_VALUE>

指定密钥属性值。

必需：是

<KEY_REFERENCE>

密钥的十六进制或十进制表现形式（例如密钥处理程序）。

必需：否

相关 主题

- 使用 CloudHSM CLI 筛选密钥
- CloudHSM CLI 的密钥属性

key share

该key share命令与 AWS CloudHSM 集群中的其他 CU 共享密钥。

只有创建密钥、并因此拥有密钥的 CU 才能共享密钥。共享密钥的用户可以在加密操作中使用该密钥，但不能删除、导出、共享或取消共享该密钥。此外，这些用户无法更改密钥属性。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

要运行此命令，必须以 CU 身份登录。

语法

```
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')
```

示例：与另一个 CU 共享密钥

以下示例演示了如何使用 key share 命令与 CU `alice` 共享密钥。

Example

1. 运行 key share 命令与 `alice` 共享密钥。

```
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. 运行 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. 在上面的列表中，verify `alice` 在 `shared-users` 列表中

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如`key-reference=0xabc`）或以空格分隔的按键属性列表，`attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`以选择要删除的匹配密钥。

有关支持的密钥属性的列表，请参阅 CloudHSM CLI 的密钥属性。

必需：是

<USERNAME>

为用户指定友好名称。最大长度为 31 个字符。唯一允许的特殊字符是下划线 (_)。此命令中的用户名不区分大小写，用户名始终以小写形式显示。

必需：是

<ROLE>

指定分配给该用户的角色。此参数为必需参数。要获取用户的角色，请使用 user list 命令。有关 HSM 上的用户类型的详细信息，请参阅 了解 HSM 用户。

必需：是

相关 主题

- 使用 CloudHSM CLI 筛选密钥
- CloudHSM CLI 的密钥属性

key unshare

该key unshare命令取消与 AWS CloudHSM 集群中的其他 CU 共享密钥。

只有创建密钥并因此拥有该密钥的 CU 才能取消共享该密钥。共享密钥的用户可以在加密操作中使用该密钥，但不能删除、导出、共享或取消共享该密钥。此外，这些用户无法更改密钥属性。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

要运行此命令，必须以 CU 身份登录。

语法

```
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')
```

示例：取消与其他 CU 共享密钥

以下示例演示了如何使用 key unshare 命令取消与 CU `alice` 共享密钥。

Example

1. 运行 key list 命令并按要取消与 `alice` 共享的特定密钥筛选。

```
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. 在 shared-users 输出中确认 alice，然后运行以下 key unshare 命令取消与 alice 共享密钥。

```
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. 再次运行 key list 命令以确认已取消与 alice 共享该密钥。

```
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
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以选择要删除的匹配密钥。

有关支持的密钥属性的列表，请参阅 CloudHSM CLI 的密钥属性。

必需：是

<USERNAME>

为用户指定友好名称。最大长度为 31 个字符。唯一允许的特殊字符是下划线 (_)。此命令中的用户名不区分大小写，用户名始终以小写形式显示。

必需：是

<ROLE>

指定分配给该用户的角色。此参数为必需参数。要获取用户的角色，请使用 user list 命令。有关 HSM 上的用户类型的详细信息，请参阅 了解 HSM 用户。

必需：是

相关 主题

- 使用 CloudHSM CLI 筛选密钥
- CloudHSM CLI 的密钥属性

密钥解包

CloudHSM CLI 中的key unwrap父命令将加密（封装）的对称或非对称私钥从文件导入 HSM。此命令旨在导入由该钥匙包装命令封装的加密密钥，但它也可以用来解开用其他工具封装的密钥。不过，在这些情况下，我们建议使用 PKCS#11 或 JCE 软件库来解开包装密钥。

- key unwrap aes-gcm
- 密钥解包 aes-no-pad
- key unwrap aes-pkcs5-pad
- 密钥解包 aes-zero-pad
- 密钥解包 cloudhsm-aes-gcm
- 密钥解包 rsa-aes
- key unwrap rsa-oaep
- key unwrap rsa-pkcs

key unwrap aes-gcm

该key unwrap aes-gcm命令使用 AES 封装密钥和解包机制将有效载荷密钥AES-GCM解封到集群中。

未封装的密钥的使用方式与生成的 AWS CloudHSM密钥相同。为了表示它们不是在本地生成的，它们的`local`属性设置为`false`。

要使用该key unwrap aes-gcm命令，您的 AWS CloudHSM 集群中必须有 AES 包装密钥，并且其`unwrap`属性必须设置为`true`。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明如何使用unwrap属性值设置为 AES 密钥的key unwrap aes-gcm命令true。

Example 示例：从 Base64 编码的封装密钥数据中解开有效载荷密钥

```
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 示例：解开通过数据路径提供的有效载荷密钥

```
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
      }
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择要展开的按键。

必需：是

<DATA_PATH>

包含封装密钥数据的二进制文件的路径。

必填项：是（除非通过 Base64 编码的数据提供）

<DATA>

Base64 编码的封装密钥数据。

必填项：是（除非通过数据路径提供）

<ATTRIBUTES>

以空格分隔的密钥属性列表，形式KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为封装后的密钥。

必需：否

<AAD>

Aes GCM 其他身份验证数据 (AAD) 值，以十六进制表示。

必需：否

<TAG_LENGTH_BITS>

Aes GCM 标签长度（以位为单位）。

必需：是

<KEY_TYPE_CLASS>

封装密钥的密钥类型和类别 [可能的值：aesdes3、ec-private、generic-secret、、rsa-private]。

必需：是

<LABEL>

未包装的密钥的标签。

必需：是

<SESSION>

创建仅存在于当前会话中的会话密钥。会话结束后，密钥无法恢复。

必需：否

<IV>

用于封装密钥的初始值，以十六进制表示。

必需：否

相关 主题

- 钥匙包装

- 密钥解包

密钥解包 aes-no-pad

该key unwrap aes-no-pad命令使用 AES 封装密钥和解包机制将有效载荷密钥AES-NO-PAD解封到集群中。

未包装的密钥的使用方式与生成的 AWS CloudHSM密钥相同。为了表示它们不是在本地生成的，它们的local属性设置为false。

要使用该key unwrap aes-no-pad命令，您的 AWS CloudHSM 集群中必须有 AES 包装密钥，并且其unwrap属性必须设置为true。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明如何使用unwrap属性值设置为 AES 密钥的key unwrap aes-no-pad命令true。

Example 示例：从 Base64 编码的封装密钥数据中解开有效载荷密钥

```
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 示例：解开通过数据路径提供的有效载荷密钥

```
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
      }
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择要展开的按键。

必需：是

<DATA_PATH>

包含封装密钥数据的二进制文件的路径。

必填项：是（除非通过 Base64 编码的数据提供）

<DATA>

Base64 编码的封装密钥数据。

必填项：是（除非通过数据路径提供）

<ATTRIBUTES>

以空格分隔的密钥属性列表，形式KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为封装后的密钥。

必需：否

<KEY_TYPE_CLASS>

封装密钥的密钥类型和类别 [可能的值：aesdes3、ec-private、generic-secret、、rsa-private]。

必需：是

<LABEL>

未包装的密钥的标签。

必需：是

<SESSION>

创建仅存在于当前会话中的会话密钥。会话结束后，密钥无法恢复。

必需：否

相关 主题

- 钥匙包装
- 密钥解包

key unwrap aes-pkcs5-pad

该key unwrap aes-pkcs5-pad命令使用 AES 封装密钥和解包机制AES-PKCS5-PAD解开有效载荷密钥。

未封装的密钥的使用方式与生成的 AWS CloudHSM密钥相同。为了表示它们不是在本地生成的，它们的local属性设置为false。

要使用该key unwrap aes-pkcs5-pad命令，您的 AWS CloudHSM 集群中必须有 AES 包装密钥，并且其unwrap属性必须设置为true。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明如何使用unwrap属性值设置为 AES 密钥的key unwrap aes-pkcs5-pad命令true。

Example 示例：从 Base64 编码的封装密钥数据中解开有效载荷密钥

```
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 示例：解开通过数据路径提供的有效载荷密钥

```
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
      }
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择要展开的按键。

必需：是

<DATA_PATH>

包含封装密钥数据的二进制文件的路径。

必填项：是（除非通过 Base64 编码的数据提供）

<DATA>

Base64 编码的封装密钥数据。

必填项：是（除非通过数据路径提供）

<ATTRIBUTES>

以空格分隔的密钥属性列表，形式KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为封装后的密钥。

必需：否

<KEY_TYPE_CLASS>

封装密钥的密钥类型和类别 [可能的值：aesdes3、ec-private、generic-secret、、rsa-private]。

必需：是

<LABEL>

未包装的密钥的标签。

必需：是

<SESSION>

创建仅存在于当前会话中的会话密钥。会话结束后，密钥无法恢复。

必需：否

相关 主题

- 钥匙包装
- 密钥解包

密钥解包 aes-zero-pad

该key unwrap aes-zero-pad命令使用 AES 封装密钥和解包机制将有效载荷密钥AES-ZERO-PAD解封到集群中。

未包装的密钥的使用方式与生成的 AWS CloudHSM密钥相同。为了表示它们不是在本地生成的，它们的local属性设置为false。

要使用该key unwrap aes-no-pad命令，您的 AWS CloudHSM 集群中必须有 AES 包装密钥，并且其unwrap属性必须设置为true。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明如何使用unwrap属性值设置为 AES 密钥的key unwrap aes-zero-pad命令true。

Example 示例：从 Base64 编码的封装密钥数据中解开有效载荷密钥

```
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 示例：解开通过数据路径提供的有效载荷密钥

```
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
      }
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择要展开的按键。

必需：是

<DATA_PATH>

包含封装密钥数据的二进制文件的路径。

必填项：是（除非通过 Base64 编码的数据提供）

<DATA>

Base64 编码的封装密钥数据。

必填项：是（除非通过数据路径提供）

<ATTRIBUTES>

以空格分隔的密钥属性列表，形式KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为封装后的密钥。

必需：否

<KEY_TYPE_CLASS>

封装密钥的密钥类型和类别 [可能的值：aesdes3、ec-private、generic-secret、、rsa-private]。

必需：是

<LABEL>

未包装的密钥的标签。

必需：是

<SESSION>

创建仅存在于当前会话中的会话密钥。会话结束后，密钥无法恢复。

必需：否

相关 主题

- 钥匙包装
- 密钥解包

密钥解包 cloudhsm-aes-gcm

该key unwrap cloudhsm-aes-gcm命令使用 AES 封装密钥和解包机制将有效载荷密钥CLOUDHSM-AES-GCM解封到集群中。

未封装的密钥的使用方式与生成的 AWS CloudHSM密钥相同。为了表示它们不是在本地生成的，它们的local属性设置为false。

要使用该key unwrap cloudhsm-aes-gcm命令，您的 AWS CloudHSM 集群中必须有 AES 封装密钥，并且其unwrap属性必须设置为true。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明如何使用unwrap属性值设置为 AES 密钥的key unwrap cloudhsm-aes-gcm命令true。

Example 示例：从 Base64 编码的封装密钥数据中解开有效载荷密钥

```
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 示例：解开通过数据路径提供的有效载荷密钥

```
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
      }
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择要展开的按键。

必需：是

<DATA_PATH>

包含封装密钥数据的二进制文件的路径。

必填项：是（除非通过 Base64 编码的数据提供）

<DATA>

Base64 编码的封装密钥数据。

必填项：是（除非通过数据路径提供）

<ATTRIBUTES>

以空格分隔的密钥属性列表，形式KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为封装后的密钥。

必需：否

<AAD>

Aes GCM 其他身份验证数据 (AAD) 值，以十六进制表示。

必需：否

<TAG_LENGTH_BITS>

Aes GCM 标签长度（以位为单位）。

必需：是

<KEY_TYPE_CLASS>

封装密钥的密钥类型和类别 [可能的值：aesdes3、ec-private、generic-secret、、rsa-private]。

必需：是

<LABEL>

未包装的密钥的标签。

必需：是

<SESSION>

创建仅存在于当前会话中的会话密钥。会话结束后，密钥无法恢复。

必需：否

相关 主题

- 钥匙包装
- 密钥解包

密钥解包 rsa-aes

该key unwrap rsa-aes命令使用 RSA 私钥和解包机制解封有效载荷密钥。RSA-AES

未封装的密钥的使用方式与生成的 AWS CloudHSM密钥相同。为了表示它们不是在本地生成的，它们的`local`属性设置为`false`。

要使用key unwrap rsa-aes，您的 AWS CloudHSM 集群中必须有 RSA 公用包装密钥的 RSA 私钥，并且其`unwrap`属性必须设置为。`true`

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明如何使用unwrap属性值设置为 RSA 私钥的key unwrap rsa-aes命令。`true`

Example 示例：从 Base64 编码的封装密钥数据中解开有效载荷密钥

```
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 示例：解开通过数据路径提供的有效载荷密钥

```
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
      }
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择要展开的按键。

必需：是

<DATA_PATH>

包含封装密钥数据的二进制文件的路径。

必填项：是（除非通过 Base64 编码的数据提供）

<DATA>

Base64 编码的封装密钥数据。

必填项：是（除非通过数据路径提供）

<ATTRIBUTES>

以空格分隔的密钥属性列表，形式KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为封装后的密钥。

必需：否

<KEY_TYPE_CLASS>

封装密钥的密钥类型和类别 [可能的值：aesdes3、ec-private、generic-secret、、rsa-private]。

必需：是

<HASH_FUNCTION>

指定哈希函数。

有效值：

- sha1
- sha224
- sha256
- sha384
- sha512

必需：是

<MGF>

指定蒙版生成函数。

> **Note**
>
> 掩码生成函数哈希函数必须与签名机制哈希函数相匹配。

有效值：

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

必需：是

<LABEL>

未包装的密钥的标签。

必需：是

<SESSION>

创建仅存在于当前会话中的会话密钥。会话结束后，密钥无法恢复。

必需：否

相关 主题

- 钥匙包装
- 密钥解包

key unwrap rsa-oaep

该key unwrap rsa-oaep命令使用 RSA 私钥和解包机制解封有效载荷密钥。RSA-OAEP

未封装的密钥的使用方式与生成的 AWS CloudHSM密钥相同。为了表示它们不是在本地生成的，它们的local属性设置为false。

要使用该key unwrap rsa-oaep命令，您的 AWS CloudHSM 集群中必须有 RSA 公用包装密钥的 RSA 私钥，并且其unwrap属性必须设置为。true

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明如何使用unwrap属性值设置为 RSA 私钥的key unwrap rsa-oaep命令。`true`

Example 示例：从 Base64 编码的封装密钥数据中解开有效载荷密钥

```
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 示例：解开通过数据路径提供的有效载荷密钥

```
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
      }
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择要展开的按键。

必需：是

<DATA_PATH>

包含封装密钥数据的二进制文件的路径。

必填项：是（除非通过 Base64 编码的数据提供）

<DATA>

Base64 编码的封装密钥数据。

必填项：是（除非通过数据路径提供）

<ATTRIBUTES>

以空格分隔的密钥属性列表，形式KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为封装后的密钥。

必需：否

<KEY_TYPE_CLASS>

封装密钥的密钥类型和类别 [可能的值：aesdes3、ec-private、generic-secret、、rsa-private]。

必需：是

<HASH_FUNCTION>

指定哈希函数。

有效值：

- sha1

- sha224
- sha256
- sha384
- sha512

必需：是

<MGF>

指定蒙版生成函数。

> **Note**
>
> 掩码生成函数哈希函数必须与签名机制哈希函数相匹配。

有效值：

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

必需：是

<LABEL>

未包装的密钥的标签。

必需：是

<SESSION>

创建仅存在于当前会话中的会话密钥。会话结束后，密钥无法恢复。

必需：否

相关 主题

- 钥匙包装

- 密钥解包

key unwrap rsa-pkcs

该key unwrap rsa-pkcs命令使用 RSA 私钥和解包机制解封有效载荷密钥。RSA-PKCS

未封装的密钥的使用方式与生成的 AWS CloudHSM密钥相同。为了表示它们不是在本地生成的，它们的local属性设置为false。

要使用 key unwrap rsa-pkcs 命令，您的 AWS CloudHSM 集群中必须有 RSA 公用包装密钥的 RSA 私钥，并且其unwrap属性必须设置为。true

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

这些示例说明如何使用unwrap属性值设置为 AES 密钥的key unwrap rsa-oaep命令true。

Example 示例：从 Base64 编码的封装密钥数据中解开有效载荷密钥

```
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 示例：解开通过数据路径提供的有效载荷密钥

```
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
      }
    }
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择要展开的按键。

必需：是

<DATA_PATH>

包含封装密钥数据的二进制文件的路径。

必填项：是（除非通过 Base64 编码的数据提供）

<DATA>

Base64 编码的封装密钥数据。

必填项：是（除非通过数据路径提供）

<ATTRIBUTES>

以空格分隔的密钥属性列表，形式KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为封装后的密钥。

必需：否

<KEY_TYPE_CLASS>

封装密钥的密钥类型和类别 [可能的值：aesdes3、ec-private、generic-secret、、rsa-private]。

必需：是

<LABEL>

未包装的密钥的标签。

必需：是

<SESSION>

创建仅存在于当前会话中的会话密钥。会话结束后，密钥无法恢复。

必需：否

相关 主题

- 钥匙包装

- 密钥解包

钥匙包装

CloudHSM CLI 中的key wrap命令将对称或非对称私钥的加密副本从 HSM 导出到文件中。运行时key wrap，需要指定两项内容：要导出的密钥和输出文件。要导出的密钥是 HSM 上的密钥，它将对要导出的密钥进行加密（封装）。

该key wrap命令不会从 HSM 中删除密钥，也不会阻止您在加密操作中使用该密钥。您可以多次导出相同的密钥。要将加密密钥导回 HSM，请使用密钥解包。只有密钥的所有者，即创建密钥的加密用户 (CU)，才能封装密钥。与之共享密钥的用户只能在加密操作中使用该密钥。

该key wrap命令由以下子命令组成：

- 钥匙包装 aes-gcm
- 钥匙包装 aes-no-pad
- 钥匙包裹 aes-pkcs5-pad
- 钥匙包装 aes-zero-pad
- 钥匙包装 cloudhsm-aes-gcm
- 密钥包装 rsa-aes
- 密钥包装 rsa-oaep
- 密钥包装 rsa-pkcs

钥匙包装 aes-gcm

该key wrap aes-gcm命令使用 HSM 上的 AES 密钥和AES-GCM包装机制封装有效载荷密钥。有效载荷密钥的`extractable`属性必须设置为`true`。

只有密钥的所有者，即创建密钥的加密用户 (CU)，才能封装密钥。共享密钥的用户可以在加密操作中使用该密钥。

要使用该key wrap aes-gcm命令，您必须先在 AWS CloudHSM 集群中拥有 AES 密钥。您可以生成 AES 密钥进行封装，key generate-symmetric aes命令和`wrap`属性设置为`true`。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

此示例说明如何使用 AES 密钥使用该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"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<PAYLOAD_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择有效载荷密钥。

必需：是

<PATH>

保存封装密钥数据的二进制文件路径。

必需：否

<WRAPPING_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择换行密钥。

必需：是

<AAD>

AES GCM 其他身份验证数据 (AAD) 值，以十六进制表示。

必需：否

<TAG_LENGTH_BITS>

AES GCM 标签长度（以位为单位）。

必需：是

相关 主题

- 钥匙包装
- 密钥解包

钥匙包装 aes-no-pad

该key wrap aes-no-pad命令使用 HSM 上的 AES 密钥和AES-NO-PAD包装机制封装有效载荷密钥。有效载荷密钥的`extractable`属性必须设置为`true`。

只有密钥的所有者，即创建密钥的加密用户 (CU)，才能封装密钥。共享密钥的用户可以在加密操作中使用该密钥。

要使用该key wrap aes-no-pad命令，您必须先在 AWS CloudHSM 集群中拥有 AES 密钥。您可以使用key generate-symmetric aes命令和设置为的`wrap`属性生成 AES 密钥进行封装`true`。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

此示例说明如何使用wrap属性值设置为 AES 密钥的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"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<PAYLOAD_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择有效载荷密钥。

必需：是

<PATH>

保存封装密钥数据的二进制文件路径。

必需：否

<WRAPPING_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以选择换行密钥。

必需：是

相关 主题

- 钥匙包装
- 密钥解包

钥匙包裹 aes-pkcs5-pad

该key wrap aes-pkcs5-pad命令使用 HSM 上的 AES 密钥和AES-PKCS5-PAD包装机制封装有效载荷密钥。有效载荷密钥的extractable属性必须设置为true。

只有密钥的所有者，即创建密钥的加密用户 (CU)，才能封装密钥。共享密钥的用户可以在加密操作中使用该密钥。

要使用该key wrap aes-pkcs5-pad命令，您必须先在 AWS CloudHSM 集群中拥有 AES 密钥。您可以使用key generate-symmetric aes命令和设置为的wrap属性生成 AES 密钥进行封装true。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

此示例说明如何使用`wrap`属性值设置为 AES 密钥的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="
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<PAYLOAD_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择有效载荷密钥。

必需：是

<PATH>

保存封装密钥数据的二进制文件路径。

必需：否

<WRAPPING_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择换行密钥。

必需：是

相关 主题

- 钥匙包装
- 密钥解包

钥匙包装 aes-zero-pad

该key wrap aes-zero-pad命令使用 HSM 上的 AES 密钥和AES-ZERO-PAD包装机制封装有效载荷密钥。有效载荷密钥的extractable属性必须设置为true。

只有密钥的所有者，即创建密钥的加密用户 (CU)，才能封装密钥。共享密钥的用户可以在加密操作中使用该密钥。

要使用该key wrap aes-zero-pad命令，您必须先在 AWS CloudHSM 集群中拥有 AES 密钥。您可以使用wrap属性设置为的key generate-symmetric aes命令生成 AES 密钥进行封装true。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

此示例说明如何使用`wrap`属性值设置为 AES 密钥的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"
```

```
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<PAYLOAD_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择有效载荷密钥。

必需：是

<PATH>

保存封装密钥数据的二进制文件路径。

必需：否

<WRAPPING_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以选择换行密钥。

必需：是

相关 主题

- 钥匙包装
- 密钥解包

钥匙包装 cloudhsm-aes-gcm

该key wrap cloudhsm-aes-gcm命令使用 HSM 上的 AES 密钥和CLOUDHSM-AES-GCM包装机制封装有效载荷密钥。有效载荷密钥的extractable属性必须设置为true。

只有密钥的所有者，即创建密钥的加密用户 (CU)，才能封装密钥。共享密钥的用户可以在加密操作中使用该密钥。

要使用该key wrap cloudhsm-aes-gcm命令，您必须先在 AWS CloudHSM 集群中拥有 AES 密钥。您可以生成 AES 密钥进行封装，key generate-symmetric aes命令和wrap属性设置为true。

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

此示例说明如何使用 AES 密钥使用该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"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<PAYLOAD_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择有效载荷密钥。

必需：是

<PATH>

保存封装密钥数据的二进制文件路径。

必需：否

<WRAPPING_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择换行密钥。

必需：是

<AAD>

AES GCM 其他身份验证数据 (AAD) 值，以十六进制表示。

必需：否

<TAG_LENGTH_BITS>

AES GCM 标签长度（以位为单位）。

必需：是

相关 主题

- 钥匙包装
- 密钥解包

密钥包装 rsa-aes

该key wrap rsa-aes命令使用 HSM 上的 RSA 公钥和 RSA-AES 包装机制封装有效载荷密钥。有效载荷密钥的`extractable`属性必须设置为`true`。

只有密钥的所有者，即创建密钥的加密用户 (CU)，才能封装密钥。共享密钥的用户可以在加密操作中使用该密钥。

要使用该key wrap rsa-aes命令，您必须先在 AWS CloudHSM 集群中拥有 RSA 密钥。您可以使用钥匙 generate-asymmetric-pair 命令和设置为的`wrap`属性生成 RSA 密钥对。`true`

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

此示例说明如何使用`wrap`属性值设置为 RSA 公钥的key wrap rsa-ae命令。`true`

Example

```
aws-cloudhsm > key wrap rsa-aes --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example --hash-function sha256 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "payload-key-reference": "0x00000000001c08f1",
    "wrapping-key-reference": "0x00000000007008da",
    "wrapped-key-data": "HrSE1DEyLjIeyGdPa9R+ebiqB5TIJGyamPker31ZebPwRA
+NcerbAJO8DJ1lXPygZcI21vIFSZJuWMEiWpe1R9D/5WSYgxLVKex30xCFqebtEzxbKuv4DOmU4meSofqREYvtb3EoIKwjy
+RL5WGXKe4nAboAkC5GO7veI5yHL1SaKlssSJtTL/CFpbSLsAFuYbv/NUCWwMY5mwyVTCSlw+HlgKK
+5TH1MzBaSi8fpfyepLT8sHy2Q/VRl6ifb49p6m0KQFbRVvz/OWUd6l4d97BdgtaEz6ueg=="
  }
```

```
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<PAYLOAD_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择有效载荷密钥。

必需：是

<PATH>

保存封装密钥数据的二进制文件路径。

必需：否

<WRAPPING_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以选择换行密钥。

必需：是

<MGF>

指定蒙版生成函数。

> **Note**
>
> 掩码生成函数哈希函数必须与签名机制哈希函数相匹配。

有效值

- mgf1-sha1
- mgf1-sha224

- mgf1-sha256
- mgf1-sha384
- mgf1-sha512

必需：是

相关 主题

- 钥匙包装
- 密钥解包

密钥包装 rsa-oaep

该key wrap rsa-oaep命令使用 HSM 上的 RSA 公钥和包装机制封装有效载荷密钥。RSA-OAEP有效载荷密钥的`extractable`属性必须设置为`true`。

只有密钥的所有者，即创建密钥的加密用户 (CU)，才能封装密钥。共享密钥的用户可以在加密操作中使用该密钥。

要使用该key wrap rsa-oaep命令，您必须先在 AWS CloudHSM 集群中拥有 RSA 密钥。您可以使用钥匙 generate-asymmetric-pair 命令和设置为的`wrap`属性生成 RSA 密钥对。`true`

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

此示例说明如何使用`wrap`属性值设置为 RSA 公钥的key wrap rsa-oaep命令。`true`

Example

```
aws-cloudhsm > key wrap rsa-oaep --payload-filter attr.label=payload-key --wrapping-
filter attr.label=rsa-public-key-example --hash-function sha256 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "payload-key-reference": "0x00000000001c08f1",
    "wrapping-key-reference": "0x00000000007008da",
    "wrapped-key-data": "OjJe4msobPLz9TuSAdULEu17T5rMDWtSlLyBSkLbaZnYzzpdrhsbGLbwZJCtB/
jGkDNdB4qyTAOQwEpggGf6v+Yx6JcesNeKKNU8XZal/YBoHC8noTGUSDI2qr+u2tDc84NPv6d
+F2KOONXsSxMhmxzzNG/gzTVIJhOuy/B1yHjGP4mOXoDZf5+7f5M1CjxBmz4Vva/
wrWHGCSG0yOaWblEvOiHAIt3UBdyKmU+/
My4xjfJv7WGGu3DFUUIZ06TihRtKQhUYU1M9u6NPf9riJJfHsk6QCuSZ9yWThDT9as6i7e3htnyDhIhGWaoK8JU855cN/
YNKAUqkNpC4FPL3iw=="
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<PAYLOAD_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择有效载荷密钥。

必需：是

<PATH>

保存封装密钥数据的二进制文件路径。

必需：否

<WRAPPING_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE以选择换行密钥。

必需：是

<MGF>

指定蒙版生成函数。

> **Note**
>
> 掩码生成函数哈希函数必须匹配签名机制哈希函数。

有效值

- mgf1-sha1
- mgf1-sha224
- mgf1-sha256

- mgf1-sha384
- mgf1-sha512

必需：是

相关 主题

- 钥匙包装
- 密钥解包

密钥包装 rsa-pkcs

该key wrap rsa-pkcs命令使用 HSM 上的 RSA 公钥和包装机制封装有效载荷密钥。RSA-PKCS有效载荷密钥的`extractable`属性必须设置为`true`。

只有密钥的所有者，即创建密钥的加密用户 (CU)，才能封装密钥。共享密钥的用户可以在加密操作中使用该密钥。

要使用该key wrap rsa-pkcs命令，您必须先在 AWS CloudHSM 集群中拥有 RSA 密钥。您可以使用钥匙 generate-asymmetric-pair 命令和设置为的`wrap`属性生成 RSA 密钥对。`true`

用户类型

以下类型的用户均可运行此命令。

- 加密用户（CU）

要求

- 要运行此命令，必须以 CU 身份登录。

语法

```
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
```

示例

此示例说明如何使用 RSA 公钥使用该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=="
  }
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<PAYLOAD_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择有效载荷密钥。

必需：是

<PATH>

保存封装密钥数据的二进制文件路径。

必需：否

<WRAPPING_FILTER>

按键引用（例如key-reference=0xabc）或以空格分隔的按键属性列表，形式attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE为，用于选择换行密钥。

必需：是

相关 主题

- 钥匙包装
- 密钥解包

login

可使用 CloudHSM CLI 中的 login 命令登录到集群中的每个 HSM 和从其中注销。

> **Note**
>
> 如果错误登录尝试满五次，则将锁定账户。要解锁账户，管理员必须使用 cloudhsm_cli 中的 user change-password 命令重置您的密码。

对登录和注销进行故障排除

如果您的集群中有多个 HSM，在锁定账户前，可能会允许您尝试更多次错误登录。这是因为 CloudHSM 客户端跨各 HSM 均衡负载。因此，您的登录尝试可能不是每次都在相同的 HSM 上开始的。如果您要测试此功能，我们建议您在仅具有一个活动 HSM 的集群上测试。

如果您的集群是在 2018 年 2 月前创建的，则在错误登录尝试满 20 次后锁定账户。

用户类型

以下用户均可运行这些命令。

- 未激活的管理员
- Admin
- 加密用户 (CU)

语法

```
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')
```

示例

Example

此命令使用名为 admin1 的管理员用户凭证登录到集群中的所有 HSM。

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<USERNAME>

为用户指定友好名称。最大长度为 31 个字符。唯一允许的特殊字符是下划线 (_)。此命令中的用户名不区分大小写，用户名始终以小写形式显示。

必需：是

<ROLE>

指定分配给该用户的角色。此参数为必需参数。有效值为 admin、crypto-user。

要获取用户的角色，请使用 user list 命令。有关 HSM 上的用户类型的详细信息，请参阅了解 HSM 用户。

<PASSWORD>

指定登录到 HSM 的用户的密码。

相关 主题

- CloudHSM CLI 入门
- 激活集群

登录 mfa-token-sign

使用 AWS CloudHSM CloudHSM CLI 中的 login mfa-token-sign 命令使用多重身份验证登录。要使用此命令，您必须先为 CloudHSM CLI 设置 MFA。

用户类型

以下用户均可运行这些命令。

- Admin
- 加密用户 (CU)

语法

```
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
```

示例

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"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<TOKEN>

将写入未签名令牌文件的文件路径。

必需：是

相关 主题

- CloudHSM CLI 入门
- 激活集群
- 使用 CloudHSM CLI 管理 MFA

注销

可使用 CloudHSM CLI 中的 logout 命令注销集群中的每个 HSM。

用户类型

以下用户均可运行此命令。

- Admin
- 加密用户 (CU)

语法

```
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
```

示例

Example

此命令将您从集群中的所有 HSM 中注销。

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

相关 主题

- CloudHSM CLI 入门
- 激活集群

用户

user 是一组命令的父类别，当这些命令与父类别结合使用时，会创建特定于用户的命令。当前，用户类别由以下命令组成：

- user change-mfa
- user change-password
- user create
- user delete
- user list

user change-mfa

目前，此类别由以下子命令组成：

- user change-mfa token-sign

user change-mfa token-sign

使用 CloudHSM CLI 中的 user change-mfa 命令更新用户账户的多重身份验证（MFA）设置。任何用户账户均可运行此命令。担当管理员角色的账户可为其他用户运行此命令。

用户类型

以下用户均可运行此命令。

- Admin
- 加密用户

语法

目前，只有一种多重策略可供用户使用：令牌签名。

```
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)
```

令牌签名策略要求提供一个用于写入未签名令牌的令牌文件。

```
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')
```

示例

此命令将集群中的每个 HSM 的一个未签名令牌写入 token 指定的文件中。出现系统提示时，对文件中的令牌签名。

Example ：将集群中的每个 HSM 中的一个未签名令牌写入

```
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"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<ROLE>

指定赋予用户账户的角色。此参数为必需参数。有关 HSM 上的用户类型的详细信息，请参阅了解 HSM 用户。

有效值

- 管理员：管理员可以管理用户，却无法管理密钥。
- 加密用户：加密用户可以创建管理密钥并在加密操作中使用密钥。

<USERNAME>

为用户指定友好名称。最大长度为 31 个字符。唯一允许的特殊字符是下划线 (_)。

用户名在创建之后无法更改。在 CloudHSM CLI 命令中，角色和密码区分大小写，但用户名不区分大小写。

必需：是

<CHANGE_PASSWORD>

指定要注册/注销 MFA 的用户的纯文本新密码。

必需：是

<TOKEN>

将写入未签名令牌文件的文件路径。

必需：是

<APPROVAL>

指定要批准操作的已签名仲裁令牌文件的文件路径。仅当仲裁用户服务仲裁值大于 1 时才需要。

<DEREGISTER>

注销 MFA 公有密钥（如有）。

<CHANGE-QUORUM>

更改仲裁公有密钥和 MFA 密钥。

相关 主题

- 了解 HSM 用户的 2FA

user change-password

使用 CloudHSM CLI 中的user change-password命令更改集群中现有用户的 AWS CloudHSM 密码。要对用户启用 MFA，请使用 `user change-mfa` 命令。

所有用户都可更改自己的密码。此外，具有管理员角色的用户可以更改集群中其他用户的密码。您无需输入当前密码即可进行此更改。

> **Note**
>
> 您无法更改当前已登录集群的用户的密码。

用户类型

以下用户均可运行此命令。

- Admin

- 加密用户 (CU)

语法

> **Note**
>
> 要对用户启用多重身份验证（MFA），请使用 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')
```

示例

以下示例说明如何使用 user change-password 重置当前用户或集群中的任何其他用户的密码。

Example ：更改您的密码

集群中的任何用户都可使用 user change-password 更改自己的密码。

以下输出显示 Bob 目前已以加密用户 (CU) 身份登录。

```
aws-cloudhsm > user change-password --username bob --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "bob",
    "role": "crypto-user"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<APPROVAL>

指定要批准操作的已签名仲裁令牌文件的文件路径。仅当仲裁用户服务仲裁值大于 1 时才需要。

<DEREGISTER-MFA>

注销 MFA 公有密钥（如有）。

<DEREGISTER-QUORUM>

注销仲裁公有密钥（如有）。

<PASSWORD>

指定用户的纯文本新密码。

必需：是

<ROLE>

指定赋予用户账户的角色。此参数为必需参数。有关 HSM 上的用户类型的详细信息，请参阅了解 HSM 用户。

有效值

- 管理员：管理员可以管理用户，却无法管理密钥。
- 加密用户：加密用户可以创建管理密钥并在加密操作中使用密钥。

<USERNAME>

为用户指定友好名称。最大长度为 31 个字符。唯一允许的特殊字符是下划线 (_)。

用户名在创建之后无法更改。在 CloudHSM CLI 命令中，角色和密码区分大小写，但用户名不区分大小写。

必需：是

相关 主题

- user list
- user create
- user delete

user change-quorum

user change-quorum 是一组命令的父类别，当这些命令与父类别结合使用时，会创建专门用于更改用户仲裁的命令。

user change-quorum 用于使用指定的仲裁策略注册用户仲裁身份验证。从 SDK 5.8.0 版开始，只有一种仲裁策略可供用户使用，如下所示。

目前，此类别由以下类别和子命令组成：

- token-sign
  - register

user change-quorum token-sign

user change-quorum token-sign 是某些命令的父类别，当这些命令与该父类别结合使用时，会创建特定于令牌签名仲裁操作的命令。

当前，此类别由以下命令组成：

- register

user change-quorum token-sign register

使用 CloudHSM CLI 中的 user change-quorum token-sign register 命令为管理员用户注册令牌签名仲裁策略。

用户类型

以下用户均可运行此命令。

- Admin

语法

```
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')
```

示例

Example

要运行此命令，您需要以您想要 register quorum token-sign 的用户身份登录。

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

该 user change-quorum token-sign register 命令将向 HSM 注册您的公有密钥。因此，它将使您有资格成为仲裁批准者，以执行需要用户获取仲裁签名才能满足必要的仲裁阈值的仲裁要求的操作。

```
aws-cloudhsm > user change-quorum token-sign register \
    --public-key /home/mypemfile \
    --signed-token /home/mysignedtoken
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

现在，您可以运行 user list 命令并确认已为此用户注册了仲裁令牌签名。

```
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"
      }
    ]
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<PUBLIC-KEY>

公有密钥 PEM 文件的文件路径。

必需：是

<SIGNED-TOKEN>

带有由用户私有密钥签名的令牌的文件路径。

必需：是

相关 主题

- 使用 CloudHSM CLI 管理仲裁身份验证
- 使用面向管理员的仲裁身份验证：首次设置
- 更改管理员的仲裁最小值
- 支持仲裁身份验证的服务名称和类型

user create

CloudHSM CLI 中的user create命令会在您的集群中创建一个用户 AWS CloudHSM 。只有具有管理员角色的用户帐户才能运行此命令。

用户类型

以下类型的用户均可运行此命令。

- Admin

要求

若要运行此命令，您必须以管理员身份登录

语法

```
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')
```

示例

这些示例说明如何使用 user create 在 HSM 中创建新用户。

Example ：创建加密用户

此示例在您的 AWS CloudHSM 集群中创建了一个具有加密用户角色的账户。

```
aws-cloudhsm > user create --username alice --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<USERNAME>

为用户指定友好名称。最大长度为 31 个字符。唯一允许的特殊字符是下划线 (_)。此命令中的用户名不区分大小写，用户名始终以小写形式显示。

必需：是

<ROLE>

指定分配给该用户的角色。此参数为必需参数。有效值为 admin、crypto-user。

要获取用户的角色，请使用 user list 命令。有关 HSM 上的用户类型的详细信息，请参阅了解 HSM 用户。

<PASSWORD>

指定登录到 HSM 的用户的密码。

必需：是

<APPROVAL>

指定要批准操作的已签名仲裁令牌文件的文件路径。仅当仲裁用户服务仲裁值大于 1 时才需要。

相关 主题

- user list
- user delete
- user change-password

user delete

CloudHSM CLI 中的user delete命令将用户 AWS CloudHSM 从您的集群中删除。只有担当管理员角色的用户账户可运行此命令。您无法删除当前登录到 HSM 的用户。

用户类型

以下类型的用户均可运行此命令。

- Admin

要求

- 您无法删除拥有密钥的用户帐户。
- 您的用户帐户必须具有管理员角色才能运行此命令。

语法

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
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
```

示例

```
aws-cloudhsm > user delete --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<USERNAME>

为用户指定友好名称。最大长度为 31 个字符。唯一允许的特殊字符是下划线 (_)。此命令中的用户名不区分大小写，用户名始终以小写形式显示。

必需：是

<ROLE>

指定分配给该用户的角色。此参数为必需参数。有效值为 admin、crypto-user。

要获取用户的角色，请使用 user list 命令。有关 HSM 上的用户类型的详细信息，请参阅了解 HSM 用户。

必需：是

<APPROVAL>

指定要批准操作的已签名仲裁令牌文件的文件路径。仅当仲裁用户服务仲裁值大于 1 时才需要。

必需：是

相关 主题

- user list
- user create
- user change-password

user list

CloudHSM CLI 中的 user list 命令列出了您的 CloudHSM 集群中存在的用户帐户。您不必登录 CloudHSM CLI 即可运行此命令。

> **Note**
>
> 如果您添加或删除 HSM，请更新 AWS CloudHSM 客户端和命令行工具使用的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下类型的用户均可运行此命令。

- 所有用户。您不必登录即可运行此命令。

语法

```
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
```

示例

此命令列出了您的 CloudHSM 集群中存在的用户。

```
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"
      }
    ]
  }
}
```

输出包含以下用户属性：

- 用户名：显示用户的用户定义友好名称。用户名始终以小写形式显示。
- 角色：确定用户可在 HSM 上执行的操作。
- 已锁定：表示此用户帐户是否已被锁定。
- MFA：表示此用户账户支持的多重身份验证机制。
- 集群覆盖范围：表示此用户帐户在集群范围内的可用性。

相关 主题

- key_mgmt_util 中的 listUsers
- user create
- user delete
- user change-password

仲裁

quorum 是一组命令的父类别，当其与 quorum 一起使用时，将创建特定于仲裁身份验证或 M of N 操作的命令。目前，该类别由token-sign子类别组成，该子类别包含其自己的命令。有关详细信息，请单击以下链接。

- token-sign

管理员服务：仲裁身份验证用于管理员特权服务，例如创建用户、删除用户、更改用户密码、设置仲裁值以及停用仲裁和 MFA 功能。

每个服务类型都进一步细分为限定服务名称，其中包含一组特定的、可执行的仲裁支持服务操作。

服务名称	服务类型	服务操作
用户	Admin	• user create • user delete • user change-password • user change-mfa
仲裁	Admin	• 法定代币符号 set-quorum-value

相关 主题

- 使用面向管理者的仲裁身份验证：首次设置
- 使用 CloudHSM CLI 管理仲裁身份验证（M of N 访问控制）

quorum token-sign

quorum token-sign 是一组命令的类别，当其与 quorum token-sign 一起使用时，将创建特定于仲裁身份验证或 M of N 操作的命令。

目前，此类别由以下命令组成：

- 删除
- 生成
- list
- list-quorum-values
- list-timeouts
- set-quorum-value
- 设置超时

quorum token-sign delete

使用 CloudHSM CLI 中的 quorum token-sign delete 命令删除仲裁授权服务的一个或多个令牌。

用户类型

以下用户均可运行此命令。

- Admin

语法

```
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')
```

示例

下面的示例显示了如何使用 CloudHSM CLI 中的 quorum token-sign delete 命令删除仲裁授权服务的一个或多个令牌。

Example ：删除仲裁授权服务的一个或多个令牌

```
aws-cloudhsm > quorum token-sign delete --scope all
{
  "error_code": 0,
```

```
  "data": "Deletion of quorum token(s) successful"
}
```

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<SCOPE>

AWS CloudHSM 集群中将删除令牌的范围。

有效值

- 用户：仅用于删除已登录用户所拥有的令牌。
- 全部：用于删除 AWS CloudHSM 集群中的所有令牌。

相关 主题

- user list
- user create
- user delete

quorum token-sign generate

使用 CloudHSM CLI 中的 quorum token-sign generate 命令为仲裁授权服务生成令牌。

对于服务用户和仲裁，在 HSM 集群中每个服务每位用户只能获得一个活动令牌。

> **Note**
>
> 只有管理员才能生成服务令牌。

管理员服务：仲裁身份验证用于管理员特权服务，例如创建用户、删除用户、更改用户密码、设置仲裁值以及停用仲裁和 MFA 功能。

每个服务类型都进一步细分为限定服务名称，其中包含一组特定的、可执行的仲裁支持服务操作。

服务名称	服务类型	服务操作
用户	Admin	• user create • user delete • user change-password • user change-mfa
仲裁	Admin	• 法定代币符号 set-quorum-value

用户类型

以下用户均可运行此命令。

- Admin
- 加密用户 (CU)

语法

```
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
```

示例

此命令将集群中的每个 HSM 的一个未签名令牌写入 token 指定的文件中。

Example ：将集群中的每个 HSM 中的一个未签名令牌写入

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

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<SERVICE>

指定要为其生成令牌的仲裁授权服务。此参数为必需参数。

有效值

- 用户：用于执行仲裁授权用户管理操作的用户管理服务。
- 仲裁：用于为任何仲裁授权服务设置仲裁授权仲裁值的仲裁管理服务。
- 注册：生成用于注册仲裁授权公有密钥的未签名令牌。

必需：是

<TOKEN>

将写入未签名令牌文件的文件路径。

必需：是

相关 主题

- 支持仲裁身份验证的服务名称和类型

quorum token-sign list

使用 CloudHSM CLI 中的quorum token-sign list命令列出集群中存在的所有令牌签名法定令牌。 AWS CloudHSM

用户类型

以下用户均可运行此命令。

- Admin
- 加密用户 (CU)

语法

```
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
```

示例

此命令将列出集群中存在的所有令牌签名令牌。 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"
      }
    ]
  }
}
```

相关 主题

- quorum token-sign generate

法定代币符号 list-quorum-values

使用 CloudHSM CLI 中的quorum token-sign list-quorum-values命令列出集群中设置的法定值。 AWS CloudHSM

用户类型

以下用户均可运行此命令。

- 所有用户。您不必登录即可运行此命令。

语法

```
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
```

示例

此命令列出 AWS CloudHSM 集群中为每项服务设置的法定值。

Example

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

相关 主题

- 支持仲裁身份验证的服务名称和类型

quorum token-sign list-timeouts

使用 CloudHSM CLI 中的 quorum token-sign list-timeouts 命令为所有令牌类型获取令牌超时期限（以秒为单位）。

用户类型

以下用户均可运行此命令。

- 所有用户。您不必登录即可运行此命令。

语法

```
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
```

示例

Example

```
aws-cloudhsm > quorum token-sign list-timeouts
{
  "error_code": 0,
  "data": {
    "generated": 600,
    "approved": 600
  }
}
```

输出包括以下内容：

- 已生成：批准已生成令牌的超时期限（以秒为单位）。
- 已批准：执行仲裁授权操作的已批准令牌的超时期限（以秒为单位）。

相关 主题

- quorum token-sign set-timeout

法定代币符号 set-quorum-value

使用 CloudHSM CLI 中的 quorum token-sign set-quorum-value 命令为仲裁授权服务设置新的仲裁值。

用户类型

以下用户均可运行此命令。

- Admin

语法

```
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')
```

示例

Example

在下面的示例中，此命令会将集群中每个 HSM 的一个未签名令牌写入令牌指定的文件中。出现系统提示时，对文件中的令牌签名。

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

然后，您可以运行 list-quorum-values 命令来确认是否已设置仲裁管理服务的仲裁值：

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

参数

<CLUSTER_ID>

要运行此操作的集群的 ID。

必需：如果已配置多个集群。

<APPROVAL>

要在 HSM 上批准已签名令牌文件的文件路径。

<SERVICE>

指定要为其生成令牌的仲裁授权服务。此参数为必需参数。有关服务类型和名称的更多信息，请参阅 支持仲裁身份验证的服务名称和类型。

有效值

- 用户：用户管理服务。用于执行仲裁授权用户管理操作的服务。

- 仲裁：仲裁管理服务。用于为任何仲裁授权服务设置仲裁授权仲裁值的服务。
- 注册：生成用于注册仲裁授权公有密钥的未签名令牌。

必需：是

<VALUE>

指定要设置的仲裁值。最大仲裁值为八 (8)。

必需：是

相关 主题

- 法定代币符号 list-quorum-values
- 支持仲裁身份验证的服务名称和类型

quorum token-sign set-timeout

使用 CloudHSM CLI 中的 quorum token-sign set-timeout 命令为每种令牌类型设置令牌超时期限（以秒为单位）。

用户类型

以下用户均可运行此命令。

- Admin

语法

```
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')
```

示例

以下示例将显示如何使用 quorum token-sign set-timeout 命令设置令牌超时期限。

```
aws-cloudhsm > quorum token-sign set-timeout --generated 900
{
  "error_code": 0,
  "data": "Set token timeout successful"
}
```

相关 主题

- quorum token-sign list-timeouts

CloudHSM 管理实用程序 (CMU)

cloudhsm_mgmt_util 命令行工具帮助加密员（CO）管理 HSM 中的用户。它包含用于创建、删除和列出用户以及更改用户密码的工具。

KMU 和 CMU 是客户端软件开发工具包 3 套件的一部分。客户端 SDK 3 及其相关的命令行工具（密钥管理实用程序和 CloudHSM 管理实用程序）仅在 HSM 类型 hsm1.medium 中可用。

cloudhsm_mgmt_util 还包含允许加密用户（CU）共享密钥、获取并设置密钥属性的命令。这些命令将对主要密钥管理工具 key_mgmt_util 中的密钥管理命令起到补充作用。

要快速开始使用，请参阅 管理克隆的集群。有关 cloudhsm_mgmt_util 命令的详细信息以及使用这些命令的示例，请参阅 cloudhsm_mgmt_util 命令引用。

主题

- AWS CloudHSM 管理实用程序支持的平台
- CloudHSM 管理实用程序 (CMU) 入门
- 安装和配置 AWS CloudHSM 客户端 (Linux)
- 安装和配置 AWS CloudHSM 客户端 (Windows)
- cloudhsm_mgmt_util 命令引用

AWS CloudHSM 管理实用程序支持的平台

Linux 支持

- 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 支持

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

CloudHSM 管理实用程序 (CMU) 入门

CloudHSM 管理实用程序 (CMU) 可帮助您管理硬件安全模块 (HSM) 用户。使用本主题开始执行基础 HSM 用户管理任务，例如创建用户、列出用户以及将 CMU 连接至集群。

1. 若要使用 CMU，必须先使用配置工具，更新本地 CMU 配置中的 `--cmu` 参数和集群 HSM 中的其中一项 IP 地址。当您使用 CMU 完成此操作，确保您在管理集群所有 HSM 的 HSM 用户。

   Linux

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

Windows

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

2. 使用以下命令以交互模式启动 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
```

输出应与以下内容类似，具体取决于您拥有的 HSM 的数量。

```
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.
```

当 cloudhsm_mgmt_util 运行时，提示符将变为 `aws-cloudhsm>`。

3. 使用 loginHSM 命令登录到集群。任何类型的任何用户均可使用此命令登录集群。

以下示例日志中的命令将登录 admin，即默认加密员 (CO)。您在激活集群时设置该用户的密码。您可以使用 `-hpswd` 参数隐藏您的密码。

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

系统将会提示您输入密码。您输入密码后，系统隐藏密码，输出显示命令成功，且您已连接至集群上的所有 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. 使用 listUsers 列出集群上的所有用户。

```
aws-cloudhsm>listUsers
```

CMU 会列出集群上的所有用户。

```
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 创建名为 **example_user** 的 CU 用户，密码为 **password1**。

   您通过应用程序中的 CU 用户执行加密和密钥管理操作。由于您在第 3 步中以 CO 用户身份登录，因此您可以创建 CU 用户。只有 CO 用户可通过 CMU 执行用户管理任务，例如创建和删除用户以及更改其他用户密码。

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

   CMU 会提示您关于创建用户操作的信息。

   ```
   
   ************************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. 若要创建 CU 用户**example_user**，请键入 **y**。

7. 使用 listUsers 列出集群上的所有用户。

   ```
   aws-cloudhsm>listUsers
   ```

   CMU 列出了集群中的所有用户，包括您刚创建的新 CU 用户。

   ```
   
   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 命令可从 HSM 退出。

```
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 命令停止 cloudhsm_mgmt_util。

```
aws-cloudhsm>quit


disconnecting from servers, please wait...
```

安装和配置 AWS CloudHSM 客户端 (Linux)

要与 AWS CloudHSM 集群中的 HSM 进行交互，您需要适用于 Linux 的 AWS CloudHSM 客户端软件。您应在之前创建的 Linux EC2 客户端实例上安装该软件。如果您使用的是 Windows，也可以安装客户端。有关更多信息，请参阅 安装和配置 AWS CloudHSM 客户端 (Windows)。

任务

- 安装 AWS CloudHSM 客户端和命令行工具
- 编辑客户端配置

安装 AWS CloudHSM 客户端和命令行工具

连接到您的客户端实例并运行以下命令来下载和安装 AWS CloudHSM 客户端和命令行工具。

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
```

编辑客户端配置

必须先编辑 AWS CloudHSM 客户端配置，然后才能使用客户端连接到集群。

编辑客户端配置

1. 如果在 cloudhsm_mgmt_util 上安装客户端软件开发工具包 3，请完成以下步骤以确保集群中的所有节点都已同步。

   a. 运行 configure -a ***<IP of one of the HSMs>***。

   b. 重新启动客户端服务。

   c. 运行 config -m。

2. 将您的颁发证书 — 用于签署集群证书的证书 — 复制到客户端实例上的以下位置：`/opt/cloudhsm/etc/customerCA.crt`。您需要在客户端实例上具有实例根用户权限才能将您的证书复制到该位置。

3. 使用以下 configure 命令更新 AWS CloudHSM 客户端和命令行工具的配置文件，指定集群中 HSM 的 IP 地址。要获取 HSM 的 IP 地址，请在AWS CloudHSM 控制台中查看您的集群，或者运行describe-clusters AWS CLI 命令。在命令输出中，HSM 的 IP 地址为 `EniIp` 字段的值。如果您有多个 HSM，请选择其中任一 HSM 的 IP 地址；选择哪个 HSM 并不重要。

   ```
   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. 转到 激活集群。

安装和配置 AWS CloudHSM 客户端 (Windows)

要在 Windows 上使用 AWS CloudHSM 集群中的 HSM，你需要适用于 Windows 的 AWS CloudHSM 客户端软件。您应在之前创建的 Windows Server 实例上安装它。

安装（或更新）最新的 Windows 客户端和命令行工具

1. 连接到您的 Windows Server 实例。
2. 下载 AWSCloudHSMClient-latest.msi 安装程序。
3. 如果在 cloudhsm_mgmt_util 上安装客户端软件开发工具包 3，请完成以下步骤以确保集群中的所有节点都已同步。

   a. 运行 configure -a ***<IP of one of the HSMs>***。

   b. 重新启动客户端服务。

   c. 运行 config -m。
4. 前往您的下载位置并使用管理权限运行安装程序 (AWSCloudHSMClient-latest.msi)。
5. 按照安装程序说明进行操作，然后在安装程序完成后选择关闭。
6. 将您的自签名颁发证书 — 用于签署集群证书的证书 — 复制到 `C:\ProgramData\Amazon\CloudHSM` 文件夹。
7. 运行以下命令以更新您的配置文件。如果要更新，在重新配置期间务必停止并启动客户端：

   ```
   C:\Program Files\Amazon\CloudHSM\bin\ configure.exe -a <HSM IP address>
   ```
8. 转到 激活集群。

备注：

- 如果更新客户端，以前安装的现有配置文件不会被覆盖。
- 适用于 Windows 的 AWS CloudHSM 客户端安装程序会自动注册加密 API：下一代 (CNG) 和密钥存储提供程序 (KSP)。要卸载客户端，请再次运行安装程序并按照卸载说明操作。
- 如果您使用的是 Linux，也可以安装 Linux 客户端。有关更多信息，请参阅 安装和配置 AWS CloudHSM 客户端 (Linux)。

cloudhsm_mgmt_util 命令引用

cloudhsm_mgmt_util 命令行工具帮助加密员（CO）管理 HSM 中的用户。它还包含允许加密用户 (CU) 共享密钥、获取并设置密钥属性的命令。这些命令将对 key_mgmt_util 命令行工具中的主要密钥管理命令起到补充作用。

要快速开始使用，请参阅 管理克隆的集群。

在运行任何 cloudhsm_mgmt_util 命令之前，必须启动 cloudhsm_mgmt_util 并登录。请确保使用可运行您计划使用的命令的用户账户类型登录。

若要列出现有 cloudhsm_mgmt_util 命令，请运行以下命令：

```
aws-cloudhsm> help
```

要获取 cloudhsm_mgmt_util 命令的语法，请运行以下命令：

```
aws-cloudhsm> help <command-name>
```

> **Note**
>
> 根据文档使用语法。虽然内置的软件帮助可能提供额外选项，但不应将这些选项视为支持的选项，也不应在生产代码中使用这些选项。

要运行命令，请输入完整或部分命令名称（应足以与其他 cloudhsm_mgmt_util 命令的名称互相区分）。

例如，要获取 HSM 上用户的列表，请输入 listUsers 或 listU。

```
aws-cloudhsm> listUsers
```

要结束 cloudhsm_mgmt_util 会话，请运行以下命令：

```
aws-cloudhsm> quit
```

有关解释密钥属性的帮助，请参阅 密钥属性引用。

以下主题介绍 cloudhsm_mgmt_util 中的命令。

> **Note**
>
> key_mgmt_util 和 cloudhsm_mgmt_util 中的部分命令名称相同。但是，这些命令通常具有不同的语法、不同的输出和轻微不同的功能。

命令	描述	用户类型
changePswd	更改用户在 HSM 上的密码。所有用户都可更改自己的密码。CO 可以更改任何人的密码。	CO
createUser	在 HSM 上创建所有类型的用户。	CO
deleteUser	从 HSM 中删除所有类型的用户。	CO
findAllKeys	获取用户拥有或共享的密钥。此外获取每个 HSM 上所有密钥的密钥所有权和共享数据的哈希。	CO、AU
getAttribute	获取 AWS CloudHSM 密钥的属性值并将其写入文件或 stdout（标准输出）。	CU
getCert	获取某个特定 HSM 的证书并采用所需证书格式将其保存。	所有。
getHSMInfo	获取运行 HSM 的硬件的相关信息。	所有。不需要登录名。
getKeyInfo	获取密钥的所有者、共享用户和仲裁身份验证状态。	所有。不需要登录名。

命令	描述	用户类型
info	获取有关 HSM 的信息，包括 IP 地址、主机名、端口和当前用户。	所有。不需要登录名。
listUsers	获取每个 HSM 中的用户、其用户类型和 ID 以及其他属性。	所有。不需要登录名。
loginHSM 和 logoutHSM	登录和退出 HSM。	所有。
退出	退出 cloudhsm_mgmt_util.	所有。不需要登录名。
服务器	进入和退出 HSM 上的服务器模式。	所有。
registerQuorumPub钥匙	将 HSM 用户与非对称 RSA-2048 密钥对关联。	CO
setAttribute	更改现有密钥的标签、加密、解密、包装和解开包装属性的值。	CU
shareKey	与其他用户共享现有密钥。	CU
syncKey	跨克隆 AWS CloudHSM 群集同步密钥。	CU、CO
syncUser	跨克隆 AWS CloudHSM 群集同步用户。	CO

changePswd

cloudhsm_mgmt_util 中的 changePswd 命令可更改集群中的 HSM 上现有用户的密码。

所有用户都可更改自己的密码。此外，加密员（CO 和 PCO）可以更改另一个 CO 或加密用户（CU）的密码。您无需输入当前密码即可进行此更改。

> **Note**
>
> 您无法更改当前登录 AWS CloudHSM 客户端的用户的密码或 key_mgmt_util。

要对 ChangepsWD 进行问题排查

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下用户均可运行此命令。

- 加密员（CO）
- 加密用户 (CU)

语法

按照语法图表中指定的顺序输入参数。使用 `-hpswd` 参数来掩盖您的密码。要为 CO 用户启用双重身份验证 (2FA, two-factor authentication)，请使用 `-2fa` 参数并添加文件路径。有关更多信息，请参阅the section called “参数”。

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

示例

以下示例说明如何使用 changePassword 重置当前用户或 HSM 中的任何其他用户的密码。

Example ：更改您的密码

HSM 上的任何用户都可使用 changePswd 更改自己的密码。在更改密码之前，请使用 info 获取有关集群中每个 HSM 的信息，包括已登录用户的用户名和用户类型。

以下输出显示 Bob 目前已以加密用户 (CU) 身份登录。

```
        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)'
```

为了更改密码，Bob 运行后跟用户类型、用户名和新密码的 changePswd。

```
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 ：更改其他用户的密码

您必须是 CO 或 PCO 才能更改 HSM 上另一个 CO 或 CU 的密码。在更改其他用户的密码之前，请使用 info 命令以确认您的用户类型为 CO 或 PCO。

以下输出确认作为 CO 的 Alice 当前已登录。

```
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 想重置另一个用户 John 的密码。她在更改密码之前，使用 listUsers 命令验证 John 的用户类型。

以下输出将 John 列出为 CO 用户。

```
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
```

为了更改密码，Alice 运行后跟用户类型、用户名和新密码的 changePswd。

```
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
```

参数

按照语法图表中指定的顺序输入参数。使用 -hpswd 参数来掩盖您的密码。要为 CO 用户启用 2FA，请使用 -2fa 参数并添加文件路径。有关使用 2FA 的更多信息，请参阅 使用 CMU 管理 2FA。

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user-type>

指定您要更改其密码的用户的当前类型。您不能使用 changePswd 来更改用户类型。

有效值为 CO、CU、PCO 和 PRECO。

要获取用户类型，请使用 listUsers。有关 HSM 上的用户类型的详细信息，请参阅 了解 HSM 用户。

必需：是

<user-name>

指定用户的友好名称。此参数不区分大小写。您不能使用 changePswd 来更改用户名。

必需：是

<password | -hpswd >

为用户指定新密码。输入一个包含 7 到 32 个字符的字符串。此值区分大小写。密码在键入时将明文显示。要隐藏密码，请使用 -hpswd 参数代替密码，然后按照提示操作。

必需：是

[-2fa </path/to/authdata>]

指定为该 CO 用户启用 2FA。要获取设置 2FA 所需的数据，请在 -2fa 参数后加上文件系统中带有文件名的位置的路径。有关使用 2FA 的更多信息，请参阅 使用 CMU 管理 2FA。

必需：否

相关 主题

- info
- listUsers
- createUser
- deleteUser

createUser

使用 cloudhsm_mgmt_util 中的 createUser 命令可在 HSM 上创建一个用户。只有加密员（CO 和 PRECO）才能运行此命令。如果命令成功，它将在群集的所有 HSM 中创建该用户。

对 createUser 进行问题排查

如果您的 HSM 配置不正确，可能不会在所有 HSM 上创建该用户。要将该用户添加到其上缺少该用户的任何 HSM，请仅在缺少该用户的 HSM 上使用 syncUser 或 createUser 命令。要防止配置错误，请运行配置工具与 -m 选项。

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下类型的用户均可运行此命令。

- 加密员（CO 和 PRECO）

语法

按照语法图表中指定的顺序输入参数。使用 -hpswd 参数来掩盖您的密码。要创建采用双重身份验证 (2FA, two-factor authentication) 的 CO 用户，请使用 -2fa 参数并添加文件路径。有关更多信息，请参阅 the section called “参数”。

```
createUser <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

示例

这些示例说明如何使用 createUser 在 HSM 中创建新用户。

Example ：创建加密员

此示例在群集的 HSM 上创建加密员 (CO)。第一个命令使用 loginHSM 以加密管理者身份登录 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)
```

第二个命令使用 createUser 命令在 HSM 上创建新的加密管理者 alice。

警告消息说明，此命令将在群集的所有 HSM 上创建用户。但是，如果此命令在任何 HSM 上失败，这些 HSM 上将不存在用户。要继续，请键入 y。

输出显示，已在群集的所有三个 HSM 上创建新用户。

```
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
```

当此命令完成后，alice 将在 HSM 上具有与 admin CO 用户相同的权限，包括更改 HSM 上任何用户的密码。

最后一个命令使用 listUsers 命令验证群集的所有三个 HSM 上是否存在 alice。输出还显示，已为 alice 分配用户 ID 3。.您可以使用用户 ID alice 在其他命令中进行识别，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 ：创建加密用户

此示例将在 HSM 上创建一个加密用户 (CU) bob。加密用户可以创建和管理密钥，但无法管理用户。

在键入 y 以响应警告消息之后，输出显示，已在群集的所有三个 HSM 上创建 bob。新 CU 可以登录 HSM 来创建和管理密钥。

此命令使用了密码值 `defaultPassword`。之后，bob 或任何 CO 可以使用 changePswd 命令来更改其密码。

```
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
```

参数

按照语法图表中指定的顺序输入参数。使用 `-hpswd` 参数来掩盖您的密码。要创建启用 2FA 的 CO 用户，请使用 `-2fa` 参数并添加文件路径。有关 2FA 的更多信息，请参阅 使用 CMU 管理 2FA。

```
createUser <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user-type>

指定用户类型。此参数为必需参数。

有关 HSM 上的用户类型的详细信息，请参阅 了解 HSM 用户。

有效值：

- CO：加密员可以管理用户，但无法管理密钥。
- CU：加密用户可以创建管理密钥并在加密操作中使用密钥。

当您在 HSM 激活期间分配密码时，PRECO 将转换为 CO。

必需：是

<user-name>

为用户指定友好名称。最大长度为 31 个字符。唯一允许的特殊字符是下划线 (_)。

用户名在创建之后无法更改。在 cloudhsm_mgmt_util 命令中，用户类型和密码都区分大小写，但用户名不区分大小写。

必需：是

<password | -hpswd >

为用户指定密码。输入一个包含 7 到 32 个字符的字符串。此值区分大小写。密码在键入时将明文显示。要隐藏密码，请使用 `-hpswd` 参数代替密码，然后按照提示操作。

要更改用户密码，请使用 changePswd。任何 HSM 用户都可以更改自己的密码，但 CO 用户可以更改 HSM 上任何类型的任何用户的密码。

必需：是

[-2fa </path/to/authdata>]

指定创建启用了 2FA 的 CO 用户。要获取设置 2FA 身份验证所需的数据，请在 `-2fa` 参数后加上文件系统中带有文件名的位置的路径。有关设置和使用 2FA 的更多信息，请参阅 使用 CMU 管理 2FA。

必需：否

相关 主题

- listUsers
- deleteUser
- syncUser

- changePswd

deleteUser

cloudhsm_mgmt_util 中的 deleteUser 命令将用户从硬件安全模块 (HSM, hardware security modules) 中删除。只有加密员（CO）才能运行此命令。您无法删除当前登录到 HSM 的用户。有关删除用户的更多信息，请参阅如何删除 HSM 用户。

> **Tip**
>
> 您无法删除拥有密钥的加密用户 (CU)。

用户类型

以下类型的用户均可运行此命令。

- CO

语法

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
deleteUser <user-type> <user-name>
```

示例

本示例从集群内的 HSM 中删除加密管理者 (CO)。第一个命令使用 listUsers 列出 HSM 上的所有用户。

输出表明用户 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
```

第二个命令使用 deleteUser 命令从 HSM 中删除 `alice`。

输出表明该命令在集群中的所有三个 HSM 上已成功执行。

```
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)
```

最后一个命令使用 listUsers 命令验证是否从集群上的所有三个 HSM 中删除了 `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
```

参数

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
deleteUser <user-type> <user-name>
```

<user-type>

指定用户类型。此参数为必需参数。

> **Tip**
>
> 您无法删除拥有密钥的加密用户 (CU)。

有效值为 CO 和 CU。

要获取用户类型，请使用 listUsers。有关 HSM 上的用户类型的详细信息，请参阅 了解 HSM 用户。

必需：是

<user-name>

为用户指定友好名称。最大长度为 31 个字符。唯一允许的特殊字符是下划线 (_)。

用户名在创建之后无法更改。在 cloudhsm_mgmt_util 命令中，用户类型和密码都区分大小写，但用户名不区分大小写。

必需：是

相关 主题

- listUsers
- createUser
- syncUser
- changePswd

findAllKeys

cloudhsm_mgmt_util 中的 findAllKeys 命令用于获取指定加密用户（CU）所拥有或共享的密钥。它还会返回每个 HSM 上的用户数据的哈希。您可以使用哈希来大致确定用户、密钥所有权和密钥共享数据在集群中的所有 HSM 上是否相同。在输出中，用户拥有的密钥由（o）进行注释，而共享密钥由（s）进行注释。

仅当指定 CU 拥有公有密钥时，findAllKeys 才会返回该密钥，即使 HSM 上的所有 CU 都可以使用任何公有密钥也是如此。此行为不同于 key_mgmt_util 中的 findKey，前者将为所有 CU 用户返回公有密钥。

只有加密管理者（CO 和 PCO）和设备用户 (AU) 才能运行此命令。加密用户 (CU) 可以运行以下命令：

- listUsers，用于查找所有用户
- 通过 key_mgmt_util 中的 findKey 可以找到其可以使用的密钥
- getKeyInfo在 key_mgmt_util 中查找他们拥有或共享的特定密钥的所有者和共享用户

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下用户均可运行此命令。

- 加密员 (CO、PCO)
- 设备用户 (AU)

语法

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

示例

以下示例演示如何使用 `findAllKeys` 来查找某个用户的所有密钥并获取每个 HSM 上的密钥用户信息的哈希。

Example ：查找 CU 的密钥

本示例使用 findAllKeys 来查找 HSM 中由用户 4 拥有和共享的密钥。该命令使用第二个参数的值 `0` 来隐藏哈希值。由于本示例忽略了可选的文件名，因此该命令将写入到 stdout（标准输出）。

输出表明用户 4 可使用 6 个密钥：8、9、17、262162、19 和 31。输出使用（s）指示用户显式共享的密钥。用户拥有的密钥由（o）表示，包含用户未共享的对称密钥和私有密钥，以及对所有加密用户可用的公有密钥。

```
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 ：验证用户数据是否已同步

本示例使用 findAllKeys 来确认集群中的所有 HSM 都包含相同的用户、密钥所有权和密钥共享值。为了执行此操作，它获取了每个 HSM 上的密钥用户数据的哈希并比较了哈希值。

为了获取密钥哈希，该命令在第二个参数中使用了值 1。由于可选文件名已忽略，因此该命令将密钥哈希写入到 stdout。

本示例指定了用户 6，但哈希值对于拥有或共享 HSM 上的任一密钥的任何用户都是相同的。如果指定的用户不拥有或共享任何密钥 (如 CO)，则该命令不会返回哈希值。

输出表明集群中的两个 HSM 的密钥哈希是相同的。如果其中一个 HSM 具有不同的用户、不同的密钥所有者或不同的共享用户，密钥哈希值将不相等。

```
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)
```

此命令演示了哈希值表示 HSM 上的所有密钥的用户数据。此命令对用户 3 使用了 findAllKeys。与仅拥有或共享 3 个密钥的用户 6 不同，用户 3 拥有或共享 17 个密钥，但它们的密钥哈希值是相同的。

```
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)
```

参数

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

<user id>

获取指定用户拥有或共享的所有密钥。输入 HSM 上的用户的用户 ID。要查找所有用户的用户 ID，请使用 listUsers。

所有用户 ID 都有效，但 `findAllKeys` 仅为加密用户 (CU) 返回密钥。

必需：是

<key hash>

包含 (1) 或排除 (0) 用户所有权以及每个 HSM 中的所有密钥的共享数据的哈希。

当 `user id` 参数表示拥有或共享密钥的用户时，系统将填充密钥哈希。密钥哈希值对拥有或共享 HSM 上的密钥的所有用户都是相同的，即使它们拥有和共享不同的密钥。但是，当 `user id` 表示不拥有或共享任何密钥的用户 (如 CO) 时，将不会填充哈希值。

必需：是

<output file>

将输出写入到指定文件。

必需：否

默认值：Stdout

相关 主题

- changePswd
- deleteUser
- listUsers
- syncUser
- key_mgmt_util 中的 findKey
- getKeyInfo在 key_mgmt_util 中

getAttribute

cloudhsm_mgmt_util 中的 getAttribute 命令从集群中的所有 HSM 获取某个密钥的属性值并将其写入 stdout（标准输出）或文件。仅加密用户 (CU) 可以运行此命令。

密钥属性是密钥的属性。它们包括特征 (如密钥类型、类、标签和 ID) 和表示您可对密钥执行的操作 (如加密、解密、包装、签名和验证) 的值。

您只能对您拥有的密钥和与您共享的密钥使用 getAttribute。您可运行此命令或 key_mgmt_util 中的 getAttribute 命令 (将密钥的一个或所有属性值写入文件)。

要获取包含属性和表示属性的常量的列表，请使用 listAttributes 命令。要更改现有密钥的属性值，请使用 key_mgmt_util 中的 setAttribute 和 cloudhsm_mgmt_util 中的 setAttribute。有关解释密钥属性的帮助，请参阅 密钥属性引用。

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下用户均可运行此命令。

- 加密用户 (CU)

语法

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
getAttribute <key handle> <attribute id> [<filename>]
```

示例

此示例获取 HSM 中的密钥的可提取属性值。您可以使用与此类似的命令来确定您是否能从 HSM 导出密钥。

第一条命令使用 listAttributes 查找表示可提取属性的常量。输出表明，OBJ_ATTR_EXTRACTABLE 的常量为 354。您也可以在 密钥属性引用 中找到此信息与属性的描述以及属性值。

```
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
```

第二条命令使用 getAttribute 在 HSM 中获取密钥句柄为 262170 的密钥的可提取属性值。要指定可提取属性，此命令使用 354 (表示此属性的常量)。由于此命令未指定文件名，getAttribute 会将输出写入 stdout。

输出表明，所有 HSM 上的可提取属性值为 1。此值指示密钥的所有者可导出密钥。当值为 0 (0x0) 时，无法从 HSM 导出它。可在创建密钥时设置可提取属性的值，但无法更改此值。

```
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
```

参数

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
getAttribute <key handle> <attribute id> [<filename>]
```

<key-handle>

指定目标密钥的密钥句柄。您在每个命令中只能指定一个密钥。要获取密钥的密钥句柄，请使用 key_mgmt_util 中的 findKey。

您必须拥有指定密钥或必须与您共享此密钥。要查找密钥的用户，请在 key_mgmt_uti getKeyInfol 中使用。

必需：是

<attribute id>

标识属性。输入表示属性的常量，或表示所有属性的 512。例如，要获取密钥类型，请输入 256（OBJ_ATTR_KEY_TYPE 属性的常量）。

要列出属性及其常量，请使用 listAttributes。有关解释密钥属性的帮助，请参阅 密钥属性引用。

必需：是

<filename>

将输出写入到指定文件。输入文件路径。

如果指定文件存在，getAttribute 将覆盖此文件而不发出警告。

必需：否

默认值：Stdout

相关 主题

- key_mgmt_util 中的 getAttribute
- listAttributes
- cloudhsm_mgmt_util 中的 setAttribute
- key_mgmt_util 中的 setAttribute
- 密钥属性引用

getCert

使用 cloudhsm_mgmt_util 中的 getCert 命令，您可以检索集群中某个特定 HSM 的证书。当您运行此命令时，请指定要检索的证书的类型。要执行此操作，请使用其中一个相应的整数，如下列参数部分所述。要了解这些证书中的每个证书的角色，请参阅验证 HSM 身份。

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下用户均可运行此命令。

- 所有用户。

先决条件

在开始之前，您必须进入目标 HSM 上的服务器模式。有关更多信息，请参阅服务器。

语法

在服务器模式下使用 getCert 命令一次：

```
server> getCert <file-name> <certificate-type>
```

示例

首选，输入服务器模式。此命令将进入 HSM 上的服务器模式，服务器编号为 0。

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

然后，使用 getCert 命令。在本示例中，我们使用 /tmp/P0.crt 作为证书将保存到的文件的名称并且使用 4（客户根证书）作为所需证书类型：

```
server0> getCert /tmp/P0.crt 4
getCert Success
```

参数

```
getCert <file-name> <certificate-type>
```

<file-name>

指定证书将保存到的文件的名称。

必需：是

<certificate-type>

指定要检索的证书的类型的整数。整数及其相应的证书类型如下所示：

- 1 – 制造商根证书
- 2 - 制造商硬件证书
- 4 - 客户根证书
- 8 - 集群证书（由客户根证书签署）
- 16 - 集群证书（链接到制造商根证书）

必需：是

相关 主题

- 服务器

getHSMInfo

cloudhsm_mgmt_util 中的 getHSMInfo 命令将获取运行每个 HSM 的硬件的相关信息，包括硬件和固件的型号、序列号、FIPS 状态、内存、温度和版本号。该信息还包括 cloudhsm_mgmt_util 用来引用 HSM 的服务器 ID。

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下类型的用户均可运行此命令。

- 所有用户。您不必登录即可运行此命令。

语法

此命令没有输出参数。

```
getHSMInfo
```

示例

此示例使用 getHSMInfo 获取有关群集中的 HSM 的信息。

```
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

...
```

相关 主题

- info

getKeyInfo

key_mgmt_util 中的 getKeyInfo 命令将返回可使用密钥的 HSM 用户的用户 ID，这些用户包括所有者和共享密钥的加密用户（CU）。如果对某个密钥启用了仲裁身份验证，getKeyInfo 也将返回必须批准使用该密钥的加密操作的用户数。您只能在您拥有的密钥和与您共享的密钥上运行 getKeyInfo。

当您在公有密钥上运行 getKeyInfo 时，getKeyInfo 将仅返回密钥所有者，即使 HSM 的所有用户都可以使用该公有密钥。要在您的 HSM 中查找 HSM 用户的用户 ID，请使用 listUsers。要查找特定用户的密钥，请在 key_mgmt_util 中使用 findKey -u。加密官员可以findAllKeys在 cloudhsm_mgmt_util 中使用。

您拥有您创建的密钥。在创建密钥时，您可以与其他用户共享它。然后，要共享或取消共享现有密钥，请在 cloudhsm_mgmt_util 中使用 shareKey。

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下类型的用户均可运行此命令。

- 加密用户 (CU)

语法

```
getKeyInfo -k <key-handle> [<output file>]
```

示例

这些示例说明如何使用 getKeyInfo 获取有关密钥用户的信息。

Example ：获取非对称密钥的用户

此命令将获取可使用密钥句柄为 262162 的 AES (非对称) 密钥的用户。输出表明，用户 3 拥有密钥并将密钥与用户 4 和 6 共享。

仅用户 3、4 和 6 可以在密钥 262162 上运行 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 ：获取对称密钥对的用户

这些命令使用 getKeyInfo 获取可使用 ECC（对称）密钥对中的密钥的用户。公有密钥具有密钥句柄 262179。私有密钥具有密钥句柄 262177。

当您在私有密钥 (262177) 上运行 getKeyInfo 时，它会返回密钥所有者 (3) 和共享密钥的加密用户 (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
```

当您在公有密钥 (262179) 上运行 getKeyInfo 时，它只会返回密钥所有者，即用户 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
```

要确认用户 4 可以使用公有密钥（和 HSM 上的所有公有密钥），请使用 key_mgmt_util 中 findKey 的 -u 参数。

输出显示，用户 4 可以使用密钥对中的公有密钥 (262179) 和私有密钥 (262177)。用户 4 还可以使用所有其他公有密钥以及他们创建的或与他们共享的任何私有密钥。

```
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 ：获取密钥的仲裁身份验证值 (m_value)

此示例介绍如何获取密钥的 m_value。m_value 是仲裁中的用户数，这些用户必须批准使用密钥的任何加密操作和共享/取消共享密钥的操作。

如果对某个密钥启用了仲裁身份验证，用户的仲裁必须批准使用该密钥的任何加密操作。要启用仲裁身份验证并设置仲裁大小，请在创建密钥时使用 -m_value 参数。

此命令用于创建genSymKey与用户 4 共享的 256 位 AES 密钥。它使用 m_value 参数启用仲裁身份验证并将仲裁大小设置为两个用户。用户的数量必须足够大，才能提供所需的批准。

输出表明，该命令创建了密钥 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 获取有关密钥 10 的用户的信息。输出表明，该密钥由用户 3 所有并与用户 4 共享。它还表明，这两个用户的仲裁必须批准使用该密钥的所有加密操作。

```
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
```

参数

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
getKeyInfo -k <key-handle> <output file>
```

<key-handle>

指定 HSM 中的一个密钥的密钥句柄。输入您拥有或共享的密钥的密钥句柄。此参数为必需参数。

必需：是

<output file>

将输出写入指定文件，而不是 stdout。如果该文件存在，则命令将覆盖该文件而不发出警告。

必需：否

默认值：stdout

相关 主题

- getKeyInfo在 key_mgmt_util 中
- key_mgmt_util 中的 findKey
- findAllKeys在 cloudhsm_mgmt_util
- listUsers
- shareKey

info

cloudhsm_mgmt_util 中的 info 命令可获取有关集群中的每个 HSM 的信息，包括主机名称、端口、IP 地址和已登录 HSM 上的 cloudhsm_mgmt_util 的用户的名称和类型。

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下类型的用户均可运行此命令。

- 所有用户。您不必登录即可运行此命令。

语法

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
info server <server ID>
```

示例

此示例使用 info 获取有关集群中 HSM 的信息。此命令使用 0 来引用集群中的第一个 HSM。输出显示了 IP 地址、端口以及当前用户的类型和名称。

```
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)'
```

参数

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
info server <server ID>
```

<server id>

指定 HSM 的服务器 ID。HSM 均分配有序号 (以 0 开始) 来表示它们添加到集群的顺序。要查找 HSM 的服务器 ID，请使用 getHSMInfo。

必需：是

相关 主题

- getHSMInfo
- loginHSM 和 logoutHSM

listAttributes

cloudhsm_mgmt_util 中的listAttributes命令列出了密钥的属性以及代表它们的常量。 AWS CloudHSM 您使用这些常量来标识 getAttribute 和 setAttribute 命令中的属性。

有关解释密钥属性的帮助，请参阅 密钥属性引用。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

用户类型

以下用户均可运行此命令。

- 所有用户。您不必登录即可运行此命令。

语法

```
listAttributes [-h]
```

示例

此命令将列出您可以在 key_mgmt_util 中获取和更改的密钥属性和表示它们的常量。有关解释密钥属性的帮助，请参阅 密钥属性引用。要表示所有属性，请使用 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
```

参数

-h

显示该命令的帮助信息。

必需：是

相关 主题

- getAttribute
- setAttribute
- 密钥属性引用

listUsers

cloudhsm_mgmt_util 中的 listUsers 命令可获取每个 HSM 中的用户以及用户的类型和其他属性。所有类型的用户都可以运行此命令。您甚至不必登录 cloudhsm_mgmt_util 即可运行此命令。

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下类型的用户均可运行此命令。

- 所有用户。您不必登录即可运行此命令。

语法

此命令没有输出参数。

```
listUsers
```

示例

此命令将列出集群中每个 HSM 上的用户并显示用户的属性。您可以使用 `User ID` 属性来识别其他命令（如 deleteUser、changePswd 和 findAllKeys）中的用户。

```
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
```

输出包含以下用户属性：

- 用户 ID：标识 key_mgmt_util 和 cloudhsm_mgmt_util 命令中的用户。
- User type：确定用户可在 HSM 上执行的操作。
- User Name：显示用户的用户定义友好名称。
- MofnPubKey：表示用户是否注册了用于签署法定身份验证令牌的密钥对。
- LoginFailureCnt：表示用户登录失败的次数。
- 2FA：指示用户已启用多因素验证。

相关 主题

- 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 命令登录和登出集群中的每个 HSM。任何类型的任何用户都可以使用这些命令。

> **Note**
>
> 如果错误登录尝试满五次，则将锁定账户。要解锁账户，加密员 (CO) 必须在 cloudhsm_mgmt_util 中使用 changePswd 命令重置您的密码。

若要对 loginHSM 和 LogouthSM 进行故障排除

运行这些 cloudhsm_mgmt_util 命令前，必须启动 cloudhsm_mgmt_util。

如果您添加或删除 HSM，请更新 AWS CloudHSM 客户端和命令行工具使用的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

如果您的集群中有多个 HSM，在锁定账户前，可能会允许您尝试更多次错误登录。这是因为 CloudHSM 客户端跨各 HSM 均衡负载。因此，您的登录尝试可能不是每次都在相同的 HSM 上开始的。如果您要测试此功能，我们建议您在仅具有一个活动 HSM 的集群上测试。

如果您的集群是在 2018 年 2 月前创建的，则在错误登录尝试满 20 次后锁定账户。

用户类型

以下用户均可运行这些命令。

- 准加密员 (PRECO)
- 加密员 (CO, Crypto officer)
- 加密用户 (CU)

语法

按照语法图表中指定的顺序输入参数。使用 -hpswd 参数来掩盖您的密码。要使用双重身份验证 (2FA) 登录，请使用 -2fa 参数并纳入文件路径。有关更多信息，请参阅 the section called “参数”。

```
loginHSM <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

```
logoutHSM
```

示例

这些示例演示如何使用 loginHSM 和 logoutHSM 登录到集群中的所有 HSM 和从其中注销。

Example ：登录到集群中的 HSM

此命令使用名为 admin、密码为 co12345 的 CO 用户的凭证让您登录到集群中的所有 HSM。输出显示命令已成功，并且您已连接到 HSM（在这种情况下，为 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 ：使用隐藏密码登录

此命令与上述示例相同，但是这一次您指定了系统应隐藏密码。

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

系统将会提示您输入密码。您输入密码后，系统隐藏密码，输出显示命令成功，且您已连接至 HSM。

```
Enter password:

loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)

aws-cloudhsm>
```

Example ：从 HSM 中注销

此命令从您当前登录到的 HSM 中注销（在这种情况下，是 `server 0` 和 `server 1`）。输出显示命令已成功，并且您已从 HSM 断开连接。

```
aws-cloudhsm>logoutHSM

logoutHSM success on server 0(10.0.2.9)
logoutHSM success on server 1(10.0.3.11)
```

参数

按照语法图表中指定的顺序输入参数。使用 `-hpswd` 参数来掩盖您的密码。要使用双重身份验证 (2FA) 登录，请使用 `-2fa` 参数并纳入文件路径。有关使用 2FA 的更多信息，请参阅 使用 CMU 管理 2FA。

```
loginHSM <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<user type>

指定登录到 HSM 的用户的类型。有关更多信息，请参阅上面的用户类型。

必需：是

<user name>

指定登录到 HSM 的用户的用户名。

必需：是

<password | -hpswd >

指定登录到 HSM 的用户的密码。若要隐藏密码，请使用 `-hpswd` 参数代替密码，然后按照提示进行操作。

必需：是

[-2fa </path/to/authdata>]

指定系统应使用第二项因素对此 2FA CO 用户进行身份验证。要获取使用 2FA 登录所需的数据，请在 `-2fa` 参数后面添加文件名，以指向文件系统中的某个位置的路径。有关使用 2FA 的更多信息，请参阅 使用 CMU 管理 2FA。

必需：否

相关 主题

- cloudhsm_mgmt_util 入门
- 激活集群

registerQuorumPub钥匙

cloudhsm_mgmt_util 中的 registerQuorumPubKey 命令将硬件安全模块 (HSM) 用户与非对称 RSA-2048 密钥对关联。将 HSM 用户与密钥关联后，这些用户就可以使用私钥批准仲裁人数请求，集群可以使用注册的公钥验证签名是否来自该用户。有关仲裁身份验证的更多信息，请参阅管理仲裁身份验证（M of N 访问控制）。

> **Tip**
>
> 在 AWS CloudHSM 文档中，法定人数身份验证有时被称为 M of N (MoFN)，这意味着在总数 N 个批准者中，至少有 M 个批准者。

用户类型

以下类型的用户均可运行此命令。

- 加密员（CO）

语法

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

示例

此示例说明如何使用 registerQuorumPubKey 将加密员（CO）注册为仲裁身份验证请求的批准者。若要运行此命令，您必须拥有非对称 RSA-2048 密钥对、已签名令牌和未签名令牌。有关要求的更多信息，请参阅 the section called “参数”。

Example : 注册 HSM 用户以进行法定人数身份验证

此示例将名为 `quorum_officer` 的 CO 注册为仲裁身份验证批准者。

```
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)
```

最后一个命令使用 listUsers 命令验证 quorum_officer 是否以 MofN 用户身份注册。

```
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
```

参数

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-
token> <public-key>
```

<user-type>

指定用户类型。此参数为必需参数。

有关 HSM 上的用户类型的详细信息，请参阅 了解 HSM 用户。

有效值：

- CO：加密员可以管理用户，但无法管理密钥。

必需：是

<user-name>

为用户指定友好名称。最大长度为 31 个字符。唯一允许的特殊字符是下划线 (_)。

用户名在创建之后无法更改。在 cloudhsm_mgmt_util 命令中，用户类型和密码都区分大小写，但用户名不区分大小写。

必需：是

<registration-token>

指定包含未签名注册令牌的文件路径。可以有最大文件大小为 245 字节的任何随机数据。有关创建未签名注册令牌的更多信息，请参阅创建和签名注册令牌。

必需：是

<signed-registration-token>

指定包含注册令牌 SHA256_PKCS 机制签名哈希的文件路径。有关更多信息，请参阅创建和签名注册令牌。

必需：是

<public-key>

指定包含非对称 RSA-2048 密钥对公钥的文件路径。使用私有密钥对注册令牌进行签名。有关更多信息，请参阅创建 RSA 密钥对。

必需：是

> **Note**
>
> 集群使用相同的密钥进行仲裁身份验证和双重身份验证 (2FA, two-factor authentication)。这意味着您无法对使用 registerQuorumPubKey 进行双因素身份验证的用户轮换仲裁密钥。若要要轮换密钥，必须使用 changePswd。有关使用仲裁身份验证和双因素身份验证的更多信息，请参阅仲裁身份验证和双因素身份验证。

相关 主题

- 创建 RSA 密钥对

- 创建注册令牌并签名
- 通过 HSM 注册公钥
- 管理仲裁身份验证（M of N 访问控制）
- 仲裁身份验证和双因素身份验证
- listUsers

服务器

通常，如果发出 cloudhsm_mgmt_util 中的某个命令，则该命令会影响指定集群（全局模式）中的所有 HSM。但是，在某些情况下，您需要向单个 HSM 发布命令。例如，在自动同步失败的情况下，您可能需要同步 HSM 上的密钥和用户，以跨集群保持一致性。您可以使用 cloudhsm_mgmt_util 中的 server 命令进入服务器模式，并与特定 HSM 实例直接交互。

成功启动后，`aws-cloudhsm>` 命令提示符将替换为 `server>` 命令提示符。

为了退出服务器模式，请使用 `exit` 命令。成功退出后，您将返回到 cloudhsm_mgmt_util 命令提示符。

运行任何 cloudhsm_mgmt_util 命令前，必须启动 cloudhsm_mgmt_util。

用户类型

以下用户均可运行此命令。

- 所有用户。

先决条件

为了进入服务器模式，您必须首先知道目标 HSM 的服务器编号。服务器编号将在启动后在由 cloudhsm_mgmt_util 生成的跟踪输出中列出。服务器编号的分配顺序与 HSM 在配置文件中显示的顺序相同。在此示例中，我们假设 `server 0` 是对应于所需 HSM 的服务器。

语法

启动服务器模式：

```
server <server-number>
```

退出服务器模式：

```
server> exit
```

示例

此命令将进入 HSM 上的服务器模式，服务器编号为 0。

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

为了退出服务器模式，请使用 exit 命令。

```
server0> exit
```

参数

```
server <server-number>
```

<server-number>

指定目标 HSM 的服务器编号。

必需：是

没有 exit 命令的参数。

相关 主题

- syncKey
- createUser
- deleteUser

setAttribute

cloudhsm_mgmt_util 中的 setAttribute 命令更改 HSM 中的密钥的标签、加密、解密、包装和解开包装属性的值。您也可以使用 key_mgmt_util 中的 setAttribute 命令来将会话密钥转换为持久密钥。您只能更改自己拥有的密钥的属性。

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下用户均可运行此命令。

- 加密用户 (CU)

语法

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
setAttribute <key handle> <attribute id>
```

示例

此示例说明如何禁用对称密钥的解密功能。您可以使用与此类似的命令来配置包装密钥，该密钥应能够将其他密钥进行包装和解开包装，但无法加密或解密数据。

第一步是创建包装密钥。此命令genSymKey在 key_mgmt_util 中使用来生成 256 位 AES 对称密钥。输出显示新密钥具有密钥句柄 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
```

接下来，我们想要确认解密属性的当前值。要获取解密属性的属性 ID，请使用 listAttributes。输出显示表示 OBJ_ATTR_DECRYPT 属性的常量为 261。有关解释密钥属性的帮助，请参阅 密钥属性引用。

```
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
```

要获取密钥 14 的解密属性的当前值，请使用 cloudhsm_mgmt_util 中的 getAttribute。

输出显示解密属性的值在集群中的两个 HSM 上均为 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 将密钥 14 的解密属性（属性 261）的值更改为 0。这将禁用密钥上的解密功能。

输出显示该命令在集群中的两个 HSM 上已成功执行。

```
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)
```

最后一个命令重复执行 getAttribute 命令。同样，它会获得密钥 14 的解密属性 (属性 261)。

此时，输出显示解密属性的值在两个 HSM 上均为 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
```

参数

```
setAttribute <key handle> <attribute id>
```

<key-handle>

指定您拥有的密钥的密钥句柄。您在每个命令中只能指定一个密钥。要获取密钥的密钥句柄，请使用 key_mgmt_util 中的 findKey。要查找密钥的用户，请使用getKeyInfo。

必需：是

<attribute id>

指定表示您要更改的属性的常量。您在每个命令中只能指定一个属性。要获取属性及其整数值，请使用 listAttributes。有关解释密钥属性的帮助，请参阅 密钥属性引用。

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。

必需：是

相关 主题

- key_mgmt_util 中的 setAttribute
- getAttribute
- listAttributes
- 密钥属性引用

退出

cloudhsm_mgmt_util 中的 quit 命令可退出 cloudhsm_mgmt_util。任何类型的任何用户都可以使用此命令。

运行任何 cloudhsm_mgmt_util 命令前，必须启动 cloudhsm_mgmt_util。

用户类型

以下用户均可运行此命令。

- 所有用户。您不必登录即可运行此命令。

语法

```
quit
```

示例

此命令将退出 cloudhsm_mgmt_util。成功完成后，您将返回到您的常规命令行。此命令没有输出参数。

```
aws-cloudhsm> quit

disconnecting from servers, please wait...
```

相关 主题

- cloudhsm_mgmt_util 入门

shareKey

cloudhsm_mgmt_util 中的 shareKey 命令将与其他加密用户共享和取消共享您拥有的密钥。只有密钥所有者才能共享和取消共享密钥。您还可以在创建密钥时共享它。

共享密钥的用户可在加密操作中使用密钥，但他们无法删除、导出、共享或取消共享密钥，也无法更改密钥的属性。对某个密钥启用仲裁身份验证后，仲裁必须对共享或取消共享该密钥的任何操作进行审批。

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下类型的用户均可运行此命令。

- 加密用户 (CU)

语法

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

用户类型：加密用户 (CU)

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

示例

以下示例说明如何使用 shareKey 与其他加密用户共享和取消共享您拥有的密钥。

Example ：共享密钥

本示例使用 shareKey 与 HSM 上的另一个加密用户共享当前用户拥有的 ECC 私有密钥。公有密钥对 HSM 的所有用户都可用，因此您无法共享或取消共享它们。

第一个命令getKeyInfo用于获取密262177钥（HSM 上的 ECC 私钥）的用户信息。

输出表明密钥 262177 由用户 3 拥有但未共享。

```
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 与用户 4（HSM 上的另一个加密用户）共享密钥 262177。最后一个参数使用值 1 来指示共享操作。

输出表明该操作在集群中的两个 HSM 上已成功执行。

```
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)
```

为了验证该操作是否成功，本示例再次执行了第一个 getKeyInfo 命令。

输出表明密钥 262177 现在与用户 4 共享。

```
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 ：取消共享密钥

本示例将取消共享一个对称密钥，即从该密钥的共享用户列表中删除某个加密用户。

此命令使用 shareKey 从密钥 6 的共享用户列表中删除用户 4。最后一个参数使用值 0 来指示取消共享操作。

输出表明命令在两个 HSM 上都已成功执行。因此，用户 4 无法再在加密操作中使用密钥 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)
```

参数

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

<key-handle>

指定您拥有的密钥的密钥句柄。您在每个命令中只能指定一个密钥。要获取密钥的密钥句柄，请使用 key_mgmt_util 中的 findKey。要验证您是否拥有密钥，请使用getKeyInfo。

必需：是

<user id>

指定您要与之共享或取消共享密钥的加密用户 (CU) 的用户 ID。要查找某个用户的用户 ID，请使用 listUsers。

必需：是

<share 1 or unshare 0>

要与指定用户共享密钥，请键入 1。要取消共享密钥，即从密钥的共享用户列表中删除指定用户，请键入 0。

必需：是

相关 主题

- getKeyInfo

syncKey

您可以使用 cloudhsm_mgmt_util 中的 syncKey 命令可跨集群中的 HSM 实例或跨克隆的集群同步密钥。通常，您不需要使用此命令，因为 HSM 集群中的实例会自动同步密钥。但是，跨克隆集群的密钥同步必须手动完成。为了简化全球扩展和灾难恢复流程，克隆集群通常在不同的 AWS 区域创建。

您不能使用 syncKey 跨任意集群同步密钥：一个集群必须已从其他集群的备份中创建。此外，这两个集群必须具有一致的 CO 和 CU 凭证，操作才能成功。有关更多信息，请参阅 HSM 用户。

要使用syncKey，必须先创建一个 AWS CloudHSM 配置文件，指定源集群中的一个 HSM 和目标集群中的一个 HSM。这将允许 cloudhsm_mgmt_util 连接到这两个 HSM 实例。使用此配置文件启动 cloudhsm_mgmt_util。然后，使用拥有您要同步的密钥的 CO 或 CU 的凭证登录。

用户类型

以下类型的用户均可运行此命令。

- 加密员（CO）
- 加密用户 (CU)

> **Note**
>
> CO 可以在任何密钥上使用 syncKey，而 CU 只能在自己的密钥上使用此命令。有关更多信息，请参阅 the section called “了解 HSM 用户”。

先决条件

在开始之前，您必须知道要与目标 HSM 密钥同步的源 HSM 上的密钥的 `key handle`。要查找 `key handle`，请使用 listUsers 命令列出命名用户的所有标识符。然后，使用findAllKeys命令查找属于特定用户的所有密钥。

您还需要知道分配给源和目标 HSM 的 `server IDs`，这些值显示在 cloudhsm_mgmt_util 启动时返回的跟踪输出中。这些分配的顺序与 HSM 在配置文件中显示的顺序相同。

按照跨克隆的集群使用 CMU Across Cloned Clusters 中的说明进行操作，并使用新的配置文件初始化 cloudhsm_mgmt_util。然后，通过发出 server 命令在源 HSM 上进入服务器模式。

语法

> **Note**
>
> 要运行 syncKey，请先在 HSM 上进入服务器模式，其中包含要同步的密钥。

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

用户类型：加密用户 (CU)

```
syncKey <key handle> <destination hsm>
```

示例

运行 server 命令登录到源 HSM，并进入服务器模式。对于此示例，我们假设 `server 0` 是源 HSM。

```
aws-cloudhsm> server 0
```

现在运行 syncKey 命令。在本示例中，我们假定密钥 261251 将同步到 `server 1`。

```
aws-cloudhsm> syncKey 261251 1
syncKey success
```

参数

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
syncKey <key handle> <destination hsm>
```

<key handle>

指定要同步的密钥的密钥句柄。您在每个命令中只能指定一个密钥。要获取密钥的密钥句柄，请在登录 HSM 服务器findAllKeys时使用。

必需：是

<destination hsm>

指定要同步密钥的服务器的编号。

必需：是

相关 主题

- listUsers
- findAllKeys
- 描述中的集群 AWS CLI
- 服务器

syncUser

您可以使用 cloudhsm_mgmt_util 中的syncUser命令在集群内的 HSM 实例之间或克隆的集群之间手动同步加密用户 (CU) 或加密官员 (CO)。 AWS CloudHSM 不会自动同步用户。通常，您以全局模式管理用户，以便集群中的所有 HSM 一起更新。如果 HSM 意外不同步（例如，由于密码更改）或者您希望在克隆的集群之间轮换用户凭证，则可能需要使用 syncUser。克隆集群通常在不同的 AWS 区域创建，以简化全球扩展和灾难恢复流程。

在运行任何 CMU 命令之前，必须启动 CMU 并登录 HSM。请确保使用可运行您计划使用的命令的用户类型登录。

如果您要添加或删除 HSM，请更新 CMU 的配置文件。否则，您所做的更改可能不会对集群中的所有 HSM 生效。

用户类型

以下类型的用户均可运行此命令。

- 加密员（CO）

先决条件

在开始之前，您必须知道要与目标 HSM 同步的源 HSM 上的用户的 `user ID`。要查找 `user ID`，请使用 listUsers 命令以列出集群中的 HSM 上的所有用户。

您还需要知道分配给源和目标 HSM 的 `server ID`，这些值显示在 cloudhsm_mgmt_util 启动时返回的跟踪输出中。这些分配的顺序与 HSM 在配置文件中显示的顺序相同。

如果要跨克隆的集群同步 HSM，请按照跨克隆的集群使用 CMU Across Cloned Clusters 中的说明进行操作，并使用新的配置文件初始化 cloudhsm_mgmt_util。

当您准备好运行 syncUser 时，通过发出 server 命令在源 HSM 上进入服务器模式。

语法

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
syncUser <user ID> <server ID>
```

示例

运行 server 命令登录到源 HSM，并进入服务器模式。对于此示例，我们假设 `server 0` 是源 HSM。

```
aws-cloudhsm> server 0
```

立即运行 syncUser 命令。对于此示例，我们假定用户 6 是要同步的用户，`server 1` 是目标 HSM。

```
server 0> syncUser 6 1
ExtractMaskedObject: 0x0 !
InsertMaskedObject: 0x0 !
syncUser success
```

参数

由于此命令没有命名参数，因此您必须按语法图中指定的顺序输入参数。

```
syncUser <user ID> <server ID>
```

<用户 ID>

指定要同步的用户的 ID。您在每个命令中只能指定一个用户。要获取用户的 ID，请使用 listUsers。

必需：是

<服务器 ID>

指定要将用户同步到的 HSM 的服务器编号。

必需：是

相关 主题

- listUsers
- 描述中的集群 AWS CLI
- 服务器

密钥管理实用程序（KMU）

密钥管理实用程序（KMU）是一种命令行工具，可帮助加密用户（CU）管理硬件安全模块（HSM）上的密钥。KMU 包括多个命令，这些命令用于生成、删除、导入和导出密钥、获取和设置属性、查找密钥以及执行加密操作。

KMU 和 CMU 是客户端软件开发工具包 3 套件的一部分。

要快速开始使用，请参阅 key_mgmt_util 入门。有关这些命令的详细信息，请参阅 key_mgmt_util 命令引用。有关解释密钥属性的帮助，请参阅 密钥属性引用。

如果您使用的是 Linux，要使用 key_mgmt_util，请连接到您的客户端实例，然后参见安装和配置 AWS CloudHSM 客户端 (Linux)。如果您使用的是 Windows，请参见安装和配置 AWS CloudHSM 客户端 (Windows)。

主题

- key_mgmt_util 入门
- 安装和配置 AWS CloudHSM 客户端 (Linux)
- 安装和配置 AWS CloudHSM 客户端 (Windows)
- key_mgmt_util 命令引用

key_mgmt_util 入门

AWS CloudHSM 包括两个带有AWS CloudHSM 客户端软件的命令行工具。cloudhsm_mgmt_util 工具包括管理 HSM 用户的命令。key_mgmt_util 工具包含管理密钥的命令。要开始使用 key_mgmt_util 命令行工具，请参阅以下主题。

主题

- 设置 key_mgmt_util
- key_mgmt_util 的基本用法

如果命令出现错误消息或意外结果，请参见故障排除 AWS CloudHSM主题获取帮助。有关 key_mgmt_util 命令的详细信息，请参阅 key_mgmt_util 命令引用。

设置 key_mgmt_util

在使用 key_mgmt_util 之前，请完成以下设置。

启动 AWS CloudHSM 客户端

在使用 key_mgmt_util 之前，必须启动客户端。 AWS CloudHSM 客户端是一个守护程序，用于与集群中的 HSM 建立 end-to-end 加密通信。key_mgmt_util 工具使用客户端连接与群集中的 HSM 进行通信。如果没有此连接，key_mgmt_util 将不起作用。

启动 AWS CloudHSM 客户端

使用以下命令启动 AWS CloudHSM 客户端。

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

- 对于 Windows 客户端 1.1.2 以上版本：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 对于 Windows 客户端 1.1.1 及更低版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

启动 key_mgmt_util

启动 AWS CloudHSM 客户端后，使用以下命令启动 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
```

当 key_mgmt_util 运行时，提示符将变为 `Command:`。

如果此命令失败 (如返回 `Daemon socket connection error` 消息)，请尝试更新您的配置文件。

key_mgmt_util 的基本用法

请参阅以下主题以了解 key_mgmt_util 工具的基本用法。

主题

- 登录到 HSM
- 从 HSM 退出
- 停止 key_mgmt_util

登录到 HSM

使用 loginHSM 命令登录到 HSM。以下命令以名为 `example_user` 的加密用户（CU）身份登录。输出指明针对群集中的所有三个 HSM 的成功登录。

```
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
```

下面说明 loginHSM 命令的语法。

```
Command:  loginHSM -u <USER TYPE> -s <USERNAME> -p <PASSWORD>
```

从 HSM 退出

使用 logoutHSM 命令可从 HSM 注销。

```
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 命令停止 key_mgmt_util。

```
Command:  exit
```

安装和配置 AWS CloudHSM 客户端 (Linux)

要与 AWS CloudHSM 集群中的 HSM 进行交互，您需要适用于 Linux 的 AWS CloudHSM 客户端软件。您应在之前创建的 Linux EC2 客户端实例上安装该软件。如果您使用的是 Windows，也可以安装客户端。有关更多信息，请参阅 安装和配置 AWS CloudHSM 客户端 (Windows)。

任务

- 安装 AWS CloudHSM 客户端和命令行工具

- 编辑客户端配置

安装 AWS CloudHSM 客户端和命令行工具

连接到您的客户端实例并运行以下命令来下载和安装 AWS CloudHSM 客户端和命令行工具。

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
```

编辑客户端配置

必须先编辑 AWS CloudHSM 客户端配置，然后才能使用客户端连接到集群。

编辑客户端配置

1. 将您的颁发证书 — 用于签署集群证书的证书 — 复制到客户端实例上的以下位置：/opt/cloudhsm/etc/customerCA.crt。您需要在客户端实例上具有实例根用户权限才能将您的证书复制到该位置。
2. 使用以下 configure 命令更新 AWS CloudHSM 客户端和命令行工具的配置文件，指定集群中 HSM 的 IP 地址。要获取 HSM 的 IP 地址，请在AWS CloudHSM 控制台中查看您的集群，或者运行describe-clusters AWS CLI 命令。在命令输出中，HSM 的 IP 地址为 EniIp 字段的值。如果您有多个 HSM，请选择其中任一 HSM 的 IP 地址；选择哪个 HSM 并不重要。

```
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. 转到 激活集群。

安装和配置 AWS CloudHSM 客户端 (Windows)

要在 Windows 上使用 AWS CloudHSM 集群中的 HSM，你需要适用于 Windows 的 AWS CloudHSM 客户端软件。您应在之前创建的 Windows Server 实例上安装它。

安装（或更新）最新的 Windows 客户端和命令行工具

1. 连接到您的 Windows Server 实例。
2. 从下载页面下载最新AWSCloudHSMClient版本 (-latest.msi)。
3. 前往您的下载位置并使用管理权限运行安装程序 (AWSCloudHSMClient-latest.msi)。
4. 按照安装程序说明进行操作，然后在安装程序完成后选择关闭。
5. 将您的自签名颁发证书 — 用于签署集群证书的证书 — 复制到 C:\ProgramData\Amazon\CloudHSM 文件夹。
6. 运行以下命令以更新您的配置文件。如果要更新，在重新配置期间务必停止并启动客户端：

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

7. 转到 激活集群。

备注：

- 如果更新客户端，以前安装的现有配置文件不会被覆盖。
- 适用于 Windows 的 AWS CloudHSM 客户端安装程序会自动注册加密 API：下一代 (CNG) 和密钥存储提供程序 (KSP)。要卸载客户端，请再次运行安装程序并按照卸载说明操作。
- 如果您使用的是 Linux，也可以安装 Linux 客户端。有关更多信息，请参阅 安装和配置 AWS CloudHSM 客户端 (Linux)。

key_mgmt_util 命令引用

key_mgmt_util 命令行工具可帮助您在集群的 HSM 中管理密钥，包括创建、删除和查找密钥及其属性。它包括多个命令，其中每个命令在本主题中有详细介绍。

要快速开始使用，请参阅 key_mgmt_util 入门。有关解释密钥属性的帮助，请参阅 密钥属性引用。有关 cloudhsm_mgmt_util 命令行工具的信息，其中包括在您的集群中管理 HSM 和用户的命令，请参阅 CloudHSM 管理实用程序 (CMU)。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

要列出所有 key_mgmt_util 命令，请键入：

```
Command: help
```

要获取某个 key_mgmt_util 命令的帮助，请键入：

```
Command: <command-name> -h
```

要结束您的 key_mgmt_util 会话，请键入：

```
Command: exit
```

以下主题介绍了 key_mgmt_util 中的命令。

> **Note**
>
> key_mgmt_util 和 cloudhsm_mgmt_util 中的部分命令名称相同。但是，这些命令通常具有不同的语法、不同的输出和轻微不同的功能。

命令	描述
aesWrapUnwrap	加密和解密文件中的密钥的内容。
deleteKey	从 HSM 中删除密钥。
Error2String	获取对应于 key_mgmt_util 十六进制错误代码的错误。
exit	退出 key_mgmt_util。
exportPrivateKey	将私有密钥的副本从 HSM 导出到磁盘上的文件。
exportPubKey	将公有密钥的副本从 HSM 导出到文件中。
exSymKey	将对称密钥的明文副本从 HSM 导出到文件中。
extractMaskedObject	从 HSM 提取密钥作为遮蔽对象文件。
findKey	按密钥属性值搜索密钥。
findSingleKey	验证密钥是否存在于集群中的所有 HSM 上。
GendSA KeyPair	在您的 HSM 中生成数字签名算法 (DSA) 密钥对。
GenecC KeyPair	在您的 HSM 中生成椭圆曲线加密 (ECC) 密钥对。
GenrSA KeyPair	在您的 HSM 中生成 RSA 非对称密钥对。
genSymKey	在您的 HSM 中生成对称密钥
getAttribute	获取 AWS CloudHSM 密钥的属性值并将其写入文件中。
getCaviumPriv钥匙	创建私有密钥的伪造 PEM 格式版本，并将其导出到文件中。
getCert	检索 HSM 的分区证书并将其保存到文件中。

命令	描述
getKeyInfo	获取可使用该密钥的用户的 HSM 用户 ID。 如果密钥是仲裁控制型的，则它会得到仲裁用户的数量。
help	显示 key_mgmt_util 中有关命令的可用帮助信息。
importPrivateKey	将私有密钥导入到 HSM 中。
importPubKey	将公有密钥导入到 HSM 中。
imSymKey	将对称密钥的明文副本从文件导入 HSM 中。
insertMaskedObject	将磁盘上文件中的遮蔽对象插入到相关集群包含的 HSM 中，并将其插入到对象的原始集群中。相关集群是从原始集群备份生成的任意集群。
???	确定给定文件是包含一个真实的私有密钥，还是伪造的 PEM 密钥。
listAttributes	列出 AWS CloudHSM 密钥的属性以及代表它们的常量。
listUsers	获取 HSM 中的用户、用户类型和 ID 以及其他属性。
loginHSM 和 logoutHSM	登录到集群中的 HSM 和从其中注销。
setAttribute	将会话密钥转换为持久密钥。
sign	使用选定的私有密钥生成文件的签名。
unWrapKey	将已包装 (已加密) 密钥从文件导入 HSM 中。
verify	验证给定密钥是否用于对给定文件签名。
wrapKey	将密钥的加密副本从 HSM 导出到文件中。

aesWrapUnwrap

aesWrapUnwrap 命令可为磁盘上的文件内容加密或解密。此命令可用于对加密密钥进行包装和解开包装，但您可对包含小于 4 KB (4096 字节) 数据的任何文件使用它。

aesWrapUnwrap 使用 AES 密钥包装。它使用 HSM 上的 AES 密钥作为包装密钥或解开包装密钥。然后，它将结果写入到磁盘上的另一个文件中。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
aesWrapUnwrap -h

aesWrapUnwrap -m <wrap-unwrap mode>
              -f <file-to-wrap-unwrap>
              -w <wrapping-key-handle>
              [-i <wrapping-IV>]
              [-out <output-file>]
```

示例

这些示例演示如何使用 aesWrapUnwrap 加密和解密文件中的加密密钥。

Example ：包装加密密钥

此命令使用 aesWrapUnwrap 包装三重 DES 对称密钥（以明文形式从 HSM 导出至 3DES.key 文件）。您可以使用类似命令包装保存在文件中的任何密钥。

此命令使用值为 1 的 -m 参数来指示包装模式。它使用 -w 参数指定 HSM 中的 AES 密钥 (密钥句柄 6) 作为包装密钥。它将生成的包装密钥写入到 3DES.key.wrapped 文件。

输出显示，此命令已成功运行并且操作使用了默认 IV (首选设置)。

```
 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 ：将加密密钥解开包装

此示例演示如何使用 aesWrapUnwrap 将文件中的已包装（已加密）密钥解开包装（解密）。在将密钥导入到 HSM 中之前，您可能需要执行与此类似的操作。例如，如果您尝试使用imSymKey命令导入加密密钥，则会返回错误，因为加密密钥的格式不符合该类型的纯文本密钥所需的格式。

此命令将 3DES.key.wrapped 文件中的密钥解开包装并将明文写入到 3DES.key.unwrapped 文件。此命令使用值为 0 的 -m 参数指示解开包装模式。它使用 -w 参数指定 HSM 中的 AES 密钥 (密钥句柄 6) 作为包装密钥。它将生成的包装密钥写入到 3DES.key.unwrapped 文件。

```
 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
```

参数

-h

显示该命令的帮助信息。

必需：是

-m

指定模式。要包装 (加密) 文件内容，键入 1；要将文件内容进行解开包装 (解密)，键入 0。

必需：是

-f

指定要包装的文件。输入一个包含小于 4 KB (4096 字节) 数据的文件。此操作可用于将加密密钥进行包装和解开包装。

必需：是

-w

指定包装密钥。在 HSM 上输入 AES 密钥的密钥句柄。此参数为必需参数。要查找密钥句柄，请使用 findKey 命令。

要创建包装密钥，genSymKey请使用生成 AES 密钥（类型 31）。

必需：是

-i

指定算法的备用初始值 (IV)。除非您有需要备用值的特殊条件，否则请使用默认值。

默认值：0xA6A6A6A6A6A6A6A6。默认值是在 AES 密钥包装算法规范中定义的。

必需：否

-out

为包含已包装或已解开包装密钥的输出文件指定备用名称。本地目录中的默认名称为 wrapped_key (对于包装操作) 和 unwrapped_key (对于解开包装操作)。

如果该文件存在，则 aesWrapUnwrap 将覆盖该文件而不发出警告。如果此命令失败，则 aesWrapUnwrap 将创建一个无内容的输出文件。

默认名称：wrapped_key (对于包装)。unwrapped_key (对于解开包装)。

必需：否

相关 主题

- exSymKey
- imSymKey
- unWrapKey

- wrapKey

deleteKey

key_mgmt_util 中的 deleteKey 命令可从 HSM 中删除密钥。您一次只能删除一个密钥。删除某密钥对中的一个密钥不会影响该密钥对中的另一个密钥。

只有密钥所有者才能删除密钥。共享密钥的用户可以在加密操作中使用密钥，但无法将其删除。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
deleteKey -h

deleteKey -k
```

示例

这些示例演示如何使用 deleteKey 从您的 HSM 中删除密钥。

Example ：删除密钥

此命令删除密钥句柄为 6 的密钥。当该命令成功时，deleteKey 将从集群中的每个 HSM 返回成功消息。

```
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 ：删除密钥 (失败)

当由于密钥没有指定的密钥句柄导致该命令失败时，deleteKey 将返回对象句柄无效的错误消息。

```
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
```

当由于当前用户不是密钥的所有者导致该命令失败时，该命令会返回拒绝访问错误。

```
Command:  deleteKey -k 262152

Cfm3DeleteKey returned: 0xc6 : HSM Error: Key Access is denied.
```

参数

-h

显示该命令的命令行帮助。

必需：是

-k

指定要删除的密钥的密钥句柄。要在 HSM 中查找密钥的密钥句柄，请使用 findKey。

必需：是

相关 主题

- findKey

Error2String

Error2String 中的 key_mgmt_util 辅助标记命令返回对应于 key_mgmt_util 十六进制错误代码的错误。您可以在对您的命令和脚本进行故障排除时使用此命令。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
Error2String -h

Error2String -r <response-code>
```

示例

这些示例演示如何使用 Error2String 获取 key_mgmt_util 错误代码的错误字符串。

Example ：获取错误描述

此命令获取 0xdb 错误代码的错误描述。该描述说明用户尝试登录 key_mgmt_util 失败是由于用户类型错误。仅加密用户 (CU) 可以登录。

```
        Command:  Error2String -r 0xdb

        Error Code db maps to HSM Error: Invalid User Type.
```

Example ：查找错误代码

此示例显示要在 key_mgmt_util 错误中查找错误代码的位置。错误代码 0xc6 显示在字符串 Cfm3*command-name* returned: 之后。

在此示例中，getKeyInfo表示当前用户（用户 4）可以在加密操作中使用该密钥。然而，当用户尝试使用 deleteKey 删除该密钥时，该命令会返回错误代码 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 错误，则可以使用与此类似的 Error2String 命令来查找该错误。在这种情况下，由于该密钥与当前用户共享但由另一个用户所有，因此 deleteKey 命令失败，出现拒绝访问错误。只有密钥所有者才有删除密钥的权限。

```
        Command:  Error2String -r 0xa8

        Error Code c6 maps to HSM Error: Key Access is denied
```

参数

-h

显示该命令的帮助信息。

必需：是

-r

指定十六进制错误代码。0x 十六进制指示符是必需的。

必需：是

exit

key_mgmt_util 中的 exit 命令可退出 key_mgmt_util。成功退出后，您将返回到标准命令行。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util。

语法

```
exit
```

参数

此命令无任何参数。

相关 主题

- 启动 key_mgmt_util

exportPrivateKey

key_mgmt_util 中的 exportPrivateKey 命令将 HSM 中非对称私有密钥导出到文件中。HSM 不允许以明文形式直接导出密钥。该命令使用您指定的 AES 包装私钥，解密包装后的字节，然后将明文私钥复制到文件中。

exportPrivateKey 命令不会从 HSM 中删除此密钥，不会更改其密钥属性，也不会阻止您将密钥用于将来的加密操作。您可以多次导出相同的密钥。

您只能导出 OBJ_ATTR_EXTRACTABLE 属性值为 1 的私有密钥。必须指定具有 OBJ_ATTR_WRAP 和 OBJ_ATTR_DECRYPT 属性值 1 的 AES 包装密钥。要查找密钥的属性，请使用 getAttribute 命令。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
exportPrivateKey -h

exportPrivateKey -k <private-key-handle
                 -w <wrapping-key-handle>
                 -out <key-file>
                 [-m <wrapping-mechanism>]
                 [-wk <wrapping-key-file>]
```

示例

此示例显示了如何使用 exportPrivateKey 将私有密钥导出 HSM。

Example ：导出私有密钥

此命令使用具有 16 句柄的包装密钥，将具有 15 句柄的私有密钥导出到名为 exportKey.pem 的 PEM 文件。当该命令成功时，exportPrivateKey 将返回成功消息。

```
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
```

参数

此命令采用以下参数。

-h

显示该命令的命令行帮助。

必需：是

-k

指定要导出的私有密钥的密钥句柄。

必需：是

-w

指定包装密钥的密钥句柄。此参数为必需参数。要查找密钥句柄，请使用 findKey 命令。

要确定密钥是否可用作包装密钥，请使用 getAttribute 获取 OBJ_ATTR_WRAP 属性 (262) 的值。要创建包装密钥，请使用 genSymKey 创建 AES 密钥（类型 31）。

如果您使用 -wk 参数指定外部解开包装密钥，则 -w 包装密钥用于在导出过程中包装密钥而不是将其解开包装。

必需：是

-out

指定要将导出的私有密钥写入的文件的名称。

必需：是

-m

指定包装机制来包装正在导出的私有密钥。唯一有效值为 4，它表示 NIST_AES_WRAP mechanism.

默认值：4 (NIST_AES_WRAP)

必需：否

-wk

指定用于解开正在导出的密钥包装的密钥。输入包含明文 AES 密钥的文件的路径和名称。

当您包含此参数时，exportPrivateKey 使用 -w 文件中的密钥包装导出的密钥，并使用由 -wk 参数指定的密钥对导入的密钥解开包装。

默认值：使用 -w 参数中指定的包装密钥包装和解开包装。

必需：否

相关 主题

- importPrivateKey
- wrapKey
- unWrapKey
- genSymKey

exportPubKey

key_mgmt_util 中的 exportPubKey 命令将 HSM 中的公有密钥导出到一个文件。您可以使用它来导出 HSM 中生成的公有密钥。您还可以使用此命令来导出已经导入到 HSM 的公有密钥，如使用 importPubKey 命令导入的公有密钥。

此 exportPubKey 操作会将密钥资料复制到指定的文件。但它不会从 HSM 中删除此密钥，不会更改其密钥属性，也不会阻止您将密钥用于将来的加密操作。您可以多次导出相同的密钥。

您只能导出 OBJ_ATTR_EXTRACTABLE 值为 1 的公有密钥。要查找密钥的属性，请使用 getAttribute 命令。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户 (CU) 身份登录到 HSM。

语法

```
exportPubKey -h

exportPubKey -k <public-key-handle
             -out <key-file>
```

示例

此示例显示了如何使用 exportPubKey 将公有密钥从 HSM 导出。

Example ：导出公有密钥

此命令将具有句柄 10 的公有密钥导出到名为 public.pem 的文件中。当该命令成功时，exportPubKey 将返回成功消息。

```
Command: exportPubKey -k 10 -out public.pem

PEM formatted public key is written to public.pem

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

参数

此命令采用以下参数。

-h

显示该命令的命令行帮助。

必需：是

-k

指定要导出的公有密钥的密钥句柄。

必需：是

-out

指定要将导出的公有密钥写入的文件的名称。

必需：是

相关 主题

- importPubKey
- 生成密钥

exSymKey

key_mgmt_util 工具中的 exSymKey 命令从 HSM 中导出对称密钥的明文副本并将其保存在磁盘上的文件中。要导出密钥的加密 (已包装) 副本，请使用 wrapKey。要导入纯文本密钥（如导出的密钥），请exSymKey使用。imSymKey

在导出过程中，exSymKey 使用您指定的 AES 密钥（包装密钥）进行包装（加密），然后将要导出的密钥解开包装（解密）。然而，导出操作的结果是磁盘上的明文 (解开包装) 密钥。

只有密钥的所有者 (即创建该密钥的 CU 用户) 才能导出它。共享密钥的用户可以在加密操作中使用密钥，但无法导出它。

exSymKey 操作将密钥材料复制到您指定的文件，但它不会从 HSM 中删除密钥，不会更改其密钥属性，也不会阻止您在加密操作中使用密钥。您可以多次导出相同的密钥。

exSymKey 仅导出对称密钥。要导出公有密钥，请使用 exportPubKey。要导出私有密钥，请使用 exportPrivateKey。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
exSymKey -h

exSymKey -k <key-to-export>
         -w <wrapping-key>
         -out <key-file>
         [-m 4]
         [-wk <unwrapping-key-file> ]
```

示例

这些示例演示如何使用 exSymKey 从您的 HSM 中导出您拥有的对称密钥。

Example ：导出 3DES 对称密钥

此命令将导出三重 DES (3DES) 对称密钥 (密钥句柄 7)。它使用 HSM 中的现有 AES 密钥 (密钥句柄 6) 作为包装密钥。然后，它将 3DES 明文密钥写入 3DES.key 文件。

输出显示密钥 7 (3DES 密钥) 已成功包装和解开包装，然后写入 3DES.key 文件。

> **Warning**
>
> 尽管输出表明“包装的对称密钥”已写入输出文件，但输出文件包含明文 (解开包装的) 密钥。

```
        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 ：导出仅会话型包装密钥

此示例显示如何使用仅存在于会话中的密钥作为包装密钥。由于要导出的密钥会进行包装、立即解开包装并以明文形式提供，因此无需保留包装密钥。

此系列命令从 HSM 中导出密钥句柄为 8 的 AES 密钥。它使用专为该用途创建的 AES 会话密钥。

第一个命令用于genSymKey创建 256 位 AES 密钥。它使用 `-sess` 参数创建仅在当前会话中存在的密钥。

输出显示 HSM 创建密钥 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
```

然后，此示例验证密钥 8 (要导出的密钥) 是否为可提取的对称密钥。它还会验证包装密钥 (密钥 262168) 是否为仅在会话中存在的 AES 密钥。您可以使用 findKey 命令，但此示例会将两个密钥的属性都导出到文件，然后使用 `grep` 在文件中查找相关属性值。

这些命令使用 `-a` 值为 512 (全部) 的 `getAttribute` 来获取密钥 8 和 262168 的所有属性。有关密钥属性的信息，请参阅the section called “密钥属性引用”。

```
getAttribute -o 8 -a 512 -out attributes/attr_8
getAttribute -o 262168 -a 512 -out attributes/attr_262168
```

这些命令使用 grep 验证要导出的密钥 (密钥 8) 和仅会话型包装密钥 (密钥 262168) 的属性。

```
    // 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
```

最后，我们使用 exSymKey 命令通过使用会话密钥（密钥 262168）作为包装密钥来导出密钥 8。

当会话结束时，密钥 262168 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 ：使用外部解开包装密钥

此示例显示如何使用外部解开包装密钥来从 HSM 中导出密钥。

当从 HSM 中导出密钥时，您可在 HSM 上指定要成为包装密钥的 AES 密钥。默认情况下，包装密钥用于将要导出的密钥进行包装和解开包装。但是，您可以使用 -wk 参数来告知 exSymKey 使用磁盘上文件中的外部密钥解开包装。执行此操作时，由 -w 参数指定的密钥会包装目标密钥，而由 -wk 参数指定的文件中的密钥将为该密钥解开包装。

由于包装密钥必须是 AES 对称密钥，因此 HSM 中的包装密钥和磁盘上的解开包装密钥必须具有相同的密钥材料。为此，您必须将包装密钥导入 HSM 或在导出操作之前从 HSM 中导出包装密钥。

此示例在 HSM 外部创建密钥并将其导入到 HSM 中。它使用密钥的内部副本将要导出的对称密钥进行包装，然后使用文件中的密钥副本将其解开包装。

第一个命令使用 OpenSSL 生成 256 位 AES 密钥。它将密钥保存到 aes256-forImport.key 文件。OpenSSL 命令不会返回任何输出，但您可以使用多个命令来确认其是否运行成功。此示例使用了用于确认文件包含 32 字节数据的 wc (wordcount) 工具。

```
$  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命令将 AES 密钥从aes256-forImport.key文件导入 HSM。该命令完成后，密钥句柄为 262167 的密钥将存在于 HSM 和 aes256-forImport.key 文件中。

```
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
```

此命令在导出操作中使用该密钥。此命令使用 exSymKey 来导出密钥 21（192 位 AES 密钥）。为了包装该密钥，它会使用密钥 262167，这是已导入 HSM 中的副本。为了解开包装密钥，它会使用 aes256-forImport.key 文件中的相同密钥材料。该命令完成后，密钥 21 将会导出到 aes192_h21.key 文件。

```
        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"
```

参数

-h

显示该命令的帮助信息。

必需：是

-k

指定要导出的密钥的密钥句柄。此参数为必需参数。输入您拥有的对称密钥的密钥句柄。此参数为必需参数。要查找密钥句柄，请使用 findKey 命令。

要验证是否能导出密钥，请使用 getAttribute 命令获取由常量 354 表示的 OBJ_ATTR_EXTRACTABLE 属性的值。此外，您还只能导出您拥有的密钥。要查找密钥的所有者，请使用getKeyInfo命令。

必需：是

-w

指定包装密钥的密钥句柄。此参数为必需参数。要查找密钥句柄，请使用 findKey 命令。

包装密钥 是 HSM 中的密钥，用于对要导出的密钥进行加密 (包装) 和解密 (解开包装)。只有 AES 密钥才能用作包装密钥。

您可以使用任何 AES 密钥 (任何大小) 作为包装密钥。由于包装密钥将对目标密钥进行包装，然后紧接着将其解开包装，因此您可以使用仅会话型 AES 密钥作为包装密钥。要确定某个密钥是否可以用作包装密钥，请使用 getAttribute 获取 OBJ_ATTR_WRAP 属性的值，该值由常量 262 表示。要创建包装密钥，genSymKey请使用创建 AES 密钥（类型 31）。

如果您使用 -wk 参数指定外部解开包装密钥，则 -w 包装密钥用于在导出过程中包装密钥而不是将其解开包装。

> **Note**
>
> 密钥 4 表示不受支持的内部密钥。建议您使用作为包装密钥创建和管理的 AES 密钥。

必需：是

-out

指定输出文件的路径和名称。此命令成功后，此文件将包含以明文形式的已导出密钥。如果该文件已存在，则命令将覆盖该文件而不发出警告。

必需：是

-m

指定包装机制。唯一有效值为 4，它表示 NIST_AES_WRAP 机制。

必需：否

默认值：4

-wk

使用指定文件中的 AES 密钥将正在导出的密钥解开包装。输入包含明文 AES 密钥的文件的路径和名称。

当您包含此参数时。exSymKey 使用 HSM 中由 -w 参数指定的密钥包装正在导出的密钥，并且使用 -wk 文件中的密钥将其解开包装。-w 和 -wk 参数值必须解析为相同的明文密钥。

必需：否

默认：在 HSM 上使用包装密钥解开包装。

相关 主题

- genSymKey
- imSymKey
- wrapKey

extractMaskedObject

key_mgmt_util 中的 extractMaskedObject 命令从 HSM 提取密钥，并将其作为遮蔽对象另存到一个文件中。遮蔽对象是克隆 对象，只能在使用 insertMaskedObject 命令将它们插入回原始集群后才能使用。您只能将遮蔽对象插入到生成该对象的同一集群或克隆集群。这包括通过复制跨区域的备份生成的任何克隆版本的集群，并使用该备份来创建新的集群。

遮蔽对象是一种用来卸载和同步密钥的有效方式，包括不可提取密钥（即 OBJ_ATTR_EXTRACTABLE 值为 0 的密钥）。这样，无需更新 AWS CloudHSM 配置文件即可在不同区域的相关集群之间安全地同步密钥。

> **Important**
>
> 在插入时，解密遮蔽对象，并赋予与原始密钥的密钥句柄不同的密钥句柄。一个遮蔽对象包括与原始密钥关联的所有元数据，包含属性、所有权和共享信息，以及 quorum 设置。如果您需要在应用程序中跨集群同步密钥，可改用 cloudhsm_mgmt_util 中的 syncKey。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并登录到 HSM。此 extractMaskedObject 命令可通过拥有密钥的 CU 或任意 CO 使用。

语法

```
extractMaskedObject -h

extractMaskedObject -o <object-handle>
                    -out <object-file>
```

示例

此示例显示了如何使用 extractMaskedObject 从一个 HSM 提取密钥作为遮蔽对象。

Example ：提取遮蔽对象

此命令从具有句柄 524295 的密钥提取 HSM 的遮蔽对象，并将其保存为名为 maskedObj 的文件。当该命令成功时，extractMaskedObject 将返回成功消息。

```
Command: extractMaskedObject -o 524295 -out maskedObj

Object was masked and written to file "maskedObj"
```

```
        Cfm3ExtractMaskedObject returned: 0x00 : HSM Return: SUCCESS
```

参数

此命令采用以下参数。

-h

显示该命令的命令行帮助。

必需：是

-o

指定要提取的密钥句柄作为遮蔽对象。

必需：是

-out

指定要将遮蔽对象保存到的文件的名称。

必需：是

相关 主题

- insertMaskedObject
- syncKey
- 跨区域复制备份
- 使用先前的 Backup 创建 AWS CloudHSM 集群

findKey

使用 key_mgmt_util 中的 findKey 命令可通过使用密钥属性的值来搜索密钥。当某个密钥与您设置的所有条件都匹配时，findKey 将返回密钥句柄。如果没有参数，findKey 将返回您可在 HSM 中使用的所有密钥的密钥句柄。要查找特定密钥的属性值，请使用 getAttribute。

与所有 key_mgmt_util 命令相似，findKey 是特定于用户的。它只会返回当前用户可在加密操作中使用的密钥。这包括当前用户拥有的密钥和已与当前用户共享的密钥。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
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>]
```

示例

这些示例演示了如何使用 findKey 查找和识别您的 HSM 中的密钥。

Example ：查找所有密钥

此命令可查找 HSM 中当前用户的所有密钥。输出包含用户拥有和共享的密钥，以及 HSM 中的所有公有密钥。

要获取具有特定密钥句柄的密钥的属性，请使用 getAttribute。要确定当前用户是否拥有或共享特定密钥，请在 cloudhsm_mgmt_uti findAllKeysl 中使用getKeyInfo或。

```
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 ：按类型、用户和会话查找密钥

此命令用于查找当前用户和用户 3 可使用的持久性 AES 密钥。(用户 3 可能能够使用当前用户看不到的其他密钥。)

```
Command: findKey -t 31 -sess 0 -u 3
```

Example ：按类别和标签查找密钥

此命令可查找具有 2018-sept 标签的当前用户的所有公有密钥。

```
Command: findKey -c 2 -l 2018-sept
```

Example ：按 Modulus 查找 RSA 密钥

此命令可查找当前用户的 RSA 密钥 (类型 0)，该密钥是通过使用 m4.txt 文件中的模数创建的。

```
Command: findKey -t 0 -m m4.txt
```

参数

-h

显示该命令的帮助信息。

必需：是

-t

查找指定类型的密钥。输入表示密钥类别的常量。例如，要查找 3DES 密钥，请键入 -t 21。

有效值：

- 0：RSA
- 1：DSA
- 3：EC
- 16：GENERIC_SECRET
- 18：RC4
- 21：三重 DES (3DES)
- 31：AES

必需：否

-c

查找指定类别中的密钥。输入表示密钥类别的常量。例如，要查找公有密钥，请键入 -c 2。

每种密钥类型的有效值如下：

- 2：公有。此类别包含公有私有密钥对的公有密钥。
- 3：私有。此类别包含公有私有密钥对的私有密钥。
- 4：机密。此类别包含所有对称密钥。

必需：否

-l

查找具有指定标签的密钥。键入确切的标签。您不能在 `--l` 值中使用通配符或正则表达式。

必需：否

-id

查找具有指定 ID 的密钥。键入确切的 ID 字符串。您不能在 `-id` 值中使用通配符或正则表达式。

必需：否

-sess

按会话状态查找密钥。要查找仅在当前会话中有效的密钥，请键入 `1`。要查找持久性密钥，请键入 `0`。

必需：否

-u

查找指定用户和当前用户共享的密钥。键入一个逗号分隔的 HSM 用户 ID 列表，例如 `-u 3` 或 `-u 4,7`。要查找 HSM 上的用户的 ID，请使用 listUsers。

当您指定一个用户 ID 时，findKey 将返回该用户的密钥。当您指定多个用户 ID 时，findKey 将返回所有指定用户可使用的密钥。

由于 findKey 仅返回当前用户可使用的密钥，`-u` 结果始终与当前用户的部分密钥相同。要获取任何用户拥有或与任何用户共享的所有密钥，加密官员 (CO) 可以findAllKeys在 cloudhsm_mgmt_util 中使用。

必需：否

-m

查找通过在指定文件中使用 RSA 模数创建的密钥。键入用于存储模数的文件的路径。

-m 表示包含待匹配 RSA 模块的二进制文件（可选）。

必需：否

-kcv

查找具有指定密钥检查值的密钥。

密钥检查值 (KCV) 是 HSM 导入或生成密钥时所产生密钥的 3 字节哈希值或校验和。您也可以在 HSM 之外计算 KCV，例如导出密钥之后。然后，您可对比 KCV 值，以确认密钥的标识和完整性。若要获取密钥 KCV，请使用 getAttribute。

AWS CloudHSM 使用以下标准方法生成密钥检查值：

- 对称密钥：密钥加密零块结果的前 3 个字节。
- 非对称密钥对：公钥 SHA-1 哈希值的前 3 个字节。
- HMAC 密钥：目前不支持 HMAC 密钥 KCV。

必需：否

输出

findKey 输出列出了匹配的密钥及其密钥句柄的总数。

```
        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
```

相关 主题

- findSingleKey
- getKeyInfo
- getAttribute

- findAllKeys在 cloudhsm_mgmt_util
- 密钥属性引用

findSingleKey

key_mgmt_util 工具中的 findSingleKey 命令可验证某个密钥是否存在于集群中的所有 HSM 上。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
findSingleKey -h

findSingleKey -k <key-handle>
```

示例

Example

此命令将验证密钥 252136 是否存在于集群中的所有三个 HSM 上。

```
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
```

参数

-h

显示该命令的帮助信息。

必需：是

-k

指定 HSM 中的一个密钥的密钥句柄。此参数为必需参数。

要查找密钥句柄，请使用 findKey 命令。

必需：是

相关 主题

- findKey
- getKeyInfo
- getAttribute

GendSA KeyPair

key_mgmt_util 工具中的 genDSAKeyPair 命令在 HSM 中生成数字签名算法 (DSA) 密钥对。您必须指定模数长度；该命令将生成模数值。您也可以分配一个 ID，与其他 HSM 用户共享密钥，创建不可提取的密钥，以及创建在会话结束时过期的密钥。如果命令成功，则它会返回 HSM 分配到公有和私有密钥的密钥句柄。您可以使用密钥句柄来区分这些密钥与其他命令。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

> **Tip**
>
> 要查找您所创建密钥的属性（例如类型、长度、标签和 ID），请使用 getAttribute。要查找特定用户的密钥，请使用getKeyInfo。要根据属性值查找密钥，请使用 findKey。

语法

```
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]
```

示例

这些示例演示如何使用 genDSAKeyPair 创建 DSA 密钥对。

Example ：创建 DSA 密钥对

此命令会创建一个具有 DSA 标签的 DSA 密钥对。输出显示公有密钥的密钥句柄为 19 且私有密钥的句柄为 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 ：创建仅会话型 DSA 密钥对

此命令会创建仅在当前会话中有效的 DSA 密钥对。该命令除了必需的 (不唯一) 标签外还将分配 DSA_temp_pair 的唯一 ID。您可能希望创建诸如此类的密钥对并且验证仅会话型令牌。输出显示公有密钥的密钥句柄为 12 且私有密钥的句柄为 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
```

要确认密钥对仅在会话中存在，请使用具有值为 1 (true) 的 findKey 的 -sess 参数。

```
  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 ：创建共享、不可提取的 DSA 密钥对

此命令会创建一个 DSA 密钥对。私有密钥与三个其他用户共享，而且它无法从 HSM 中导出。公有密钥可由任何用户使用且可始终被提取。

```
        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 ：创建仲裁控制型密钥对

此命令会创建一个具有 DSA-mV2 标签的 DSA 密钥对。该命令使用 -u 参数与用户 4 和 6 共享私有密钥。它使用 -m_value 参数来要求使用私有密钥的任何加密操作需要至少两个批准的仲裁。此命令还使用 -attest 参数来验证生成密钥对的固件的完整性。

输出显示该命令生成一个具有密钥句柄 12 的公有密钥和一个具有密钥句柄 17 的私有密钥，并且显示通过了对于集群固件的鉴证检查。

```
        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在私钥（密钥句柄17）上使用。输出确认密钥由当前用户 (用户 3) 所有且与用户 4 和 6 (无其他用户) 共享。输出还显示已启用仲裁身份验证且仲裁大小为 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
```

参数

-h

显示该命令的帮助信息。

必需：是

-m

指定模数的长度 (以位为单位)。唯一有效值为 2048。

必需：是

-l

指定用户定义的密钥对标签。键入字符串。同样的标签适用于配对中的两个密钥。label 的最大大小为 127 个字符。

您可使用任何短语，以帮助您识别密钥。由于标签并非唯一，因此您可将其用于分组和分类密钥。

必需：是

-id

为密钥对指定用户定义标识符。在集群中键入唯一字符串。默认值是空字符串。您指定的 ID 适用于密钥对中的两个密钥。

默认值：无 ID 值。

必需：否

-min_srv

指定在 `-timeout` 参数的值到期之前密钥在其上同步的 HSM 数量的最小值。如果密钥在分配的时间内未同步到指定数量的服务器，则不会创建它。

AWS CloudHSM 自动将每个密钥同步到集群中的每个 HSM。要加快此过程，请将 `min_srv` 的值设置为小于集群中的 HSM 数，并设置一个较低的超时值。但请注意，一些请求可能无法生成密钥。

默认值：1

必需：否

-m_value

指定必须批准使用密钥对中的私有密钥进行任何加密操作的用户数量。键入 `0` 至 `8` 范围的值。

此参数规定了私有密钥的仲裁身份验证要求。默认值 `0` 表明禁用密钥的仲裁身份验验证功能。启用仲裁身份验证后，指定数量的用户必须签署令牌才能批准私有密钥加密操作以及共享或取消共享私有密钥操作。

要查找密钥`m_value`的，请使用getKeyInfo。

只有当命令中的 `-u` 参数与足够多的用户共享密钥对以满足 `m_value` 要求时，此参数才有效。

默认：0

必需：否

-nex

使私钥无法提取。生成的私钥无法从 HSM 中导出。公有密钥始终可提取。

默认：密钥对中的公有密钥和私有密钥均可提取。

必需：否

-sess

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。

如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

若要将会话密钥更改为永久（令牌）密钥，请使用 setAttribute。

默认：密钥永久有效。

必需：否

-timeout

指定命令等待密钥同步到 `min_srv` 参数指定数量的 HSM 所需的时间（以秒为单位）。

仅当 `min_srv` 参数也用于此命令时，该参数才有效。

默认：没有超时 该命令无限期等待，仅当密钥同步至最少数量的服务器时才返回。

必需：否

-u

与指定用户共享密钥对中的私钥。此参数向其他 HSM 加密用户（CU）授予在加密操作中使用私有密钥的权限。任何用户都可以在不共享的情况下使用公有密钥。

键入一个逗号分隔的 HSM 用户 ID 列表，例如 `-u 5,6`。请勿包括当前用户的 HSM 用户 ID。要查找 HSM 上的 CU 的 HSM 用户 ID，请使用 listUsers。要共享或取消共享现有密钥，请在 cloudhsm_mgmt_util 中使用 shareKey。

默认：只有当前用户可使用私有密钥。

必需：否

-attest

运行完整性检查，以验证运行集群的固件是否被篡改。

默认：不执行认证检查。

必需：否

相关 主题

- GenrSA KeyPair
- genSymKey
- GenecC KeyPair

GenecC KeyPair

key_mgmt_util 工具中的 genECCKeyPair 命令将在您的 HSM 中生成一个椭圆曲线加密 (ECC) 密钥对。运行 genECCKeyPair 命令时，您必须为密钥对指定椭圆曲线标识符和标签。您也可以与其他 CU 用户共享私有密钥，创建不可提取的密钥、仲裁控制的密钥和会话结束时过期的密钥。如果命令成功，则它会返回 HSM 分配到公有和私有 ECC 密钥的密钥句柄。您可以使用密钥句柄来区分这些密钥与其他命令。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

> **Tip**
>
> 要查找您所创建密钥的属性（例如类型、长度、标签和 ID），请使用 getAttribute。要查找特定用户的密钥，请使用getKeyInfo。要根据属性值查找密钥，请使用 findKey。

语法

```
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]
```

示例

以下示例演示如何在您的 HSM 中使用 genECCKeyPair 创建 ECC 密钥对。

Example ：创建并检查 ECC 密钥对

此命令使用 NID_secp384r1 椭圆曲线和 ecc14 标签创建 ECC 密钥对。输出表明，私有密钥的密钥句柄为 262177，并且公有密钥的密钥句柄为 262179。该标签同时应用于公有密钥和私有密钥。

```
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
```

生成密钥后，您可以检查其属性。使用 getAttribute 将新 ECC 私有密钥的所有属性（由常量 512 表示）写入 attr_262177 文件。

```
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
```

然后，使用 cat 命令查看 attr_262177 属性文件的内容。输出表明，该密钥是一个椭圆曲线私有密钥，它可用于签名，但不可用于加密、解密、包装、解开包装或验证。该密钥是持久性的，并且可导出。

```
$  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 使用无效的 EEC 曲线

此命令将尝试使用 NID_X9_62_prime192v1 曲线创建 ECC 密钥对。由于此椭圆曲线对 FIPS 模式 HSM 无效，因此该命令将失败。消息报告集群中的某个服务器不可用，但这通常不表示集群中的 HSM 有问题。

```
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
```

参数

-h

显示该命令的帮助信息。

必需：是

-i

指定椭圆曲线的标识符。输入标识符。

有效值：

- 2：NID_X9_62_prime256v1
- 14：NID_secp384r1
- 16：NID_secp256k1

必需：是

-l

指定用户定义的密钥对标签。键入字符串。同样的标签适用于配对中的两个密钥。label 的最大大小为 127 个字符。

您可使用任何短语，以帮助您识别密钥。由于标签并非唯一，因此您可将其用于分组和分类密钥。

必需：是

-id

为密钥对指定用户定义标识符。在集群中键入唯一字符串。默认值是空字符串。您指定的 ID 适用于密钥对中的两个密钥。

默认值：无 ID 值。

必需：否

-min_srv

指定在 -timeout 参数的值到期之前密钥在其上同步的 HSM 数量的最小值。如果密钥在分配的时间内未同步到指定数量的服务器，则不会创建它。

AWS CloudHSM 自动将每个密钥同步到集群中的每个 HSM。要加快此过程，请将 `min_srv` 的值设置为小于集群中的 HSM 数，并设置一个较低的超时值。但请注意，一些请求可能无法生成密钥。

默认值：1

必需：否

-m_value

指定必须批准使用密钥对中的私有密钥进行任何加密操作的用户数量。键入 `0` 至 `8` 范围的值。

此参数规定了私有密钥的仲裁身份验证要求。默认值 `0` 表明禁用密钥的仲裁身份验验证功能。启用仲裁身份验证后，指定数量的用户必须签署令牌才能批准私有密钥加密操作以及共享或取消共享私有密钥操作。

要查找密钥`m_value`的，请使用getKeyInfo。

只有当命令中的 `-u` 参数与足够多的用户共享密钥对以满足 `m_value` 要求时，此参数才有效。

默认：0

必需：否

-nex

使私钥无法提取。生成的私钥无法从 HSM 中导出。公有密钥始终可提取。

默认：密钥对中的公有密钥和私有密钥均可提取。

必需：否

-sess

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。

如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

若要将会话密钥更改为永久（令牌）密钥，请使用 setAttribute。

默认：密钥永久有效。

必需：否

-timeout

指定命令等待密钥同步到 `min_srv` 参数指定数量的 HSM 所需的时间（以秒为单位）。

仅当 min_srv 参数也用于此命令时，该参数才有效。

默认：没有超时 该命令无限期等待，仅当密钥同步至最少数量的服务器时才返回。

必需：否

-u

与指定用户共享密钥对中的私钥。此参数向其他 HSM 加密用户（CU）授予在加密操作中使用私有密钥的权限。任何用户都可以在不共享的情况下使用公有密钥。

键入一个逗号分隔的 HSM 用户 ID 列表，例如 -u 5,6。请勿包括当前用户的 HSM 用户 ID。要查找 HSM 上的 CU 的 HSM 用户 ID，请使用 listUsers。要共享或取消共享现有密钥，请在 cloudhsm_mgmt_util 中使用 shareKey。

默认：只有当前用户可使用私有密钥。

必需：否

-attest

运行完整性检查，以验证运行集群的固件是否被篡改。

默认：不执行认证检查。

必需：否

相关 主题

- genSymKey
- GenrSA KeyPair
- GendSA KeyPair

GenrSA KeyPair

key_mgmt_util 工具中的 genRSAKeyPair 命令生成一个 RSA 非对称密钥对。您指定密钥类型、模数长度和公有指数。该命令会生成一个指定长度的模数并创建密钥对。您可以分配一个 ID，与其他 HSM 用户共享密钥，创建不可提取的密钥和会话结束时过期的密钥。如果命令成功，则返回 HSM 分配给密钥的密钥句柄。您可以使用该密钥句柄来为其他命令标识密钥。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

> **Tip**
>
> 要查找您所创建密钥的属性（例如类型、长度、标签和 ID），请使用 getAttribute。要查找特定用户的密钥，请使用getKeyInfo。要根据属性值查找密钥，请使用 findKey。

语法

```
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]
```

示例

这些示例演示如何在您的 HSM 中使用 genRSAKeyPair 创建非对称密钥对。

Example ：创建并检查 RSA 密钥对

此命令将创建一个带 2048 位模数和指数 65537 的 RSA 密钥对。输出表明，公有密钥句柄为 2100177，私有密钥句柄为 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
```

下一条命令使用 getAttribute 获取我们刚刚创建的公有密钥的属性。它将输出写入 attr_2100177 文件。后跟 cat 命令，该命令将获取属性文件的内容。有关解释密钥属性的帮助，请参阅 密钥属性引用。

生成的十六进制值确认它是一个类型为 RSA (OBJ_ATTR_CLASS 0x02) 的公有密钥 (OBJ_ATTR_KEY_TYPE 0x00)。您可以使用此公有密钥进行加密 (OBJ_ATTR_ENCRYPT 0x01)，但不能进行解密 (OBJ_ATTR_DECRYPT 0x00)。该结果还包含密钥长度 (512，0x200)、模数、模数长度 (2048，0x800) 和公有指数 (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 ：生成共享 RSA 密钥对

此命令将生成一个 RSA 密钥对并与用户 4 (HSM 上的另一个 CU) 共享私有密钥。该命令使用 m_value 参数以要求至少两次批准，之后密钥对中的私有密钥才能用于加密操作。在使用 m_value 参数时，还必须在命令中使用 -u，并且 m_value 不能超过用户总数 (-u 中的值数 + 所有者)。

```
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
```

参数

-h

显示该命令的帮助信息。

必需：是

-m

指定模数的长度 (以位为单位)。最小值为 2048。

必需：是

-环

指定公有指数。此值必须为大于或等于 65537 的奇数。

必需：是

-l

指定用户定义的密钥对标签。键入字符串。同样的标签适用于配对中的两个密钥。label 的最大大小为 127 个字符。

您可使用任何短语，以帮助您识别密钥。由于标签并非唯一，因此您可将其用于分组和分类密钥。

必需：是

-id

为密钥对指定用户定义标识符。在集群中键入唯一字符串。默认值是空字符串。您指定的 ID 适用于密钥对中的两个密钥。

默认值：无 ID 值。

必需：否

-min_srv

指定在 -timeout 参数的值到期之前密钥在其上同步的 HSM 数量的最小值。如果密钥在分配的时间内未同步到指定数量的服务器，则不会创建它。

AWS CloudHSM 自动将每个密钥同步到集群中的每个 HSM。要加快此过程，请将 `min_srv` 的值设置为小于集群中的 HSM 数，并设置一个较低的超时值。但请注意，一些请求可能无法生成密钥。

默认值：1

必需：否

-m_value

指定必须批准使用密钥对中的私有密钥进行任何加密操作的用户数量。键入 `0` 至 `8` 范围的值。

此参数规定了私有密钥的仲裁身份验证要求。默认值 `0` 表明禁用密钥的仲裁身份验验证功能。启用仲裁身份验证后，指定数量的用户必须签署令牌才能批准私有密钥加密操作以及共享或取消共享私有密钥操作。

要查找密钥`m_value`的，请使用getKeyInfo。

只有当命令中的 `-u` 参数与足够多的用户共享密钥对以满足 `m_value` 要求时，此参数才有效。

默认：0

必需：否

-nex

使私钥无法提取。生成的私钥无法从 HSM 中导出。公有密钥始终可提取。

默认：密钥对中的公有密钥和私有密钥均可提取。

必需：否

-sess

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。

如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

若要将会话密钥更改为永久（令牌）密钥，请使用 setAttribute。

默认：密钥永久有效。

必需：否

-timeout

指定命令等待密钥同步到 `min_srv` 参数指定数量的 HSM 所需的时间（以秒为单位）。

仅当 `min_srv` 参数也用于此命令时，该参数才有效。

默认：没有超时 该命令无限期等待，仅当密钥同步至最少数量的服务器时才返回。

必需：否

-u

与指定用户共享密钥对中的私钥。此参数向其他 HSM 加密用户（CU）授予在加密操作中使用私有密钥的权限。任何用户都可以在不共享的情况下使用公有密钥。

键入一个逗号分隔的 HSM 用户 ID 列表，例如 `-u 5,6`。请勿包括当前用户的 HSM 用户 ID。要查找 HSM 上的 CU 的 HSM 用户 ID，请使用 listUsers。要共享或取消共享现有密钥，请在 cloudhsm_mgmt_util 中使用 shareKey。

默认：只有当前用户可使用私有密钥。

必需：否

-attest

运行完整性检查，以验证运行集群的固件是否被篡改。

默认：不执行认证检查。

必需：否

相关 主题

- genSymKey
- GendSA KeyPair
- GenecC KeyPair

genSymKey

key_mgmt_util 工具中的 genSymKey 命令可在 HSM 中生成对称密钥。您可以指定密钥类型和大小、分配 ID 和标签以及与其他 HSM 用户共享密钥。您还可以创建不可提取的密钥以及将在会话结束时过期的密钥。如果命令成功，则返回 HSM 分配给密钥的密钥句柄。您可以使用该密钥句柄来为其他命令标识密钥。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
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]
```

示例

这些示例演示如何使用 genSymKey 在 HSM 中创建对称密钥。

> **Tip**
>
> 要将您在这些示例中创建的密钥用于 HMAC 操作，则必须在生成密钥后将 OBJ_ATTR_SIGN 和 OBJ_ATTR_VERIFY 设置为 TRUE。要设置这些值，请在 CloudHSM 管理实用程（CMU）中使用 setAttribute。更多信息，请参阅 setAttribute。

Example ：生成 AES 密钥

此命令将创建一个具有 aes256 标签的 256 位 AES 密钥。输出显示，新密钥的密钥句柄为 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 ：创建会话密钥

此命令将创建仅在当前会话中有效的不可提取的 192 位 AES 密钥。您可能希望创建一个与此类似的密钥来包装 (然后立即解开包装) 正导出的密钥。

```
Command: genSymKey -t 31 -s 24 -l tmpAES -id wrap01 -nex -sess
```

Example ：快速返回

此命令将创建一个具有 IT_test_key 标签的通用 512 字节密钥。此命令不会等待该密钥同步到群集中的所有 HSM。相反，此命令将在任何一个 HSM (-min_srv 1) 上创建该密钥或在 1 秒 (-timeout 1) 后返回，以时间较短者为准。如果该密钥在超时前未同步到指定最少数量的 HSM，则不会生成。您可能希望在脚本中使用与此类似的命令创建许多密钥，如以下示例中的 for 循环。

```
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 ：创建仲裁授权通用密钥

此命令将创建一个具有 generic-mV2 标签的 2048 位通用机密密钥。此命令使用 -u 参数与另一 CU (用户 6) 共享此密钥。对于使用此密钥的任何加密操作，它使用 -m_value 参数以要求至少有两个审批者的仲裁。此命令还使用 -attest 参数来验证生成密钥的固件的完整性。

输出显示，此命令已生成一个具有密钥句柄 9 的密钥以及已通过群集固件上的鉴证检查。

```
                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 ：创建并检查密钥

此命令将创建一个具有 3DES_shared 标签和 IT-02 ID 的三重 DES 密钥。此密钥可由当前用户、用户 4 和用户 5 使用。如果该 ID 在群集中不唯一或当前用户为用户 4 或 5，则此命令将失败。

输出显示，新密钥具有密钥句柄 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
```

要验证新的 3DES 密钥是否由当前用户拥有并与用户 4 和 5 共享，请使用 getKeyInfo。此命令使用分配给新密钥的句柄 (Key Handle: 7)。

输出确认，此密钥由用户 3 所有并与用户 4 和 5 共享。

```
Command:  getKeyInfo -k 7

        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

        Owned by user 3

        also, shared to following 2 user(s):

                4, 5
```

要确认密钥的其他属性，请使用 getAttribute。第一个命令使用 getAttribute 获取密钥句柄 7 (-o 7) 的所有属性 (-a 512)。它将这些属性写入 attr_7 文件。第二个命令使用 cat 获取 attr_7 文件的内容。

此命令确认，密钥 7 是具有标签 3DES_shared (OBJ_ATTR_LABEL 3DES_shared) 和 IT_02 (OBJ_ATTR_ID IT-02) ID 的 192 位 (OBJ_ATTR_VALUE_LEN 0x00000018 或 24 字节)

3DES (OBJ_ATTR_KEY_TYPE 0x15) 对称密钥 (OBJ_ATTR_CLASS 0x04)。此密钥是永久密钥 (OBJ_ATTR_TOKEN 0x01) 和可提取密钥 (OBJ_ATTR_EXTRACTABLE 0x01)，并且可用于加密、解密和包装。

> **Tip**
>
> 要查找您所创建密钥的属性（例如类型、长度、标签和 ID），请使用 getAttribute。要查找特定用户的密钥，请使用getKeyInfo。要根据属性值查找密钥，请使用 findKey。

有关解释密钥属性的帮助，请参阅 密钥属性引用。

```
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**
>
> 要将您在这些示例中创建的密钥用于 HMAC 操作，则必须在生成密钥后将 OBJ_ATTR_SIGN 和 OBJ_ATTR_VERIFY 设置为 TRUE。要设置这些值，请在 CMU 中使用 setAttribute。更多信息，请参阅 setAttribute。

参数

-h

显示该命令的帮助信息。

必需：是

-t

指定对称密钥的类型。输入表示密钥类型的常量。例如，要创建 AES 密钥，请键入 `-t 31`。

有效值：

- 16：GENERIC_SECRET。通用机密密钥 是一个字节数组，不符合任何特定标准 (如 AES 密钥的要求)。
- 18：RC4。RC4 密钥在 FIPS 模式的 HSM 上无效
- 21：三重 DES (3DES)。根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

- 31：AES

必需：是

-s

指定以字节为单位的密钥大小。例如，要创建 192 位密钥，请键入 24。

每种密钥类型的有效值如下：

- AES：16 (128 位)、24 (192 位)、32 (256 位)
- 3DES：24 (192 位)
- 通用密钥：<3584（28672 位）

必需：是

-l

指定用户定义的密钥标签。键入字符串。

您可使用任何短语，以帮助您识别密钥。由于标签并非唯一，因此您可将其用于分组和分类密钥。

必需：是

-attest

运行完整性检查，以验证运行集群的固件是否被篡改。

默认：不执行认证检查。

必需：否

-id

为密钥指定用户定义标识符。在集群中键入唯一字符串。默认值是空字符串。

默认值：无 ID 值。

必需：否

-min_srv

指定在 `-timeout` 参数的值到期之前密钥在其上同步的 HSM 数量的最小值。如果密钥在分配的时间内未同步到指定数量的服务器，则不会创建它。

AWS CloudHSM 自动将每个密钥同步到集群中的每个 HSM。要加快此过程，请将 `min_srv` 的值设置为小于集群中的 HSM 数，并设置一个较低的超时值。但请注意，一些请求可能无法生成密钥。

默认值：1

必需：否

-m_value

指定必须批准使用密钥进行任何加密操作的用户数量。键入 0 至 8 范围的值。

此参数规定了密钥的仲裁身份验证要求。默认值 0 表明禁用密钥的仲裁身份验验证功能。启用仲裁身份验证后，必须有指定数量的用户签署令牌，才能批准使用密钥的加密操作，以及共享或不共享密钥的操作。

要查找密钥m_value的，请使用getKeyInfo。

只有当命令中的 -u 参数与足够多的用户共享密钥以满足 m_value 要求时，此参数才有效。

默认：0

必需：否

-nex

使密钥无法提取。生成的私有密钥无法从 HSM 导出。

默认：密钥可提取。

必需：否

-sess

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。

如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

若要将会话密钥更改为永久（令牌）密钥，请使用 setAttribute。

默认：密钥永久有效。

必需：否

-timeout

指定命令等待密钥同步到 min_srv 参数指定数量的 HSM 所需的时间（以秒为单位）。

仅当 min_srv 参数也用于此命令时，该参数才有效。

默认：没有超时 该命令无限期等待，仅当密钥同步至最少数量的服务器时才返回。

必需：否

-u

与指定用户共享密钥。此参数向其他 HSM 加密用户 (CU) 授予在加密操作中使用此密钥的权限。

键入一个逗号分隔的 HSM 用户 ID 列表，例如 -u 5,6。请勿包括当前用户的 HSM 用户 ID。要查找 HSM 上的 CU 的 HSM 用户 ID，请使用 listUsers。要共享或取消共享现有密钥，请在 cloudhsm_mgmt_util 中使用 shareKey。

默认：只有当前用户可使用密钥。

必需：否

相关 主题

- exSymKey
- GenrSA KeyPair
- GendSA KeyPair
- GenecC KeyPair
- setAttribute

getAttribute

key_mgmt_util 中的getAttribute命令将密钥的一个或全部属性值写入文件。 AWS CloudHSM 如果密钥类型不存在您指定的属性（如 AES 密钥的模数），则 getAttribute 将返回错误。

密钥属性是密钥的属性。它们包括各种特征 (如密钥类型、类、标签和 ID) 以及表示您可使用密钥执行的操作 (如加密、解密、包装、签名和验证) 的值。

您只能对您拥有的密钥和与您共享的密钥使用 getAttribute。您可运行此命令或 cloudhsm_mgmt_util 中的 getAttribute 命令，以从集群的所有 HSM 中获取密钥的一个属性值并将其写入到 stdout 或文件。

要获取包含属性和表示属性的常量的列表，请使用 listAttributes 命令。要更改现有密钥的属性值，请使用 key_mgmt_util 中的 setAttribute 和 cloudhsm_mgmt_util 中的 setAttribute。有关解释密钥属性的帮助，请参阅 密钥属性引用。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
getAttribute -h

getAttribute -o <key handle>
             -a <attribute constant>
             -out <file>
```

示例

以下示例演示如何使用 getAttribute 获取 HSM 中的密钥的属性。

Example ：获取密钥类型

此示例将获取密钥的类型 (如 AES、3DES、通用密钥、RSA 或椭圆曲线密钥对)。

第一个命令运行 listAttributes，以获取密钥属性和表示它们的常量。输出显示，密钥类型的常量为 256。有关解释密钥属性的帮助，请参阅 密钥属性引用。

```
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
```

第二个命令运行 getAttribute。它请求密钥句柄 524296 的密钥类型 (属性 256) 并将其写入到 `attribute.txt` 文件。

```
Command: getAttribute -o 524296 -a 256 -out attribute.txt
Attributes dumped into attribute.txt file
```

最后一个命令获取密钥文件的内容。输出显示，密钥类型为 `0x15` 或 21，这是一个三重 DES (3DES) 密钥。有关类和类型值的定义，请参阅密钥属性参考。

```
$  cat attribute.txt
OBJ_ATTR_KEY_TYPE
0x00000015
```

Example ：获取密钥的所有属性

此命令将获取密钥句柄为 6 的密钥的所有属性并将这些属性写入到 `attr_6` 文件中。它使用表示所有属性的属性值 512。

```
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>
```

此命令将显示包含所有属性值的示例属性文件的内容。在这些值中，它报告密钥是一个具有 ID `test_01` 和标签 `aes256` 的 256 位 AES 密钥。此密钥是可提取的永久密钥 (也就是说，它不是仅会话密钥)。有关解释密钥属性的帮助，请参阅 密钥属性引用。

```
$  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
```

参数

-h

显示该命令的帮助信息。

必需：是

-o

指定目标密钥的密钥句柄。您在每个命令中只能指定一个密钥。要获取密钥的密钥句柄，请使用 findKey。

此外，您必须拥有指定密钥或必须与您共享此密钥。要查找密钥的用户，请使用getKeyInfo。

必需：是

-a

标识属性。输入表示属性的常量，或表示所有属性的 512。例如，要获取密钥类型，请键入 256 (OBJ_ATTR_KEY_TYPE 属性的常量)。

要列出属性及其常量，请使用 listAttributes。有关解释密钥属性的帮助，请参阅 密钥属性引用。

必需：是

-out

将输出写入到指定文件。键入文件路径。您不能将输出写入到 stdout。

如果指定文件存在，getAttribute 将覆盖此文件而不发出警告。

必需：是

相关 主题

- cloudhsm_mgmt_util 中的 getAttribute
- listAttributes
- setAttribute
- findKey
- 密钥属性引用

getCaviumPriv钥匙

key_mgmt_util 中的密getCaviumPriv钥命令以假的 PEM 格式从 HSM 中导出私钥。伪造 PEM 文件不包含实际的私有密钥资料，而是引用 HSM 中的私有密钥，随后可用于建立从 Web 服务器到 AWS CloudHSM的 SSL/TLS 卸载。有关更多信息，请参阅 Linux 上的 SSL/TLS 卸载。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
getCaviumPrivKey -h

getCaviumPrivKey -k <private-key-handle
                 -out <fake-PEM-file>
```

示例

此示例显示了如何使用 getCaviumPrivKey 将私有密钥以伪造 PEM 格式导出。

Example ：导出伪造 PEM 文件

此命令创建和导出具有句柄 15 的私有密钥的伪造 PEM 版本，并将其保存到名为 `cavKey.pem` 的文件中。当该命令成功时，exportPrivateKey 将返回成功消息。

```
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
```

参数

此命令采用以下参数。

-h

显示该命令的命令行帮助。

必需：是

-k

指定要以伪造 PEM 格式导出的私有密钥的密钥句柄。

必需：是

-out

指定要将伪造 PEM 密钥写入的文件的名称。

必需：是

相关 主题

- importPrivateKey
- Linux 上的 SSL/TLS 卸载

getCert

key_mgmt_util 中的 getCert 命令检索 HSM 的分区证书并将其保存到一个文件中。当您运行此命令时，请指定要检索的证书的类型。要执行此操作，请使用其中一个相应的整数，如下面的参数部分中所述。要了解这些证书中的每个证书的角色，请参阅验证 HSM 身份。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
getCert -h

getCert -f <file-name>
        -t <certificate-type>
```

示例

此示例说明如何使用 getCert 检索集群的客户根证书并将其另存为文件。

Example ：检索客户根证书

此命令导出客户根证书（由整数 4 表示）并将其保存到一个名为 `userRoot.crt` 的文件中。当该命令成功时，getCert 将返回成功消息。

```
Command: getCert -f userRoot.crt -s 4

Cfm3GetCert() returned 0 :HSM Return: SUCCESS
```

参数

此命令采用以下参数。

`-h`

显示该命令的命令行帮助。

必需：是

`-f`

指定要将检索到的证书保存到的文件的名称。

必需：是

-s

一个指定要检索的分区证书的类型的整数。整数及其相应的证书类型如下所示：

- 1 - 制造商根证书
- 2 - 制造商硬件证书
- 4 - 客户根证书
- 8 - 集群证书（由客户根证书签署）
- 16 - 集群证书（链接到制造商根证书）

必需：是

相关 主题

- 验证 HSM 身份
- getCert（在 cloudhsm_mgmt_util 中）

getKeyInfo

key_mgmt_util 中的 getKeyInfo 命令将返回可使用密钥的 HSM 用户的用户 ID，这些用户包括所有者和共享密钥的加密用户（CU）。如果对某个密钥启用了仲裁身份验证，getKeyInfo 也将返回必须批准使用该密钥的加密操作的用户数。您只能在您拥有的密钥和与您共享的密钥上运行 getKeyInfo。

当您在公有密钥上运行 getKeyInfo 时，getKeyInfo 将仅返回密钥所有者，即使 HSM 的所有用户都可以使用该公有密钥。要在您的 HSM 中查找 HSM 用户的用户 ID，请使用 listUsers。要查找特定用户的密钥，请使用 findKey -u。

您拥有您创建的密钥。在创建密钥时，您可以与其他用户共享它。然后，要共享或取消共享现有密钥，请在 cloudhsm_mgmt_util 中使用 shareKey。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
getKeyInfo -h

getKeyInfo -k <key-handle>
```

示例

这些示例说明如何使用 getKeyInfo 获取有关密钥用户的信息。

Example ：获取对称密钥的用户

此命令将获取可使用密钥句柄为 9 的 AES (对称) 密钥的用户。输出显示，用户 3 拥有此密钥并已将它与用户 4 共享。

```
Command:  getKeyInfo -k 9

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 1 user(s):

                4
```

Example ：获取非对称密钥对的用户

这些命令使用 getKeyInfo 获取可使用 RSA（非对称）密钥对中的密钥的用户。公有密钥具有密钥句柄 21。私有密钥具有密钥句柄 20。

当您在私有密钥 (20) 上运行 getKeyInfo 时，它将返回密钥所有者 (3) 和共享密钥的加密用户 (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
```

当您在公有密钥 (21) 上运行 getKeyInfo 时，它仅返回密钥所有者 (3)。

```
Command:  getKeyInfo -k 21
```

```
        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

        Owned by user 3
```

要确认用户 4 可以使用公有密钥 (和 HSM 上的所有公有密钥)，请使用 -ufindKey 的 参数。

输出显示，用户 4 可以使用密钥对中的公有密钥 (21) 和私有密钥 (20)。用户 4 还可以使用所有其他公有密钥以及他们创建的或与他们共享的任何私有密钥。

```
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 ：获取密钥的仲裁身份验证值 (m_value)

此示例演示如何获取密钥的 m_value，即，必须批准使用密钥的任何加密操作的仲裁中的用户数量。

如果对某个密钥启用了仲裁身份验证，用户的仲裁必须批准使用该密钥的任何加密操作。要启用仲裁身份验证并设置仲裁大小，请在创建密钥时使用 -m_value 参数。

此命令使用 GenRSA 创建KeyPair与用户 4 共享的 RSA 密钥对。它使用 m_value 参数对密钥对中的私有密钥启用仲裁身份验证并将仲裁大小设置为两个用户。用户的数量必须足够大，才能提供所需的批准。

输出显示，此命令已创建公有密钥 27 和私有密钥 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
```

此命令将使用 getKeyInfo 获取有关私有密钥用户的信息。输出显示，该密钥由用户 3 所有并与用户 4 共享。它还表明，这两个用户的仲裁必须批准使用该密钥的所有加密操作。

```
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
```

参数

-h

显示该命令的命令行帮助。

必需：是

-k

指定 HSM 中的一个密钥的密钥句柄。输入您拥有或共享的密钥的密钥句柄。此参数为必需参数。

要查找密钥句柄，请使用 findKey 命令。

必需：是

相关 主题

- getKeyInfo在 cloudhsm_mgmt_util
- listUsers
- findKey
- findAllKeys在 cloudhsm_mgmt_util

help

key_mgmt_util 中的 help 命令显示有关所有可用 key_mgmt_util 命令的信息。

在运行 help 之前，您必须启动 key_mgmt_util。

语法

```
help
```

示例

此示例显示了 help 命令的输出。

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
```

参数

此命令无任何参数。

相关 主题

- loginHSM 和 logoutHSM

importPrivateKey

key_mgmt_util 中的 importPrivateKey 命令将非对称私有密钥从文件导入到 HSM。HSM 不允许以明文形式直接导入密钥。该命令使用您指定的 AES 包装密钥对私有密钥进行加密，并在 HSM 中解包密钥。如果您正在尝试将 AWS CloudHSM 密钥与证书关联，请参阅此主题。

> **Note**
>
> 您不能使用对称密钥或私有密钥导入受密码保护的 PEM 密钥。

必须指定具有 OBJ_ATTR_UNWRAP 和 OBJ_ATTR_ENCRYPT 属性值 1 的 AES 包装密钥。要查找密钥的属性，请使用 getAttribute 命令。

> **Note**
>
> 此命令不提供将导入的密钥标记为不可导出的选项。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
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]
```

示例

此示例显示了如何使用 importPrivateKey 将私有密钥导入到 HSM。

Example ：导入私有密钥

此命令从名为 rsa2048.key 的文件导入私有密钥，标签为 rsa2048-imported，包含句柄为 524299 的包装密钥。如果命令成功，importPrivateKey 将返回导入密钥的密钥句柄和成功消息。

```
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
```

参数

此命令采用以下参数。

-h

显示该命令的命令行帮助。

必需：是

-l

指定用户定义的私有密钥标签。

必需：是

-f

指定要导入的密钥的文件名。

必需：是

-w

指定包装密钥的密钥句柄。此参数为必需参数。要查找密钥句柄，请使用 findKey 命令。

要确定密钥是否可用作包装密钥，请使用 getAttribute 获取 OBJ_ATTR_WRAP 属性 (262) 的值。要创建包装密钥，请使用 genSymKey 创建 AES 密钥（类型 31）。

如果您使用 -wk 参数指定外部解开包装密钥，则 -w 包装密钥用于在导入过程中包装密钥而不是将其解开包装。

必需：是

-sess

指定导入的密钥作为会话密钥。

默认值：导入密钥保存为集群中的持久（令牌）密钥。

必需：否

-id

指定要导入的密钥的 ID。

默认值：无 ID 值。

必需：否

-m_value

指定必须批准使用导入密钥的任何加密操作的用户数量。输入一个 **0** 和 **8** 之间的值。

只有当命令中的 -u 参数与足够多的用户共享密钥以满足 m_value 要求时，此参数才有效。

默认：0

必需：否

-min_srv

指定在 -timeout 参数的值到期之前导入密钥在其上同步的 HSM 数量的最小值。如果密钥在分配的时间内未同步到指定数量的服务器，则不会创建它。

AWS CloudHSM 自动将每个密钥同步到集群中的每个 HSM。要加快此过程，请将 min_srv 的值设置为小于集群中的 HSM 数，并设置一个较低的超时值。但请注意，一些请求可能无法生成密钥。

默认值：1

必需：否

-timout

指定在包含 min-serv 参数时，等待密钥在 HSM 间同步的秒数。如果没有指定数字，轮询永远继续。

默认值：无限制

必需：否

-u

指定要与之共享导入的私有密钥的用户列表。此参数向其他 HSM 加密用户 (CU) 授予在加密操作中使用导入密钥的权限。

输入一个逗号分隔的 HSM 用户 ID 列表，例如 -u 5,6。请勿包括当前用户的 HSM 用户 ID。要查找 HSM 上的 CU 的 HSM 用户 ID，请使用 listUsers。

默认值：只有当前用户可使用导入密钥。

必需：否

-wk

指定用于包装正在导入密钥的密钥。输入包含明文 AES 密钥的文件的路径和名称。

当您包含此参数时，importPrivateKey 将使用 -wk 文件中的密钥包装正在导入的密钥。它还使用 -w 参数指定的密钥解开包装。

默认值：使用 -w 参数中指定的包装密钥包装和解开包装。

必需：否

-attest

对固件响应执行证明检查，以确保运行集群的固件未受损。

必需：否

相关 主题

- wrapKey
- unWrapKey
- genSymKey
- exportPrivateKey

importPubKey

key_mgmt_util 中的 importPubKey 命令将 PEM 格式公有密钥导入到 HSM。您可以使用它来导入在 HSM 外生成的公有密钥。您也可以使用命令导入从 HSM 导出的密钥，例如exportPubKey命令导出的密钥。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
importPubKey -h

importPubKey -l <label>
             -f <key-file>
             [-sess]
             [-id <key-id>]
             [min_srv <minimum-number-of-servers>]
             [-timeout <number-of-seconds>]
```

示例

此示例显示了如何使用 importPubKey 将公有密钥导入到 HSM。

Example ：导入公有密钥

此命令从名为 public.pem 的文件导入公有密钥，标签为 importedPublicKey。如果命令成功，importPubKey 将返回导入密钥的密钥句柄和成功消息。

```
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
```

参数

此命令采用以下参数。

-h

显示该命令的命令行帮助。

必需：是

-l

指定用户定义的公有密钥标签。

必需：是

-f

指定要导入的密钥的文件名。

必需：是

-sess

指定导入的密钥作为会话密钥。

默认值：导入密钥保存为集群中的持久（令牌）密钥。

必需：否

-id

指定要导入的密钥的 ID。

默认值：无 ID 值。

必需：否

-min_srv

指定在 `-timeout` 参数的值到期之前，导入密钥与其同步的 HSM 数量的最小值。如果密钥在分配的时间内未同步到指定数量的服务器，则不会创建它。

AWS CloudHSM 自动将每个密钥同步到集群中的每个 HSM。要加快此过程，请将 `min_srv` 的值设置为小于集群中的 HSM 数，并设置一个较低的超时值。但请注意，一些请求可能无法生成密钥。

默认值：1

必需：否

-timeout

指定在包含 `min-serv` 参数时，等待密钥在 HSM 间同步的秒数。如果没有指定数字，轮询永远继续。

默认值：无限制

必需：否

相关 主题

- exportPubKey
- 生成密钥

imSymKey

key_mgmt_util 工具中的 imSymKey 命令将文件中对称密钥的明文副本导入 HSM 中。您可以使用它来导入通过 HSM 之外的任何方法生成的密钥以及从 HSM 导出的密钥，例如命令写入文件的密钥。exSymKey

在导入过程中，imSymKey 将您选择的 AES 密钥（包装密钥）用于包装（加密），然后解开包装（解密）要导入的密钥。但是，imSymKey 仅适用于包含纯文本密钥的文件。要导出和导入加密密钥，请使用 WrapKey 和命令。unWrapKey

此外，imSymKey 命令仅导入对称密钥。要导入公有密钥，请使用 importPubKey。要导入私钥，请使用importPrivateKey或 wrap Key。

> **Note**
>
> 您不能使用对称密钥或私有密钥导入受密码保护的 PEM 密钥。

导入的密钥在工作方式方面与在 HSM 中生成的密钥非常相似。但是，OBJ_ATTR_LOCAL 属性的值为零，这表示该属性不是本地生成的。您可以在导入对称密钥时，使用以下命令共享该密钥。导入密钥后，您可以在 shareKeycloudhsm_mgmt_util 中使用 命令来共享它。

```
imSymKey -l aesShared -t 31 -f kms.key -w 3296 -u 5
```

导入密钥后，请确保标记或删除密钥文件。此命令不会阻止您多次导入相同密钥材料。因此，具有不同密钥句柄和相同密钥材料的多个密钥会让跟踪密钥材料的使用情况变得困难并防止超出其加密限制。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
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>]
```

示例

这些示例说明如何使用 imSymKey 将对称密钥导入您的 HSM 中。

Example ：导入 AES 对称密钥

此示例使用 imSymKey 将 AES 对称密钥导入 HSM 中。

第一条命令使用 OpenSSL 生成随机 256 位 AES 对称密钥。它将密钥保存在 aes256.key 文件中。

```
$  openssl rand -out aes256-forImport.key 32
```

第二条命令使用 imSymKey 将 aes256.key 文件中的 AES 密钥导入 HSM 中。它使用密钥 20 (HSM 中的 AES 密钥) 作为包装密钥并指定 imported 的标签。与 ID 不同，标签无需在群集中是唯一的。-t (类型) 参数的值为 31，该值表示 AES。

输出表明，文件中的密钥已经过包装和解开包装，然后导入 HSM 中，其中为密钥分配了密钥句柄 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
```

下一条命令使用 getAttribute 获取新导入密钥的 OBJ_ATTR_LOCAL 属性 (属性 355) 并将它写入 attr_262180 文件中。

```
Command:  getAttribute -o 262180 -a 355 -out attributes/attr_262180
Attributes dumped into attributes/attr_262180_imported file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

在您检查属性文件时，您会看到 OBJ_ATTR_LOCAL 属性的值为零，这指示 HSM 中未生成密钥材料。

```
$  cat attributes/attr_262180_local
OBJ_ATTR_LOCAL
0x00000000
```

Example ：在集群之间移动对称密钥

此示例说明如何在集群之间使用exSymKey和imSymKey移动纯文本 AES 密钥。您可使用与此类似的过程，创建两个集群中的 HSM 上存在的 AES 包装密钥。共享包装密钥到位后，您可以使用 wrapKey and 在集unWrapKey群之间移动加密密钥。

执行此操作的 CU 用户必须有权登录这两个集群上的 HSM。

第一个命令用于exSymKey将密钥 14（一个 32 位 AES 密钥）从集群 1 导出到aes.key文件中。它使用密钥 6（集群 1 中的 HSM 上的 AES 密钥）作为包装密钥。

```
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"
```

随后，用户登录集群 2 中的 key_mgmt_util 并运行 imSymKey 命令以将 aes.key 文件中的密钥导入集群 2 中的 HSM 中。此命令使用密钥 252152（集群 2 中 HSM 上的 AES 密钥）作为包装密钥。

由于exSymKey和imSymKey使用的包装密钥会包装并立即解开目标密钥，因此不同集群上的封装密钥不必相同。

输出表明，此密钥已成功导入集群 2 中并已获得密钥句柄 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
```

要证明群集 1 的密钥 14 和群集 2 中的密钥 21 具有相同的密钥材料，请获取每个密钥的密钥检查值 (KCV)。如果 KCV 值相同，则密钥材料相同。

以下命令使用群集 1 中的 getAttribute 将密钥 14 的 KCV 属性 (属性 371) 的值写入 attr_14_kcv 文件中。然后，它使用 cat 命令获取 attr_14_kcv 文件的内容。

```
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
```

此类似命令使用群集 2 中的 getAttribute 将密钥 21 的 KCV 属性 (属性 371) 的值写入 attr_21_kcv 文件中。然后，它使用 cat 命令获取 attr_21_kcv 文件的内容。

```
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
```

输出表明，两个密钥的 KCV 值相同，这证明密钥材料相同。

由于两个群集的 HSM 中存在相同的密钥材料，因此，您现在可以在群集之间共享已加密密钥，而无需公开纯文本密钥。例如，您可以将 wrapKey 命令与包装密钥 14 结合使用以导出群集 1 中的已加密密钥，然后将 unWrapKey 与包装密钥 21 结合使用以将已加密密钥导入群集 2 中。

Example ：导入会话密钥

此命令使用 imSymKey 的 -sess 参数导入仅在当前会话中有效的 192 位三重 DES 密钥。

此命令使用 -f 参数指定包含要导入的密钥的文件，使用 -t 参数指定密钥类型，并使用 -w 参数指定包装密钥。它使用 -l 参数指定用于对密钥进行分类的标签，并使用 -id 参数为密钥创建友好的唯一标识符。它还使用 -attest 参数验证导入密钥的固件。

输出表明，密钥已成功包装和解开包装，导入 HSM 中，并且分配有密钥句柄 37。此外，通过了鉴证检查，这指示固件未遭篡改。

```
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
```

接下来，您可以使用 getAttribute 或 findKey 命令验证新导入密钥的属性。以下命令使用 findKey 验证密钥 37 是否具有此命令指定的类型、标签和 ID，并验证此密钥是否为会话密钥。如输出的第 5 行所示，findKey 报告，与所有属性匹配的唯一密钥为密钥 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
```

参数

-attest

运行完整性检查，以验证运行集群的固件是否被篡改。

默认：不执行认证检查。

必需：否

-f

指定包含要导入的密钥的文件。

该文件必须包含指定长度的 AES 或三重 DES 密钥的纯文本副本。RC4 和 DES 密钥在 FIPS 模式的 HSM 上无效。

- AES：16、24 或 32 个字节
- 三重 DES (3DES)：24 个字节

必需：是

-h

显示该命令的帮助信息。

必需：是

-id

为密钥指定用户定义标识符。在集群中键入唯一字符串。默认值是空字符串。

默认值：无 ID 值。

必需：否

-l

指定用户定义的密钥标签。键入字符串。

您可使用任何短语，以帮助您识别密钥。由于标签并非唯一，因此您可将其用于分组和分类密钥。

必需：是

-min_srv

指定在 `-timeout` 参数的值到期之前密钥在其上同步的 HSM 数量的最小值。如果密钥在分配的时间内未同步到指定数量的服务器，则不会创建它。

AWS CloudHSM 自动将每个密钥同步到集群中的每个 HSM。要加快此过程，请将 `min_srv` 的值设置为小于集群中的 HSM 数，并设置一个较低的超时值。但请注意，一些请求可能无法生成密钥。

默认值：1

必需：否

-sess

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。

如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

若要将会话密钥更改为永久（令牌）密钥，请使用 setAttribute。

默认：密钥永久有效。

必需：否

-timeout

指定命令等待密钥同步到 `min_srv` 参数指定数量的 HSM 所需的时间（以秒为单位）。

仅当 `min_srv` 参数也用于此命令时，该参数才有效。

默认：没有超时 该命令无限期等待，仅当密钥同步至最少数量的服务器时才返回。

必需：否

-t

指定对称密钥的类型。输入表示密钥类型的常量。例如，要创建 AES 密钥，请输入 `-t 31`。

有效值：

- 21：三重 DES (3DES)。
- 31：AES

必需：是

-u

与指定用户共享导入的密钥。此参数向其他 HSM 加密用户 (CU) 授予在加密操作中使用此密钥的权限。

键入一个 ID 或逗号分隔的 HSM 用户 ID 列表，如 5,6。请勿包括当前用户的 HSM 用户 ID。要查找 ID，您可以使用 cloudhsm_mgmt_util 命令行工具中的 listUsers 命令或 key_mgmt_util 命令行工具中的 listUsers 命令。

必需：否

-w

指定包装密钥的密钥句柄。此参数为必需参数。要查找密钥句柄，请使用 findKey 命令。

包装密钥 是 HSM 中的密钥，在导入过程中用于依次加密 (“包装”) 和解密 (“解开包装”) 密钥。只有 AES 密钥才能用作包装密钥。

您可以使用任何 AES 密钥 (任何大小) 作为包装密钥。由于包装密钥将对目标密钥进行包装，然后紧接着将其解开包装，因此您可以使用仅会话型 AES 密钥作为包装密钥。要确定密钥是否可用作包装密钥，请使用 getAttribute 获取 OBJ_ATTR_WRAP 属性 (262) 的值。要创建包装密钥，genSymKey请使用创建 AES 密钥（类型 31）。

如果您使用 -wk 参数指定外部包装密钥，则 -w 包装密钥用于解开包装，而不是包装正在导入的密钥。

> **Note**
>
> 密钥 4 是不受支持的内部密钥。建议您使用作为包装密钥创建和管理的 AES 密钥。

必需：是

-wk

使用指定文件中的 AES 密钥包装正在导入的密钥。输入包含明文 AES 密钥的文件的路径和名称。

当您包含此参数时。imSymKey 使用 -wk 文件中的密钥包装导入的密钥，并使用 HSM 中由 -w 参数指定的密钥对导入的密钥解开包装。-w 和 -wk 参数值必须解析为相同的明文密钥。

默认：在 HSM 上使用包装密钥解开包装。

必需：否

相关 主题

- genSymKey
- exSymKey

- wrapKey
- unWrapKey
- exportPrivateKey
- exportPubKey

insertMaskedObject

key_mgmt_util 中的 insertMaskedObject 命令将文件的遮蔽对象插入到指定 HSM。遮蔽对象是使用 extractMaskedObject 命令从 HSM 中提取的克隆对象。它们只能在插回原始集群后使用。您只能将遮蔽对象插入到生成该对象的同一集群或克隆集群。这包括通过复制跨区域的备份生成的任何克隆版本的原始集群，并使用该备份来创建新的集群。

遮蔽对象是一种用来卸载和同步密钥的有效方式，包括不可提取密钥（即 OBJ_ATTR_EXTRACTABLE 值为 0 的密钥）。这样，无需更新 AWS CloudHSM 配置文件即可在不同区域的相关集群之间安全地同步密钥。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
insertMaskedObject -h

insertMaskedObject -f <filename>
                   [-min_srv <minimum-number-of-servers>]
                   [-timeout <number-of-seconds>]
```

示例

此示例显示了如何使用 insertMaskedObject 将遮蔽对象文件插入到 HSM。

Example ：插入遮蔽对象

此命令从名为 maskedObj 的文件将遮蔽对象插入到 HSM。如果命令成功，insertMaskedObject 将返回从遮蔽对象解密的密钥的密钥句柄和成功消息。

```
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
```

参数

此命令采用以下参数。

-h

显示该命令的命令行帮助。

必需：是

-f

指定遮蔽对象要插入的文件名。

必需：是

-min_srv

指定在 -timeout 参数的值到期之前，同步插入遮蔽对象的服务器的最小数量。如果对象在分配的时间内未同步到指定数量的服务器，则不会插入。

默认值：1

必需：否

-timeout

指定在包含 min-serv 参数时，等待密钥在服务器间同步的秒数。如果没有指定数字，轮询永远继续。

默认值：无限制

必需：否

相关 主题

- extractMaskedObject
- syncKey
- 跨区域复制备份

- 使用先前的 Backup 创建 AWS CloudHSM 集群

IsValidKeyHandlefile

使用 key_mgmt_util 中的 IsValidKeyHandlefile 命令，查明密钥文件是否包含一个真实的私有密钥，还是伪造的 RSA PEM 密钥。伪造 PEM 文件不包含实际的私有密钥资料，而是引用 HSM 中的私有密钥。此类文件可用于建立从您的 Web 服务器到 AWS CloudHSM的 SSL/TLS 卸载。有关更多信息，请参阅 Linux 上的 SSL/TLS 卸载。

> **Note**
>
> IsValidKeyHandlefile 仅适用于 RSA 密钥。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
IsValidKeyHandlefile -h

IsValidKeyHandlefile -f <rsa-private-key-file>
```

示例

这些示例演示如何使用 IsValidKeyHandlefile，确定给定密钥文件是包含真实密钥资料，还是伪造 PEM 密钥资料。

Example ：验证真实的私有密钥

此命令确认名为 `privateKey.pem` 的文件包含真实密钥资料。

```
Command: IsValidKeyHandlefile -f privateKey.pem

Input key file has real private key
```

Example ：使伪造 PEM 密钥失效

此命令从密钥句柄 15 确认名为 `caviumKey.pem` 的文件包含伪造 PEM 密钥资料。

```
Command: IsValidKeyHandlefile -f caviumKey.pem

Input file has invalid key handle: 15
```

参数

此命令采用以下参数。

-h

显示该命令的命令行帮助。

必需：是

-f

指定要检查有效密钥资料的 RSA 私有密钥文件。

必需：是

相关 主题

- getCaviumPriv钥匙
- Linux 上的 SSL/TLS 卸载

listAttributes

key_mgmt_util 中的listAttributes命令列出了 AWS CloudHSM 密钥的属性以及代表它们的常量。您使用这些常量来标识 getAttribute 和 setAttribute 命令中的属性。有关解释密钥属性的帮助，请参阅 密钥属性引用。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

此命令没有输出参数。

```
listAttributes
```

示例

此命令将列出您可以在 key_mgmt_util 中获取和更改的密钥属性和表示它们的常量。有关解释密钥属性的帮助，请参阅 密钥属性引用。

要表示 key_mgmt_util 中 getAttribute 命令中的所有属性，请使用 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
```

相关 主题

- cloudhsm_mgmt_util 中的 listAttributes
- getAttribute
- setAttribute
- 密钥属性引用

listUsers

key_mgmt_util 中的 listUsers 命令可获取 HSM 中的用户以及用户的类型和其他属性。

在 key_mgmt_util 中，listUsers 将返回表示集群中所有 HSM 的输出，即使它们不一致也是如此。要获取有关每个 HSM 中的用户的信息，请使用 cloudhsm_mgmt_util 中的 listUsers 命令。

key_mgmt_util listUsers 和getKeyInfo、中的用户命令是加密用户 (CU) 有权运行的只读命令。剩余的用户管理命令是 cloudhsm_mgmt_util 的一部分。它们由具有用户管理权限的加密员 (CO) 运行。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
listUsers

listUsers -h
```

示例

此命令列出群集中的 HSM 的用户及其属性。您可以使用该User ID属性在其他命令中识别用户，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
```

输出包含以下用户属性：

- 用户 ID：标识 key_mgmt_util 和 cloudhsm_mgmt_util 命令中的用户。

- User type：确定用户可在 HSM 上执行的操作。
- User Name：显示用户的用户定义友好名称。
- MofnPubKey：表示用户是否注册了用于签署法定身份验证令牌的密钥对。
- LoginFailureCnt：表示用户登录失败的次数。
- 2FA：指示用户已启用多因素验证。

参数

-h

显示该命令的帮助信息。

必需：是

相关 主题

- 在 cloudhsm_mgmt_util 中的 listUsers
- findKey
- getAttribute
- getKeyInfo

loginHSM 和 logoutHSM

您可以使用 key_mgmt_util 中的 loginHSM 和 logoutHSM 命令登录到集群中的 HSM 和从其中注销。登录到 HSM 后，您可以使用 key_mgmt_util 来执行多种密钥管理操作，包括公有和私有密钥生成、同步和包装。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util。为了使用 key_mgmt_util 来管理密钥，您必须以加密用户（CU）身份登录到 HSM。

> **Note**
>
> 如果错误登录尝试满五次，则将锁定账户。如果您的集群是在 2018 年 2 月前创建的，则在错误登录尝试满 20 次后锁定账户。要解锁账户，加密员 (CO) 必须在 cloudhsm_mgmt_util 中使用 changePswd 命令重置您的密码。

如果您的集群中有多个 HSM，在锁定账户前，可能会允许您尝试更多次错误登录。这是因为 CloudHSM 客户端跨各 HSM 均衡负载。因此，您的登录尝试可能不是每次都在相同的 HSM 上开始的。如果您要测试此功能，我们建议您在仅具有一个活动 HSM 的集群上测试。

语法

```
loginHSM -h

loginHSM -u <user type>
         { -p | -hpswd } <password>
         -s <username>
```

示例

此示例演示了如何使用 `loginHSM` 和 `logoutHSM` 命令登录和注销集群中的 HSM。

Example ：登录到 HSM

此命令使用用户名 `example_user` 和密码 `aws` 让您以加密用户 (CU) 身份登录到 HSM。输出显示您已登录集群内的所有 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 ：使用隐藏密码登录

此命令与上述示例相同，但是这一次您指定了系统应隐藏密码。

```
Command:  loginHSM -u CU -s example_user -hpswd
```

系统将会提示您输入密码。您输入密码后，系统隐藏密码，输出显示命令成功，且您已连接至 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 ：从 HSM 注销

此命令让您从 HSM 注销。输出显示您已登出集群内的所有 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
```

参数

-h

显示该命令的帮助信息。

-u

指定登录用户类型。要使用 key_mgmt_util，您必须以 CU 身份登录。

必需：是

-s

指定登录用户名。

必需：是

{ -p | -hpswd }

使用 -p指定登录密码。密码在键入时将明文显示。若要隐藏密码，请使用可选 -hpswd 参数代替 -p，然后按照提示进行操作。

必需：是

相关 主题

- exit

setAttribute

key_mgmt_util 中的 setAttribute 命令将仅在当前会话中有效的密钥转换为持久密钥（该密钥在您将其删除前始终存在）。它通过将密钥的令牌属性 (OBJ_ATTR_TOKEN) 的值从 false (0) 更改为 true (1) 来实现此目的。您只能更改自己拥有的密钥的属性。

您还可以使用 cloudhsm_mgmt_util 中的 setAttribute 命令更改标签、包装、解开包装、加密和解密属性。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
setAttribute -h

setAttribute -o <object handle>
             -a 1
```

示例

此示例说明如何将会话密钥转换为持久密钥。

第一个命令使用-sess参数创建仅在当前会话中有效的 192 位 AES 密钥。genSymKey输出显示新会话密钥的密钥句柄为 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 查找当前会话中的会话密钥。输出将验证密钥 262154 是否为会话密钥。

```
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 将密钥 262154 从会话密钥转换为持久密钥。为此，它将密钥令牌属性 (OBJ_ATTR_TOKEN) 的值从 (0) (false) 更改为 (1) (true)。有关解释密钥属性的帮助，请参阅 密钥属性引用。

此命令使用 -o 参数指定密钥句柄 (262154)，并使用 -a 参数指定表示令牌属性 (1) 的常量。当您运行此命令时，它将提示您输入令牌属性的值。唯一有效值是 1 (true)；持久密钥的值。

```
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
```

为确认密钥 262154 现在是持久密钥，此命令将使用 findKey 搜索会话密钥 (-sess 1) 和持久密钥 (-sess 0)。此时，此命令未找到任何会话密钥，但它将返回持久密钥列表中的 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
```

参数

-h

显示该命令的帮助信息。

必需：是

-o

指定目标密钥的密钥句柄。您在每个命令中只能指定一个密钥。要获取密钥的密钥句柄，请使用 findKey。

必需：是

-a

指定表示您要更改的属性的常量。唯一有效值为 1，它表示令牌属性 OBJ_ATTR_TOKEN。

要获取属性及其整数值，请使用 listAttributes。

必需：是

相关 主题

- cloudhsm_mgmt_util 中的 setAttribute
- getAttribute
- listAttributes

- 密钥属性引用

签名

key_mgmt_util 中的 sign 命令使用选定的私有密钥生成文件的签名。

要使用 sign，您必须先拥有 HSM 中的私有密钥。您可以使用 genSymKey、genRSAKeyPair 或 genECCKeyPair 命令生成私有密钥。您也可以使用 importPrivateKey 命令导入私有密钥。有关更多信息，请参阅生成密钥。

sign 命令使用用户指定的签名机制，由一个整数表示，用于对消息文件签名。有关可能的签名机制的列表，请参阅参数。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
sign -h

sign -f <file name>
     -k <private key handle>
     -m <signature mechanism>
     -out <signed file name>
```

示例

该示例演示了如何使用 sign 对文件签名。

Example ：对文件签名

此命令使用句柄为 266309 的私有密钥对名为 messageFile 的文件签名。该命令使用 SHA256_RSA_PKCS (1) 签名机制，并将结果签名文件另存为 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
```

参数

此命令采用以下参数。

-f

要签名的文件的名称

必需：是

-k

要用于签名的私有密钥的句柄。

必需：是

-m

一个表示签名机制的整数，用于签名。可能的机制对应于以下整数：

签名机制	对应整数
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

签名机制	对应整数
ECDSA_SHA1	15
ECDSA_SHA224	16
ECDSA_SHA256	17
ECDSA_SHA384	18
ECDSA_SHA512	19

必需：是

-out

要将签名文件保存的文件的名称。

必需：是

相关 主题

- verify
- importPrivateKey
- GenrSA KeyPair
- GenecC KeyPair
- genSymKey
- 生成密钥

unWrapKey

key_mgmt_util 工具中的 unWrapKey 命令将已包装（已加密）对称密钥或私有密钥从文件导入到 HSM 中。它旨在导入由 key_mgmt_util 中的 wrapKey 命令封装的加密密钥，但它也可以用来解开用其他工具封装的密钥。不过，在这些情况下，我们建议使用 PKCS#11 或 JCE 软件库来解开包装密钥。

导入的密钥的工作原理与生成的密钥类似 AWS CloudHSM。但是，其 OBJ_ATTR_LOCAL attribute 值为零，这表示它们不是在本地生成的。

导入密钥后，请确保标记或删除密钥文件。此命令不会阻止您多次导入相同密钥材料。因此具有不同密钥句柄和相同密钥材料的多个密钥会让跟踪密钥材料的使用情况变得困难并防止其超出加密限制。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
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>]
```

示例

这些示例说明如何使用 unWrapKey 将包装密钥从文件导入 HSM 中。在第一个例子中，我们解开了使用 wrapKey key_mgmt_util 命令包装的密钥，因此该密钥有一个标头。在第二个例子中，我们解开了包装在 key_mgmt_util 之外的密钥，因此没有标头。

Example ：解开包装密钥（带标头）

此命令将 3DES 对称密钥的包装副本导入 HSM 中。该密钥用带有标签 6 的 AES 密钥解开包装，后者与用于包装 3DES 密钥的密钥的加密方式相同。输出显示已解开包装并导入文件中的密钥，并且已导入密钥的句柄是 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 ：解开包装密钥（不带标头）

此命令将 3DES 对称密钥的包装副本导入 HSM 中。该密钥用带有标签 6 的 AES 密钥解开包装，后者与用于包装 3DES 密钥的密钥的加密方式相同。由于未用 key_mgmt_util 包装此 3DES 密钥，因此指定了 noheader 参数及其所需的附加参数：密钥标签 (unwrapped3DES)、密钥类 (4) 和密钥类型 (21)。输出显示已解开包装并导入文件中的密钥，并且已导入密钥的句柄是 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
```

参数

-h

显示该命令的帮助信息。

必需：是

-f

包含包装密钥的文件的路径和名称。

必需：是

-w

指定包装密钥。在 HSM 上输入 AES 密钥或 RSA 密钥的密钥句柄。此参数为必需参数。要查找密钥句柄，请使用 findKey 命令。

要创建包装密钥，请使用genSymKey生成 AES 密钥（类型 31）或使用 GenrSA KeyPair 生成 RSA 密钥对（类型 0）。如果您使用的是 RSA 密钥对，请确保用其中的一个密钥包装密钥，然后用另一个密钥解包。要验证某个密钥是否可以用作包装密钥，请使用 getAttribute 获取 OBJ_ATTR_WRAP 属性的值，该值由常量 262 表示。

必需：是

-sess

创建仅在当前会话中存在的密钥。会话结束后，密钥无法恢复。

如果您只需要一个短暂的密钥，例如用于加密然后快速解密另一个密钥的包装密钥，请使用此参数。对于会话结束后可能需要解密的加密数据，切勿使用会话密钥。

若要将会话密钥更改为永久（令牌）密钥，请使用 setAttribute。

默认：密钥永久有效。

必需：否

-min_srv

指定在 -timeout 参数的值到期之前密钥在其上同步的 HSM 数量的最小值。如果密钥在分配的时间内未同步到指定数量的服务器，则不会创建它。

AWS CloudHSM 自动将每个密钥同步到集群中的每个 HSM。要加快此过程，请将 min_srv 的值设置为小于集群中的 HSM 数，并设置一个较低的超时值。但请注意，一些请求可能无法生成密钥。

默认值：1

必需：否

-timeout

指定命令等待密钥同步到 min_srv 参数指定数量的 HSM 所需的时间（以秒为单位）。

仅当 min_srv 参数也用于此命令时，该参数才有效。

默认：没有超时 该命令无限期等待，仅当密钥同步至最少数量的服务器时才返回。

必需：否

-attest

运行完整性检查，以验证运行集群的固件是否被篡改。

默认：不执行认证检查。

必需：否

-nex

使密钥无法提取。生成的私有密钥无法从 HSM 导出。

默认：密钥可提取。

必需：否

-m

表示包装机制的值。CloudHSM 支持以下机制：

机制	值
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP（有关最大数据大小，请参阅本节后面的说明）	8
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS（有关最大数据大小，请参阅本节后面的说明）。有关即将发生的更改，请参阅下面的注释 1。	12

必需：是

> **Note**
>
> 使用RSA_OAEP包装机制时，可以封装的最大密钥大小由 RSA 密钥的模数和指定哈希的长度决定，如下所示：最大密钥大小 = modulusLengthIn Bytes-(2* Byt hashLengthIn es) -2。
> 使用 RSA_PKCS 封装机制时，可以封装的最大密钥大小由 RSA 密钥的模数决定，如下所示：最大密钥大小 =（字节 -11）。modulusLengthIn

-t

哈希算法	值
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224（对 RSA_AES 和 RSA_OAEP 有效）	6

必需：否

-noheader

如果要解开包装在 key_mgmt_util 之外的密钥，您必须指定此参数以及所有其他关联的参数。

必需：否

> **Note**
>
> 如果您指定此参数，则还必须指定以下 -noheader 参数：

- -l

指定要添加到解开包装的密钥的标签。

必需：是

- -kc

  指定要解开包装的密钥的类。以下是可接受的值：

  3 = 公有-私有密钥对中的私有密钥

  4 = 私有（对称）密钥

  必需：是

- -kt

  指定要解开包装的密钥的类型。以下是可接受的值：

  0 = RSA

  1 = DSA

  3 = ECC

  16 = GENERIC_SECRET

  21 = DES3

  31 = AES

  必需：是

您还可以选择指定以下 -noheader 参数：

- -id

  要添加到解开包装的密钥的 ID。

  必需：否

- -i

  要使用的解开包装初始化向量 (IV)。

  必需：否

[1] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

相关 主题

- wrapKey
- exSymKey
- imSymKey

确认

key_mgmt_util 中的 verify 命令确认文件是否已通过给定密钥签名。要执行此操作，verify 命令将签名文件与源文件进行比较，并根据给定的公有密钥和签名机制分析它们是否与加密相关。可以通过sign操作登录 AWS CloudHSM 文件。

签名机制用参数部分中列出的整数表示。

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
verify -h

verify -f <message-file>
       -s <signature-file>
       -k <public-key-handle>
       -m <signature-mechanism>
```

示例

这些示例演示如何使用 verify 检查特定公有密钥是否用于对给定文件签名。

Example ：验证文件签名

此命令尝试使用 SHA256_RSA_PKCS 签名机制验证名为 hardwarCert.crt 的文件是否通过公有密钥 262276 签名，以生成 hardwareCertSigned 签名文件。由于给定参数表示一个真正的签名关系，所以命令会返回成功消息。

```
Command: verify -f hardwareCert.crt -s hardwareCertSigned -k 262276 -m 1
```

```
Signature verification successful

Cfm3Verify returned: 0x00 : HSM Return: SUCCESS
```

Example ：证明为错误的签名关系

此命令使用 SHA256_RSA_PKCS 签名机制验证名为 hardwareCert.crt 的文件是否通过公有密钥 262276 签名，以生成 userCertSigned 签名文件。由于给定参数不构成真正的签名关系，所以命令会返回错误消息。

```
Command: verify -f hardwarecert.crt -s usercertsigned -k 262276 -m 1
Cfm3Verify returned: 0x1b

CSP Error: ERR_BAD_PKCS_DATA
```

参数

此命令采用以下参数。

-f

原始消息文件的名称。

必需：是

-s

签名文件名称。

必需：是

-k

公有密钥的句柄会被认为用于对文件签名。

必需：是

-m

表示建议签名机制的一个整数，用于对文件签名。可能的机制对应于以下整数：

签名机制	对应整数
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

必需：是

相关 主题

- sign
- getCert
- 生成密钥

wrapKey

key_mgmt_util 中的 wrapKey 命令将对称密钥或私有密钥的加密副本从 HSM 导出到文件中。当您运行 wrapKey 时，指定要导出的密钥、HSM 上用于加密（包装）要导出的密钥的密钥和输出文件。

`wrapKey` 命令将已加密密钥写入您指定的文件，但它不会从 HSM 中删除此密钥，也不会阻止您将它用于加密操作。您可以多次导出相同的密钥。

只有密钥的拥有者（即创建该密钥的 CU 用户）才能导出它。共享密钥的用户可以在加密操作中使用密钥，但无法导出它。

要将加密密钥导回 HSM，请使用unWrapKey。要从 HSM 导出纯文本密钥，请根据需要使用exSymKey或exportPrivateKey。该aesWrapUnwrap命令无法解密（解包）加密的密钥。wrapKey

在运行任何 key_mgmt_util 命令之前，您必须启动 key_mgmt_util 并以加密用户（CU）身份登录到 HSM。

语法

```
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>>]
```

示例

Example

此命令将导出 192 位三重 DES (3DES) 对称密钥 (密钥句柄 7)。它使用 HSM 中的 256 位 AES 密钥 (密钥句柄 14) 包装密钥 7。然后，它将加密的 3DES 密钥写入 `3DES-encrypted.key` 文件。

输出表明，密钥 7 (3DES 密钥) 已成功包装并写入指定文件。加密密钥的长度为 307 个字节。

```
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
```

参数

-h

显示该命令的帮助信息。

必需：是

-k

要导出的密钥的密钥句柄。输入您拥有的对称密钥或私有密钥的密钥句柄。要查找密钥句柄，请使用 findKey 命令。

要验证是否能导出密钥，请使用 getAttribute 命令获取由常量 354 表示的 OBJ_ATTR_EXTRACTABLE 属性的值。有关解释密钥属性的帮助，请参阅 密钥属性引用。

您只能导出您拥有的那些密钥。要查找密钥的所有者，请使用getKeyInfo命令。

必需：是

-w

指定包装密钥。在 HSM 上输入 AES 密钥或 RSA 密钥的密钥句柄。此参数为必需参数。要查找密钥句柄，请使用 findKey 命令。

要创建包装密钥，请使用genSymKey生成 AES 密钥（类型 31）或使用 GenrSA KeyPair 生成 RSA 密钥对（类型 0）。如果您使用的是 RSA 密钥对，请确保用其中的一个密钥包装密钥，然后用另一个密钥解包。要验证某个密钥是否可以用作包装密钥，请使用 getAttribute 获取 OBJ_ATTR_WRAP 属性的值，该值由常量 262 表示。

必需：是

-out

输出文件的路径和名称。当此命令成功时，此文件将包含已导出密钥的加密副本。如果该文件已存在，则命令将覆盖该文件而不发出警告。

必需：是

-m

表示包装机制的值。CloudHSM 支持以下机制：

机制	值
AES_KEY_WRAP_PAD_PKCS5	4
NIST_AES_WRAP_NO_PAD	5
NIST_AES_WRAP_PAD	6
RSA_AES	7
RSA_OAEP（有关最大数据大小，请参阅本节后面的说明）	8
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS（有关最大数据大小，请参阅本节后面的说明）。有关即将发生的更改，请参阅下面的注释 1。	12

必需：是

> **Note**
>
> 使用RSA_OAEP包装机制时，可以封装的最大密钥大小由 RSA 密钥的模数和指定哈希的长度决定，如下所示：最大密钥大小 = (modulusLengthInBytes-2* Bytes hashLengthIn -2)。使用 RSA_PKCS 封装机制时，可以封装的最大密钥大小由 RSA 密钥的模数决定，如下所示：最大密钥大小 =（字节 -11）。modulusLengthIn

-t

表示哈希算法的值。CloudHSM 支持以下算法：

哈希算法	值
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224（对 RSA_AES 和 RSA_OAEP 有效）	6

必需：否

-aad

包含 AAD 的文件名。

> **Note**
>
> 仅对 AES_GCM 和 CLOUDHSM_AES_GCM 机制有效。

必需：否

-noheader

忽略指定 CloudHSM 特定的密钥属性的标头。仅在您希望使用 key_mgmt_util 外的工具解开包装密钥时使用此参数。

必需：否

-i

初始化向量 (IV)（十六进制值）。

> **Note**
>
> 仅当通过 CLOUDHSM_AES_KEY_WRAP 和 NIST_AES_WRAP 机制的 -noheader 参数传递时有效。

必需：否

-iv_file

要写入在响应中获得的 IV 值的文件。

> **Note**
>
> 仅当通过 AES_GCM 机制的 -noheader 参数传递时有效。

必需：否

-tag_size

要与包装的 Blob 一起保存的标签的大小。

> **Note**
>
> 仅当通过 AES_GCM 和 CLOUDHSM_AES_GCM 机制的 -noheader 参数传递时有效。最小标签大小为 8。

必需：否

[1] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

相关 主题

- exSymKey
- imSymKey
- unWrapKey

密钥属性引用

key_mgmt_util 命令使用常量表示 HSM 中密钥的属性。本主题可帮助您标识属性、查找在命令中表示它们的常量以及了解属性值。

密钥的属性是在创建密钥时设置的。要更改令牌属性 (指示密钥是永久密钥还是只存在于会话中)，请使用 key_mgmt_util 中的 setAttribute command in key_mgmt_util. 命令。要更改标签、包装、解开包装、加密和解密属性，使用 cloudhsm_mgmt_util 中的 `setAttribute` 命令。

要获取属性的列表及其常量，请使用 listAttributes。要获取密钥的属性值，请使用 getAttribute。

下表列出了密钥属性、其常量及其有效值。

属性	常量	值
OBJ_ATTR_ALL	512	表示所有属性。
OBJ_ATTR_ALWAYS_SENSITIVE	357	0：False。 1：True。
OBJ_ATTR_CLASS	0	2：公有私有密钥对中的公有密钥。 3：公有私有密钥对中的私有密钥。 4：私有 (对称) 密钥。
OBJ_ATTR_DECRYPT	261	0：False。 1：True。密钥可用于解密数据。
OBJ_ATTR_DERIVE	268	0：False。 1：True。该函数派生密钥。
OBJ_ATTR_DESTROYABLE	370	0：False。 1：True。
OBJ_ATTR_ENCRYPT	260	0：False。 1：True。密钥可用于加密数据。
OBJ_ATTR_EXTRACTABLE	354	0：False。

属性	常量	值
		1：True。密钥可从 HSM 中导出。
OBJ_ATTR_ID	258	用户定义的字符串。在集群中必须是唯一的。默认值是空字符串。
OBJ_ATTR_KCV	371	密钥的密钥检查值。有关更多信息，请参阅其他详细信息。
OBJ_ATTR_KEY_TYPE	256	0：RSA。 1：DSA。 3：EC。 16：普通机密。 18：RC4。 21：三重 DES (3DES)。 31：AES。
OBJ_ATTR_LABEL	3	用户定义的字符串。它在集群中不必是唯一的。
OBJ_ATTR_LOCAL	355	0。False。密钥已导入到 HSM 中。 1：True。
OBJ_ATTR_MODULUS	288	用于生成 RSA 密钥对的模数。对于 EC 密钥，此值表示十六进制格式的 ANSI X9.62 ecPoint 值 Q 的 DER 编码。 对于其他密钥类型，此属性不存在。

属性	常量	值
OBJ_ATTR_MODULUS_BITS	289	用于生成 RSA 密钥对的模数的长度。对于 EC 密钥，这表示用于生成密钥的椭圆曲线 ID。 对于其他密钥类型，此属性不存在。
OBJ_ATTR_NEVER_EXTRACTABLE	356	0：False。 1：True。密钥无法从 HSM 中导出。
OBJ_ATTR_PUBLIC_EXPONENT	290	用于生成 RSA 密钥对的公有指数。 对于其他密钥类型，此属性不存在。
OBJ_ATTR_PRIVATE	2	0：False。 1：True。此属性指示未经身份验证的用户是否可以列出密钥的属性。由于 CloudHSM PKCS#11 提供程序当前不支持公有会话，所有密钥（包括公有/私有密钥对中的公有密钥）的此属性设置为 1。
OBJ_ATTR_SENSITIVE	259	0：False。公有私有密钥对中的公有密钥。 1：True。
OBJ_ATTR_SIGN	264	0：False。 1：True。密钥可用于签名 (私有密钥)。

属性	常量	值
OBJ_ATTR_TOKEN	1	0：False。会话密钥。 1：True。永久密钥。
OBJ_ATTR_TRUSTED	134	0：False。 1：True。
OBJ_ATTR_UNWRAP	263	0：False。 1：True。密钥可用于解密密钥。
OBJ_ATTR_UNWRAP_TEMPLATE	1073742354	值应使用应用于使用此包装密钥解开包装的任何密钥的属性模板。
OBJ_ATTR_VALUE_LEN	353	密钥长度（字节）。
OBJ_ATTR_VERIFY	266	0：False。 1：True。密钥可用于验证 (公有密钥)。
OBJ_ATTR_WRAP	262	0：False。 1：True。密钥可用于加密密钥。
OBJ_ATTR_WRAP_TEMPLATE	1073742353	值应使用属性模板来匹配使用此包装密钥包装的密钥。
OBJ_ATTR_WRAP_WITH_TRUSTED	528	0：False。 1：True。

其他详细信息

密钥检查值 (KCV)

密钥检查值 (KCV) 是 HSM 导入或生成密钥时所产生密钥的 3 字节哈希值或校验和。您也可以在 HSM 之外计算 KCV，例如导出密钥之后。然后，您可对比 KCV 值，以确认密钥的标识和完整性。若要获取密钥 KCV，请使用 getAttribute。

AWS CloudHSM 使用以下标准方法生成密钥检查值：

- 对称密钥：密钥加密零块结果的前 3 个字节。
- 非对称密钥对：公钥 SHA-1 哈希值的前 3 个字节。
- HMAC 密钥：目前不支持 HMAC 密钥 KCV。

AWS CloudHSM 客户端 SDK

使用客户端软件开发工具包，将加密操作从平台或语言应用分流至硬件安全模块 (HSM)。

AWS CloudHSM 提供两个主要版本，客户端 SDK 5 是最新版本。与客户端软件开发工具包 3（之前的系列）相比，它具有多种优势。有关更多信息，请参阅客户端软件开发工具包 5 的优势。有关支持的平台的更多信息，请参阅 客户端软件开发工具包 5 支持的平台。

有关使用客户端软件开发工具包 3 的信息，请参阅 以前的客户端 SDK（客户端 SDK 3）。

the section called “PKCS #11 库”

PKCS #11 是在硬件安全模块 (HSM) 上执行加密操作的标准。 AWS CloudHSM 实施符合 PKCS #11 2.40 版要求的 PKCS #11 库。

the section called “OpenSSL 动态引擎”

AWS CloudHSM OpenSSL 动态引擎允许您通过 OpenSSL API 将加密操作卸载到 CloudHSM 集群。

the section called “JCE 提供程序”

AWS CloudHSM JCE 提供程序符合 Java 加密架构 (JCA)。该提供程序有助于在 HSM 上执行加密操作。

the section called “KSP 和 CNG 提供程序”

Windows AWS CloudHSM 客户端包括 CNG 和 KSP 提供商。目前，只有客户端 SDK 3 支持 CNG 和 KSP 提供程序。

客户端软件开发工具包 5 支持的平台

每个版本的 AWS CloudHSM 客户端 SDK 的基础支持都不同。软件开发工具包中组件的平台支持通常与基本支持相匹配，但并非总是如此。要确定给定组件的平台支持，请首先确保您想要的平台显示在 SDK 的基础部分中，然后在组件部分中检查是否存在任何排除项或任何其他相关信息。

AWS CloudHSM 仅支持 64 位操作系统。

平台支持会随着时间的推移而变化。CloudHSM 客户端软件开发工具包的早期版本可能不支持此处列出的所有操作系统。根据发行说明确定 CloudHSM 客户端软件开发工具包早期版本的操作系统支持情况。有关更多信息，请参阅 AWS CloudHSM 客户端 SDK 的下载内容。

有关此前客户端软件开发工具包支持平台的信息，请参阅 客户端软件开发工具包 3 支持的平台

客户端 SDK 5 不需要客户端进程守护程序。

客户端软件开发工具包 5 对 Linux 的支持情况

支持的平台	架构：x86_64	ARM 架构
Amazon Linux 2	是	是
Amazon Linux 2023	是	是
CentOS 7 (7.8+)	是	不支持
红帽企业 Linux 7 (7.8+)	是	不支持
红帽企业 Linux 8 (8.3+)	是	不支持
红帽企业 Linux 9 (9.2+)	是	是
Ubuntu 20.04 LTS	是	不支持
Ubuntu 22.04 LTS	是	是

注意：SDK 5.4.2 是最后一个提供 CentOS 8 平台支持的版本。有关更多信息，请参阅 CentOS 网站。

客户端软件开发工具包 5 对 Windows 的支持情况

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

客户端软件开发工具包 5 的无服务器支持

- AWS Lambda
- Docker/ECS

客户端软件开发工具包 5 的 HSM 兼容性

hsm1.medium	hsm2m.medium
与客户端 SDK 版本 5.0.0 及更高版本兼容。	与客户端 SDK 版本 5.12.0 及更高版本兼容。

组件支持

CloudHSM CLI

CloudHSM CLI 是一款命令行工具，可帮助管理员管理其集群中的用户。有关更多信息，请参阅 CloudHSM 命令行界面（CLI）。

PKCS #11 库

PKCS #11 库是跨平台组件，可与 Linux 和 Windows 客户端软件开发工具包 5 基础支持相匹配。有关更多信息，请参阅 the section called “客户端软件开发工具包 5 对 Linux 的支持情况” 和 the section called “客户端软件开发工具包 5 对 Windows 的支持情况”。

OpenSSL 动态引擎

OpenSSL 动态引擎是仅限 Linux 的组件，需要 OpenSSL 1.0.2、1.1.1 或 3.x。

JCE 提供程序

JCE 提供者是一个 Java SDK，在所有支持的平台上与 OpenJDK 8、OpenJDK 11、OpenJDK 17 和 OpenJDK 21 兼容。

客户端软件开发工具包 5 的优点

与客户端软件开发工具包 3 相比，客户端软件开发工具包 5 更易于管理，具有卓越的可配置性和更高的可靠性。客户端软件开发工具包 5 还在客户端软件开发工具包 3 的基础之上提供了一些重要优势。

专为无服务器架构设计

客户端软件开发工具包 5 不需要客户端进程守护程序，因此您无需再管理后台服务。为用户提供以下重要帮助：

- 简化应用程序启动进程。若要开始使用 CloudHSM，您只需在运行应用程序之前配置 SDK 即可。
- 您不需要持续运行进程，这使得与 Lambda 和弹性容器服务 (ECS) 等无服务器组件的集成变得更加容易。

更好的第三方集成和更轻松的可移植性

客户端软件开发工具包 5 严格遵循 JCE 规范，在不同 JCE 提供商之间提供更便捷的可移植性和更好的第三方集成

改善用户体验和可配置性

客户端软件开发工具包 5 改善了日志消息的可读性，并提供了更清晰的异常和错误处理机制，使用户可以更轻松进行自助分类。SDK 5 还提供各种配置，这些配置参见配置工具页面。

更广泛的平台支持

客户端软件开发工具包 5 为现代操作平台提供了更多支持。这包括对 ARM 技术的支持，以及对 JCE、PKCS #11 和 Op enSSL 的优化支持。有关更多信息，请参阅支持平台。

其他功能和机制

客户端软件开发工具包 5 包括客户端软件开发工具包 3 中没有的其他功能和机制，客户端软件开发工具包 5 未来还将继续添加更多机制。

从客户端软件开发工具包 3 迁移到客户端软件开发工具包 5

有关从客户端 SDK 3 迁移到客户端 SDK 5 的详细说明，请参阅每个客户端 SDK 的迁移说明：

- 将你的 PKCS #11 库从客户端 SDK 3 迁移到客户端 SDK 5
- 将 OpenSSL 动态引擎从客户端 SDK 3 迁移到客户端 SDK 5
- 将您的 JCE 提供程序从客户端 SDK 3 迁移到客户端 SDK 5
- 从客户端 SDK 3 CMU 和 KMU 迁移到客户端 SDK 5 CloudHSM CLI

如需了解 CloudHSM CLI 不支持的功能或用例，请联系支持人员。

> **Note**
>
> Windows 平台现在支持客户端 SDK 5 PKCS #11 库。它可以处理压缩天然气和KSP提供商可以而且应该被视为替代品的大多数用例。KSP 目前仅在客户端 SDK 3 中可用。

PKCS #11 库

PKCS #11 是在硬件安全模块 (HSM) 上执行加密操作的标准。 AWS CloudHSM 实施符合 PKCS #11 2.40 版要求的 PKCS #11 库。

有关引导的信息，请参阅 连接到集群。有关疑难解答，请参阅PKCS#11 库的已知问题。

有关使用客户端软件开发工具包 3 的信息，请参阅 以前的客户端 SDK（客户端 SDK 3）。

主题

- 为客户端 SDK 5 安装 PKCS #11 库
- 向 PKCS #11 库进行身份验证
- PKCS #11 库支持的密钥类型
- PKCS #11 库支持的机制
- PKCS #11 库支持的 API 操作
- PKCS #11 库支持的密钥属性
- PKCS #11 库的代码示例
- 将你的 PKCS #11 库从客户端 SDK 3 迁移到客户端 SDK 5
- PKCS #11 高级配置

为客户端 SDK 5 安装 PKCS #11 库

本主题提供安装适用于 git 客户端 SDK 5 版本系列的最新版本的 PKCS #11 库的说明。有关客户端软件开发工具包 或 PKCS #11 库的更多信息，请参阅使用客户端软件开发工具包 和 PKCS #11 库。

安装

使用客户端软件开发工具包 5，您无需安装或运行客户端进程守护程序。

要使用客户端软件开发工具包 5 运行单个 HSM 集群，必须首先通过将 `disable_key_availability_check` 设置为 `True`，以管理客户端密钥持久性设置。有关更多信息，请参阅密钥同步和客户端软件开发工具包 5 配置工具。

有关客户端软件开发工具包 5 中的 PKCS #11 库的更多信息，请参阅 PKCS #11 库。

> **Note**
>
> 要使用客户端软件开发工具包 5 运行单个 HSM 集群，必须首先通过将 `disable_key_availability_check` 设置为 `True`，以管理客户端密钥持久性设置。有关更多信息，请参阅密钥同步和客户端软件开发工具包 5 配置工具。

安装和配置 PKCS #11 库

1. 使用以下命令下载和安装 PKCS #11 库。

   Amazon Linux 2

   在 X86_64 架构上安装适用于 Amazon Linux 2 的 PKCS #11 库：

   ```
   $ 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
   ```

   在 ARM64 架构上安装适用于 Amazon Linux 2 的 PKCS #11 库：

   ```
   $ 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 架构上安装适用于亚马逊 Linux 2023 的 PKCS #11 库：

   ```
   $ 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 架构上安装适用于亚马逊 Linux 2023 的 PKCS #11 库：

```
$ 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+)

在 X86_64 架构上安装适用于 CentOS 7.8+ 的 PKCS #11 库：

```
$ 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 架构上安装适用于 RHEL 7 的 PKCS #11 库：

```
$ 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 架构上安装适用于 RHEL 8 的 PKCS #11 库：

```
$ 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 架构上安装适用于 RHEL 9 的 PKCS #11 库：

```
$ 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 架构上安装适用于 RHEL 9 的 PKCS #11 库：

```
$ 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

在 X86_64 架构上安装适用于 Ubuntu 20.04 LTS 的 PKCS #11 库：

```
$ 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

在 X86_64 架构上安装适用于 Ubuntu 22.04 LTS 的 PKCS #11 库：

```
$ 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 架构上安装适用于 Ubuntu 22.04 LTS 的 PKCS #11 库：

```
$ 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

在 X86_64 架构上安装适用于 Windows 客户管理型策略 2016 的 PKCS #11 库：

1. 下载 适用于客户端软件开发工具包 5 的 PKCS #11 库。
2. 使用 Windows 管理权限运行 PKCS #11 库安装程序 (AWSCloudHSMPKCS11-latest.msi)。

Windows Server 2019

在 X86_64 架构上安装适用于 Windows 客户管理型策略 2019 的 PKCS #11 库：

1. 下载 适用于客户端软件开发工具包 5 的 PKCS #11 库。
2. 使用 Windows 管理权限运行 PKCS #11 库安装程序 (AWSCloudHSMPKCS11-latest.msi)。

2. 使用配置工具指定证书发放位置。有关说明，请参阅指定颁发证书的位置。
3. 要连接到集群，请参阅引导客户端软件开发工具包。
4. 您可在以下位置找到 PKCS #11 库的文件：

- Linux 二进制文件、配置脚本和日志文件：

```
/opt/cloudhsm
```

Windows 二进制文件：

```
C:\ProgramFiles\Amazon\CloudHSM
```

Windows 配置脚本与日志文件：

```
C:\ProgramData\Amazon\CloudHSM
```

向 PKCS #11 库进行身份验证

当您使用 PKCS #11 库时，您的应用程序将作为 HSM 中的特定加密用户（CU）运行。您的应用程序只能查看和管理 CU 拥有和共享的密钥。您可以在 HSM 中使用现有 CU，也可以为应用程序创建新 CU。有关 CU 管理的信息，请参见通过 CloudHSM CLI 管理 HSM 用户 以及 通过 CloudHSM 管理实用程序 (CMU) 管理 HSM 用户。

要将 CU 指定给 PKCS #11 库，请使用 PKCS #11 C_Login 函数的 PIN 参数。对于 AWS CloudHSM，pin 参数采用以下格式：

```
<CU_user_name>:<password>
```

例如，以下命令会将 PKCS #11 库 PIN 设置为用户名为 `CryptoUser`、密码为 `CUPassword123!` 的 CU。

```
CryptoUser:CUPassword123!
```

PKCS #11 库支持的密钥类型

PKCS #11 库支持以下密钥类型。

密钥类型	描述
AES	生成 128 位、192 位和 256 位 AES 密钥。
三重 DES（3DES、DESede）	生成 192 位三重 DES 密钥。有关即将发生的更改，请参阅下面的注释 1。
EC	通过 secp224r1 (P-224)、secp256r1 (P-256)、secp256k1（区块链）、secp384r1 (P-384) 和 secp521r1 (P-521) 曲线生成密钥。
GENERIC_SECRET	生成 1 至 800 字节的通用密钥。
RSA	生成 2048 位到 4096 位的 RSA 密钥，增量为 256 位。

[1] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

PKCS #11 库支持的机制

PKCS #11 库符合 2.40 版 PKCS #11 规格。要使用 PKCS # 11 调用加密功能，请使用给定机制调用函数。以下章节汇总了 AWS CloudHSM支持的函数和机制的组合。

PKCS #11 库支持以下算法：

- 加密和解密 - AES-CBC、AES-CTR、AES-ECB、AES-GCM、DES3-CBC、DES3-ECB、RSA-OAEP 和 RSA-PKCS
- 签名和验证 - RSA、HMAC 和 ECDSA；带和不带哈希
- 哈希/摘要 - SHA1、SHA224、SHA256、SHA384 和 SHA512
- 密钥包装 - AES 密钥包装[1]、AES-GCM、RSA-AES 和 RSA-OAEP

主题

生成密钥与密钥对功能

PKCS #11 库的 AWS CloudHSM 软件库允许您使用以下机制生成密钥和密钥对功能。

- `CKM_RSA_PKCS_KEY_PAIR_GEN`
- `CKM_RSA_X9_31_KEY_PAIR_GEN` – 此机制的功能与 `CKM_RSA_PKCS_KEY_PAIR_GEN` 机制相同，但为生成 p 和 q 提供了更强有力的保证。

- CKM_EC_KEY_PAIR_GEN
- CKM_GENERIC_SECRET_KEY_GEN
- CKM_AES_KEY_GEN
- CKM_DES3_KEY_GEN – 即将在 5 脚注中列出的更改。

签署并验证功能

PKCS #11 库的 AWS CloudHSM 软件库允许您将以下机制用于签名和验证功能。通过客户端软件开发工具包 5，数据将在软件中进行本地哈希处理。这意味着可由 SDK 执行哈希处理的数据大小没有限制。

通过客户端软件开发工具包 5 RSA 和 ECDSA，哈希处理在本地执行，无数据限制。使用 HMAC 时会有数据限制。有关更多信息，请参阅脚注 2。

RSA

- CKM_RSA_X_509
- CKM_RSA_PKCS – 仅限单部分操作。
- CKM_RSA_PKCS_PSS – 仅限单部分操作。
- 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 – 仅限单部分操作。
- 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

签名、恢复和验证恢复功能

客户端软件开发工具包 5 不支持 “签名恢复” 和 “验证恢复” 功能。

摘要功能

PKCS #11 库的 AWS CloudHSM 软件库允许您将以下机制用于摘要函数。通过客户端软件开发工具包 5，数据将在软件中进行本地哈希处理。这意味着可由 SDK 执行哈希处理的数据大小没有限制。

- CKM_SHA_1
- CKM_SHA224
- CKM_SHA256
- CKM_SHA384
- CKM_SHA512

加密和解密功能

PKCS #11 库的 AWS CloudHSM 软件库允许您将以下机制用于加密和解密功能。

- CKM_RSA_X_509
- CKM_RSA_PKCS – 仅限单部分操作。脚注 5 中列出了即将进行的更改。
- CKM_RSA_PKCS_OAEP – 仅限单部分操作。
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_DES3_CBC – 即将在 5 脚注中列出的更改。
- CKM_DES3_ECB – 即将在 5 脚注中列出的更改。
- CKM_DES3_CBC_PAD – 即将在 5 脚注中列出的更改。
- CKM_AES_GCM [1, 2]
- CKM_CLOUDHSM_AES_GCM[3]

派生密钥功能

PKCS #11 库的 AWS CloudHSM 软件库允许您将以下机制用于派生函数。

- CKM_SP800_108_COUNTER_KDF

包装和解包功能

PKCS #11 库的 AWS CloudHSM 软件库允许您使用以下机制来实现 Wrap 和 Unwrap 函数。

有关其他 AES 密钥包装的其他信息，请参阅 AES 密钥包装。

- CKM_RSA_PKCS – 仅限单部分操作。脚注 5 中列出了即将进行的更改。
- 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]

每种机制的最大数据大小

下表列出了每个机制的最大数据大小设置：

最大数据集大小

机制	以字节为单位的最大数据大小
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

机制注释

- [1] 在执行 AES-GCM 加密时，HSM 不会接受应用程序中的初始化向量 (IV) 数据。您必须使用其生成的 IV。HSM 提供的 12 字节 IV 将写入您提供的 CK_GCM_PARAMS 参数结构的 pIV 元素所指向的内存参考。为了防止用户混淆，版本 1.1.1 及更高版本中的 PKCS #11 开发工具包将在初始化 AES-GCM 加密时确保该 pIV 指向已清零的缓冲区。

- [2] 使用以下任何机制对数据进行操作时，如果数据缓冲区超出最大数据大小，则操作会导致错误。对此机制，所有数据处理均应在 HSM 内发生。有关每种机制的最大数据大小集的信息，请参阅 每种机制的最大数据大小。
- [3] 供应商定义的机制。为了使用 CloudHSM 供应商定义的机制，PKCS #11 应用程序必须在编译过程中包含 `/opt/cloudhsm/include/pkcs11t.h`。

  CKM_CLOUDHSM_AES_GCM：这种专有机制是标准 CKM_AES_GCM 的编程更安全的替代方案。它将 HSM 生成的 IV 附加到密文，而不是将其写回密码初始化期间提供的 CK_GCM_PARAMS 结构中。您可以将此机制与 C_Encrypt、C_WrapKey、C_Decrypt 和 C_UnwrapKey 函数一起使用。使用此机制时，CK_GCM_PARAMS 结构中的 pIV 变量必须设置为 NULL。将此机制与 C_Decrypt 和 C_UnwrapKey 一起使用时，IV 预计会被放在正在解开包装的密文之前。

  CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD：带 PKCS #5 填充的 AES 密钥包装。

  CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD：零填充的 AES 密钥包装。
- [4] 以下 CK_MECHANISM_TYPE 和 CK_RSA_PKCS_MGF_TYPE 出于 CKM_RSA_PKCS_OAEP 作为 CK_RSA_PKCS_OAEP_PARAMS 受到支持：
  - 使用 CKG_MGF1_SHA1 的 CKM_SHA_1
  - 使用 CKG_MGF1_SHA224 的 CKM_SHA224
  - 使用 CKG_MGF1_SHA256 的 CKM_SHA256
  - 使用 CKM_MGF1_SHA384 的 CKM_SHA384
  - 使用 CKM_MGF1_SHA512 的 CKM_SHA512
- [5] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

PKCS #11 库支持的 API 操作

适用于 PKCS #11 库支持以下 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 库支持的密钥属性

密钥对象可以是公有、私有或秘密密钥。通过属性指定密钥对象上允许的操作。属性是在创建密钥对象时定义的。当您使用 CloudHSM 的 PKCS#11 开发工具包时，我们将按照 PKCS #11 标准的要求分配默认值。

AWS CloudHSM 不支持 PKCS #11 规范中列出的所有属性。对于我们支持的所有属性，我们符合此规范。这些属性列在各自的表中。

创建、修改或复制对象的加密函数（如 C_CreateObject、C_GenerateKey、C_GenerateKeyPair、C_UnwrapKey 和 C_DeriveKey）将属性模板作为它们的一个参数。有关在对象创建期间传递属性模板的更多信息，请参阅通过 PKCS #11 库生成密钥了解样例。

解释 PKCS#11 库属性表格

PKCS#11 库表格包含依密钥类型而不同的属性的列表。它表示在使用特定加密函数时，特定密钥类型是否支持给定属性。 AWS CloudHSM

图例：

- ✓ 表示 CloudHSM 对特定密钥类型支持此属性。
- ✖ 表示 CloudHSM 对特定密钥类型不支持此属性。
- R 表示特定密钥类型的属性值设置为只读的。
- S 表示无法通过 `GetAttributeValue` 读取该属性，因为它是敏感属性。
- 默认值列中的空单元格表示没有给该属性分配特定的默认值。

GenerateKeyPair

属性	密钥类型				默认值
	EC 私有	EC 公有	RSA 私有	RSA 公有	
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

属性	密钥类型				默认值
CKA_DERIVE	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	True
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

属性	密钥类型				默认值
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	
CKA_SENSITIVE	✓[1]	✖	✓[1]	✖	True
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	
CKA_EXTRACTABLE	✓	✖	✓	✖	True
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	
CKA_MODULUS	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✓[2]	
CKA_PRIME_1	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	

属性	密钥类型				默认值
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

属性	密钥类型			默认值
	AES	DES3	普通机密	
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	

属性	密钥类型			默认值
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✓	✓	✖	False
CKA_DECRYPT	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	True
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	True
CKA_VERIFY_RECOVER	✖	✖	✖	

属性	密钥类型			默认值
CKA_WRAP	✓	✓	✖	False
CKA_WRAP_TEMPLATE	✓	✓	✖	
CKA_TRUSTED	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✓	✓	False
CKA_UNWRAP	✓	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✓	✖	
CKA_SENSITIVE	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	✖	✖	✖	
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_MODULUS	✖	✖	✖	

属性	密钥类型			默认值
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]	

属性	密钥类型			默认值
CKA_CHECK_VALUE	R	R	R	

CreateObject

属性	密钥类型							默认值
	EC 私有	EC 公有	RSA 私有	RSA 公有	AES	DES3	普通机密	
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

属性	密钥类型							默认值
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[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
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖	
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓	False
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖	False

属性	密钥类型							默认值
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	R	R	R	
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	R	R	R	
CKA_MODULUS	✖	✖	✓[2]	✓[2]	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✓	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✓	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✓	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✓	✖	✖	✖	✖	

属性	密钥类型							默认值
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

属性	密钥类型					默认值
	EC 私有	RSA 私有	AES	DES3	普通机密	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	

属性	密钥类型					默认值
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	False
CKA_TOKEN	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✖	✖	✓	✓	✖	False
CKA_DECRYPT	✖	✓	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	False

属性	密钥类型					默认值
CKA_VERIFY	✖	✖	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✓	✓	✖	False
CKA_UNWRAP	✖	✓	✓	✓	✖	False
CKA_SENSITIVE	✓	✓	✓	✓	✓	True
CKA_EXTRACTABLE	✓	✓	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	
CKA_MODULUS	✖	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	✖	

属性	密钥类型					默认值
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	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	

DeriveKey

属性	密钥类型			默认值
	AES	DES3	普通机密	
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
CKA_DESTROYABLE	✓[1]	✓[1]	✓[1]	True
CKA_SIGN	✓	✓	✓	False

属性	密钥类型			默认值
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	False
CKA_UNWRAP	✓	✓	✖	False
CKA_SENSITIVE	R	R	R	True
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	

属性	密钥类型			默认值
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

属性	密钥类型						
	EC 私有	EC 公有	RSA 私有	RSA 公有	AES	DES3	普通机密
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	✖	✖	✓	✖	✓	✓	✖
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓

属性	密钥类型						
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓
CKA_SIGN_RECOVER	✖	✖	✓	✖	✖	✖	✖
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓
CKA_VERIFY_RECOVER	✖	✖	✖	✓	✖	✖	✖
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✓
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓

属性	密钥类型						
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_NEVER_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	R	R
CKA_MODULUS	✖	✖	✓	✓	✖	✖	✖
CKA_MODULUS_BITS	✖	✖	✖	✓	✖	✖	✖
CKA_PRIME_1	✖	✖	S	✖	✖	✖	✖
CKA_PRIME_2	✖	✖	S	✖	✖	✖	✖
CKA_COEFFICIENT	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_1	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_2	✖	✖	S	✖	✖	✖	✖
CKA_PRIVATE_EXPONENT	✖	✖	S	✖	✖	✖	✖

属性	密钥类型						
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
CKA_VALUE	S	✖	✖	✖	✓	✓	✓
CKA_VALUE_LEN	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	✖

属性注释

- [1] 此属性由固件部分支持且必须明确地仅设置为默认值。
- [2] 必需属性。

修改属性

对象的一些属性可以在对象创建后修改，有一些则不能。若要修改属性，请使用 cloudhsm_mgmt_util 中的 setAttribute 命令。您还可通过 cloudhsm_mgmt_util 中的 listAttribute 命令派生出属性列表和代表这些属性的常量。

以下列表显示了对象创建后允许修改的属性：

- CKA_LABEL
- CKA_TOKEN

> **Note**
>
> 修改仅允许将会话密钥更改为令牌密钥。使用 key_mgmt_util 中的 setAttribute 命令更改属性值。

- CKA_ENCRYPT
- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

> **Note**
>
> 此属性支持密钥派生。它对所有公有密钥必须为 False，并且无法设置为 True。对于秘密和 EC 私有密钥，它可以设置为 True 或 False。

- CKA_TRUSTED

> **Note**
>
> 此属性仅可通过加密员（CO）设置为 True 或 False。

- CKA_WRAP_WITH_TRUSTED

> **Note**
>
> 将此属性应用于可导出的数据密钥，以表示此密钥只能更换为标记CKA_TRUSTED的密钥。将 CKA_WRAP_WITH_TRUSTED 设置为 true 后，该属性将变为只读，并且您无法更改或移除该属性。

解释错误代码

在模板中指定特定密钥不支持的属性会导致错误。下表包含当您违反规范时生成的错误代码：

错误代码	描述
CKR_TEMPLATE_INCONSISTENT	当您在属性模板中指定一个属性而该属性符合 PKCS #11 规范但不受 CloudHSM 支持时，您会收到此错误。
CKR_ATTRIBUTE_TYPE_INVALID	当您在属性模板中检索符合 PKCS #11 规范但不受 CloudHSM 支持的属性的值时，您会收到此错误。
CKR_ATTRIBUTE_INCOMPLETE	当您未在属性模板中指定必需属性时，您会收到此错误。
CKR_ATTRIBUTE_READ_ONLY	当您在属性模板中指定只读属性时，您会收到此错误。

PKCS #11 库的代码示例

上的代码示例 GitHub 向您展示了如何使用 PKCS #11 库完成基本任务。

先决条件

在运行示例之前，请执行以下步骤以设置您的环境：

- 为客户端软件开发工具包 5 安装和配置 PKCS #11 库。
- 设置加密用户 (CU)。您的应用程序使用此 HSM 账户在 HSM 运行代码示例。

代码示例

PKCS #11 AWS CloudHSM 软件库的代码样本可在上找到。GitHub此存储库包括有关如何使用 PKCS#11 执行常见操作的示例，包括加密、解密、签名和验证。

- 生成密钥（AES、RSA、EC）
- 列出密钥属性

- 使用 AES GCM 加密和解密数据
- 使用 AES_CTR 加密和解密数据
- 使用 3DES 加密和解密数据
- 使用 RSA 对数据进行签名和验证
- 使用 HMAC KDF 派生密钥
- 使用 AES 对密钥进行包装和解开包装（使用 PKCS #5 填充）
- 使用 AES 对密钥进行包装和解开包装（无填充）
- 使用 AES 包装和解开密钥（使用零填充）
- 使用 AES-GCM 对密钥进行包装和解开包装
- 使用 RSA 对密钥进行包装和解开包装

将你的 PKCS #11 库从客户端 SDK 3 迁移到客户端 SDK 5

使用本主题将您的 PKCS #11 库从客户端 SDK 3 迁移到客户端 SDK 5。有关迁移的好处，请参阅客户端软件开发工具包 5 的优点。

在中 AWS CloudHSM，客户应用程序使用 AWS CloudHSM 客户端软件开发套件 (SDK) 执行加密操作。客户端 SDK 5 是主要 SDK，它继续添加新功能和平台支持。

要查看所有提供商的迁移说明，请参阅从客户端软件开发工具包 3 迁移到客户端软件开发工具包 5。

通过解决重大变化做好准备

查看这些重大更改，并在开发环境中相应地更新您的应用程序。

包装机制已改变

客户端 SDK 3 机制	等效客户端 SDK 5 机制
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

客户端 SDK 3 机制	等效客户端 SDK 5 机制
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

在客户端 SDK 3 中，您可以使用 ECDH 并指定 KDF。此功能目前在客户端 SDK 5 中不可用。如果您的应用程序需要此功能，请联系支持人员。

按键手柄现在是特定于会话的

要在客户端软件开发工具包 5 中成功使用密钥句柄，您必须每次运行应用程序时均获得密钥句柄。如果您现有的应用程序将在不同的会话中使用相同的密钥句柄，则每次运行应用程序时都必须修改代码以获取密钥句柄。有关检索密钥句柄的信息，请参阅此 AWS CloudHSM PKCS #11 示例。此更改符合 PKCS #11 2.40 规范。

迁移到客户端 SDK 5

按照本节中的说明从客户端 SDK 3 迁移到客户端 SDK 5。

> **Note**
>
> 客户端 SDK 5 目前不支持亚马逊 Linux、Ubuntu 16.04、Ubuntu 18.04、CentOS 6、CentOS 8 和 RHEL 6。如果您目前正在将其中一个平台与客户端 SDK 3 一起使用，则在迁移到客户端 SDK 5 时需要选择其他平台。

1. 卸载客户端 SDK 3 的 PKCS #11 库。

   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. 卸载客户端 SDK 的客户端守护程序 3。

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-client
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-client
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-client
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-client
   ```

   > **Note**
   >
   > 需要再次启用自定义配置。

3. 按照中的步骤安装客户端 SDK PKCS #11 库。为客户端 SDK 5 安装 PKCS #11 库
4. 客户端 SDK 5 引入了一种新的配置文件格式和命令行引导工具。要引导您的 Client SDK 5 PKCS #11 库，请按照用户指南中列出的说明进行操作。引导客户端软件开发工具包
5. 在您的开发环境中，测试您的应用程序。在最终迁移之前，对现有代码进行更新，以解决重大更改。

相关 主题

- 以下方面的最佳实践 AWS CloudHSM

PKCS #11 高级配置

P AWS CloudHSM KCS #11 提供程序包括以下高级配置，这不是大多数客户使用的常规配置的一部分。这些配置还提供了其他功能。

- 连接至多个 PKCS#11 插槽
- PKCS #11 重试配置

连接至多个 PKCS#11 插槽

客户端软件开发工具包 5 PKCS #11 库内的单个插槽表示单独连接至 AWS CloudHSM中的集群。您可通过客户端软件开发工具包 5 配置 PKCS11 库，以通过多个插槽将用户连接至单个 PKCS#11 应用的多个 CloudHSM 集群。

按照本主题中的说明，让您的应用程序使用多插槽功能连接多个集群。

主题

- 多插槽使用先决条件
- 配置 PKCS #11 库以实现多插槽功能
- configure-pkcs11 添加集群
- configure-pkcs11 remove-cluster

多插槽使用先决条件

- 您要连接的两个或更多 AWS CloudHSM 集群及其集群证书。
- 已正确配置安全组的 EC2 实例，可连接至上述所有集群。有关如何设置集群和客户端实例的更多信息，请参阅入门 AWS CloudHSM。
- 若要设置多插槽功能，您必须已经下载并安装了 PKCS #11 库。如果您尚未执行此操作，请参阅 ??? 中的说明。

配置 PKCS #11 库以实现多插槽功能

若要配置 PKCS #11 库以实现多插槽功能，请执行以下操作：

1. 确定要通过多插槽功能连接的集群。
2. 按照 ??? 中的说明，将这些集群添加至 PKCS #11 配置
3. 下次运行 PKCS #11 应用程序时，它将包含多插槽功能。

configure-pkcs11 添加集群

当连接至多个 PKCS #11 插槽时，使用 configure-pkcs11 add-cluster 命令将集群添加至您的配置中。

语法

```
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]
```

示例

使用 **cluster-id** 参数添加集群

Example

使用 configure-pkcs11 add-cluster 和 `cluster-id` 参数，将集群添加至 (ID 为 `cluster-1234567`) 您的配置。

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**
>
> 如果使用 `cluster-id` 参数和 configure-pkcs11 add-cluster 无法添加集群，请参见以下示例，以获取也需要通过 `--region` 和 `--endpoint` 参数识别待添加集群的、更长版本的命令。例如，如果集群的区域与配置默认 AWS CLI 配置区域不同，则应使用 `--region` 参数使用正确区域。此外，您还可以指定用于调用的 AWS CloudHSM API 终端节点，这可能是各种网络设置所必需的，例如使用不使用默认 DNS 主机名的 VPC 接口终端节点。 AWS CloudHSM

使用 **`cluster-id`**、**`endpoint`** 和 **`region`** 参数添加集群

Example

使用 configure-pkcs11 add-cluster、`cluster-id`、`endpoint` 和 `region` 参数，将集群 (ID 为 `cluster-1234567`) 添加至您的配置。

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
```

有关 `--cluster-id`、`--region` 和 `--endpoint` 参数的更多信息，请参阅 the section called “参数”。

参数

--cluster-id ***<Cluster ID>***

进行 DescribeClusters 调用以查找集群中与集群 ID 关联的所有 HSM 弹性网络接口（ENI）IP 地址。系统将 ENI IP 地址添加到 AWS CloudHSM 配置文件中。

> **Note**
>
> 如果您在无法访问公共互联网的 VPC 中使用来自 EC2 实例的--cluster-id参数，则必须创建要连接的接口 VPC 终端节点 AWS CloudHSM。有关 VPC 端点的更多信息，请参阅???。

必需：是

--endpoint ***<Endpoint>***

指定用于进行DescribeClusters呼叫的 AWS CloudHSM API 端点。设置此选项时，您必须与 --cluster-id 结合。

必需：否

--hsm-ca-cert <HsmCA Certificate Filepath>

指定 HSM CA 证书的文件路径。

必需：否

--region ***<Region>***

指定集群所在区域。设置此选项时，您必须与 --cluster-id 结合。

如果您不提供 --region 参数，则系统会通过尝试读取 AWS_DEFAULT_REGION 或 AWS_REGION 环境变量选择区域。如果未设置这些变量，则系统会在 AWS Config 文件中检查与您的配置文件关联的区域（通常 ~/.aws/config），除非您在 AWS_CONFIG_FILE 环境变量中指定了其他文件。如果以上均未设置，则系统默认为 us-east-1 区域。

必需：否

--server-client-cert-file <Client Certificate Filepath>

用于 TLS 客户端服务器相互身份验证的客户端认证路径。

只有当您不希望使用客户端软件开发工具包 5 中包含的默认密钥和 SSL/TLS 证书时，才使用此选项。设置此选项时，您必须与 `--server-client-key-file` 结合。

必需：否

--server-client-key-file <Client Key Filepath>

用于 TLS 客户端-服务器相互身份验证的客户端密钥的路径。

只有当您不希望使用客户端软件开发工具包 5 中包含的默认密钥和 SSL/TLS 证书时，才使用此选项。设置此选项时，您必须与 `--server-client-cert-file` 结合。

必需：否

configure-pkcs11 remove-cluster

当连接至多个 PKCS #11 插槽时，使用 configure-pkcs11 remove-cluster 命令将集群从可用的 PKCS #11 插槽中移除。

语法

```
configure-pkcs11 remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

示例

使用 **`cluster-id`** 参数移除集群

Example

使用 configure-pkcs11 remove-cluster 和 `cluster-id` 参数，从您的配置中移除集群 (ID 为 `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
```

有关 `--cluster-id` 参数的更多信息，请参阅 the section called “参数”。

参数

--cluster-id *<Cluster ID>*

要从配置中删除的集群 ID

必需：是

PKCS #11 重试命令

客户端软件开发工具包 5.8.0 及更高版本具有内置的自动重试策略，该策略将从客户端重试 HSM 节流的操作。当 HSM 因忙于执行之前的操作而无法接受更多请求而对操作进行节流时，客户端软件开发工具包会尝试重试节流操作最多 3 次，同时进行指数级退避。这种自动重试策略可设置为以下两种模式之一：关闭和标准。

- 关闭：客户端软件开发工具包 不会对 HSM 的任何节流操作执行任何重试策略。
- 标准：这是客户端软件开发工具包版本 5.8.0 及更高版本的默认模式。在此模式下，客户端软件开发工具包将通过指数级退避自动重试节流操作。

有关更多信息，请参阅 HSM 节流。

将重试命令设置为关闭模式

Linux

在 Linux 上将客户端软件开发工具包 5 的重试命令设置为 off

- 您可以使用以下命令将重试配置设置为 off 模式：

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --default-retry-mode off
```

Windows

在 Windows 上将客户端软件开发工具包 5 的重试命令设置为 off

- 您可以使用以下命令将重试配置设置为 off 模式：

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-pkcs11.exe --default-retry-mode off
```

OpenSSL 动态引擎

AWS CloudHSM OpenSSL 动态引擎允许您通过 OpenSSL API 将加密操作卸载到 CloudHSM 集群。

AWS CloudHSM 提供了 OpenSSL 动态引擎，您可以在中阅读该引擎。Linux 上的 SSL/TLS 分载有关 AWS CloudHSM 与 OpenSSL 配合使用的示例，请参阅此 AWS 安全博客。有关 SDK 平台支持的更多信息，请参阅 the section called “支持的平台”。有关疑难解答，请参阅OpenSSL Dynamic Engine 的已知问题。

有关使用客户端软件开发工具包 3 的信息，请参阅 以前的客户端 SDK（客户端 SDK 3）。

有关更多信息，请参阅以下主题。

主题

- 安装 OpenSSL 动态引擎
- OpenSSL 动态引擎密钥类型
- OpenSSL 动态引擎机制
- 将 OpenSSL 动态引擎从客户端 SDK 3 迁移到客户端 SDK 5
- OpenSSL 的高级配置

安装 OpenSSL 动态引擎

> **Note**
>
> 要使用客户端软件开发工具包 5 运行单个 HSM 集群，必须首先通过将 `disable_key_availability_check` 设置为 `True`，以管理客户端密钥持久性设置。有关更多信息，请参阅密钥同步和客户端软件开发工具包 5 配置工具。

安装和配置 OpenSSL 动态引擎

1. 使用以下命令下载和安装 OpenSSL 引擎。

   Amazon Linux 2

   在 x86_64 架构上安装适用于亚马逊 Linux 2 的 OpenSSL 动态引擎：

   ```
   $ 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 架构上安装适用于亚马逊 Linux 2 的 OpenSSL 动态引擎：

   ```
   $ 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 x86_64 架构上安装适用于亚马逊 Linux 2023 的 OpenSSL 动态引擎：

   ```
   $ 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 架构上安装适用于亚马逊 Linux 2023 的 OpenSSL 动态引擎：

   ```
   $ 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 架构上安装适用于 CentOS 7 的 OpenSSL 动态引擎：

```
$ 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 x86_64 架构上安装适用于 RHEL 7 的 OpenSSL 动态引擎：

```
$ 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 x86_64 架构上安装适用于 RHEL 8 的 OpenSSL 动态引擎：

```
$ 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 x86_64 架构上安装适用于 RHEL 9 的 OpenSSL 动态引擎：

```
$ 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 架构上安装适用于 RHEL 9 的 OpenSSL 动态引擎：

```
$ 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 x86_64 架构上安装适用于 Ubuntu 20.04 LTS 的 OpenSSL 动态引擎：

```
$ 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 x86_64 架构上安装适用于 Ubuntu 22.04 LTS 的 OpenSSL 动态引擎：

```
$ 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 架构上安装适用于 Ubuntu 22.04 LTS 的 OpenSSL 动态引擎：

```
$ 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` 安装了共享动态引擎库。

2. 引导客户端软件开发工具包 5。有关引导程序的更多信息，请参阅 引导客户端软件开发工具包。
3. 使用加密用户 (CU) 凭据设置环境变量。有关创建用户的信息，请参阅 使用 CMU 管理用户。

```
$ export CLOUDHSM_PIN=<HSM user name>:<password>
```

> **Note**
>
> 客户端软件开发工具包 5 引入了用于存储 CU 凭证的 `CLOUDHSM_PIN` 环境变量。在客户端软件开发工具包 3 中，您可以将 CU 凭据存储至 `n3fips_password` 环境变量中。客户端软件开发工具包 5 支持这两个环境变量，但我们建议使用 `CLOUDHSM_PIN`。

4. 将安装的 OpenSSL 动态引擎连接至集群。有关更多信息，请参阅连接至集群。
5. 引导程序客户端软件开发工具包 5 有关更多信息，请参阅 the section called “引导客户端软件开发工具包”。

验证适用于客户端软件开发工具包 5 的 OpenSSL 动态引擎

使用以下命令验证您所安装的 OpenSSL 动态引擎。

```
$ openssl engine -t cloudhsm
```

以下输出值用于验证您的配置：

```
(cloudhsm) CloudHSM OpenSSL Engine
     [ available ]
```

OpenSSL 动态引擎密钥类型

AWS CloudHSM OpenSSL 动态引擎支持以下密钥类型。

密钥类型	描述
EC	ECDSA 对 P-256、P-384 和 secp256k1 密钥类型执行签名/验证。要生成可与 OpenSSL 引擎互操作的 EC 密钥，请参阅 key generate-file。
RSA	为 2048、3072 和 4096 位密钥生成 RSA 密钥。rsa 签名/验证。验证已分流至 OpenSSL 软件。

OpenSSL 动态引擎机制

学习如何使用 AWS CloudHSM OpenSSL 动态引擎机制。

签署并验证功能

O AWS CloudHSM penSSL 动态引擎允许您将以下机制用于签名和验证功能。

通过客户端软件开发工具包 5，数据将在软件中进行本地哈希处理。这意味着可以进行哈希处理的数据大小没有限制。

RSA 签名类型

- SHA1withRSA
- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA

ECDSA 签名类型

- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

将 OpenSSL 动态引擎从客户端 SDK 3 迁移到客户端 SDK 5

使用本主题将您的 OpenSSL 动态引擎从客户端 SDK 3 迁移到客户端 SDK 5。有关迁移的好处，请参阅客户端软件开发工具包 5 的优点。

在中 AWS CloudHSM，客户应用程序使用 AWS CloudHSM 客户端软件开发套件 (SDK) 执行加密操作。客户端 SDK 5 是主要 SDK，它继续添加新功能和平台支持。

> **Note**
>
> 采用 OpenSSL 动态引擎的客户端 SDK 5 目前不支持随机数生成。

要查看所有提供商的迁移说明，请参阅从客户端软件开发工具包 3 迁移到客户端软件开发工具包 5。

迁移到客户端 SDK 5

按照本节中的说明从客户端 SDK 3 迁移到客户端 SDK 5。

> **Note**
>
> 客户端 SDK 5 目前不支持亚马逊 Linux、Ubuntu 16.04、Ubuntu 18.04、CentOS 6、CentOS 8 和 RHEL 6。如果您目前正在将其中一个平台与客户端 SDK 3 一起使用，则在迁移到客户端 SDK 5 时需要选择其他平台。

1. 卸载适用于客户端 SDK 的 OpenSSL 动态引擎 3.

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

   RHEL 7

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

   RHEL 8

   ```
   $ sudo yum remove cloudhsm-dyn
   ```

2. 卸载客户端 SDK 的客户端守护程序 3。

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client
```

CentOS 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 8

```
$ sudo yum remove cloudhsm-client
```

> **Note**
>
> 需要再次启用自定义配置。

3. 按照中的步骤安装客户端 SDK OpenSSL 动态引擎。安装 OpenSSL 动态引擎
4. 客户端 SDK 5 引入了一种新的配置文件格式和命令行引导工具。要启动您的客户端 SDK 5 OpenSSL 动态引擎，请按照用户指南中列出的说明进行操作。引导客户端软件开发工具包
5. 在您的开发环境中，测试您的应用程序。在最终迁移之前，请更新现有代码以解决重大更改。

相关 主题

- 以下方面的最佳实践 AWS CloudHSM

OpenSSL 的高级配置

AWS CloudHSM OpenSSL 提供程序包括以下高级配置，这不是大多数客户使用的常规配置的一部分。这些配置还提供了其他功能。

- OpenSSL 重试命令

OpenSSL 重试命令

客户端软件开发工具包 5.8.0 及更高版本具有内置的自动重试策略，该策略将从客户端重试 HSM 节流的操作。当 HSM 因忙于执行之前的操作而无法接受更多请求而对操作进行节流时，客户端软件开发工具包会尝试重试节流操作最多 3 次，同时进行指数级退避。这种自动重试策略可设置为以下两种模式之一：关闭和标准。

- 关闭：客户端软件开发工具包 不会对 HSM 的任何节流操作执行任何重试策略。
- 标准：这是客户端软件开发工具包版本 5.8.0 及更高版本的默认模式。在此模式下，客户端软件开发工具包将通过指数级退避自动重试节流操作。

有关更多信息，请参阅 HSM 节流。

将重试命令设置为关闭模式

Linux

在 Linux 上将客户端软件开发工具包 5 的重试命令设置为 off

- 您可以使用以下命令将重试命令设置为 off 模式：

```
$ sudo /opt/cloudhsm/bin/configure-dyn --default-retry-mode off
```

Windows

在 Windows 上将客户端软件开发工具包 5 的重试命令设置为 off

- 您可以使用以下命令将重试命令设置为 off 模式：

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-dyn.exe --default-retry-mode
 off
```

JCE 提供程序

AWS CloudHSM JCE 提供程序是基于 Java 加密扩展 (JCE) 提供程序框架构建的提供程序实现。您可通过 JCE，使用 Java 开发工具包 (JDK) 执行加密操作。在本指南中， AWS CloudHSM JCE 提供者有时被称为 JCE 提供者。使用 JCE 提供程序和 JDK 将加密操作分流至 HSM。有关疑难解答，请参阅JCE 开发工具包的已知问题。

有关使用客户端软件开发工具包 3 的信息，请参阅 以前的客户端 SDK（客户端 SDK 3）。

主题

安装并使用客户端 SDK 5 的 AWS CloudHSM JCE 提供程序

JCE 提供程序与 OpenJDK 8、OpenJDK 11、OpenJDK 17 和 OpenJDK 21 兼容。您可以从 OpenJDK 网站下载两者。

> **Note**
>
> 要使用客户端软件开发工具包 5 运行单个 HSM 集群，必须首先通过将 `disable_key_availability_check` 设置为 `True`，以管理客户端密钥持久性设置。有关更多信息，请参阅密钥同步和客户端软件开发工具包 5 配置工具。

主题

安装 JCE 提供程序

1. 使用以下命令下载和安装 JCE 提供程序。

Amazon Linux 2

在 x86_64 架构上安装适用于亚马逊 Linux 2 的 JCE 提供程序：

```
$ 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 架构上安装适用于亚马逊 Linux 2 的 JCE 提供程序：

```
$ 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 架构上安装适用于亚马逊 Linux 2023 的 JCE 提供程序：

```
$ 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 架构上安装适用于亚马逊 Linux 2023 的 JCE 提供程序：

```
$ 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 架构上安装 CentOS 7 的 JCE 提供程序：

```
$ 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 架构上安装 RHEL 7 的 JCE 提供程序：

```
$ 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 架构上安装 RHEL 8 的 JCE 提供程序：

```
$ 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 架构上安装 RHEL 9 (9.2+) 的 JCE 提供程序：

```
$ 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 架构上安装 RHEL 9 (9.2+) 的 JCE 提供程序：

```
$ 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 架构上安装 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 x86_64 架构上安装 Ubuntu 22.04 LTS 的 JCE 提供程序：

```
$ 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 架构上安装 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 架构上安装适用于 Windows Server 2016 的 JCE 提供程序，以管理员 PowerShell 身份打开并运行以下命令：

```
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
```

Windows Server 2019

在 x86_64 架构上安装适用于 Windows Server 2019 的 JCE 提供程序，以管理员 PowerShell 身份打开并运行以下命令：

```
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. 引导客户端软件开发工具包 5。有关引导程序的更多信息，请参阅 引导客户端软件开发工具包。
3. 找到以下 JCE 提供程序文件：

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

向 JCE 提供程序提供凭证

HSM 需要先验证 Java 应用程序，之后您的 Java 应用程序才能使用 HSM。HSM 使用显式登录或隐式登录方法来进行验证。

显式登录：此方法可让您直接在应用程序中提供 AWS CloudHSM 凭证。它使用 `AuthProvider` 中的方法，以 PIN 模式传入 CU 用户名和密码。有关更多信息，请参阅登录 HSM 代码示例。

隐式登录：此方法可让您在新属性文件或系统属性中设置 AWS CloudHSM 凭证，或将该凭证设置为环境变量。

- 系统属性：在运行应用程序时通过系统属性设置凭证。以下示例演示了可执行此操作的两种不同方式：

  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>");
  ```

- 环境变量：将凭证设置为环境变量。

  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>"
  ```

如果应用程序未提供凭证，或者在 HSM 验证会话前尝试执行操作，则凭证可能不可用。在这些情况下，适用于 Java 的 CloudHSM 软件库按以下顺序搜索凭证：

1. 系统属性
2. 环境变量

JCE 提供程序中的密钥管理基础知识

JCE 提供程序中密钥管理的基础知识涉及导入密钥、导出密钥、通过句柄加载密钥或删除密钥。有关管理密钥的更多信息，请参阅管理密钥代码示例。

另外，您可以在 代码示例 中找到更多 JCE 提供程序代码示例。

JCE 提供程序支持的密钥类型

Java AWS CloudHSM 软件库允许您生成以下密钥类型。

密钥类型	描述
AES	生成 128 位、192 位和 256 位 AES 密钥。
三重 DES（3DES、DESede）	生成 192 位的三重 DES 密钥[1]有关即将进行的更改，请参阅脚注1。
EC	生成 EC 密钥对：NIST 曲线 secp224r1 (P-224)、secp256r1 (P-256)、secp256k1 (区块链)、secp384r1 (P-384) 和 secp521r1 (P-521)。
GENERIC_SECRET	生成 1 至 800 字节的通用密钥。
HMAC	对 SHA1、SHA224、SHA256、SHA384、SHA512 的哈希支持。
RSA	生成 2048 位到 4096 位的 RSA 密钥，增量为 256 位。

[1] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

JCE 提供程序支持的机制

本主题提供有关客户端 SDK 5 支持的 JCE 提供程序的机制的信息。有关 Java 加密架构 (JCA) 接口和支持的引擎类的信息 AWS CloudHSM，请参阅以下主题。

主题

- 生成密钥与密钥对功能
- 密码函数
- 签署并验证功能
- 摘要功能
- HMAC 散列消息认证码函数
- 基于密码的消息身份验证码 (CMAC) 函数
- 使用密钥工厂将密钥转换为密钥规格
- 机制注释

生成密钥与密钥对功能

Java AWS CloudHSM 软件库允许您使用以下操作生成密钥和密钥对函数。

- RSA
- EC
- AES
- DESede (Triple DES)[请参阅备注1]
- GenericSecret

密码函数

Java AWS CloudHSM 软件库支持以下算法、模式和填充组合。

算法	Mode	Padding	注意
AES	CBC	AES/CBC/NoPadding AES/CBC/PKCS5Padding	实施 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。 实施 Cipher.UNWRAP_MODE for AES/CBC NoPadding

算法	Mode	Padding	注意
AES	ECB	AES/ECB/PKCS5Padding AES/ECB/NoPadding	实施 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。
AES	CTR	AES/CTR/NoPadding	实施 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。
AES	GCM	AES/GCM/NoPadding	实施 Cipher.WRAP_MODE 、Cipher.UNWRAP_MODE 、Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。 执行 AES-GCM 加密时，HSM 会忽略请求中的初始化向量 (IV) 并使用其生成的 IV。当该操作完成时，您必须调用 Cipher.getIV() 以获取 IV。
AESWrap	ECB	AESWrap/ECB/NoPadding AESWrap/ECB/PKCS5Padding AESWrap/ECB/ZeroPadding	实施 Cipher.WRAP_MODE 和 Cipher.UNWRAP_MODE 。

算法	Mode	Padding	注意
DESede（三重 DES）	CBC	DESede/CBC/PKCS5Padding DESede/CBC/NoPadding	实施 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。有关即将发生的更改，请参阅下面的注释 1。
DESede（三重 DES）	ECB	DESede/ECB/NoPadding DESede/ECB/PKCS5Padding	实施 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。有关即将发生的更改，请参阅下面的注释 1。

算法	Mode	Padding	注意
RSA	ECB	RSA/ECB/PKCS1Padding 请参阅备注 1 RSA/ECB/OAEPPadding RSA/ECB/OAEPWithSHA-1ANDMGF1Padding RSA/ECB/OAEPWithSHA-224ANDMGF1Padding RSA/ECB/OAEPWithSHA-256ANDMGF1Padding RSA/ECB/OAEPWithSHA-384ANDMGF1Padding RSA/ECB/OAEPWithSHA-512ANDMGF1Padding	实施 Cipher.WRAP_MODE 、Cipher.UNWRAP_MODE 、Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。
RSA	ECB	RSA/ECB/NoPadding	实施 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。

算法	Mode	Padding	注意
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	实施 Cipher.WRAP_MODE 和 Cipher.UNWRAP_MODE 。

签署并验证功能

Java AWS CloudHSM 软件库支持以下类型的签名和验证。使用客户端软件开发工具包 5 和带哈希功能的签名算法，在数据发送至 HSM 进行签名/验证之前，会在软件中对数据进行本地哈希处理。这意味着可由 SDK 执行哈希处理的数据大小没有限制。

RSA 签名类型

- NONEwithRSA
- RSASSA-PSS
- SHA1withRSA
- SHA1withRSA/PSS
- SHA1withRSAandMGF1
- SHA224withRSA
- SHA224withRSAandMGF1
- SHA224withRSA/PSS
- SHA256withRSA
- SHA256withRSAandMGF1
- SHA256withRSA/PSS
- SHA384withRSA
- SHA384withRSAandMGF1
- SHA384withRSA/PSS
- SHA512withRSA
- SHA512withRSAandMGF1
- SHA512withRSA/PSS

ECDSA 签名类型

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

摘要功能

Java AWS CloudHSM 软件库支持以下消息摘要。通过客户端软件开发工具包 5，数据将在软件中进行本地哈希处理。这意味着可由 SDK 执行哈希处理的数据大小没有限制。

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

HMAC 散列消息认证码函数

Java AWS CloudHSM 软件库支持以下 HMAC 算法。

- HmacSHA1（以字节为单位的最大数据大小：16288）
- HmacSHA224（以字节为单位的最大数据大小：16256）
- HmacSHA256（以字节为单位的最大数据大小：16288）
- HmacSHA384（以字节为单位的最大数据大小：16224）
- HmacSHA512（以字节为单位的最大数据大小：16224）

基于密码的消息身份验证码 (CMAC) 函数

CMAC（基于密码的消息身份验证码）使用数据块密码和机密密钥创建消息身份验证码（MAC）。其与 HMAC 的不同之处在于，其对 MAC 使用数据块对称密钥方法，而不是哈希方法。

Java 的 AWS CloudHSM 软件库支持以下 CMAC 算法。

- AESCMAC

使用密钥工厂将密钥转换为密钥规格

您可以使用密钥工厂将密钥转换为密钥规格。 AWS CloudHSM JCE 有两种类型的关键工厂：

SecretKeyFactory：用于导入或派生对称密钥。使用 SecretKeyFactory，您可以传递支持的密钥或支持的密钥 KeySpec 以导入或派生对称密钥。 AWS CloudHSM以下是支持的规格 KeyFactory：

- F SecretKeyFactory o generateSecret r 的方法支持以下KeySpec类：
  - KeyAttributesMap可用于将带有附加属性的密钥字节作为 CloudHSM 密钥导入。可以在此处找到一个示例。

- SecretKeySpec可用于将对称密钥规范作为 CloudHSM 密钥导入。
- AesCmacKdfParameterSpec可用于使用另一个 CloudHSM AES 密钥派生对称密钥。

> **Note**
>
> SecretKeyFactory的`translateKey`方法接受任何实现密钥接口的密钥。

KeyFactory：用于导入非对称密钥。使用 KeyFactory，您可以传递支持的密钥或支持的密钥 KeySpec 以将非对称密钥导入其中 AWS CloudHSM。更多信息，请参阅以下资源：

- KeyFactory对于`generatePublic`的方法，支持以下KeySpec类：
- 适用于 RSA 和 EC 的 KeyAttributesMap CloudHSM，包括： KeyTypes
  - Cloud KeyAttributesMap HSM 适用于 RSA 和 EC 公众。 KeyTypes可以在此处找到示例。
  - X509 同时EncodedKeySpec适用于 RSA 和 EC 公钥
  - 适用于 RSA PublicKeySpec 的 RSA 公钥
  - EC PublicKeySpec 代表欧共体公钥
- KeyFactory对于`generatePrivate`的方法，支持以下KeySpec类：
- 适用于 RSA 和 EC 的 KeyAttributesMap CloudHSM，包括： KeyTypes
  - Cloud KeyAttributesMap HSM 适用于 RSA 和 EC 公众。 KeyTypes可以在此处找到示例。
  - PKCS8 同时EncodedKeySpec适用于 EC 和 RSA 私钥
  - 适用于 RSA PrivateCrtKeySpec 私钥的 RSA
  - EC PrivateKeySpec 代表欧共体私钥

对于 KeyFactory的`translateKey`方法来说，它接受任何实现密钥接口的密钥。

机制注释

[1] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

受支持的 Java 密钥属性

本主题提供有关客户端 SDK 5 支持的 Java 密钥属性的信息。本主题介绍了如何使用 JCE 提供程序的专有扩展来设置密钥属性。使用此扩展可在以下操作期间设置受支持的密钥属性及其值：

- 密钥生成
- 密钥导入

有关如何使用密钥属性的示例，请参阅 the section called “代码示例”。

主题

了解属性

可以使用密钥属性指定允许对密钥对象（包括公有密钥或私有密钥）执行哪些操作。可以在创建密钥对象的过程中定义密钥属性和值。

Java Cryptography Extension (JCE) 不指定如何设置密钥属性值，因此，默认情况下允许执行大多数操作。相比之下，PKCS #11 标准定义了一组具有更受限的默认值的综合属性。从 JCE 提供程序 3.1 开始， AWS CloudHSM 提供了一个专有扩展，允许您为常用属性设置更严格的值。

支持的属性

可以为下表中列出的属性设置值。作为最佳实践，仅为应受限的属性设置值。如果未指定值，则 AWS CloudHSM 使用下表中指定的默认值。默认值列中的空单元格表示未向该属性分配特定的默认值。

属性	默认值			注意
	对称密钥	密钥对中的公有密钥	密钥对中的私有密钥	
DECRYPT	TRUE		TRUE	True 表示可使用密钥对任何缓冲区进行解密。对于其 WRAP 设置为 true 的密钥，通常将此项设置为 FALSE。

属性	默认值			注意
	对称密钥	密钥对中的公有密钥	密钥对中的私有密钥	
DERIVE				允许使用密钥派生其他密钥。
ENCRYPT	TRUE	TRUE		True 表示可使用密钥对任何缓冲区进行加密。
EXTRACTABLE	TRUE		TRUE	True 表示可从 HSM 中导出此密钥。
ID				用于标识密钥的用户定义值。
KEY_TYPE				用于识别密钥的类型（AES 密钥、DESede 密钥、通用机密密钥、EC 密钥 或 RSA 密钥）。
LABEL				便于您识别 HSM 上的密钥的用户定义字符串。要遵循最佳实践，请为每个密钥使用唯一的标签，以便日后查找。
LOCAL				表示 HSM 生成的密钥。

属性	默认值			注意
	对称密钥	密钥对中的公有密钥	密钥对中的私有密钥	
OBJECT_CLASS				用于标识密钥的对象类（SecretKey、 PublicKey 或 PrivateKey）。
PRIVATE	TRUE	TRUE	TRUE	True 表示用户在通过身份验证之前将无法访问密钥。为清楚起见，即使此属性设置为 FALSE，用户在通过身份验证 AWS CloudHSM 之前也无法访问任何密钥。
SIGN	TRUE		TRUE	True 表示可使用密钥对消息摘要进行签名。对于已存档的公有密钥和私有密钥，此项通常设置为 FALSE。

属性	默认值			注意
	对称密钥	密钥对中的公有密钥	密钥对中的私有密钥	
SIZE				定义密钥大小的属性。有关支持的密钥大小的更多详细信息，请参阅客户端开发工具包 5 支持的机制。
TOKEN	FALSE	FALSE	FALSE	跨集群中的所有 HSM 复制并包含在备份中的永久密钥。TOKEN = FALSE 表示一个临时密钥，当与 HSM 的连接中断或注销时会自动清除。
UNWRAP	TRUE		TRUE	True 表示可使用密钥解开包装（导入）另一个密钥。
VERIFY	TRUE	TRUE		True 表示可使用密钥验证签名。对于私有密钥，此项通常设置为 FALSE。

属性	默认值			注意
	对称密钥	密钥对中的公有密钥	密钥对中的私有密钥	
WRAP	TRUE	TRUE		True 表示可使用密钥包装另一个密钥。对于私有密钥，通常将此项设置为 FALSE。
WRAP_WITH_TRUSTED	FALSE		FALSE	True 表示只有将 TRUSTED 属性设置为 true 的密钥才能包装和解包的密钥。将密钥 WRAP_WITH_TRUSTED 设置为 true 后，该属性即为只读且不能设置为 false。要了解信任包装，请参阅使用信任的密钥控制密钥解包。

> **Note**
>
> 您将获得对 PKCS #11 库中属性的更广泛支持。有关更多信息，请参阅支持的 PKCS #11 属性。

设置密钥的属性

KeyAttributesMap 是一个类似于 Java Map 的对象，可以使用它设置密钥对象的属性值。KeyAttributesMap 函数的方法与用于 Java 映射操作的方法类似。

可以通过下面两种方式为属性设置自定义值：

- 使用下表中列出的方法
- 使用本文档后面演示的生成器模式

属性映射对象支持通过以下方法来设置属性：

操作	返回值	**KeyAttributesMap** 方法
获取现有密钥的密钥属性值	对象（包含值）或 null	get(keyAttribute)
填充一个密钥属性的值	与密钥属性关联的上一个值，或 null（如果没有密钥属性的映射）	put(keyAttribute, value)
填充多个密钥属性的值	不适用	putall () keyAttributesMap
从属性映射中删除密钥/值对	与密钥属性关联的上一个值，或 null（如果没有密钥属性的映射）	remove(keyAttribute)

> **Note**
>
> 未明确指定的任何属性都将设置为 the section called “支持的属性” 中前面的表中列出的默认值。

设置密钥对的属性

使用 Java 类 KeyPairAttributesMap 处理密钥对的密钥属性。KeyPairAttributesMap 封装了两个 KeyAttributesMap 对象；一个用于公有密钥，另一个用于私有密钥。

要分别为公有密钥和私有密钥设置单个属性，您可以对该密钥的相应 KeyAttributes 映射对象使用 put() 方法。使用 getPublic() 方法可检索公有密钥的属性映射，使用 getPrivate() 可检索私有密钥的属性映射。使用 putAll() 填充公有密钥和私有密钥对的多个密钥属性的值，并将密钥对属性映射作为其参数。

适用于 Java 的 AWS CloudHSM 软件库的代码示例

本主题提供有关客户端 SDK 5 的 Java 代码示例的资源和信息。

先决条件

运行示例前，必须设置环境：

- 安装和配置 Java 加密扩展（JCE）提供程序。
- 设置有效的 HSM 用户名和密码。加密用户 (CU) 权限足以执行这些任务。在每个示例中，您的应用程序都将使用这些凭证登录 HSM。
- 决定如何向 JCE 提供程序 提供凭证。

代码示例

以下代码示例向您展示了如何使用 AWS CloudHSM JCE 提供程序 执行基本任务。有关更多代码示例，请访问GitHub。

- 登录 HSM
- 管理密钥
- 生成对称密钥
- 生成非对称密钥
- 使用 AES-GCM 加密和解密
- 使用 AES-CTR 加密和解密
- 使用 DESede-ECB 加密和解密[请参阅备注1]
- 使用 RSA 密钥签名和验证
- 使用 EC 密钥签名和验证
- 使用支持的关键属性
- 使用 CloudHSM 密钥存储

[1] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

AWS CloudHSM JCE 提供商 Javadocs

使用 JCE 提供程序 Javadocs 获取有关 AWS CloudHSM JCE SDK 中定义的 Java 类型和方法的使用信息。要下载最新的 Javadocs AWS CloudHSM，请参阅 “下载最新版本” 页面上的部分。

您可以将 Javadocs 导入集成式开发环境（IDE）或在 Web 浏览器中查看它们。

使用 AWS CloudHSM KeyStore Java 类

该 AWS CloudHSM `KeyStore`类提供了一个特殊用途的 PKCS12 密钥存储库。该密钥库可以将证书与您的密钥数据一起存储，并将它们与存储在 AWS CloudHSM上的密钥数据相关联。该 AWS CloudHSM `KeyStore`类实现了 Java 密码学扩展 (JCE) 的`KeyStore`服务提供者接口 (SPI)。有关使用的更多信息`KeyStore`，请参阅 C lass KeyStore。

> **Note**
>
> 由于证书是公共信息，并且为了最大限度地提高加密密钥的存储容量，因此 AWS CloudHSM 不支持在 HSM 上存储证书。

选择适当的密钥库

AWS CloudHSM Java 加密扩展 (JCE) 提供商提供特殊用途的 AWS CloudHSM。 KeyStore该 AWS CloudHSM `KeyStore`类支持将密钥操作卸载到 HSM、证书的本地存储和基于证书的操作。

按如下方式加载特殊用途 KeyStore CloudHSM：

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

正在初始化 AWS CloudHSM KeyStore

使用与登录 JCE 提供程序相同的方式登录。 AWS CloudHSM KeyStore 您可以使用环境变量或系统属性文件，并且应该在开始使用 CloudHSM KeyStore 之前登录。有关使用 JCE 提供程序登录 HSM 的示例，请参阅登录到 HSM。

如果需要，您可以指定密码以加密保存密钥库数据的本地 PKCS12 文件。创建 AWS CloudHSM 密钥库时，需要设置密码，并在使用加载、设置和获取方法时提供密码。

按如下方式实例化一个新的 CloudHSM 对象 KeyStore ：

```
ks.load(null, null);
```

使用 `store` 方法将密钥库数据写入文件。从那时起，您可以使用带有源文件和密码的 `load` 方法加载现有密钥库，如下所示：

```
ks.load(inputStream, password);
```

使用 AWS CloudHSM KeyStore

AWS CloudHSM KeyStore 符合 JCE 类KeyStore规范，并提供以下功能。

- `load`

  从给定输入流加载密钥库。如果在保存密钥库时设置了密码，则必须提供相同的密码才能成功加载。将两个参数都设置为 null 可以初始化一个新的空密钥库。

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  ks.load(inputStream, password);
  ```

- `aliases`

  返回给定密钥库实例中所有条目的别名的枚举。结果包括本地存储在 PKCS12 文件中的对象和驻留在 HSM 上的对象。

  示例代码：

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();) {
      String label = entry.nextElement();
      System.out.println(label);
  }
  ```

- `containsalias`

  如果密钥库可以访问至少一个具有指定别名的对象，则返回 true。密钥库检查本地存储在 PKCS12 文件中的对象和驻留在 HSM 上的对象。

- `deleteEntry`

  从本地 PKCS12 文件中删除证书条目。不支持使用删除存储在 HSM 中的密钥数据。 AWS CloudHSM KeyStore您可以使用 Destroyable 接口的 `destroy` 方法删除密钥。

  ```
  ((Destroyable) key).destroy();
  ```

- getCertificate

  返回与别名关联的证书（如果可用）。如果别名不存在或引用的对象不是证书，则该函数返回 NULL。

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
Certificate cert = ks.getCertificate(alias);
```

- getCertificateAlias

  返回其数据与给定证书匹配的第一个密钥库条目的名称（别名）。

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
String alias = ks.getCertificateAlias(cert);
```

- getCertificateChain

  返回与给定别名关联的证书链。如果别名不存在或引用的对象不是证书，则该函数返回 NULL。

- getCreationDate

  返回由给定别名标识的条目的创建日期。如果创建日期不可用，则函数返回证书生效的日期。

- getKey

  GetKey 传递给 HSM 并返回与给定标签对应的密钥对象。当getKey直接查询 HSM 时，它可以用于 HSM 上的任何密钥，无论该密钥是否由生成。 KeyStore

```
Key key = ks.getKey(keyLabel, null);
```

- isCertificateEntry

  检查具有给定别名的条目是否表示证书条目。

- isKeyEntry

  检查具有给定别名的条目是否表示密钥条目。该操作同时搜索 PKCS12 文件和 HSM 以查找别名。

- setCertificateEntry

  将给定证书分配给给定别名。如果给定的别名已被用于标识密钥或证书，则会引发 KeyStoreException。您可以使用 JCE 代码获取密钥对象，然后使用 KeyStore SetKeyEntry方法将证书与密钥相关联。

- 使用 byte[] 密钥执行 setKeyEntry

客户端软件开发工具包 5 目前不支持此 API。

- 使用 Key 对象执行 setKeyEntry

  将给定密钥分配到给定别名并将其存储在 HSM 中。如果 HSM 中尚不存在该密钥，则会将其作为可提取的会话密钥导入 HSM。

  如果 Key 对象属于类型 PrivateKey，则它必须伴随相应的证书链。

  如果别名已存在，则 SetKeyEntry 调用会引发 KeyStoreException，并阻止该密钥被覆盖。如果密钥必须被覆盖，请为此目的使用 KMU 或 JCE。

- engineSize

  返回密钥库中的条目数。

- store

  将密钥库存储作为 PKCS12 文件存储到给定输出流，并使用给定的密码保护它。此外，它会保留所有加载的密钥（使用 setKey 调用进行设置）。

将您的 JCE 提供程序从客户端 SDK 3 迁移到客户端 SDK 5

使用本主题将您的 JCE 提供程序从客户端 SDK 3 迁移到客户端 SDK 5。有关迁移的好处，请参阅客户端软件开发工具包 5 的优点。

在中 AWS CloudHSM，客户应用程序使用 AWS CloudHSM 客户端软件开发套件 (SDK) 执行加密操作。客户端 SDK 5 是主要 SDK，它继续添加新功能和平台支持。

客户端 SDK 3 JCE 提供程序使用不属于标准 JCE 规范的自定义类和 API。JCE 提供商的客户端 SDK 5 不符合 JCE 规范，并且在某些方面与客户端 SDK 3 向后不兼容。在迁移到客户端 SDK 5 的过程中，可能需要对客户应用程序进行更改。本节概述成功迁移所需的更改。

要查看所有提供商的迁移说明，请参阅从客户端软件开发工具包 3 迁移到客户端软件开发工具包 5。

主题

- 通过解决重大变化做好准备
- 迁移到客户端 SDK 5
- 相关 主题

通过解决重大变化做好准备

查看这些重大更改，并在开发环境中相应地更新您的应用程序。

提供程序的类和名称已更改

发生了什么变化	客户端 SDK 3 里有什么	客户端 SDK 5 中的内容	示例
提供者类别和名称	客户端 SDK 3 中的 JCE 提供程序类被调用`CaviumProvider`，其名称`Cavium`为提供程序。	在客户端 SDK 5 中，调用提供者类`CloudHsmProvider`并使用提供程序名称`CloudHSM`。	示例存储库中提供了如何初始化`CloudHsmProvider`对象的 AWS CloudHSM GitHub 示例。

显式登录已更改，隐式登录未更改

发生了什么变化	客户端 SDK 3 里有什么	客户端 SDK 5 中的内容	示例
显式登录	客户端 SDK 3 使用该`LoginManager`类进行显式登录[1]。	在客户端 SDK 5 中，`CloudHSM`提供者实现`AuthProvider`了显式登录。`AuthProvider`是一个标准的 Java 类，它遵循 Java 的惯用方式登录提供程序。借助 Client SDK 5 中改进的登录状态管理，应用程序不再需要在重新连接[2]期间监控和执行登录。	有关如何在客户端软件开发工具包 5 中使用显式登录的示例，请参阅 AWS Cloud GitHub H SM LoginRunner 示例存储库中的示例。

发生了什么变化	客户端 SDK 3 里有什么	客户端 SDK 5 中的内容	示例
隐式登录	隐式登录无需进行任何更改。从客户端 SDK 3 迁移到客户端 SDK 5 时，相同的属性文件和所有环境变量将继续适用于隐式登录。		有关如何在 Client SDK 5 中使用隐式登录的示例，请参阅LoginRunner 示例存储库中的 AWS CloudHSM GitHub 示例。

- [1] 客户端 SDK 3 代码片段：

```
LoginManager lm = LoginManager.getInstance();

lm.login(partition, user, pass);
```

- [2] 客户端 SDK 5 代码片段：

```
// 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);
```

有关如何在 Client SDK 5 中使用显式登录的示例，请参阅LoginRunner 示例存储库中的 AWS CloudHSM GitHub 示例。

密钥生成已更改

发生了什么变化	客户端 SDK 3 里有什么	客户端 SDK 5 中的内容	示例
密钥生成	在客户端 SDK 3 中，`Cavium[Key-type]Al`	在客户端 SDK 5 中，`KeyAttributesMap` 用于指定	有关如何使用生成对称密钥的示例`KeyAttrib`

发生了什么变化	客户端 SDK 3 里有什么	客户端 SDK 5 中的内容	示例
	gorithmPa rameterSpec 用于指定密钥生成参数。有关代码片段，请参阅脚注1。	密钥生成属性。有关代码片段，请参阅脚注2。	utesMap ，请参阅 AWS CloudHSM Github Symmetric Keys 示例存储库中的示例。
密钥对生成	在客户端 SDK 3 中，Cavium[Ke y-type]Al gorithmpa rameterSpec 用于指定密钥对生成参数。有关代码片段，请参阅脚注3。	在客户端 SDK 5 中，KeyPairAt tributesMap 用于指定这些参数。有关代码片段，请参阅脚注4。	有关如何使用KeyAttrib utesMap 生成非对称密钥的示例，请参阅AsymmetricKeys 示例存储库中的 AWS CloudHSM GitHub 示例。

- [1] 客户端 SDK 3 密钥生成代码片段：

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
CaviumAESKeyGenParameterSpec aesSpec = new CaviumAESKeyGenParameterSpec(
keySizeInBits,
keyLabel,
isExtractable,
isPersistent);
keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [2] 客户端 SDK 5 密钥生成代码片段：

```
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] 客户端 SDK 3 key pair 生成代码片段::

```
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] 客户端 SDK 5 key pair 生成代码片段：

```
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、删除和引用密钥已更改

使用查找已生成的密钥 AWS CloudHSM 需要使用 KeyStore. 客户端 SDK 3 有两种 KeyStore 类型：`Cavium`和`CloudHSM`。客户端 SDK 5 只有一种 KeyStore 类型：`CloudHSM`。

从移动`Cavium` KeyStore 到`CloudHSM` KeyStore 需要更改 KeyStore 类型。此外，客户端 SDK 3 使用密钥句柄来引用密钥，而客户端 SDK 5 使用密钥标签。下面列出了由此产生的行为变化。

发生了什么变化	客户端 SDK 3 里有什么	客户端 SDK 5 中的内容	示例
关键参考文献	在 Client SDK 3 中，应用程序使用密钥标签或密钥句柄来引用 HSM 中的密钥。他们使用标签 KeyStore 来查找钥匙，或者他们使用手柄创建`CaviumKey` 对象。	在客户端 SDK 5 中使用 AWS CloudHSM KeyStore Java 类，应用程序可以使用按标签查找密钥。要按手柄查找按键，请 AWS CloudHSM `KeyStoreWithAttributes` 使用 with AWS CloudHSM `KeyReferenceSpec` 。	
查找多个条目	使用`getEntry`、或搜索密钥时`getKey`，如果`getCertificate` 中存在多个具有相同条件的项目 `Cavium` KeyStore，则只会返回找到的第一个条目。	使用 AWS CloudHSM `KeyStore`和`KeyStoreWithAttributes` ，同样的场景将导致引发异常。要解决此问题，建议使用 CloudHSM CLI 中的key set-attri	

发生了什么变化	客户端 SDK 3 里有什么	客户端 SDK 5 中的内容	示例
		bute命令为密钥设置唯一标签。或者使用KeyStoreWithAttributes#getKeys 返回所有符合条件的密钥。	
找到所有钥匙	在客户端 SDK 3 中，可以使用Util.findAllKeys() 查找 HSM 中的所有密钥。	客户端 SDK 5 使用该KeyStoreWithAttributes 类可以更简单、更高效地查找密钥。如果可能，请缓存您的密钥以最大限度地减少延迟。有关更多信息，请参阅 有效管理应用程序中的密钥。当您需要从 HSM 检索所有密钥时，请KeyStoreWithAttributes#getKeys 使用空的KeyAttributesMap 。	AWS CloudHSM Github 示例存储库中提供了使用该KeyStoreWithAttributes 类查找密钥的示例，其中显示了一个代码片段。1
删除密钥	客户端 SDK 3 Util.deleteKey() 用于删除密钥。	Client SDK 5 中的Key对象实现了允许使用此Destroyable 接口的destroy() 方法删除密钥的接口。	可以在 CloudHSM Github 示例存储库中找到显示删除密钥功能的示例代码。中显示了每个 SDK 的示例片段。2

- [1] 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] 删除客户端 SDK 中的密钥 3：

```
Util.deleteKey(key);
```

删除客户端 SDK 中的密钥 5：

```
((Destroyable) key).destroy();
```

密码解包操作已更改，其他密码操作未更改

> **Note**
>
> Cipher 加密/解密/封装操作无需进行任何更改。

Unwrap 操作要求将 Client SDK 3 `CaviumUnwrapParameterSpec` 类替换为以下特定于所列加密操作的类之一。

- `GCMUnwrapKeySpec`用于`AES/GCM/NoPadding`解开包装
- `IvUnwrapKeySpec`因为`AESWrap unwrap`和 `AES/CBC/NoPadding unwrap`
- 适用于 `RSA OAEP unwrap` 的 `OAEPUnwrapKeySpec`

以下示例片段：`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);
```

签名操作没有改变

签名操作无需进行任何更改。

迁移到客户端 SDK 5

按照本节中的说明从客户端 SDK 3 迁移到客户端 SDK 5。

> **Note**
>
> 客户端 SDK 5 目前不支持亚马逊 Linux、Ubuntu 16.04、Ubuntu 18.04 CentOS 6、CentOS 8 和 RHEL 6。如果您目前正在将其中一个平台与客户端 SDK 3 一起使用，则在迁移到客户端 SDK 5 时需要选择其他平台。

1. 卸载客户端 SDK 的 JCE 提供程序 3。

   Amazon Linux 2

   ```
   $ sudo yum remove cloudhsm-jce
   ```

   CentOS 7

   ```
   $ sudo yum remove cloudhsm-jce
   ```

RHEL 7

```
$ sudo yum remove cloudhsm-jce
```

RHEL 8

```
$ sudo yum remove cloudhsm-jce
```

2. 卸载客户端 SDK 的客户端守护程序 3。

Amazon Linux 2

```
$ sudo yum remove cloudhsm-client
```

CentOS 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 7

```
$ sudo yum remove cloudhsm-client
```

RHEL 8

```
$ sudo yum remove cloudhsm-client
```

> **Note**
>
> 需要再次启用自定义配置。

3. 按照中的步骤安装客户端 SDK JCE 提供程序。安装并使用客户端 SDK 5 的 AWS CloudHSM JCE 提供程序
4. 客户端 SDK 5 引入了一种新的配置文件格式和命令行引导工具。要引导您的客户端 SDK 5 JCE 提供程序，请按照用户指南中列出的说明进行操作。引导客户端软件开发工具包
5. 在您的开发环境中，测试您的应用程序。在最终迁移之前，请更新现有代码以解决重大更改。

相关 主题

- 以下方面的最佳实践 AWS CloudHSM

JCE 的高级配置

AWS CloudHSM JCE 提供商包括以下高级配置，这些配置不是大多数客户使用的常规配置的一部分。

- 连接到多个集群
- 使用 JCE 提取密钥
- JCE 重试配置

使用 JCE 提供程序连接到多个集群

此配置允许单个客户端实例与多个集群通信。对于某些用例来说，与让单个实例仅与单个集群通信相比，这可能是项节省成本的功能。该`CloudHsmProvider`类是 Java AWS CloudHSM Security 的 Provider 类的实现。此类的每个实例都代表与整个 AWS CloudHSM 集群的连接。您可以实例化该类并将其添加到 Java Security 提供程序的列表中，以便您可以使用标准 JCE 类与之交互。

以下示例实例化该类并将其添加到 Java Security 提供程序的列表中：

```
if (Security.getProvider(CloudHsmProvider.PROVIDER_NAME) == null) {
    Security.addProvider(new CloudHsmProvider());
}
```

`CloudHsmProvider` 配置

`CloudHsmProvider` 可以通过下列两种方式进行配置：

1. 使用文件进行配置（默认配置）
2. 使用代码进行配置

使用文件进行配置（默认配置）

当您使用默认构造函数实例化 CloudHsmProvider 时，默认情况下，它会在 Linux 的 /opt/cloudhsm/etc/cloudhsm-jce.cfg 路径中查找配置文件。该配置文件可使用 configure-jce 进行配置。

使用默认构造函数创建的对象将使用默认 CloudHSM 提供程序名称 CloudHSM。提供程序的名称对于与 JCE 交互非常有用，可以让 JCE 知道要使用哪个提供程序进行各类操作。使用 CloudHSM 提供程序名称进行密码操作的示例如下所示：

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHSM");
```

使用代码进行配置

从客户端 SDK 版本 5.8.0 开始，您也可以使用 Java 代码配置 CloudHsmProvider。做到这一点的方法是使用 CloudHsmProviderConfig 类的对象。您可以使用 CloudHsmProviderConfigBuilder 构建此对象。

CloudHsmProvider 还有另一个接受 CloudHsmProviderConfig 对象的构造函数，如以下示例所示。

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);
```

在此示例中，JCE 提供程序的名称是 CloudHsmCluster1。这是应用程序随后可用于与 JCE 交互的名称：

Example

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHsmCluster1");
```

或者应用程序也可以使用上面创建的提供程序对象让 JCE 知道要使用该提供程序进行操作：

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider);
```

如果未在 withClusterUniqueIdentifier 方法中指定唯一标识符，则会为您创建一个随机生成的提供程序名称。要获取这个随机生成的标识符，应用程序可以调用 provider.getName() 获取该标识符。

连接到多个集群

如上所述，每个 CloudHsmProvider 都代表与您的 CloudHSM 集群的连接。如果您想与来自同一应用程序的另一个集群通信，则可以使用另一个集群的配置创建另一个 CloudHsmProvider 对象，也可以使用提供程序对象或提供程序名称与另一个集群进行交互，如以下示例所示。

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);
}
```

配置了上面的两个提供程序（均为集群）后，就可以使用提供程序对象或提供程序名称与它们进行交互。

在此演示如何与之交谈的示例的基础上`cluster1`，您可以使用以下示例进行 AES/GCM/ NoPadding 操作：

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider1);
```

在同一个应用程序中，要使用提供程序名称在第二个集群上生成“AES”密钥，也可以使用以下示例：

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider2.getName());
```

JCE 的重试命令

客户端软件开发工具包 5.8.0 及更高版本具有内置的自动重试策略，该策略将从客户端重试 HSM 节流的操作。当 HSM 因忙于执行之前的操作而无法接受更多请求而对操作进行节流时，客户端软件开发工具包会尝试重试节流操作最多 3 次，同时进行指数级退避。这种自动重试策略可设置为以下两种模式之一：关闭和标准。

- 关闭：客户端软件开发工具包 不会对 HSM 的任何节流操作执行任何重试策略。
- 标准：这是客户端软件开发工具包版本 5.8.0 及更高版本的默认模式。在此模式下，客户端软件开发工具包将通过指数级退避自动重试节流操作。

有关更多信息，请参阅 HSM 节流。

将重试命令设置为关闭模式

Linux

在 Linux 上将客户端软件开发工具包 5 的重试命令设置为 off

- 您可以使用以下命令将重试配置设置为 off 模式：

```
$ sudo /opt/cloudhsm/bin/configure-jce --default-retry-mode off
```

Windows

在 Windows 上将客户端软件开发工具包 5 的重试命令设置为 off

- 您可以使用以下命令将重试配置设置为 off 模式：

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-jce.exe --default-retry-mode
 off
```

使用 JCE 提取密钥

Java 密码学扩展 (JCE) 使用的架构允许插入不同的加密实现。 AWS CloudHSM 发布了一个这样的 JCE 提供商，它可以将加密操作卸载到 HSM。为了让大多数其他 JCE 提供程序使用存储在 AWS CloudHSM 中的密钥，他们必须将您在 HSM 中的密钥字节以明文形式提取到您的计算机内存中以供他们使用。HSM 通常只允许将密钥提取为包装对象，而不允许以明文形式提取。但是，为了支持提供商间集成用例， AWS CloudHSM 允许使用选择加入配置选项来启用密钥字节的提取。

> **Important**
>
> AWS CloudHSM 无论何时指定 AWS CloudHSM 提供程序或 AWS CloudHSM 使用密钥对象，JCE 都会将操作卸载到任何时候。如果您希望在 HSM 内部进行操作，则无需以明文方式提取密钥。只有当您的应用程序由于第三方库或 JCE 提供程序的限制而无法使用安全机制（例如包装和解包密钥）时，才需要以明文提取密钥。

默认情况下， AWS CloudHSM JCE 提供程序允许提取公钥，以便与外部 JCE 提供程序配合使用。始终允许使用以下方法：

类	方法	格式 (getEncoded)
EcPublicKey	getEncoded()	X.509
	getW()	不适用
RSA PublicKey	getEncoded()	X.509
	getPublicExponent()	不适用
CloudHsmRsaPrivateCrtKey	getPublicExponent()	不适用

默认情况下， AWS CloudHSM JCE Provider 不允许提取私钥或私钥的密钥字节。如果您的用例需要，则可以在以下条件下启用私有密钥或机密密钥的明文密钥字节提取功能：

1. 私有密钥和机密密钥的 EXTRACTABLE 属性设置为 true。
   - 默认情况下，私有密钥和机密密钥的 EXTRACTABLE 属性设置为 true。EXTRACTABLE 密钥是允许从 HSM 中导出的密钥。有关更多信息，请参阅《客户端软件开发工具包 5 支持的 Java 属性》。
2. 私有密钥和机密密钥的 WRAP_WITH_TRUSTED 属性设置为 false。
   - getEncoded、getPrivateExponent 和 getS 不能与无法以明文导出的私有密钥一起使用。WRAP_WITH_TRUSTED 不允许您的私有密钥以明文从 HSM 中导出。有关更多信息，请参阅使用可信密钥控制密钥解包。

允许 AWS CloudHSM JCE 提供者从中提取私钥机密 AWS CloudHSM

> **Important**
>
> 此配置更改允许从 HSM 集群中以明文提取所有 EXTRACTABLE 密钥字节。为了提高安全性，您应该考虑使用密钥包装方法将密钥安全地从 HSM 中提取出来。这样可以防止无意中从 HSM 中提取密钥字节。

1. 使用以下命令从 JCE 中提取您的私有密钥或机密密钥：

   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. 一旦启用了明文密钥提取功能，就会启用以下方法将私有密钥提取到内存中。

类	方法	格式 (getEncoded)
键	getEncoded()	RAW
欧共体 PrivateKey	getEncoded()	PKCS#8
	getS()	不适用
RSA PrivateCrtKey	getEncoded()	X.509
	getPrivateExponent()	不适用
	getPrimeP()	不适用
	getPrimeQ()	不适用
	getPrimeExponentP ()	不适用
	getPrimeExponentQ ()	不适用
	getCrtCoefficient()	不适用

如果要恢复默认行为并且不允许 JCE 以明文导出密钥，请运行以下命令：

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
```

加密 API：适用于 Microsoft Windows 的新一代（CNG）和密钥存储提供程序（KSP）

Windows AWS CloudHSM 客户端包括 CNG 和 KSP 提供商。目前，只有客户端 SDK 3 支持 CNG 和 KSP 提供程序。

密钥存储提供程序 (KSP) 支持密钥存储和检索。例如，如果您向 Windows Server 添加了 Microsoft Active Directory 证书服务 (AD CS)，并且选择为证书颁发机构 (CA) 创建新的私有密钥，则可以选择将管理密钥存储的 KSP。当您配置 AD CS 角色时，您可以选择此 KSP。有关更多信息，请参阅 创建 Windows Server CA。

加密 API：下一代 (CNG) 是一种特定于 Microsoft Windows 操作系统的加密 API。CNG 使开发人员能够使用加密技术来保护基于 Windows 的应用程序。简而言之，CNG 的 AWS CloudHSM 实现提供了以下功能：

- 加密基元 - 使您能够执行基本加密操作。
- 密钥导入和导出 - 使您能够导入和导出对称和非对称密钥。
- 数据保护 API (CNG DPAPI) - 使您能够轻松加密和解密数据。
- 密钥存储和检索 - 使您能够安全地存储和隔离非对称密钥对的私有密钥。

主题

验证适用于 Windows 的 KSP 和 CNG 提供程序

KSP 和 CNG 提供程序是在你安装 Windows AWS CloudHSM 客户端时安装的。您可以按照 安装客户端 (Windows) 中的下列步骤安装客户端。

配置和运行 Windows AWS CloudHSM 客户端

您必须先满足 先决条件，然后才能启动 Windows CloudHSM 客户端。然后，更新提供程序使用的配置文件，并通过完成以下步骤启动客户端。当您首次使用 KSP 和 CNG 时以及在集群中添加或删除 HSM 后，需要执行这些步骤。这样， AWS CloudHSM 就可以同步集群中所有 HSM 的数据并保持一致性。

步骤 1：停止 AWS CloudHSM 客户端

在更新提供程序使用的配置文件之前，请停止 AWS CloudHSM 客户端。如果客户端已停止，运行停止命令也不会产生影响。

- 对于 Windows 客户端 1.1.2 以上版本：

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- 对于 Windows 客户端 1.1.1 及更低版本：

  在启动 AWS CloudHSM 客户端的命令窗口中使用 C trl + C。

步骤 2：更新 AWS CloudHSM 配置文件

此步骤使用-a配置工具的 参数将集群中的其中一个 HSM 的弹性网络接口 (ENI) IP 地址添加到配置文件。

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

要获取集群中 HSM 的 ENI IP 地址，请导航到 AWS CloudHSM 控制台，选择集群，然后选择所需的集群。您也可以使用DescribeClusters操作、desc ribe-clust ers 命令或 cmdlet。Get-HSM2Cluster PowerShell仅输入一个 ENI IP 地址。您使用哪个 HSM ENI IP 地址并不重要。

步骤 3：启动 AWS CloudHSM 客户端

接下来，启动或重新启动 AWS CloudHSM 客户端。当 AWS CloudHSM 客户端启动时，它会使用其配置文件中的 ENI IP 地址来查询集群。然后，它会将集群中所有 HSM 的 ENI IP 地址添加到集群信息文件。

- 对于 Windows 客户端 1.1.2 以上版本：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 对于 Windows 客户端 1.1.1 及更低版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

检查 KSP 和 CNG 提供程序

您可以使用以下任一命令来确定您的系统上所安装的提供程序。命令列出了已注册的 KSP 和 CNG 提供程序。 AWS CloudHSM 客户端无需运行。

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -enum
```

```
C:\Program Files\Amazon\CloudHSM>cng_config.exe -enum
```

要验证您的 Windows Server EC2 实例上是否安装了 KSP 和 CNG 提供程序，您应该在列表中查看以下条目：

```
Cavium CNG Provider
Cavium Key Storage Provider
```

如果缺少 CNG 提供程序，请运行以下命令。

```
C:\Program Files\Amazon\CloudHSM>cng_config.exe -register
```

如果缺少 KSP 提供程序，请运行以下命令。

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -register
```

Windows AWS CloudHSM 先决条件

在启动 Windows AWS CloudHSM 客户端并使用 KSP 和 CNG 提供商之前，您必须在系统上设置 HSM 的登录凭据。您可以通过 Windows Credentials Manager 或系统环境变量设置凭证。我们建议您使用 Windows Credential Manager 存储凭证。此选项适用于 AWS CloudHSM 客户端版本 2.0.4 及更高版本。使用环境变量更易于设置，但安全性低于使用 Windows Credential Manager。

Windows Credential Manager

您可以使用 `set_cloudhsm_credentials` 实用工具或 Windows Credentials Manager 界面。

- 使用 **`set_cloudhsm_credentials`** 实用工具：

  `set_cloudhsm_credentials` 实用工具包含在您的 Windows 安装程序中。您可以使用此实用工具方便地将 HSM 登录凭证传递到 Windows Credential Manager。如果要从源代码编译此实用工具，可以使用安装程序中包含的 Python 代码。

1. 转到 `C:\Program Files\Amazon\CloudHSM\tools\` 文件夹。
2. 使用 CU 用户名和密码参数运行 `set_cloudhsm_credentials.exe` 文件。

```
set_cloudhsm_credentials.exe --username <CU USER> --password <CU PASSWORD>
```

- 使用 Credential Manager 界面：

  您可以使用 Credential Manager 界面来手动管理您的凭证。

  1. 要打开 Credential Manager，请在任务栏上的搜索框中键入 `credential manager`，然后选择 Credential Manager。
  2. 选择 Windows 凭证来管理 Windows 凭证。
  3. 选择添加通用凭证，并填写详细信息，如下所示：
     - 在 Internet 或网络地址 中，输入目标名称作为 `cloudhsm_client`。
     - 在 用户名 和 密码 中，输入 CU 凭证。
     - 单击 确定。

系统环境变量

您可以为 Windows 应用程序设置标识 HSM 和加密用户 (CU) 的系统环境变量。您可以使用 setx 命令设置系统环境变量，或通过编程方式或在 Windows 系统属性 控制面板的 高级 选项卡中设置永久系统环境变量。

> **Warning**
>
> 当您通过系统环境变量设置凭证时，密码在用户系统上以纯文本形式可用。要克服此问题，请使用 Windows Credential Manager。

设置以下系统环境变量：

**`n3fips_password=`*`CU USERNAME`*`:`*`CU PASSWORD`*

在 HSM 中确定加密用户 (CU)，并提供所有必需的登录信息。您的应用程序以此 CU 的身份进行身份验证和运行。该应用程序具有此 CU 的权限，只能查看和管理 CU 拥有和共享的密钥。要创建新 CU，请使用 createUser。要查找现有的 CU，请使用 listUsers。

例如：

```
setx /m n3fips_password test_user:password123
```

将密 AWS CloudHSM 钥与证书关联

在将 AWS CloudHSM 密钥用于第三方工具（例如 Microsoft 的工具）之前 SignTool，必须将密钥的元数据导入本地证书存储区，并将元数据与证书相关联。要导入密钥的元数据，请使用包含在 CloudHSM 3.0 及更高版本中的 import_key.exe 实用程序。以下步骤提供其他信息和示例输出。

步骤 1：导入证书

在 Windows 上，您应该能够双击证书将其导入到本地证书存储。

但是，如果双击不起作用，请使用 Microsoft Certreq 工具将证书导入证书管理器中。例如：

```
certreq -accept certificatename
```

如果此操作失败，并且您收到错误 `Key not found`，请继续执行步骤 2。如果证书显示在密钥库中，则表示您已完成任务，无需进一步操作。

步骤 2：收集证书识别信息

如果上一步未成功，则需要将您的私有密钥与证书关联。但是，您必须先找到证书的唯一容器名称和序列号，然后才能创建关联。使用实用程序（例如 certutil）显示所需的证书信息。certutil 的以下示例输出显示了容器名称和序列号。

```
================ 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
```

步骤 3：将 AWS CloudHSM 私钥与证书关联

要将密钥与证书关联，请首先确保启动 AWS CloudHSM 客户端守护程序。然后，使用 import_key.exe（包含在 CloudHSM 版本 3.0 及更高版本中）将私有密钥与证书关联。指定证书时，请使用其简单容器名称。以下示例显示了命令和响应。此操作仅复制密钥的元数据；密钥保留在 HSM 上。

```
$> import_key.exe -RSA CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c

Successfully opened Microsoft Software Key Storage Provider : 0NCryptOpenKey failed :
 80090016
```

步骤 4：更新证书存储

请确保 AWS CloudHSM 客户端守护程序仍在运行。然后，使用 certutil 动词 -repairstore 更新证书序列号。以下示例显示了命令和输出。有关 -repairstore 动词的信息，请参阅 Microsoft 文档。

```
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.
```

更新证书序列号后，您可以在 Windows 上的任何第三方签名工具中使用此证书和相应的 AWS CloudHSM 私钥。

CNG 提供程序的代码示例

⚠ ** 只是示例代码 – 不用于生产用途 **
本示例代码仅用于说明。请勿在生产环境中运行此代码。

下面的示例显示了如何枚举系统中已注册的加密提供程序，以查找 Windows 版 CloudHSM 客户端安装的 CNG 提供程序。该示例还显示了如何创建非对称密钥对以及如何使用该密钥对来签署数据。

⚠ Important

在运行此示例之前，必须按照先决条件中的说明设置 HSM 凭证。有关详细信息，请参阅Windows AWS CloudHSM 先决条件。

```
// 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;
}
```

以前的客户端 SDK（客户端 SDK 3）

AWS CloudHSM 包括两个主要的客户端 SDK 版本：

- 客户端软件开发工具包 5：这是我们最新的默认客户端软件开发工具包。有关它提供的益处和优势的信息，请见 客户端软件开发工具包 5 的优点。
- 客户端软件开发工具包 3 ：这是我们的旧版客户端软件开发工具包。它包括一整套用于平台和语言应用程序的兼容性组件以及管理工具。

有关从客户端 SDK 3 迁移到客户端 SDK 5 的说明，请参阅从客户端软件开发工具包 3 迁移到客户端软件开发工具包 5。

本主题中列出了客户端软件开发工具包 3 文档。

如要下载，请参阅 Downloads。

检查您的客户端软件开发工具包版本

Amazon Linux

使用以下命令：

```
rpm -qa | grep ^cloudhsm
```

Amazon Linux 2

使用以下命令：

```
rpm -qa | grep ^cloudhsm
```

CentOS 6

使用以下命令：

```
rpm -qa | grep ^cloudhsm
```

CentOS 7

使用以下命令：

```
rpm -qa | grep ^cloudhsm
```

CentOS 8

使用以下命令：

```
rpm -qa | grep ^cloudhsm
```

RHEL 6

使用以下命令：

```
rpm -qa | grep ^cloudhsm
```

RHEL 7

使用以下命令：

```
rpm -qa | grep ^cloudhsm
```

RHEL 8

使用以下命令：

```
rpm -qa | grep ^cloudhsm
```

Ubuntu 16.04 LTS

使用以下命令：

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 18.04 LTS

使用以下命令：

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 20.04 LTS

使用以下命令：

```
apt list --installed | grep ^cloudhsm
```

Windows Server

使用以下命令：

```
wmic product get name,version
```

客户端软件开发工具包 组件对比

除了命令行工具外，客户端软件开发工具包 3 还包含将加密操作从各种平台或基于语言的应用程序分流至 HSM 的组件。客户端 SDK 5 与客户端 SDK 3 相同，唯一的不同是它还不支持 CNG 和 KSP 提供商。下表对比了客户端软件开发工具包 3 和客户端软件开发工具包 5 组件的可用性。

组件	客户端软件开发工具包 5	客户端软件开发工具包 3
PKCS #11 库	支持	是
JCE 提供程序	支持	是
OpenSSL 动态引擎	支持	是
CNG 和 KSP 提供程序		是
CloudHSM 管理实用程序 (CMU) [1]	支持	是
密钥管理实用程序（KMU）[1]	是	是
Configure 工具	支持	是

[1] CMU 和 KMU 组件包含在带有客户端软件开发工具包 5 的 CloudHSM CLI 中。

主题

- 客户端软件开发工具包 3 支持的平台
- 在 Linux 系统升级客户端软件开发工具包 3
- 用于客户端软件开发工具包 3 的 PKCS #11 库
- 安装适用于 OpenSSL 动态引擎的客户端软件开发工具包 3
- 适用于 JCE 提供程序的客户端软件开发工具包 3

客户端软件开发工具包 3 支持的平台

客户端软件开发工具包 3 需要客户端进程守护程序并提供命令行工具，包括 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility)、密钥管理实用程序 (KMU, key management utility) 和配置工具。

每个版本的 AWS CloudHSM 客户端 SDK 的基础支持都不同。通常情况下，SDK 中组件的平台支持与基础支持相匹配，但并非总是如此。要确定给定组件的平台支持，请首先确保您想要的平台显示在 SDK 的基础部分中，然后在组件部分中检查是否存在任何排除项或任何其他相关信息。

平台支持会随着时间的推移而变化。CloudHSM 客户端软件开发工具包的早期版本可能不支持此处列出的所有操作系统。根据发行说明确定 CloudHSM 客户端软件开发工具包早期版本的操作系统支持情况。有关更多信息，请参阅 AWS CloudHSM 客户端 SDK 的下载内容。

AWS CloudHSM 仅支持 64 位操作系统。

目录

Linux 支持

- 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] 不支持 OpenSSL Dynamic Engine。有关更多信息，请参阅 OpenSSL 动态引擎。

[2] 不支持客户端软件开发工具包 3.3.0 及更高版本。

[3] SDK 3.4 是 Ubuntu 16.04 上最后一个支持的版本。

[3] SDK 3.4 是 CentOS 8.3+ 上最后一个支持的版本。

Windows 支持

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

客户端 SDK 的 HSM 兼容性 3

hsm1.medium	hsm2m.medium
与客户端版本 SDK 3.1.0 及更高版本兼容。	不支持。

组件支持

PKCS #11 库

PKCS #11库是一个仅适用于 Linux 的组件，与 Linux 基础支持相匹配。有关更多信息，请参阅 the section called “Linux 支持”。

CloudHSM 管理实用程序 (CMU)

CloudHSM 管理实用程序 (CMU) 命令行工具可帮助加密官员管理 HSM 中的用户。它包含用于创建、删除和列出用户以及更改用户密码的工具。有关更多信息，请参阅 CloudHSM 管理实用程序 (CMU)。

密钥管理实用程序（KMU）

密钥管理实用程序 (KMU) 是一个命令行工具，可帮助加密用户 (CU) 管理硬件安全模块 (HSM) 上的密钥。有关更多信息，请参阅 密钥管理实用程序（KMU）。

JCE 提供程序

JCE 提供程序是一个仅适用于 Linux 的组件，与 Linux 基础支持相匹配。有关更多信息，请参阅 the section called “Linux 支持”。

- 需要 OpenJDK 1.8

OpenSSL 动态引擎

OpenSSL Dynamic Engine 是仅适用于 Linux 的组件，与 Linux 基础支持不匹配。请参见以下排除项。

- 需要 OpenSSL 1.0.2[f+]

不支持的平台：

- CentOS 8
- Red Hat Enterprise Linux (RHEL) 8
- Ubuntu 18.04 LTS

这些平台附带的 OpenSSL 版本与适用于客户端软件开发工具包 3 的 OpenSSL Dynamic Engine 不兼容。 AWS CloudHSM 通过适用于客户端软件开发工具包 5 的 OpenSSL Dynamic Engine 支持这些平台。

CNG 和 KSP 提供程序

CNG 和 KSP 提供程序是仅限 Windows 的组件，与 Windows 基础支持相匹配。有关更多信息，请参阅 Windows 支持。

在 Linux 系统升级客户端软件开发工具包 3

在 AWS CloudHSM Client SDK 3.1 及更高版本中，客户端守护程序的版本和您安装的任何组件都必须匹配才能升级。对于所有基于 Linux 的系统，必须使用单个命令、通过相同版本的 PKCS #11 库、Java 加密扩展 (JCE) 提供程序或 OpenSSL 动态引擎，批量升级客户端进程守护程序。此要求不适用于基于 Windows 的系统，因为客户端进程守护程序包已包含 CNG 和 KSP 提供程序库的二进制文件。

检查客户端进程守护程序版本

- 在基于 Red Hat 的 Linux 系统（包括 Amazon Linux 和 CentOS），使用以下命令：

```
rpm -qa | grep ^cloudhsm
```

- 在基于 Debian 的 Linux 系统上，使用以下命令：

```
apt list --installed | grep ^cloudhsm
```

- 在 Windows 系统上，使用以下命令：

```
wmic product get name,version
```

主题

先决条件

下载最新版本的 AWS CloudHSM 客户端守护程序并选择您的组件。

> **Note**
>
> 您无需安装所有的组件。对于已安装的每个组件，必须升级该组件来匹配客户端进程守护程序的版本。

最新的 Linux 客户端进程守护程序

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 库

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 动态引擎

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 提供程序

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
```

步骤 1：停止客户端进程守护程序

使用以下命令停止客户端进程守护程序。

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
```

第 2 步：升级客户端软件开发工具包

以下命令显示了升级客户端进程守护程序和组件所需的语法。在运行此命令之前，请删除所有您不打算升级的组件。

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>
```

步骤 3：启动客户端进程守护程序

使用以下命令启动客户端进程守护程序。

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

尚不支持 OpenSSL 动态引擎。

用于客户端软件开发工具包 3 的 PKCS #11 库

PKCS #11 是硬件安全模块 (HSM) 上的加密操作执行标准。

有关引导的信息，请参阅 连接到集群。

主题

- 安装适用于 PKCS #11 库的客户端软件开发工具包 3
- 正在向 PKCS #11 库（客户端软件开发工具包 3）验证身份
- 支持的密钥类型（客户端软件开发工具包 3）
- 支持的机制（客户端软件开发工具包 3）
- 支持的 API 操作（客户端软件开发工具包 3）
- 支持的密钥属性（客户端软件开发工具包 3）
- PKCS #11 库的代码示例（客户端软件开发工具包 3）

安装适用于 PKCS #11 库的客户端软件开发工具包 3

客户端软件开发工具包 3 的先决条件

PKCS #11 库需要 AWS CloudHSM 客户端。

如果您尚未安装和配置 AWS CloudHSM 客户端，请立即按照中的步骤进行操作安装客户端 (Linux)。在您安装和配置客户端之后，可以使用以下命令来启动客户端。

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
```

安装适用于客户端软件开发工具包 3 的 PKCS #11 库

以下命令将下载并安装 PKCS #11 库。

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
```

- 如果您所安装的 PKCS #11 库的 EC2 实例未安装客户端软件开发工具包 3 中的其他组件，则必须引导客户端软件开发工具包 3。您只需使用客户端软件开发工具包 3 中的组件在每个实例上执行一次此操作。
- 您可在以下位置找到 PKCS #11 库的文件：

  Linux 二进制文件、配置脚本、证书和日志文件：

```
/opt/cloudhsm/lib
```

正在向 PKCS #11 库（客户端软件开发工具包 3）验证身份

当您使用 PKCS #11 库时，您的应用程序将作为 HSM 中的特定加密用户（CU）运行。您的应用程序只能查看和管理 CU 拥有和共享的密钥。您可以在 HSM 中使用现有 CU，也可以序创建新 CU。有关 CU 管理的信息，请参见 通过 CloudHSM CLI 管理 HSM 用户 以及 CloudHSM 管理实用程序 (CMU) 管理 HSM 用户。

要将 CU 指定给 PKCS #11 库，请使用 PKCS #11 C_Login 函数的 PIN 参数。对于 AWS CloudHSM，pin 参数采用以下格式：

```
<CU_user_name>:<password>
```

例如，以下命令会将 PKCS #11 库 PIN 设置为用户名为 `CryptoUser`、密码为 `CUPassword123!` 的 CU。

```
CryptoUser:CUPassword123!
```

支持的密钥类型（客户端软件开发工具包 3）

PKCS #11 库支持以下密钥类型。

密钥类型	描述
RSA	生成 2048 位到 4096 位的 RSA 密钥，增量为 256 位。
EC	通过 secp224r1 (P-224)、secp256r1 (P-256)、secp256k1（区块链）、secp384r1 (P-384) 和 secp521r1 (P-521) 曲线生成密钥。
AES	生成 128 位、192 位和 256 位 AES 密钥。
DES3（三重数据加密标准）	生成 192 位 DES3 密钥。有关即将发生的更改，请参阅下面的注释 1。
GENERIC_SECRET	生成 1 到 64 字节的常规密钥。

- [1] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

支持的机制（客户端软件开发工具包 3）

PKCS #11 库支持以下算法：

- 加密和解密 - AES-CBC、AES-CTR、AES-ECB、AES-GCM、DES3-CBC、DES3-ECB、RSA-OAEP 和 RSA-PKCS
- 签名和验证 - RSA、HMAC 和 ECDSA；带和不带哈希
- 哈希/摘要 - SHA1、SHA224、SHA256、SHA384 和 SHA512
- 密钥包装 - AES 密钥包装、[4]AES-GCM、RSA-AES 和 RSA-OAEP
- 密钥派生 — ECDH、[5]SP800-108 CTR KDF

PKCS #11 库机制函数表

PKCS #11 库符合 2.40 版 PKCS #11 规格。要使用 PKCS # 11 调用加密功能，请使用给定机制调用函数。下表汇总了 AWS CloudHSM支持的函数和机制的组合。

解释支持的 PKCS #11 机制函数表

✓ 标记表示 AWS CloudHSM 支持该功能的机制。我们不支持 PKCS #11 规范中列出的所有可能的函数。✖ 标记表示尚 AWS CloudHSM 不支持给定函数的机制，即使 PKCS #11 标准允许。空单元格表示 PKCS #11 标准不支持给定函数的机制。

支持的 PKCS #11 库机制和函数

机制	函数						
	生成密钥或密钥对	签名和验证	SR 和 VR	摘要	加密和解密	派生密钥	Wrap & UnWrap
CKM_RSA_PKCS_KEY_PAIR_GEN	✓						
CKM_RSA_X9_31_KEY_PAIR_GEN	✓[2]						

机制	函数						
CKM_RSA_X_509		✓			✓		
CKM_RSA_PKCS 请参阅备注 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]					
CKM_SHA1_RSA_PKCS_PSS		✓[3.2]					

机制	函数						
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]					
CKM_ECDSA_SHA384		✓[3.2]					
CKM_ECDSA_SHA512		✓[3.2]					

机制	函数						
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 请参阅备注 8	✓						
CKM_DES3_CBC 请参阅备注 8					✓[3.3]		✖

机制	函数						
CKM_DES3_CBC_PAD 请参阅备注 8					✓		✖
CKM_DES3_ECB 请参阅备注 8					✓		✖
CKM_AES_GCM					✓[3.3, 4]		✓[7.1]
CKM_CLOUDHSM_AES_GCM					✓[7.1]		✓[7.1]
CKM_SHA_1				✓[3.1]			
CKM_SHA_1_HMAC		✓[3.3]					
CKM_SHA224				✓[3.1]			
CKM_SHA224_HMAC		✓[3.3]					
CKM_SHA256				✓[3.1]			
CKM_SHA256_HMAC		✓[3.3]					
CKM_SHA384				✓[3.1]			

机制	函数						
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]
CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD							✓[7.1]
CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD							✓[7.1]

机制注释

- [1] 仅限单部分操作。
- [2] 机制的功能与 p 机制相同，但为生成 CKM_RSA_PKCS_KEY_PAIR_GEN 和 q 提供了更强有力的保证。
- [3.1] 根据客户端 SDK，哈希 AWS CloudHSM 处理方法有所不同。对于客户端软件开发工具包 3，我们的哈希处理时机取决于数据大小以及您所用的是单个还是多个操作。

  客户端软件开发工具包 3 中的单部分操作

  表 3.1 列出了客户端软件开发工具包 3 的每种机制的最大数据集大小。整个哈希值均在 HSM 内部计算。不支持 16KB 以上的数据。

  表 3.1：单部分操作的最大数据集大小

机制	最大数据大小
CKM_SHA_1	16296
CKM_SHA224	16264
CKM_SHA256	16296
CKM_SHA384	16232
CKM_SHA512	16232

  多部分操作客户端软件开发工具包 3

  支持 16KB 以上的数据，但是数据大小决定了哈希发生的位置。小于 16KB 的数据缓冲区在 HSM 内部进行哈希处理。对于系统中的 16KB 和最大数据大小之间的缓冲区，在软件中进行本地哈希处理。请切记：哈希函数不需要加密，因此您可以在 HSM 之外安全计算。
- [3.2] 根据客户端 SDK，哈希 AWS CloudHSM 处理方法有所不同。对于客户端软件开发工具包 3，我们的哈希处理时机取决于数据大小以及您所用的是单个还是多个操作。

  单部分操作客户端软件开发工具包 3

  表 3.2 列出了客户端软件开发工具包 3 的每种机制的最大数据集大小。不支持 16KB 以上的数据。

表 3.2：单部分操作的最大数据集大小

机制	最大数据大小
CKM_SHA1_RSA_PKCS	16296
CKM_SHA224_RSA_PKCS	16264
CKM_SHA256_RSA_PKCS	16296
CKM_SHA384_RSA_PKCS	16232
CKM_SHA512_RSA_PKCS	16232
CKM_SHA1_RSA_PKCS_PSS	16296
CKM_SHA224_RSA_PKCS_PSS	16264
CKM_SHA256_RSA_PKCS_PSS	16296
CKM_SHA384_RSA_PKCS_PSS	16232
CKM_SHA512_RSA_PKCS_PSS	16232
CKM_ECDSA_SHA1	16296
CKM_ECDSA_SHA224	16264
CKM_ECDSA_SHA256	16296
CKM_ECDSA_SHA384	16232
CKM_ECDSA_SHA512	16232

多部分操作客户端软件开发工具包 3

支持 16KB 以上的数据，但是数据大小决定了哈希发生的位置。小于 16KB 的数据缓冲区在 HSM 内部进行哈希处理。对于系统中的 16KB 和最大数据大小之间的缓冲区，在软件中进行本地哈希处理。请切记：哈希函数不需要加密，因此您可以在 HSM 之外安全计算。

- [3.3] 使用以下任何机制对数据进行操作时，如果数据缓冲区超出最大数据大小，则操作会导致错误。对此机制，所有数据处理均应在 HSM 内发生。下表列出了为每个机制设置的最大数据大小：

  表 3.3：最大数据集大小

机制	最大数据大小
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] 在执行 AES-GCM 加密时，HSM 不会接受应用程序中的初始化向量 (IV) 数据。您必须使用其生成的 IV。HSM 提供的 12 字节 IV 将写入您提供的 CK_GCM_PARAMS 参数结构的 pIV 元素所指向的内存参考。为了防止用户混淆，版本 1.1.1 及更高版本中的 PKCS #11 开发工具包将在初始化 AES-GCM 加密时确保该 pIV 指向已清零的缓冲区。
- [5] 仅客户端软件开发工具包 3。机制的实施旨在支持 SSL/TLS 卸载案例，只能在 HSM 内部分执行。使用此机制之前，请参阅 PKCS#11 库的已知问题 中的“问题：ECDH 密钥派生只能在 HSM 内部分执行”。CKM_ECDH1_DERIVE 不支持 secp521r1 (P-521) 曲线。
- [6] 以下 CK_MECHANISM_TYPE 和 CK_RSA_PKCS_MGF_TYPE 出于 CKM_RSA_PKCS_OAEP 作为 CK_RSA_PKCS_OAEP_PARAMS 受到支持：
  - 使用 CKG_MGF1_SHA1 的 CKM_SHA_1
  - 使用 CKG_MGF1_SHA224 的 CKM_SHA224
  - 使用 CKG_MGF1_SHA256 的 CKM_SHA256
  - 使用 CKM_MGF1_SHA384 的 CKM_SHA384
  - 使用 CKM_MGF1_SHA512 的 CKM_SHA512

- [7.1] 供应商定义的机制。为了使用 CloudHSM 供应商定义的机制，PKCS #11 应用程序必须在编译过程中包含 /opt/cloudhsm/include/pkcs11t.h。

  CKM_CLOUDHSM_AES_GCM：这种专有机制是标准 CKM_AES_GCM 的编程更安全的替代方案。它将 HSM 生成的 IV 附加到密文，而不是将其写回密码初始化期间提供的 CK_GCM_PARAMS 结构中。您可以将此机制与 C_Encrypt、C_WrapKey、C_Decrypt 和 C_UnwrapKey 函数一起使用。使用此机制时，CK_GCM_PARAMS 结构中的 pIV 变量必须设置为 NULL。将此机制与 C_Decrypt 和 C_UnwrapKey 一起使用时，IV 预计会被放在正在解开包装的密文之前。

  CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD：带 PKCS #5 填充的 AES 密钥包装

  CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD：零填充的 AES 密钥包装

  有关其他 AES 密钥包装的其他信息，请参阅 AES 密钥包装。
- [8] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

支持的 API 操作（客户端软件开发工具包 3）

适用于 PKCS #11 库支持以下 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（仅支持客户端软件开发工具包 3）
- C_SignRecoverInit（仅支持客户端软件开发工具包 3）
- C_SignUpdate

- C_UnWrapKey
- C_Verify
- C_VerifyFinal
- C_VerifyInit
- C_VerifyRecover（仅支持客户端软件开发工具包 3）
- C_VerifyRecoverInit（仅支持客户端软件开发工具包 3）
- C_VerifyUpdate
- C_WrapKey

支持的密钥属性（客户端软件开发工具包 3）

密钥对象可以是公有、私有或秘密密钥。通过属性指定密钥对象上允许的操作。属性是在创建密钥对象时定义的。当您使用 CloudHSM 的 PKCS#11 开发工具包时，我们将按照 PKCS #11 标准的要求分配默认值。

AWS CloudHSM 不支持 PKCS #11 规范中列出的所有属性。对于我们支持的所有属性，我们符合此规范。这些属性列在各自的表中。

创建、修改或复制对象的加密函数（如 C_CreateObject、C_GenerateKey、C_GenerateKeyPair、C_UnwrapKey 和 C_DeriveKey）将属性模板作为它们的一个参数。有关在对象创建期间传递属性模板的更多信息，请参阅通过 PKCS #11 库生成密钥样例。

解释 PKCS#11 库属性表格

PKCS#11 库表格包含依密钥类型而不同的属性的列表。它表示在使用特定加密函数时，特定密钥类型是否支持给定属性。 AWS CloudHSM

图例：

- ✓ 表示 CloudHSM 对特定密钥类型支持此属性。
- ✖ 表示 CloudHSM 对特定密钥类型不支持此属性。
- R 表示特定密钥类型的属性值设置为只读的。
- S 表示无法通过 GetAttributeValue 读取该属性，因为它是敏感属性。
- 默认值列中的空单元格表示没有给该属性分配特定的默认值。

GenerateKeyPair

属性	密钥类型				默认值
	EC 私有	EC 公有	RSA 私有	RSA 公有	
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
CKA_DESTROYABLE	✓	✓	✓	✓	True

属性	密钥类型				默认值
CKA_SIGN	✓	✖	✓	✖	False
CKA_SIGN_RECOVER	✖	✖	✓[3]	✖	
CKA_VERIFY	✖	✓	✖	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✓[4]	
CKA_WRAP	✖	✓	✖	✓	False
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	
CKA_TRUSTED	✖	✓	✖	✓	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	False
CKA_UNWRAP	✓	✖	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	True
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	

属性	密钥类型				默认值
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	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	✓[2]	

属性	密钥类型				默认值
CKA_EC_PARAMS	✖	✓[2]	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	

GenerateKey

属性	密钥类型			默认值
	AES	DES3	普通机密	
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True

属性	密钥类型			默认值
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
CKA_WRAP_WITH_TRUSTED	✓	✓	✓	False

属性	密钥类型			默认值
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	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	

属性	密钥类型			默认值
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

属性	密钥类型							默认值
		EC 私有	EC 公有	RSA 私有	RSA 公有	AES	DES3	普通机密	

属性	密钥类型							默认值
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
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓	False

属性	密钥类型							默认值
CKA_SIGN_RECOVER	✖	✖	✓[3]	✖	✖	✖	✖	False
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✓[4]	✖	✖	✖	
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖	False
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖	
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓	False
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖	
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	R	✖	R	✖	R	R	R	

属性	密钥类型							默认值
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]	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✓[2]	✓[2]	✖	✖	✖	

属性	密钥类型							默认值
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

属性	密钥类型					默认值
	EC 私有	RSA 私有	AES	DES3	普通机密	
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

属性	密钥类型					默认值
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	✖	✓[3]	✖	✖	✖	False
CKA_VERIFY	✖	✖	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✓	✓	✖	False
CKA_UNWRAP	✖	✓	✓	✓	✖	False

属性	密钥类型					默认值
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	✖	✖	✖	✖	✖	

属性	密钥类型					默认值
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

属性	密钥类型				默认值
	AES	DES3	普通机密		
CKA_CLASS	✓[2]	✓[2]	✓[2]		
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]		
CKA_LABEL	✓	✓	✓		

属性	密钥类型			默认值
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

属性	密钥类型			默认值
CKA_UNWRAP	✓	✓	✖	False
CKA_SENSITIVE	✓	✓	✓	True
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	

属性	密钥类型				默认值
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

属性	密钥类型						
	EC 私有	EC 公有	RSA 私有	RSA 公有	AES	DES3	普通机密
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓

属性	密钥类型						
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	✖	✓	✖	✓	✓	✓	✓

属性	密钥类型						
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

属性	密钥类型						
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	✖	✖	✖	✓[2]	✓[2]	✓[2]

属性	密钥类型						
CKA_VALUE_LEN	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	✖

属性注释

- [1] 此属性由固件部分支持且必须明确地仅设置为默认值。
- [2] 必需属性。
- [3] 仅客户端软件开发工具包 3。CKA_SIGN_RECOVER 属性派生自 CKA_SIGN 属性。如果要设置它，只能将其设置为与为 CKA_SIGN 设置的值相同。如果不设置，它获得默认值 CKA_SIGN。由于 CloudHSM 仅支持基于 RSA 的可恢复签名机制，此属性目前仅适用于 RSA 公有密钥。
- [4] 仅客户端软件开发工具包 3。CKA_VERIFY_RECOVER 属性派生自 CKA_VERIFY 属性。如果要设置它，只能将其设置为与为 CKA_VERIFY 设置的值相同。如果不设置，它获得默认值 CKA_VERIFY。由于 CloudHSM 仅支持基于 RSA 的可恢复签名机制，此属性目前仅适用于 RSA 公有密钥。

修改属性

对象的一些属性可以在对象创建后修改，有一些则不能。若要修改属性，请使用 cloudhsm_mgmt_util 中的 setAttribute 命令。您还可通过 cloudhsm_mgmt_util 中的 listAttribute 命令派生出属性列表和代表这些属性的常量。

以下列表显示了对象创建后允许修改的属性：

- CKA_LABEL
- CKA_TOKEN

> **Note**
>
> 修改仅允许将会话密钥更改为令牌密钥。使用 key_mgmt_util 中的 setAttribute 命令更改属性值。

- CKA_ENCRYPT

- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

> **Note**
>
> 此属性支持密钥派生。它对所有公有密钥必须为 False，并且无法设置为 True。对于秘密和 EC 私有密钥，它可以设置为 True 或 False。

- CKA_TRUSTED

> **Note**
>
> 此属性仅可通过加密员（CO) 设置为 True 或 False。

- CKA_WRAP_WITH_TRUSTED

> **Note**
>
> 将此属性应用于可导出的数据密钥，以表示此密钥只能更换为标记CKA_TRUSTED的密钥。将 CKA_WRAP_WITH_TRUSTED 设置为 true 后，该属性将变为只读，并且您无法更改或移除该属性。

解释错误代码

在模板中指定特定密钥不支持的属性会导致错误。下表包含当您违反规范时生成的错误代码：

错误代码	描述
CKR_TEMPLATE_INCONSISTENT	当您在属性模板中指定一个属性而该属性符合 PKCS #11 规范但不受 CloudHSM 支持时，您会收到此错误。
CKR_ATTRIBUTE_TYPE_INVALID	当您在属性模板中检索符合 PKCS #11 规范但不受 CloudHSM 支持的属性的值时，您会收到此错误。
CKR_ATTRIBUTE_INCOMPLETE	当您未在属性模板中指定必需属性时，您会收到此错误。
CKR_ATTRIBUTE_READ_ONLY	当您在属性模板中指定只读属性时，您会收到此错误。

PKCS #11 库的代码示例（客户端软件开发工具包 3）

上的代码示例 GitHub 向您展示了如何使用 PKCS #11 库完成基本任务。

示例代码先决条件

在运行示例之前，请执行以下步骤以设置您的环境：

- 为客户端软件开发工具包 3 安装和配置 PKCS #11 库。
- 设置加密用户 (CU)。您的应用程序使用此 HSM 账户在 HSM 运行代码示例。

代码示例

PKCS #11 AWS CloudHSM 软件库的代码样本可在上找到。GitHub此存储库包括有关如何使用 PKCS#11 执行常见操作的示例，包括加密、解密、签名和验证。

- 生成密钥（AES、RSA、EC）
- 列出密钥属性
- 使用 AES GCM 加密和解密数据
- 使用 AES_CTR 加密和解密数据
- 使用 3DES 加密和解密数据

- 使用 RSA 对数据进行签名和验证
- 使用 HMAC KDF 派生密钥
- 使用 AES 对密钥进行包装和解开包装（使用 PKCS #5 填充）
- 使用 AES 对密钥进行包装和解开包装（无填充）
- 使用 AES 包装和解开密钥（使用零填充）
- 使用 AES-GCM 对密钥进行包装和解开包装
- 使用 RSA 对密钥进行包装和解开包装

安装适用于 OpenSSL 动态引擎的客户端软件开发工具包 3

客户端软件开发工具包 3 确实需要通过客户端进程守护程序连接至集群。它可支持：

- 针对 2048、3072 和 4096 位密钥的 RSA 密钥生成。
- RSA 签署/验证。
- RSA 加密/解密。
- 通过加密进行保护且经 FIPS 验证的随机数生成。

主题

- 使用带有客户端软件开发工具包 3 的 OpenSSL 动态引擎的先决条件
- 安装适用于客户端软件开发工具包 3 的 OpenSSL 动态引擎
- 使用适用于客户端软件开发工具包的 OpenSSL 动态引擎 3

使用带有客户端软件开发工具包 3 的 OpenSSL 动态引擎的先决条件

有关支持的平台的更多信息，请参阅 客户端软件开发工具包 3 支持的平台。

在使用 OpenSSL 的 AWS CloudHSM 动态引擎之前，您需要客户端。 AWS CloudHSM

客户端是一个守护程序，用于与集群中的 HSM 建立 end-to-end 加密通信，OpenSSL 引擎与客户端进行本地通信。要安装和配置 AWS CloudHSM 客户端，请参见安装客户端 (Linux)。使用以下命令可启动。

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
```

安装适用于客户端软件开发工具包 3 的 OpenSSL 动态引擎

以下步骤介绍如何安装和配置 OpenSS AWS CloudHSM L 的动态引擎。有关升级的信息，请参阅 升级客户端软件开发工具包 3。

安装和配置 OpenSSL 引擎

1. 使用以下命令下载和安装 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
```

将 OpenSSL 引擎安装至 /opt/cloudhsm/lib/libcloudhsm_openssl.so

2. 使用以下命令设置名为 n3fips_password 的环境变量，该变量包含加密用户 (CU) 的凭证。

```
$ export n3fips_password=<HSM user name>:<password>
```

使用适用于客户端软件开发工具包的 OpenSSL 动态引擎 3

要使用集成 OpenSSL 的应用程序中的 OpenSSL AWS CloudHSM 动态引擎，请确保您的应用程序使用名为的 OpenSSL 动态引擎。cloudhsm动态引擎的共享库位于 /opt/cloudhsm/lib/libcloudhsm_openssl.so。

要从 OpenSSL 命令行使用适用于 OpenSSL 的 AWS CloudHSM 动态引擎，请使用-engine选项指定名为的 OpenSSL 动态引擎。cloudhsm例如：

```
$ openssl s_server -cert server.crt -key server.key -engine cloudhsm
```

适用于 JCE 提供程序的客户端软件开发工具包 3

AWS CloudHSM JCE 提供程序是基于 Java 加密扩展 (JCE) 提供程序框架构建的提供程序实现。您可通过 JCE，使用 Java 开发工具包 (JDK) 执行加密操作。在本指南中， AWS CloudHSM JCE 提供者有时被称为 JCE 提供者。使用 JCE 提供程序和 JDK 将加密操作分流至 HSM。

主题

- 安装并使用客户端 SDK 3 的 AWS CloudHSM JCE 提供程序
- 客户端软件开发工具包 3 的支持机制
- 支持的 Java 密钥属性（客户端开发工具包 3）
- 适用于客户端 SDK 的 Java AWS CloudHSM 软件库的代码示例 3

- 为客户端 SDK 使用 AWS CloudHSM KeyStore Java 类 3

安装并使用客户端 SDK 3 的 AWS CloudHSM JCE 提供程序

在使用 JCE 提供程序之前，需要 AWS CloudHSM 客户端。

客户端是一个守护程序，用于与集群中的 HSM 建立 end-to-end 加密通信。JCE 提供程序在本地与客户端通信。如果您尚未安装和配置 AWS CloudHSM 客户端软件包，请立即按照中的步骤进行操作安装客户端 (Linux)。在您安装和配置客户端之后，可以使用以下命令来启动客户端。

请注意 JCE 提供程序仅在 Linux 和兼容的操作系统中受支持。

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
```

主题

安装 JCE 提供程序

使用以下命令下载和安装 JCE 提供程序。此提供程序仅在 Linux 和兼容的操作系统上受支持。

> **Note**
>
> 有关升级，请参阅升级客户端软件开发工具包 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
```

运行上述命令后，可以找到以下 JCE 提供程序文件：

- /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

验证安装

在 HSM 上执行基本操作来验证安装。

验证 JCE 提供程序的安装

1. （可选）如果您尚未在您的环境中安装 Java，可使用以下命令安装它。

   Linux (and compatible libraries)

   ```
   $ sudo yum install java-1.8.0-openjdk
   ```

   Ubuntu

   ```
   $ sudo apt-get install openjdk-8-jre
   ```

2. 使用以下命令可设置必要的环境变量。将 *<HSM user name>* 和 *<password>* 替换为加密用户 (CU) 的凭证。

```
$ export LD_LIBRARY_PATH=/opt/cloudhsm/lib
```

```
$ export HSM_PARTITION=PARTITION_1
```

```
$ export HSM_USER=<HSM user name>
```

```
$ export HSM_PASSWORD=<password>
```

3. 使用以下命令可运行基本功能测试。如果成功，命令的输出应该与下面的输出类似。

```
$ 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)
```

向 JCE 提供程序提供凭证

HSM 需要先验证 Java 应用程序，之后该应用程序才能使用 HSM。每个应用程序均可使用一个会话。HSM 使用显式登录或隐式登录方法来验证会话。

显式登录：此方法可让您直接在应用程序中提供 CloudHSM 凭证。它使用 `LoginManager.login()` 方法，在其中传递 CU 用户名、密码和 HSM 分区 ID。有关使用显式登录方法的更多信息，请参阅登录到 HSM 代码示例。

隐式登录：此方法可让您在新属性文件或系统属性中设置 CloudHSM 凭证，或将此类凭证设置为环境变量。

- 新属性文件：创建一个名为 `HsmCredentials.properties` 的新文件，然后将该文件添加到应用程序的 `CLASSPATH`。该文件应包含以下内容：

```
HSM_PARTITION = PARTITION_1
HSM_USER = <HSM user name>
HSM_PASSWORD = <password>
```

- 系统属性：在运行应用程序时通过系统属性设置凭证。以下示例演示了可执行此操作的两种不同方式：

```
$ 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>");
```

- 环境变量：将凭证设置为环境变量。

```
$ export HSM_PARTITION=PARTITION_1
$ export HSM_USER=<HSM user name>
$ export HSM_PASSWORD=<password>
```

如果应用程序未提供凭证，或者在 HSM 验证会话前尝试执行操作，则凭证可能不可用。在这些情况下，适用于 Java 的 CloudHSM 软件库按以下顺序搜索凭证：

1. `HsmCredentials.properties`
2. 系统属性

3. 环境变量

错误处理

通过显式登录方法处理错误比通过隐式登录方法处理错误更轻松。在使用 `LoginManager` 类时，您可以更好地控制应用程序处理失败情况的方式。当凭证无效或 HSM 在验证会话时遇到问题时，隐式登录方法使得难以理解错误处理。

JCE 提供程序中的密钥管理基础知识

JCE 提供程序中密钥管理的基础知识涉及导入密钥、导出密钥、通过句柄加载密钥或删除密钥。有关管理密钥的更多信息，请参阅管理密钥代码示例。

另外，您可以在 代码示例 中找到更多 JCE 提供程序代码示例。

客户端软件开发工具包 3 的支持机制

有关支持的 Java 密码学架构 (JCA) 接口和引擎类的信息 AWS CloudHSM，请参阅以下主题。

主题

- 支持的密钥
- 支持的密码
- 支持的摘要
- 支持 HMAC 散列消息认证码算法
- 支持的签名/验证机制
- 机制注释

支持的密钥

Java AWS CloudHSM 软件库允许您生成以下密钥类型。

- AES – 128、192 和 256 位 AES 密钥。
- DESede – 92 位 3DES 密钥。有关即将发生的更改，请参阅下面的注释 1。
- NIST 曲线 secp256r1 (P-256)、secp384r1 (P-384) 和 secp256k1（区块链）的 ECC 密钥对。
- RSA 2048 位到 4096 位 RSA 密钥，增量为 256 位。

除了标准参数以外，我们对于生成的每个密钥还支持以下参数。

- Label：可用于搜索密钥的密钥标签。
- isExtractable：指示密钥是否可从 HSM 中导出。
- isPersistent：指示在当前会话结束时密钥是否保留在 HSM 上。

> **Note**
>
> 利用 Java 库版本 3.1，可以更详细地指定参数。有关更多信息，请参阅支持的 Java 属性。

支持的密码

Java AWS CloudHSM 软件库支持以下算法、模式和填充组合。

算法	Mode	Padding	注意
AES	CBC	AES/CBC/NoPadding AES/CBC/PKCS5Padding	实施 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。
AES	ECB	AES/ECB/NoPadding AES/ECB/PKCS5Padding	实施 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。使用转换 AES。
AES	CTR	AES/CTR/NoPadding	实施 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。
AES	GCM	AES/GCM/NoPadding	实施 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE ，以及 Cipher.WRAP_MODE 和

算法	Mode	Padding	注意
			Cipher.UNWRAP_MODE 。 执行 AES-GCM 加密时，HSM 会忽略请求中的初始化向量 (IV) 并使用其生成的 IV。当该操作完成时，您必须调用 Cipher.getIV() 以获取 IV。
AESWrap	ECB	AESWrap/ECB/ZeroPadding AESWrap/ECB/NoPadding AESWrap/ECB/PKCS5Padding	实施 Cipher.WRAP_MODE 和 Cipher.UNWRAP_MODE 。使用转换 AES。
DESede（三重 DES）	CBC	DESede/CBC/NoPadding DESede/CBC/PKCS5Padding	实施 Cipher.ENCRYPT_MODE 和 Cipher.DECRYPT_MODE 。 密钥生成程序接受 168 或 192 位的大小。但是在内部，所有 DESede 密钥都是 192 位。 有关即将发生的更改，请参阅下面的注释 1。

算法	Mode	Padding	注意

DESede（三重 DES）	ECB	`DESede/ECB/NoPadding`
		
		`DESede/ECB/PKCS5Padding`
			实施 `Cipher.ENCRYPT_MODE` 和 `Cipher.DECRYPT_MODE` 。
			
			密钥生成程序接受 168 或 192 位的大小。但是在内部，所有 DESede 密钥都是 192 位。
			
			有关即将发生的更改，请参阅下面的注释 1。

RSA	ECB	`RSA/ECB/NoPadding`
		
		`RSA/ECB/PKCS1Padding`
			实施 `Cipher.ENCRYPT_MODE` 和 `Cipher.DECRYPT_MODE` 。
			
			有关即将发生的更改，请参阅下面的注释 1。

算法	Mode	Padding	注意
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	实施 Cipher.ENCRYPT_MODE 、Cipher.DECRYPT_MODE 、Cipher.WRAP_MODE 和 Cipher.UNWRAP_MODE 。 OAEPPadding 是采用 SHA-1 填充类型的 OAEP。
RSAAESWrap	ECB	OAEPPADDING	实施 Cipher.WRAP_Mode 和 Cipher.UNWRAP_MODE 。

支持的摘要

Java AWS CloudHSM 软件库支持以下消息摘要。

- SHA-1
- SHA-224
- SHA-256
- SHA-384
- SHA-512

> **Note**
>
> 长度小于 16 KB 的数据将在 HSM 上进行哈希处理，而较大的数据将在软件中本地进行哈希处理。

支持 HMAC 散列消息认证码算法

Java AWS CloudHSM 软件库支持以下 HMAC 算法。

- HmacSHA1
- HmacSHA224
- HmacSHA256
- HmacSHA384
- HmacSHA512

支持的签名/验证机制

Java AWS CloudHSM 软件库支持以下类型的签名和验证。

RSA 签名类型

- NONEwithRSA
- SHA1withRSA
- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA
- SHA1withRSA/PSS

- SHA224withRSA/PSS
- SHA256withRSA/PSS
- SHA384withRSA/PSS
- SHA512withRSA/PSS

ECDSA 签名类型

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

机制注释

[1] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

支持的 Java 密钥属性（客户端开发工具包 3）

本主题介绍如何使用 Java 库版本 3.1 的专有扩展来设置关键属性。使用此扩展可在以下操作期间设置受支持的密钥属性及其值：

- 密钥生成
- 密钥导入
- 密钥解开包装

> **Note**
>
> 用于设置自定义密钥属性的扩展是一项可选功能。如果您已有适用于 Java 库版本 3.0 的代码，则无需修改该代码。您创建的密钥仍将包含与以前相同的属性。

主题

了解属性

可以使用密钥属性指定允许对密钥对象（包括公有密钥或私有密钥）执行哪些操作。可以在创建密钥对象的过程中定义密钥属性和值。

但是，Java Cryptography Extension (JCE) 不指定如何设置密钥属性值，因此，默认情况下允许执行大多数操作。相比之下，PKCS #11 标准定义了一组具有更受限的默认值的综合属性。从 Java 库版本 3.1 开始，CloudHSM 提供了专有扩展，使您能够为常用属性设置更受限的值。

支持的属性

可以为下表中列出的属性设置值。作为最佳实践，仅为应受限的属性设置值。如果您未指定值，CloudHSM 将使用下表中指定的默认值。默认值列中的空单元格表示未向该属性分配特定的默认值。

属性	默认值			注意
	对称密钥	密钥对中的公有密钥	密钥对中的私有密钥	
CKA_TOKEN	FALSE	FALSE	FALSE	跨集群中的所有 HSM 复制并包含在备份中的永久密钥。CKA_TOKEN = FALSE 表示一个会话密钥，该密钥仅加载到一个 HSM 上，并且系统会在与 HSM 的连接断开时自动将其擦除。

属性	默认值			注意
CKA_LABEL				用户定义的字符串。您可以通过它轻松识别 HSM 上的密钥。
CKA_EXTRACTABLE	TRUE		TRUE	True 表示可从 HSM 中导出此密钥。
CKA_ENCRYPT	TRUE	TRUE		True 表示可使用密钥对任何缓冲区进行加密。
CKA_DECRYPT	TRUE		TRUE	True 表示可使用密钥对任何缓冲区进行解密。对于其 CKA_WRAP 设置为 true 的密钥，通常将此项设置为 FALSE。
CKA_WRAP	TRUE	TRUE		True 表示可使用密钥包装另一个密钥。对于私有密钥，通常将此项设置为 FALSE。
CKA_UNWRAP	TRUE		TRUE	True 表示可使用密钥解开包装（导入）另一个密钥。

属性	默认值			注意
CKA_SIGN	TRUE		TRUE	True 表示可使用密钥对消息摘要进行签名。对于已存档的公有密钥和私有密钥，此项通常设置为 FALSE。
CKA_VERIFY	TRUE	TRUE		True 表示可使用密钥验证签名。对于私有密钥，此项通常设置为 FALSE。
CKA_PRIVATE	TRUE	TRUE	TRUE	True 表示用户在通过身份验证之前将无法访问密钥。为了清楚起见，用户在通过身份验证之前将无法访问 CloudHSM 上的任何密钥，即使此属性设置为 FALSE 也是如此。

> **Note**
>
> 您将获得对 PKCS #11 库中属性的更广泛支持。有关更多信息，请参阅支持的 PKCS #11 属性。

设置密钥的属性

`CloudHsmKeyAttributesMap` 是一个类似于 Java Map 的对象，可以使用它设置密钥对象的属性值。`CloudHsmKeyAttributesMap` 函数的方法与用于 Java 映射操作的方法类似。

可以通过下面两种方式为属性设置自定义值：

- 使用下表中列出的方法
- 使用本文档后面演示的生成器模式

属性映射对象支持通过以下方法来设置属性：

操作	返回值	**`CloudHSMKeyAttributesMap`** 方法
获取现有密钥的密钥属性值	对象（包含值）或 null	get(keyAttribute)
填充一个密钥属性的值	与密钥属性关联的上一个值，或 null（如果没有密钥属性的映射）	put(keyAttribute, value)
填充多个密钥属性的值	不适用	putall () keyAttributesMap
从属性映射中删除密钥/值对	与密钥属性关联的上一个值，或 null（如果没有密钥属性的映射）	remove(keyAttribute)

> **Note**
>
> 未明确指定的任何属性都将设置为 the section called “支持的属性” 中前面的表中列出的默认值。

生成器模式示例

开发人员通常会发现，可以通过生成器模式更轻松地使用类。例如：

```
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();
```

开发人员还可以利用预定义的属性集来轻松地实施密钥模板中的最佳实践。例如：

```
//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
```

设置密钥对的属性

使用 Java 类 CloudHsmKeyPairAttributesMap 处理密钥对的密钥属性。CloudHsmKeyPairAttributesMap 封装了两个 CloudHsmKeyAttributesMap 对象；一个用于公有密钥，另一个用于私有密钥。

要分别为公有密钥和私有密钥设置单个属性，您可以对该密钥的相应 CloudHsmKeyAttributes 映射对象使用 put() 方法。使用 getPublic() 方法可检索公有密钥的属性映射，使用 getPrivate() 可检索私有密钥的属性映射。使用 putAll() 填充公有密钥和私有密钥对的多个密钥属性的值，并将密钥对属性映射作为其参数。

生成器模式示例

开发人员通常会发现，可以通过生成器模式更轻松地设置密钥属性。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**
>
> 有关此专有扩展的更多信息，请参阅 Javadoc 存档和上的示例。 GitHub要浏览 Javadoc，请下载并展开档案。

组合起来

要使用密钥操作指定密钥属性，请执行以下步骤：

1. 实例化对称密钥的 CloudHsmKeyAttributesMap 或密钥对的 CloudHsmKeyPairAttributesMap。
2. 使用所需的密钥属性和值来定义步骤 1 中的属性对象。
3. 实例化与特定密钥类型对应的 Cavium*ParameterSpec 类，并将此配置的属性对象传入其构造函数。
4. 将此 Cavium*ParameterSpec 对象传递给相应的加密类或方法。

下表包含支持自定义密钥属性的 Cavium*ParameterSpec 类和方法以供您参考。

密钥类型	参数规范类	示例构造函数
基本类	CaviumKeyGenAlgorithmParameterSpec	CaviumKeyGenAlgorithmParameterSpec(CloudHsmKeyAttributesMap keyAttributesMap)
DES	CaviumDESKeyGenParameterSpec	CaviumDESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap keyAttributesMap)
RSA	CaviumRSAKeyGenParameterSpec	CaviumRSAKeyGenParameterSpec(int keysize, BigInteger publicExponent, CloudHsmKeyPairAttributesMap keyPairAttributesMap)

密钥类型	参数规范类	示例构造函数
密钥	CaviumGenericSecretKeyGenParameterSpec	CaviumGenericSecretKeyGenParameterSpec(int size, CloudHsmKeyAttributesMap key AttributesMap)
AES	CaviumAESKeyGenParameterSpec	CaviumAESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap key AttributesMap)
EC	CaviumECGenParameterSpec	CaviumECGenParameterSpec(String stdName, CloudHsmKeyPairAttributesMap keyPairAttributesMap)

示例代码：生成和包装密钥

这些简短的代码示例演示了两项不同的操作的步骤：密钥生成和密钥包装：

```
// 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);
```

适用于客户端 SDK 的 Java AWS CloudHSM 软件库的代码示例 3

先决条件

运行示例前，必须设置环境：

- 安装和配置 Java 加密扩展（JCE）提供程序和AWS CloudHSM 客户端包。
- 设置有效的 HSM 用户名和密码。加密用户 (CU) 权限足以执行这些任务。在每个示例中，您的应用程序都将使用这些凭证登录 HSM。
- 决定如何向 JCE 提供程序 提供凭证。

代码示例

以下代码示例向您展示了如何使用 AWS CloudHSM JCE 提供程序 执行基本任务。有关更多代码示例，请访问GitHub。

- 登录 HSM
- 管理密钥
- 生成 AES 密钥
- 使用 AES-GCM 加密和解密
- 使用 AES-CTR 加密和解密
- 使用 D3DES-ECB 加密和解密[请参阅备注 1]
- 使用 AES-GCM 对密钥进行包装和解开包装
- 使用 AES 对密钥进行包装和解开包装
- 使用 RSA 对密钥进行包装和解开包装

- 使用支持的关键属性
- 枚举密钥存储中的密钥
- 使用 CloudHSM 密钥存储
- 在多线程示例中签名消息
- 使用 EC 密钥签名和验证

[1] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

为客户端 SDK 使用 AWS CloudHSM KeyStore Java 类 3

该 AWS CloudHSM **KeyStore**类提供了一个特殊用途的 PKCS12 密钥存储库，允许通过 k eytool 和 jarsigner 等应用程序访问 AWS CloudHSM 密钥。该密钥库可以将证书与您的密钥数据一起存储，并将它们与存储在 AWS CloudHSM上的密钥数据相关联。

> **Note**
>
> 由于证书是公共信息，并且为了最大限度地提高加密密钥的存储容量，因此 AWS CloudHSM 不支持在 HSM 上存储证书。

该 AWS CloudHSM `KeyStore`类实现了 Java 密码学扩展 (JCE) 的`KeyStore`服务提供者接口 (SPI)。有关使用的更多信息`KeyStore`，请参阅类KeyStore。

选择适当的密钥库

AWS CloudHSM Java 加密扩展 (JCE) 提供程序带有默认的直通式只读密钥存储，可将所有交易传递到 HSM。此默认密钥库不同于特殊用途 AWS CloudHSM KeyStore密钥库。在大多数情况下，您将通过使用默认值获得更好的运行时性能和吞吐量。除了将 AWS CloudHSM KeyStore 密钥操作卸载到 HSM 之外，您只应将它用于需要支持证书和基于证书的操作的应用程序。

尽管两个密钥库都使用 JCE 提供程序执行操作，但是它们是独立的实体，并且彼此之间不交换信息。

为您的 Java 应用程序加载默认密钥库，如下所示：

```
KeyStore ks = KeyStore.getInstance("Cavium");
```

按如下方式加载特殊用途 KeyStore CloudHSM：

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

正在初始化 AWS CloudHSM KeyStore

使用与登录 JCE 提供程序相同的方式登录。 AWS CloudHSM KeyStore 您可以使用环境变量或系统属性文件，并且应该在开始使用 CloudHSM KeyStore 之前登录。有关使用 JCE 提供程序登录 HSM 的示例，请参阅登录到 HSM。

如果需要，您可以指定密码以加密保存密钥库数据的本地 PKCS12 文件。创建 AWS CloudHSM 密钥库时，需要设置密码，并在使用加载、设置和获取方法时提供密码。

按如下方式实例化一个新的 CloudHSM 对象 KeyStore ：

```
ks.load(null, null);
```

使用 `store` 方法将密钥库数据写入文件。从那时起，您可以使用带有源文件和密码的 `load` 方法加载现有密钥库，如下所示：

```
ks.load(inputStream, password);
```

使用 AWS CloudHSM KeyStore

Cloud KeyStore HSM 对象通常通过第三方应用程序使用，例如 jarsigner 或 keytool。您也可以直接使用代码访问该对象。

AWS CloudHSM KeyStore 符合 JCE 类KeyStore规范，并提供以下功能。

- `load`

  从给定输入流加载密钥库。如果在保存密钥库时设置了密码，则必须提供相同的密码才能成功加载。将两个参数都设置为 null 可以初始化一个新的空密钥库。

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  ks.load(inputStream, password);
  ```

- `aliases`

  返回给定密钥库实例中所有条目的别名的枚举。结果包括本地存储在 PKCS12 文件中的对象和驻留在 HSM 上的对象。

  示例代码：

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();)
 {
  String label = entry.nextElement();
  System.out.println(label);
 }
```

- `ContainsAlias`

  如果密钥库可以访问至少一个具有指定别名的对象，则返回 true。密钥库检查本地存储在 PKCS12 文件中的对象和驻留在 HSM 上的对象。

- `DeleteEntry`

  从本地 PKCS12 文件中删除证书条目。不支持使用删除存储在 HSM 中的密钥数据。 AWS CloudHSM KeyStore您可以使用 CloudHSM 的 key_mgmt_util 工具删除密钥。

- `GetCertificate`

  返回与别名关联的证书（如果可用）。如果别名不存在或引用的对象不是证书，则该函数返回 NULL。

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
Certificate cert = ks.getCertificate(alias)
```

- `GetCertificateAlias`

  返回其数据与给定证书匹配的第一个密钥库条目的名称（别名）。

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
String alias = ks.getCertificateAlias(cert)
```

- `GetCertificateChain`

  返回与给定别名关联的证书链。如果别名不存在或引用的对象不是证书，则该函数返回 NULL。

- `GetCreationDate`

  返回由给定别名标识的条目的创建日期。如果创建日期不可用，则函数返回证书生效的日期。

- `GetKey`

  GetKey 传递给 HSM 并返回与给定标签对应的密钥对象。当`getKey`直接查询 HSM 时，它可以用于 HSM 上的任何密钥，无论该密钥是否由生成。 KeyStore

```
Key key = ks.getKey(keyLabel, null);
```

- IsCertificateEntry

  检查具有给定别名的条目是否表示证书条目。

- IsKeyEntry

  检查具有给定别名的条目是否表示密钥条目。该操作同时搜索 PKCS12 文件和 HSM 以查找别名。

- SetCertificateEntry

  将给定证书分配给给定别名。如果给定的别名已被用于标识密钥或证书，则会引发 KeyStoreException。您可以使用 JCE 代码获取密钥对象，然后使用 KeyStore SetKeyEntry方法将证书与密钥相关联。

- 使用 byte[] 密钥执行 SetKeyEntry

  客户端软件开发工具包 3 目前不支持此 API。

- 使用 Key 对象执行 SetKeyEntry

  将给定密钥分配到给定别名并将其存储在 HSM 中。如果 Key 对象不属于类型 CaviumKey，则该密钥将作为可提取会话密钥导入到 HSM 中。

  如果 Key 对象属于类型 PrivateKey，则它必须伴随相应的证书链。

  如果别名已存在，则 SetKeyEntry 调用会引发 KeyStoreException，并阻止该密钥被覆盖。如果密钥必须被覆盖，请为此目的使用 KMU 或 JCE。

- EngineSize

  返回密钥库中的条目数。

- Store

  将密钥库存储作为 PKCS12 文件存储到给定输出流，并使用给定的密码保护它。此外，它会保留所有加载的密钥（使用 setKey 调用进行设置）。

将第三方应用程序与 AWS CloudHSM集成

的一些用例 AWS CloudHSM 涉及将第三方软件应用程序与集群中的 HSM AWS CloudHSM 集成。通过将第三方软件与集成 AWS CloudHSM，您可以实现各种与安全相关的目标。以下主题介绍如何实现其中一些目标。

主题

- 使用 SSL/TLS 卸载来提高 Web 服务器的安全性 AWS CloudHSM
- 将 Windows Server 配置为具有 AWS CloudHSM的证书颁发机构 (CA)
- 使用 AWS CloudHSM的 Oracle Database 透明数据加密 (TDE)
- 使用 Microsof AWS CloudHSM t SignTool 与签名文件
- Java Keytool 和 Jarsigner
- 其他第三方供应商集成

使用 SSL/TLS 卸载来提高 Web 服务器的安全性 AWS CloudHSM

Web 服务器及其客户端（Web 浏览器）可以使用安全套接字层（SSL）或传输层安全性协议（TLS）来确认 Web 服务器的身份，并通过互联网建立发送和接收网页或其他数据的安全连接。这通常被称为 HTTPS。Web 服务器使用公有-私有密钥对和 SSL/TLS 公有密钥证书来与每个客户端建立 HTTPS 会话。此过程涉及大量的 Web 服务器计算，但您可以将其中一些计算卸载到 AWS CloudHSM 集群，这被称为 SSL 加速。分载将减轻 Web 服务器的计算负担，并将服务器的私有密钥存储在 HSM 中，从而提供额外的安全性。

以下主题概述了 SSL/TLS 卸载 AWS CloudHSM 的工作原理，以及在以下平台上设置 SSL/TLS 卸载的教程。 AWS CloudHSM

对于 Linux，在 NGINX 或 Apache HTTP 服务器 Web 服务器软件上使用 OpenSSL 动态引擎

Windows 使用 Internet Information Services (IIS) for Windows Server Web 服务器软件

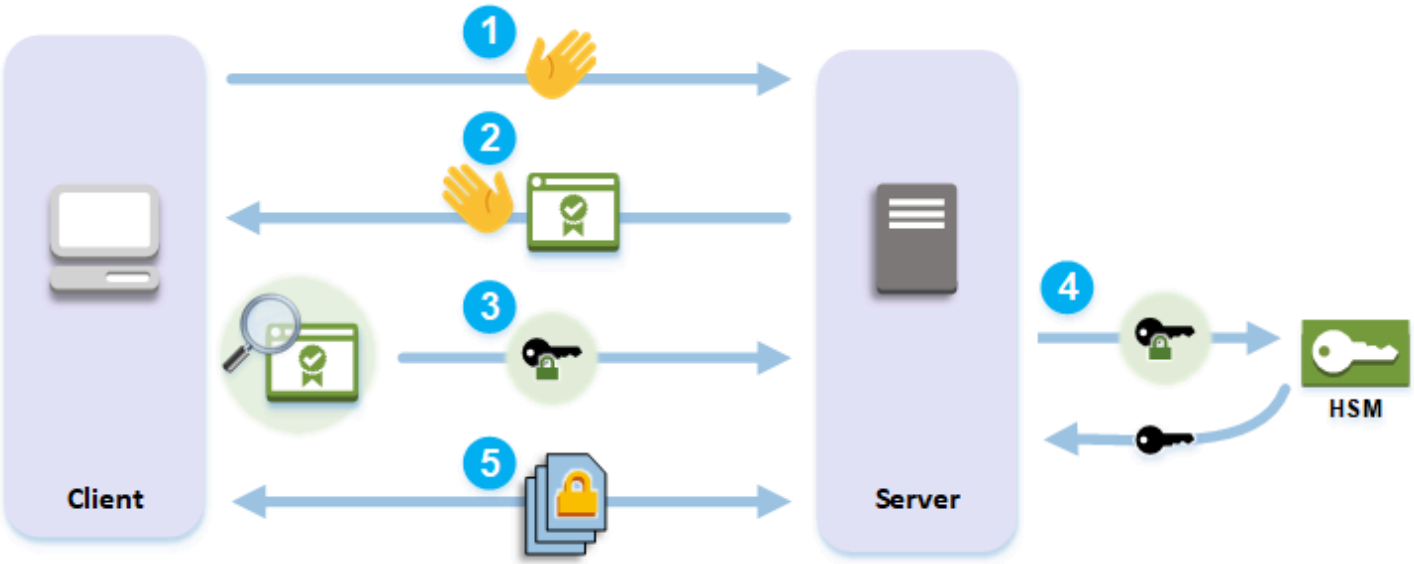
主题

- SSL/TLS 卸载的工作原理 AWS CloudHSM
- Linux 上的 SSL/TLS 分载
- 在 Windows 上使用借助 CNG 的 IIS 进行 SSL/TLS 分载
- 使用 Elastic Load Balancing 添加负载均衡器（可选）

SSL/TLS 卸载的工作原理 AWS CloudHSM

要建立 HTTPS 连接，您的 Web 服务器将执行与客户端的握手过程。在此过程中，服务器会将一些加密处理卸载到 HSM，如下图中所示。下图解释了此过程的每个步骤。

> **Note**
>
> 下面的图和过程假定 RSA 用于服务器验证和密钥交换。当使用的是 Diffie–Hellman 而不是 RSA 时，此过程稍有不同。



1. 客户端将 hello 消息发送给服务器。
2. 服务器响应 hello 消息并发送服务器的证书。
3. 客户端将执行以下操作：
   a. 验证 SSL/TLS 服务器证书是否由客户端信任的根证书签名。
   b. 从服务器证书中提取公有密钥。
   c. 生成预主密钥并使用服务器的公有密钥为它加密。
   d. 将加密的预主密钥发送给服务器。
4. 为解密客户端的预主密钥，服务器会将此密钥发送给 HSM。HSM 使用 HSM 中的私有密钥解密此预主密钥，然后它将预主密钥发送给服务器。客户端和服务器将独立使用此预主密钥和 hello 消息中的一些信息来计算主密钥。

5. 握手过程结束。在余下的会话中，客户端和服务器之间发送的所有消息将使用主密钥的派生密钥进行加密。

要了解如何使用配置 SSL/TLS 卸载 AWS CloudHSM，请参阅以下主题之一：

- Linux 上的 SSL/TLS 分载
- 在 Windows 上使用借助 CNG 的 IIS 进行 SSL/TLS 分载

Linux 上的 SSL/TLS 分载

使用 AWS CloudHSM，你可以使用 NGINX、Apache 和 Tomcat 在 Linux 上执行 SSL/TLS 卸载。有关更多信息，请参阅以下相关主题。

主题

- 在 Linux 上使用带有 OpenSSL 的 NGINX 或 Apache 进行 SSL/TLS 分载
- 在 Linux 上使用 Tomcat 和 JSSE 进行 SSL/TLS 分载

在 Linux 上使用带有 OpenSSL 的 NGINX 或 Apache 进行 SSL/TLS 分载

本主题提供在 Linux Web 服务器 AWS CloudHSM 上使用设置 SSL/TLS 卸载的 step-by-step 说明。

主题

- 概述
- 步骤 1：设置先决条件
- 步骤 2：生成或导入私有密钥和 SSL/TLS 证书
- 步骤 3：配置 Web 服务器
- 步骤 4：启用 HTTPS 流量并验证证书

概述

在 Linux 上，NGINX 和 Apache HTTP Server Web 服务器软件与 OpenSSL 集成以支持 HTTPS。适用于 OpenSSL 的AWS CloudHSM 动态引擎提供了一个接口，使 Web 服务器软件能够使用您群集中的 HSM 进行加密分载和密钥存储。OpenSSL 引擎是连接 Web 服务器与您的 AWS CloudHSM 群集的桥梁。

要完成本教程，您必须先选择在 Linux 上使用 NGINX 还是 Apache Web 服务器软件。然后，本教程将介绍如何执行以下操作：

- 在 Amazon EC2 实例上安装 Web 服务器软件。
- 使用存储在您的 AWS CloudHSM 集群中的私有密钥将 Web 服务器软件配置为支持 HTTPS。
- （可选）使用 Amazon EC2 创建第二个 Web 服务器实例，并使用 Elastic Load Balancing 创建负载均衡器。使用负载均衡器可以在多台服务器中分配负载，从而提高性能。它还能在一台或多台服务器发生故障的情况下提供冗余和更高的可用性。

在您准备好开始使用后，请转到 步骤 1：设置先决条件。

步骤 1：设置先决条件

不同的平台需要不同的先决条件。请使用以下与您的平台匹配的先决条件部分。

主题

- 客户端软件开发工具包 5 的先决条件
- 客户端软件开发工具包 3 的先决条件

客户端软件开发工具包 5 的先决条件

要使用客户端软件开发工具包 5 设置 Web 服务器 SSL/TLS 分载，需要满足以下条件：

- 具有至少两个硬件安全模块 (HSM) 的活动 AWS CloudHSM 集群

> **Note**
>
> 您可以使用单个 HSM 集群，但您必须首先禁用客户端密钥持久性。有关更多信息，请参阅管理客户端密钥持久性设置和客户端软件开发工具包 5 配置工具。

- 一个运行已安装以下软件的 Linux 操作系统的 Amazon EC2 实例：
  - Web 服务器（NGINX 或 Apache）
  - 使用适用于客户端软件开发工具包 5 的 OpenSSL 动态引擎
- 一个加密用户（CU），该用户拥有和管理 HSM 上的 Web 服务器的私有密钥。

在 HSM 上设置 Linux Web 服务器实例并创建 CU

1. 为安装和配置 OpenSSL 动态引擎。 AWS CloudHSM有关安装 OpenSSL 动态引擎的更多信息，请参阅适用于客户端软件开发工具包 5 的 OpenSSL 动态引擎。
2. 在可以访问您的集群的 EC2 Linux 实例上，安装 NGINX 或 Apache Web 服务器：

   Amazon Linux

   - NGINX

     ```
     $ sudo yum install nginx
     ```

   - Apache

     ```
     $ sudo yum install httpd24 mod24_ssl
     ```

   Amazon Linux 2

   - 有关如何在 Amazon Linux 2 上下载最新版本的 NGINX 的信息，请访问 NGINX 网站。

     适用于 Amazon Linux 2 最新版本的 NGINX 所使用的 OpenSSL 版本比 OpenSSL 的系统版本更新。安装 NGINX 后，你需要创建一个从 OpenSSL 动态引擎库到此版本的 AWS CloudHSM OpenSSL 所期望位置的符号链接

     ```
     $ 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

   - 有关如何在 CentOS 7 上下载最新版本 NGINX 的信息，请访问 NGINX 网站。

     适用于 CentOS 7 的最新版本 NGINX 所使用的 OpenSSL 版本比 OpenSSL 的系统版本更新。安装 NGINX 后，你需要创建一个从 OpenSSL 动态引擎库到此版本的 AWS CloudHSM OpenSSL 所期望位置的符号链接

```
$ 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

- 有关如何在 Red Hat 7 上下载最新版本 NGINX 的信息，请访问 NGINX 网站。

  适用于 Red Hat 7 的最新版本 NGINX 所使用的 OpenSSL 版本比 OpenSSL 的系统版本更新。安装 NGINX 后，你需要创建一个从 OpenSSL 动态引擎库到此版本的 AWS CloudHSM OpenSSL 所期望位置的符号链接

```
$ 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

尚不支持 OpenSSL 动态引擎。

3. 使用 CloudHSM CLI 创建加密用户（CU）。有关管理 HSM 用户的更多信息，请参阅使用 CloudHSM CLI 管理 HSM 用户。

> **Tip**
>
> 跟踪 CU 用户名和密码。您稍后为 Web 服务器生成或导入 HTTPS 私有密钥和证书时需要它们。

完成这些步骤后，请转到 步骤 2：生成或导入私有密钥和 SSL/TLS 证书。

注意

- 要使用安全增强型 Linux (SELinux) 和 Web 服务器，必须允许端口 2223 上的出站 TCP 连接，这是客户端软件开发工具包 5 用于与 HSM 通信的端口。
- 要创建和激活集群并授予 EC2 实例访问该集群的权限，请完成入门 AWS CloudHSM的步骤。入门 step-by-step 指南提供了使用一个 HSM 和 Amazon EC2 客户端实例创建活动集群的说明。您可使用此客户端实例作为您的 Web 服务器。
- 为避免禁用客户端密钥持久性，请向集群添加多个 HSM。有关更多信息，请参阅 添加 HSM。
- 要连接到客户端实例，可以使用 SSH 或 PuTTY。有关更多信息，请参阅 Amazon EC2 文档中的使用 SSH 连接到您的 Linux 实例或使用 PuTTY 从 Windows 连接到您的 Linux 实例。

客户端软件开发工具包 3 的先决条件

要使用客户端软件开发工具包 3 设置 Web 服务器 SSL/TLS 分载，需要满足以下条件：

- 至少有一个 HSM 的活动 AWS CloudHSM 集群。
- 一个运行已安装以下软件的 Linux 操作系统的 Amazon EC2 实例：
  - AWS CloudHSM 客户端和命令行工具。
  - NGINX 或 Apache Web 服务器应用程序。
  - Open AWS CloudHSM SSL 的动态引擎。
- 一个加密用户（CU），该用户拥有和管理 HSM 上的 Web 服务器的私有密钥。

在 HSM 上设置 Linux Web 服务器实例并创建 CU

1. 完成开始使用中的步骤。您随后将拥有一个带一个 HSM 的活动集群以及一个 Amazon EC2 客户端实例。您的 EC2 实例将用命令行工具进行配置。使用此客户端实例作为您的 Web 服务器。
2. 连接到您的客户端实例。有关更多信息，请参阅 Amazon EC2 文档中的使用 SSH 连接到您的 Linux 实例或使用 PuTTY 从 Windows 连接到您的 Linux 实例。
3. 在可以访问您的集群的 EC2 Linux 实例上，安装 NGINX 或 Apache Web 服务器：

   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，与 Amazon Linux 2 上的客户端软件开发工具包 3 引擎兼容。

  如需了解更多信息并下载 NGINX 1.19 版本，请访问 NGINX 网站。

- Apache

```
$ sudo yum install httpd mod_ssl
```

CentOS 7

- NGINX 版本 1.19 是 NGINX 的最新版本，与 CentOS 7 上的客户端软件开发工具包 3 引擎兼容。

  如需了解更多信息并下载 NGINX 1.19 版本，请访问 NGINX 网站。

- Apache

```
$ sudo yum install httpd mod_ssl
```

Red Hat 7

- NGINX 版本 1.19 是 NGINX 的最新版本，与 Red Hat 7 上的客户端软件开发工具包 3 引擎兼容。

  如需了解更多信息并下载 NGINX 1.19 版本，请访问 NGINX 网站。

- 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. (可选) 向您的集群添加更多 HSM。有关更多信息，请参阅 添加 HSM。
5. 使用 cloudhsm_mgmt_util 创建 CU。有关更多信息，请参阅 管理 HSM 用户。跟踪 CU 用户名和密码。您稍后为 Web 服务器生成或导入 HTTPS 私有密钥和证书时需要它们。

完成这些步骤后，请转到 步骤 2：生成或导入私有密钥和 SSL/TLS 证书。

步骤 2：生成或导入私有密钥和 SSL/TLS 证书

要启用 HTTPS，您的 Web 服务器应用程序（NGINX 或 Apache）需要一个私有密钥和相应的 SSL/TLS 证书。要将 Web 服务器 SSL/TLS 卸载与一起使用 AWS CloudHSM，必须将私钥存储在集群的 HSM 中。 AWS CloudHSM 您可以通过下列方式之一来完成此操作：

- 如果您没有私有密钥及相应的证书，则可以在 HSM 中生成私有密钥。使用私有密钥创建证书签名请求（CSR），并用它来创建 SSL/TLS 证书。
- 如果您已经有私有密钥及相应的证书，则可以将私有密钥导入到 HSM。

无论您选择上述哪种方法，您都可以从 HSM 中导出一个假 PEM 私有密钥，该私有密钥是 PEM 格式的私有密钥文件，其中包含对存储在 HSM 上的私有密钥的引用（它不是真正的私有密钥）。在 SSL/TLS 分载期间，您的 Web 服务器使用伪造的 PEM 私有密钥文件来识别 HSM 上的私有密钥。

请执行以下操作之一：

- 生成私有密钥和证书
- 导入现有私有密钥和证书

生成私有密钥和证书

生成私有密钥

本节将介绍如何使用客户端软件开发工具包 3 中的密钥管理实用程序（KMU）生成密钥对。在 HSM 中生成密钥对后，即可将其导出为伪造 PEM 文件，并生成相应的证书。

使用密钥管理实用程序（KMU）生成的私有密钥可以与客户端软件开发工具包 3 和客户端软件开发工具包 5 一起使用。

安装和配置密钥管理实用程序（KMU）

1. 连接到您的客户端实例。
2. 安装和配置客户端软件开发工具包 3。
3. 运行以下命令启动 AWS CloudHSM 客户端。

   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

尚不支持 OpenSSL 动态引擎。

4. 运行命令以启动 key_mgmt_util 命令行工具。

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. 运行以下命令登录 HSM。将 *<user name>* 和 *<password>* 替换为加密用户 (CU) 的用户名和密码。

```
Command: loginHSM -u CU -s <user name> -p <password>>
```

生成私有密钥

根据您的使用案例，您可以生成 RSA 或 EC 密钥对。请执行以下操作之一：

- 在 HSM 上生成 RSA 私有密钥

  使用 genRSAKeyPair 命令生成 RSA 密钥对。此示例生成一个模数为 2048、公有指数为 65537、标签为 *tls_rsa_keypair* 的 RSA 密钥对。

  ```
  Command: genRSAKeyPair -m 2048 -e 65537 -l tls_rsa_keypair
  ```

  如果命令成功，则您应该看到以下表明您已成功生成了 RSA 密钥对的输出。

  ```
  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 上生成 EC 私有密钥

  使用 genECCKeyPair 命令生成 EC 密钥对。此示例生成一个曲线 ID 为 2（对应 NID_X9_62_prime256v1 曲线）且标签为 *tls_ec_keypair* 的 EC 密钥对。

  ```
  Command: genECCKeyPair -i 2 -l tls_ec_keypair
  ```

  如果命令成功，则您应该看到以下表明您已成功生成了 EC 密钥对的输出。

  ```
  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
  ```

导出假 PEM 私有密钥文件

在 HSM 上拥有私有密钥后，您必须导出一个假 PEM 私有密钥文件。此文件不包含真密钥数据，但它允许 OpenSSL 动态引擎识别 HSM 上的私有密钥。您随后可以使用私有密钥创建证书签名请求（CSR），并签署 CSR 以创建证书。

> **Note**
>
> 使用密钥管理实用程序（KMU）生成的伪造 PEM 文件可以与客户端软件开发工具包 3 和客户端软件开发工具包 5 一起使用。

识别与您要导出为假 PEM 的密钥对应的密钥句柄，然后运行以下命令以假 PEM 格式导出私有密钥并将其保存到文件中。将以下值替换为您自己的值。

- *<private_key_handle>* – 生成的私有密钥的句柄。此句柄由上一步的密钥生成命令之一生成。在上一示例中，私有密钥的句柄为 8。
- *<web_server_fake_PEM.key>* –写入假 PEM 密钥的文件名称。

```
Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
```

Exit

运行以下命令停止 key_mgmt_util。

```
Command: exit
```

现在，您的系统上应该有一个新文件，该文件位于前述命令中*<web_server_fake_PEM.key>*所指定的路径。此文件是假 PEM 私有密钥文件。

生成自签名证书

生成伪造 PEM 私有密钥文件后，即可使用此文件生成证书签名请求（CSR）和证书。

在生产环境中，您通常使用证书颁发机构 (CA) 通过 CSR 创建证书。测试环境无需 CA。如果您使用 CA，则请将 CSR 文件发送给他们，然后使用他们在您的适用于 HTTPS的 Web 服务器中为您提供的签名 SSL/TLS 证书。

除了使用 CA 之外，您还可以使用 AWS CloudHSM OpenSSL 动态引擎创建自签名证书。自签名证书不受浏览器的信任，不应在生产环境中使用。它们可在测试环境中使用。

> **Warning**
>
> 自签名证书只应在测试环境中使用。对于生产环境，请使用更安全的方法 (如证书颁发机构) 来创建证书。

安装和配置 OpenSSL 动态引擎

1. 连接到您的客户端实例。
2. 要安装和配置，请执行以下操作之一：
   - the section called “安装 OpenSSL 动态引擎”
   - the section called “OpenSSL 动态引擎”

生成证书

1. 获取前面步骤中生成的伪造 PEM 文件的副本。
2. 创建 CSR

   运行以下命令使用 AWS CloudHSM OpenSSL 动态引擎创建证书签名请求 (CSR)。将 *<web_server_fake_PEM.key>* 替换为包含您的伪造 PEM 私有密钥的文件的名称。将 *<web_server.csr>* 替换为包含您的 CSR 的文件的名称。

   `req` 命令是交互式的。响应每个字段。字段信息将复制到您的 SSL/TLS 证书中。

   ```
   $ openssl req -engine cloudhsm -new -key <web_server_fake_PEM.key> -
   out <web_server.csr>
   ```

3. 创建自签名证书

   运行以下命令，使用 AWS CloudHSM OpenSSL 动态引擎在 HSM 上使用私钥签署您的 CSR。这会创建自签名证书。将命令中的以下值替换为您自己的值。

   - *<web_server.csr>* – 包含 CSR 的文件的名称。
   - *<web_server_fake_PEM.key>* – 包含伪造 PEM 私有密钥的文件的名称。
   - *<web_server.crt>* – 将包含您的 Web 服务器证书的文件的名称。

```
$ openssl x509 -engine cloudhsm -req -days 365 -in <web_server.csr> -
signkey <web_server_fake_PEM.key> -out <web_server.crt>
```

完成这些步骤后，请转到 步骤 3：配置 Web 服务器。

导入现有私有密钥和证书

您可能已经有一个私有密钥和可用于 Web 服务器上的 HTTPS 的相应 SSL/TLS 证书。如果是，您可以按照本部分中的步骤将密钥导入 HSM。

> **Note**
>
> 有关私有密钥导入和客户端软件开发工具包兼容性的一些说明：
>
> - 导入现有私有密钥需要客户端软件开发工具包 3。
> - 您可以结合使用客户端软件开发工具包 3 中的私有密钥与客户端软件开发工具包 5 中的私有密钥。
> - 适用于客户端软件开发工具包 3 的 OpenSSL 动态引擎不支持最新 Linux 平台，但适用于客户端软件开发工具包 5 的 OpenSSL 动态引擎的实现支持最新 Linux 平台。您可以使用客户端软件开发工具包 3 提供的密钥管理实用程序（KMU）导入现有私有密钥，然后使用该私有密钥和客户端软件开发工具包 5 的 OpenSSL 动态引擎实现来支持最新 Linux 平台上的 SSL/TLS 分载。

使用客户端软件开发工具包 3 将现有私有密钥导入到 HSM

1. 连接到您的 Amazon EC2 客户端实例。如有必要，将您的现有私有密钥和证书复制到该实例。
2. 安装和配置客户端软件开发工具包 3
3. 运行以下命令启动 AWS CloudHSM 客户端。

   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

尚不支持 OpenSSL 动态引擎。

4. 运行命令以启动 key_mgmt_util 命令行工具。

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. 运行以下命令登录 HSM。将 *<user name>* 和 *<password>* 替换为加密用户 (CU) 的用户名和密码。

```
Command: loginHSM -u CU -s <user name> -p <password>
```

6. 运行以下命令以将您的私有密钥导入到 HSM。

   a. 运行以下命令以创建仅对当前会话有效的对称包装密钥。命令和输出如下所示。

```
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. 运行以下命令以将您现有的私有密钥导入到 HSM。命令和输出如下所示。将以下值替换为您自己的值：

   - *<web_server_existing.key>* – 包含您的私有密钥的文件的名称。
   - *<web_server_imported_key>* – 您已导入私有密钥的标签。
   - *<wrapping_key_handle>* – 由上述命令生成的包装密钥句柄。在上一示例中，包装密钥句柄为 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. 运行以下命令以伪造 PEM 格式导出私有密钥并将其保存到文件。将以下值替换为您自己的值。

- *<private_key_handle>* – 已导入私有密钥的句柄。此句柄由上一步骤中的第二个命令生成。在上一示例中，私有密钥的句柄为 8。
- *<web_server_fake_PEM.key>* – 包含您已导出伪造 PEM 私有密钥的文件的名称。

```
Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
```

8. 运行以下命令停止 key_mgmt_util。

```
Command: exit
```

完成这些步骤后，请转到 步骤 3：配置 Web 服务器。

步骤 3：配置 Web 服务器

更新您的网络服务器软件的配置以使用 HTTPS 证书以及在上一步中创建的相应伪造 PEM 私有密钥。切记在开始前备份您现有的证书。这将完成 Linux网络服务器软件的设置，以便借助 AWS CloudHSM 进行 SSL/TLS 分载。

完成下述步骤之一。

主题

- 配置 NGINX网络服务器
- 配置 Apache 网络服务器

配置 NGINX网络服务器

使用此部分在支持的平台上配置 NGINX。

更新 NGINX 的网络服务器配置

1. 连接到您的客户端实例。
2. 运行以下命令以创建网络服务器证书和伪造 PEM 私有密钥所需的目录。

```
$ sudo mkdir -p /etc/pki/nginx/private
```

3. 运行以下命令以将您的网络服务器证书复制到所需位置。将 *<web_server.crt>* 替换为您的网络服务器证书的名称。

```
$ sudo cp <web_server.crt> /etc/pki/nginx/server.crt
```

4. 运行以下命令以将您的伪造 PEM 私有密钥复制到所需位置。将 *<web_server_fake_PEM.key>* 替换为包含您的伪造 PEM 私有密钥的文件的名称。

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/nginx/private/server.key
```

5. 运行以下命令更改文件所有权，以便名为 nginx 的用户可读取它们。

```
$ sudo chown nginx /etc/pki/nginx/server.crt /etc/pki/nginx/private/server.key
```

6. 运行以下命令可备份 /etc/nginx/nginx.conf 文件。

```
$ sudo cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

7. 更新 NGINX 的配置。

> **Note**
>
> 每个集群最多可在所有 NGINX 网络服务器上支持 1000 NGINX 工作进程。

Amazon Linux

使用文本编辑器编辑 /etc/nginx/nginx.conf 文件。这需要 Linux 根权限。在文件顶部，添加以下行：

- 如果使用客户端软件开发工具包 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 如果使用客户端软件开发工具包 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然后将以下内容添加至文件的 TLS 部分：

```
# 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

使用文本编辑器编辑 /etc/nginx/nginx.conf 文件。这需要 Linux 根权限。在文件顶部，添加以下行：

- 如果使用客户端软件开发工具包 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 如果使用客户端软件开发工具包 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然后将以下内容添加至文件的 TLS 部分：

```
# 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

使用文本编辑器编辑 /etc/nginx/nginx.conf 文件。这需要 Linux 根权限。在文件顶部，添加以下行：

- 如果使用客户端软件开发工具包 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 如果使用客户端软件开发工具包 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然后将以下内容添加至文件的 TLS 部分：

```
# 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

使用文本编辑器编辑 /etc/nginx/nginx.conf 文件。这需要 Linux 根权限。在文件顶部，添加以下行：

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然后将以下内容添加至文件的 TLS 部分：

```
# 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

使用文本编辑器编辑 /etc/nginx/nginx.conf 文件。这需要 Linux 根权限。在文件顶部，添加以下行：

- 如果使用客户端软件开发工具包 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- 如果使用客户端软件开发工具包 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然后将以下内容添加至文件的 TLS 部分：

```
# 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

使用文本编辑器编辑 /etc/nginx/nginx.conf 文件。这需要 Linux 根权限。在文件顶部，添加以下行：

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

然后将以下内容添加至文件的 TLS 部分：

```
# 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

使用文本编辑器编辑 /etc/nginx/nginx.conf 文件。这需要 Linux 根权限。在文件顶部，添加以下行：

```
ssl_engine cloudhsm;
    env n3fips_password;
```

然后将以下内容添加至文件的 TLS 部分：

```
# 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

使用文本编辑器编辑 /etc/nginx/nginx.conf 文件。这需要 Linux 根权限。在文件顶部，添加以下行：

```
ssl_engine cloudhsm;
    env CLOUDHSM_PIN;
```

然后将以下内容添加至文件的 TLS 部分：

```
# 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

使用文本编辑器编辑 /etc/nginx/nginx.conf 文件。这需要 Linux 根权限。在文件顶部，添加以下行：

```
ssl_engine cloudhsm;
    env CLOUDHSM_PIN;
```

然后将以下内容添加至文件的 TLS 部分：

```
# 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

尚不支持 OpenSSL 动态引擎。

保存该文件。

8. 备份 systemd 配置文件，然后设置 EnvironmentFile 路径。

Amazon Linux

无需操作。

Amazon Linux 2

1. 备份 nginx.service 文件。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 用文本编辑器打开 /lib/systemd/system/nginx.service 文件，然后在 [Service] 部分下添加下列路径：

```
EnvironmentFile=/etc/sysconfig/nginx
```

CentOS 7

无需操作。

CentOS 8

1. 备份 nginx.service 文件。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 用文本编辑器打开 /lib/systemd/system/nginx.service 文件，然后在 [Service] 部分下添加下列路径：

```
EnvironmentFile=/etc/sysconfig/nginx
```

Red Hat 7

无需操作。

Red Hat 8

1. 备份 nginx.service 文件。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 用文本编辑器打开 /lib/systemd/system/nginx.service 文件，然后在 [Service] 部分下添加下列路径：

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 16.04

1. 备份 nginx.service 文件。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 用文本编辑器打开 /lib/systemd/system/nginx.service 文件，然后在 [Service] 部分下添加下列路径：

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 18.04

1. 备份 nginx.service 文件。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 用文本编辑器打开 /lib/systemd/system/nginx.service 文件，然后在 [Service] 部分下添加下列路径：

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 20.04 LTS

1. 备份 nginx.service 文件。

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. 用文本编辑器打开 /lib/systemd/system/nginx.service 文件，然后在 [Service] 部分下添加下列路径：

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 22.04 LTS

尚不支持 OpenSSL 动态引擎。

9. 检查 /etc/sysconfig/nginx 文件是否存在，然后执行下列操作之一：

- 如果文件存在，请通过运行以下命令备份文件：

```
$ sudo cp /etc/sysconfig/nginx /etc/sysconfig/nginx.backup
```

- 如果文件不存在，请打开文本编辑器，然后在 /etc/sysconfig/ 文件夹中创建名为 nginx 的文件。

10. 配置 NGINX 环境。

> **Note**
>
> 客户端软件开发工具包 5 引入了用于存储 CU 凭证的 CLOUDHSM_PIN 环境变量。

Amazon Linux

在文本编辑器中打开 /etc/sysconfig/nginx 文件。这需要 Linux 根权限。添加加密用户 (CU) 凭证：

- 如果使用客户端软件开发工具包 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用客户端软件开发工具包 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

保存该文件。

Amazon Linux 2

在文本编辑器中打开 /etc/sysconfig/nginx 文件。这需要 Linux 根权限。添加加密用户 (CU) 凭证：

- 如果使用客户端软件开发工具包 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用客户端软件开发工具包 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

保存该文件。

CentOS 7

在文本编辑器中打开 /etc/sysconfig/nginx 文件。这需要 Linux 根权限。添加加密用户 (CU) 凭证：

- 如果使用客户端软件开发工具包 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用客户端软件开发工具包 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

保存该文件。

CentOS 8

在文本编辑器中打开 /etc/sysconfig/nginx 文件。这需要 Linux 根权限。添加加密用户 (CU) 凭证：

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

保存该文件。

Red Hat 7

在文本编辑器中打开 /etc/sysconfig/nginx 文件。这需要 Linux 根权限。添加加密用户 (CU) 凭证：

- 如果使用客户端软件开发工具包 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用客户端软件开发工具包 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

保存该文件。

Red Hat 8

在文本编辑器中打开 /etc/sysconfig/nginx 文件。这需要 Linux 根权限。添加加密用户 (CU) 凭证：

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

保存该文件。

Ubuntu 16.04 LTS

在文本编辑器中打开 /etc/sysconfig/nginx 文件。这需要 Linux 根权限。添加加密用户 (CU) 凭证：

```
n3fips_password=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

保存该文件。

Ubuntu 18.04 LTS

在文本编辑器中打开 /etc/sysconfig/nginx 文件。这需要 Linux 根权限。添加加密用户 (CU) 凭证：

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

保存该文件。

Ubuntu 20.04 LTS

在文本编辑器中打开 /etc/sysconfig/nginx 文件。这需要 Linux 根权限。添加加密用户 (CU) 凭证：

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

保存该文件。

Ubuntu 22.04 LTS

尚不支持 OpenSSL 动态引擎。

11. 启动 NGINX网络服务器。

Amazon Linux

在文本编辑器中打开 `/etc/sysconfig/nginx` 文件。这需要 Linux 根权限。添加加密用户 (CU) 凭证：

```
$ sudo service nginx start
```

Amazon Linux 2

停止任何运行中的 NGINX 进程

```
$ sudo systemctl stop nginx
```

重新加载 `systemd` 配置以获取最新更改

```
$ sudo systemctl daemon-reload
```

启动 NGINX 进程

```
$ sudo systemctl start nginx
```

CentOS 7

停止任何运行中的 NGINX 进程

```
$ sudo systemctl stop nginx
```

重新加载 `systemd` 配置以获取最新更改

```
$ sudo systemctl daemon-reload
```

启动 NGINX 进程

```
$ sudo systemctl start nginx
```

CentOS 8

停止任何运行中的 NGINX 进程

```
$ sudo systemctl stop nginx
```

重新加载 systemd 配置以获取最新更改

```
$ sudo systemctl daemon-reload
```

启动 NGINX 进程

```
$ sudo systemctl start nginx
```

Red Hat 7

停止任何运行中的 NGINX 进程

```
$ sudo systemctl stop nginx
```

重新加载 systemd 配置以获取最新更改

```
$ sudo systemctl daemon-reload
```

启动 NGINX 进程

```
$ sudo systemctl start nginx
```

Red Hat 8

停止任何运行中的 NGINX 进程

```
$ sudo systemctl stop nginx
```

重新加载 systemd 配置以获取最新更改

```
$ sudo systemctl daemon-reload
```

启动 NGINX 进程

```
$ sudo systemctl start nginx
```

Ubuntu 16.04 LTS

停止任何运行中的 NGINX 进程

```
$ sudo systemctl stop nginx
```

重新加载 systemd 配置以获取最新更改

```
$ sudo systemctl daemon-reload
```

启动 NGINX 进程

```
$ sudo systemctl start nginx
```

Ubuntu 18.04 LTS

停止任何运行中的 NGINX 进程

```
$ sudo systemctl stop nginx
```

重新加载 systemd 配置以获取最新更改

```
$ sudo systemctl daemon-reload
```

启动 NGINX 进程

```
$ sudo systemctl start nginx
```

Ubuntu 20.04 LTS

停止任何运行中的 NGINX 进程

```
$ sudo systemctl stop nginx
```

重新加载 systemd 配置以获取最新更改

```
$ sudo systemctl daemon-reload
```

启动 NGINX 进程

```
$ sudo systemctl start nginx
```

Ubuntu 22.04 LTS

尚不支持 OpenSSL 动态引擎。

12. （可选）将您的平台配置为在开启时启动 NGINX。

Amazon Linux

```
$ sudo chkconfig nginx on
```

Amazon Linux 2

```
$ sudo systemctl enable nginx
```

CentOS 7

无需操作。

CentOS 8

```
$ sudo systemctl enable nginx
```

Red Hat 7

无需操作。

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

尚不支持 OpenSSL 动态引擎。

在更新您的网络服务器配置后，转到步骤 4：启用 HTTPS 流量并验证证书。

配置 Apache 网络服务器

使用此部分在支持的平台上配置 Apache。

更新 Apache 的网络服务器配置

1. 连接到您的 Amazon EC2 客户端实例。
2. 为您的平台定义默认的凭证和私钥位置。

Amazon Linux

在 /etc/httpd/conf.d/ssl.conf 文件中，确保存在以下值：

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Amazon Linux 2

在 /etc/httpd/conf.d/ssl.conf 文件中，确保存在以下值：

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
```

```
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

CentOS 7

在 /etc/httpd/conf.d/ssl.conf 文件中，确保存在以下值：

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

CentOS 8

在 /etc/httpd/conf.d/ssl.conf 文件中，确保存在以下值：

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 7

在 /etc/httpd/conf.d/ssl.conf 文件中，确保存在以下值：

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 8

在 /etc/httpd/conf.d/ssl.conf 文件中，确保存在以下值：

```
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 文件中，确保存在以下值：

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 18.04 LTS

在 /etc/apache2/sites-available/default-ssl.conf 文件中，确保存在以下值：

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 20.04 LTS

在 /etc/apache2/sites-available/default-ssl.conf 文件中，确保存在以下值：

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 22.04 LTS

尚不支持 OpenSSL 动态引擎。

3. 将您的网络服务器证书复制到平台所需位置。

Amazon Linux

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

将 *<web_server.crt>* 替换为您的网络服务器证书的名称。

Amazon Linux 2

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

将 *<web_server.crt>* 替换为您的网络服务器证书的名称。

CentOS 7

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

将 *<web_server.crt>* 替换为您的网络服务器证书的名称。

CentOS 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

将 *<web_server.crt>* 替换为您的网络服务器证书的名称。

Red Hat 7

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

将 *<web_server.crt>* 替换为您的网络服务器证书的名称。

Red Hat 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

将 *<web_server.crt>* 替换为您的网络服务器证书的名称。

Ubuntu 16.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

将 *<web_server.crt>* 替换为您的网络服务器证书的名称。

Ubuntu 18.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

将 *<web_server.crt>* 替换为您的网络服务器证书的名称。

Ubuntu 20.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

将 *<web_server.crt>* 替换为您的网络服务器证书的名称。

Ubuntu 22.04 LTS

尚不支持 OpenSSL 动态引擎。

4. 将您的虚设 PEM 私钥复制到平台所需的位置。

Amazon Linux

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

将 *<web_server_fake_PEM.key>* 替换为包含您的伪造 PEM 私有密钥的文件的名称。

Amazon Linux 2

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

将 *<web_server_fake_PEM.key>* 替换为包含您的伪造 PEM 私有密钥的文件的名称。

CentOS 7

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

将 *<web_server_fake_PEM.key>* 替换为包含您的伪造 PEM 私有密钥的文件的名称。

CentOS 8

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

将 *<web_server_fake_PEM.key>* 替换为包含您的伪造 PEM 私有密钥的文件的名称。

Red Hat 7

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

将 *<web_server_fake_PEM.key>* 替换为包含您的伪造 PEM 私有密钥的文件的名称。

Red Hat 8

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

将 *<web_server_fake_PEM.key>* 替换为包含您的伪造 PEM 私有密钥的文件的名称。

Ubuntu 16.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

将 *<web_server_fake_PEM.key>* 替换为包含您的伪造 PEM 私有密钥的文件的名称。

Ubuntu 18.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

将 *<web_server_fake_PEM.key>* 替换为包含您的伪造 PEM 私有密钥的文件的名称。

Ubuntu 20.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

将 *<web_server_fake_PEM.key>* 替换为包含您的伪造 PEM 私有密钥的文件的名称。

Ubuntu 22.04 LTS

尚不支持 OpenSSL 动态引擎。

5. 如果您的平台需要，可以更改该文件的所有权。

Amazon Linux

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

为名为apache 的用户提供读取权限。

Amazon Linux 2

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

为名为apache 的用户提供读取权限。

CentOS 7

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

为名为apache 的用户提供读取权限。

CentOS 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

为名为apache 的用户提供读取权限。

Red Hat 7

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

为名为apache 的用户提供读取权限。

Red Hat 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

为名为apache 的用户提供读取权限。

Ubuntu 16.04 LTS

无需操作。

Ubuntu 18.04 LTS

无需操作。

Ubuntu 20.04 LTS

无需操作。

Ubuntu 22.04 LTS

尚不支持 OpenSSL 动态引擎。

6. 为您的平台配置 Apache 指令。

Amazon Linux

找到此平台的 SSL 文件：

```
/etc/httpd/conf.d/ssl.conf
```

此文件包含定义服务器运行方式的 Apache 指令。指令在左侧显示，后面跟值。使用文本编辑器编辑此文件。这需要 Linux 根权限。

通过这些值更新或输入以下指令：

```
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
```

保存该文件。

Amazon Linux 2

找到此平台的 SSL 文件：

```
/etc/httpd/conf.d/ssl.conf
```

此文件包含定义服务器运行方式的 Apache 指令。指令在左侧显示，后面跟值。使用文本编辑器编辑此文件。这需要 Linux 根权限。

通过这些值更新或输入以下指令：

```
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
```

保存该文件。

CentOS 7

找到此平台的 SSL 文件：

```
/etc/httpd/conf.d/ssl.conf
```

此文件包含定义服务器运行方式的 Apache 指令。指令在左侧显示，后面跟值。使用文本编辑器编辑此文件。这需要 Linux 根权限。

通过这些值更新或输入以下指令：

```
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
```

保存该文件。

CentOS 8

找到此平台的 SSL 文件：

```
/etc/httpd/conf.d/ssl.conf
```

此文件包含定义服务器运行方式的 Apache 指令。指令在左侧显示，后面跟值。使用文本编辑器编辑此文件。这需要 Linux 根权限。

通过这些值更新或输入以下指令：

```
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
```

保存该文件。

Red Hat 7

找到此平台的 SSL 文件：

```
/etc/httpd/conf.d/ssl.conf
```

此文件包含定义服务器运行方式的 Apache 指令。指令在左侧显示，后面跟值。使用文本编辑器编辑此文件。这需要 Linux 根权限。

通过这些值更新或输入以下指令：

```
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
```

保存该文件。

Red Hat 8

找到此平台的 SSL 文件：

```
/etc/httpd/conf.d/ssl.conf
```

此文件包含定义服务器运行方式的 Apache 指令。指令在左侧显示，后面跟值。使用文本编辑器编辑此文件。这需要 Linux 根权限。

通过这些值更新或输入以下指令：

```
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
```

保存该文件。

Ubuntu 16.04 LTS

找到此平台的 SSL 文件：

```
/etc/apache2/mods-available/ssl.conf
```

此文件包含定义服务器运行方式的 Apache 指令。指令在左侧显示，后面跟值。使用文本编辑器编辑此文件。这需要 Linux 根权限。

通过这些值更新或输入以下指令：

```
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
```

保存该文件。

启用 SSL 模块以及默认 SSL 站点配置：

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 18.04 LTS

找到此平台的 SSL 文件：

```
/etc/apache2/mods-available/ssl.conf
```

此文件包含定义服务器运行方式的 Apache 指令。指令在左侧显示，后面跟值。使用文本编辑器编辑此文件。这需要 Linux 根权限。

通过这些值更新或输入以下指令：

```
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
```

保存该文件。

启用 SSL 模块以及默认 SSL 站点配置：

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 20.04 LTS

找到此平台的 SSL 文件：

```
/etc/apache2/mods-available/ssl.conf
```

此文件包含定义服务器运行方式的 Apache 指令。指令在左侧显示，后面跟值。使用文本编辑器编辑此文件。这需要 Linux 根权限。

通过这些值更新或输入以下指令：

```
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
```

保存该文件。

启用 SSL 模块以及默认 SSL 站点配置：

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 22.04 LTS

尚不支持 OpenSSL 动态引擎。

7. 为您的平台配置环境值文件。

Amazon Linux

无需操作。环境价值被纳入 `/etc/sysconfig/httpd`

Amazon Linux 2

打开 httpd 服务文件：

```
/lib/systemd/system/httpd.service
```

在 [Service] 部分添加以下内容：

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 7

打开 httpd 服务文件：

```
/lib/systemd/system/httpd.service
```

在 [Service] 部分添加以下内容：

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 8

打开 httpd 服务文件：

```
/lib/systemd/system/httpd.service
```

在 [Service] 部分添加以下内容：

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 7

打开 httpd 服务文件：

```
/lib/systemd/system/httpd.service
```

在 [Service] 部分添加以下内容：

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 8

打开 httpd 服务文件：

```
/lib/systemd/system/httpd.service
```

在 [Service] 部分添加以下内容：

```
EnvironmentFile=/etc/sysconfig/httpd
```

Ubuntu 16.04 LTS

无需操作。环境价值被纳入 /etc/sysconfig/httpd

Ubuntu 18.04 LTS

无需操作。环境价值被纳入 /etc/sysconfig/httpd

Ubuntu 20.04 LTS

无需操作。环境价值被纳入 /etc/sysconfig/httpd

Ubuntu 22.04 LTS

尚不支持 OpenSSL 动态引擎。

8. 在存储平台环境变量的文件中，设置一个包含加密用户 (CU) 凭证的环境变量：

Amazon Linux

使用文本编辑器编辑 /etc/sysconfig/httpd。

- 如果使用客户端软件开发工具包 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用客户端软件开发工具包 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

Amazon Linux 2

使用文本编辑器编辑 /etc/sysconfig/httpd。

- 如果使用客户端软件开发工具包 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用客户端软件开发工具包 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

CentOS 7

使用文本编辑器编辑 /etc/sysconfig/httpd。

- 如果使用客户端软件开发工具包 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用客户端软件开发工具包 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

CentOS 8

使用文本编辑器编辑 /etc/sysconfig/httpd。

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

Red Hat 7

使用文本编辑器编辑 /etc/sysconfig/httpd。

- 如果使用客户端软件开发工具包 3

```
n3fips_password=<CU user name>:<password>
```

- 如果使用客户端软件开发工具包 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

Red Hat 8

使用文本编辑器编辑 /etc/sysconfig/httpd。

```
CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

> **Note**
>
> 客户端软件开发工具包 5 引入了用于存储 CU 凭证的 CLOUDHSM_PIN 环境变量。

Ubuntu 16.04 LTS

使用文本编辑器编辑 /etc/apache2/envvars。

```
export n3fips_password=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

Ubuntu 18.04 LTS

使用文本编辑器编辑 /etc/apache2/envvars。

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

> **Note**
>
> 客户端软件开发工具包 5 引入了用于存储 CU 凭证的 CLOUDHSM_PIN 环境变量。在客户端软件开发工具包 3 中，您将 CU 凭据存储在 n3fips_password 环境变量中。客户端软件开发工具包 5 支持这两个环境变量，但我们建议使用 CLOUDHSM_PIN。

Ubuntu 20.04 LTS

使用文本编辑器编辑 /etc/apache2/envvars。

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

将 *<CU user name>* 和 *<password>* 替换为 CU 凭证。

> **Note**
>
> 客户端软件开发工具包 5 引入了用于存储 CU 凭证的 CLOUDHSM_PIN 环境变量。在客户端软件开发工具包 3 中，您将 CU 凭据存储在 n3fips_password 环境变量中。客户端软件开发工具包 5 支持这两个环境变量，但我们建议使用 CLOUDHSM_PIN。

Ubuntu 22.04 LTS

尚不支持 OpenSSL 动态引擎。

9. 启动 Apache网络服务器。

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

尚不支持 OpenSSL 动态引擎。

10. （可选）将您的平台配置为在开启时启动 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

尚不支持 OpenSSL 动态引擎。

在更新您的网络服务器配置后，转到 步骤 4：启用 HTTPS 流量并验证证书。

步骤 4：启用 HTTPS 流量并验证证书

使用 SSL/TLS 卸载配置 Web 服务器后 AWS CloudHSM，将您的 Web 服务器实例添加到允许入站 HTTPS 流量的安全组中。Web 浏览器等客户端可通过 Web 服务器创建 HTTPS 连接。然后与您的 Web 服务器建立 HTTPS 连接，并确认它使用的是您为 SSL/TLS 卸载配置的证书。 AWS CloudHSM

主题

- 启用入站 HTTPS 连接
- 验证 HTTPS 使用的是您已配置的证书

启用入站 HTTPS 连接

要从客户端 (如 Web 浏览器) 连接到您的 Web 服务器，请创建一个允许入站 HTTPS 连接的安全组。具体来说，它应允许端口 443 上的入站 TCP 连接。将此安全组分配给您的网络服务器。

为 HTTPS 创建安全组并将其分配给您的网络服务器

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在导航窗格中，选择安全组。
3. 选择创建安全组。
4. 对于创建安全组，执行以下操作：
    a. 对于安全组名称，键入您要创建的安全组的名称。
    b. (可选) 键入对您要创建的安全组的描述。
    c. 对于 VPC，选择包含您的 Web 服务器 Amazon EC2 实例的 VPC。
    d. 选择添加规则。

e. 对于类型，从下拉窗口中选择 HTTPS。

f. 对于来源，输入来源位置。

g. 选择创建安全组。

5. 在导航窗格中，选择实例。
6. 选中您的 Web 服务器实例旁边的复选框。
7. 在页面顶部选择操作下拉菜单。选择安全，然后选择更改安全组。
8. 对于关联安全组，请选择搜索框，然后选择您为 HTTPS 创建的安全组。然后选择添加安全组。
9. 选择保存。

验证 HTTPS 使用的是您已配置的证书

将 Web 服务器添加到安全组后，就可以验证 SSL/TLS 分载是否使用了自签名证书。您可以使用网络浏览器或使用工具 (如 OpenSSL s_client) 执行此操作。

使用网络浏览器验证 SSL/TLS 分载

1. 使用 Web 浏览器连接到采用服务器的公共 DNS 名称或 IP 地址的 Web 服务器。确保地址栏中的 URL 以 https:// 开头。例如，**`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**。

   > **Tip**
   >
   > 您可以使用 DNS 服务 (如 Amazon Route 53) 将网站域名 (例如，https://www.example.com/) 路由到您的 Web 服务器。有关更多信息，请参阅 《Amazon Route 53 开发人员指南》或 DNS 服务文档中的将流量路由到 Amazon EC2 实例。

2. 使用您的 Web 浏览器查看 Web 服务器证书。有关更多信息，请参阅下列内容：

   - 对于 Mozilla Firefox，请参阅 Mozilla Support 网站上的查看证书。
   - 关于 Google Chrome 浏览器，请参阅 Google Tools for Web Developers 网站上的了解安全问题。

   其他网络浏览器可能具有相似的功能，可使用这些功能来查看网络服务器证书。

3. 确保 SSL/TLS 证书是您配置 Web 服务器使用的证书。

使用 OpenSSL s_client 验证 SSL/TLS 分载

1. 运行以下 OpenSSL 命令以通过 HTTPS 连接到您的 Web 服务器。将 *<server name>* 替换为您的 Web 服务器的公有 DNS 名称或 IP 地址。

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> 您可以使用 DNS 服务 (如 Amazon Route 53) 将网站域名 (例如，https://www.example.com/) 路由到您的 Web 服务器。有关更多信息，请参阅 《Amazon Route 53 开发人员指南》或 DNS 服务文档中的将流量路由到 Amazon EC2 实例。

2. 确保 SSL/TLS 证书是您配置 Web 服务器使用的证书。

您现在有一个通过 HTTPS 保护的网站。Web 服务器的私钥存储在 AWS CloudHSM 集群的 HSM 中。

若要添加负载均衡器，请参阅 使用 Elastic Load Balancing 添加负载均衡器（可选）。

在 Linux 上使用 Tomcat 和 JSSE 进行 SSL/TLS 分载

本主题提供使用 Java 安全套接字扩展 (JSSE) 和 JCE SDK 设置 SSL/TLS 卸载的 step-by-step 说明。AWS CloudHSM

主题

概述

在中 AWS CloudHSM，Tomcat 网络服务器在 Linux 上运行以支持 HTTPS。 AWS CloudHSM JCE SDK 提供了一个可与 JSSE（Java 安全套接字扩展）配合使用的接口，以允许在这类 Web 服务器上使用 HSM。 AWS CloudHSM JCE 是将 JSSE 连接到您的 AWS CloudHSM 集群的桥梁。JSSE 是用于安全套接字层（SSL）/传输层安全性协议（TLS）的 Java API。

步骤 1：设置先决条件

要在 Linux 上使用 Tomcat Web 服务器和 SSL/TLS AWS CloudHSM 卸载，请遵循以下先决条件。要使用客户端软件开发工具包 5 和 Tomcat Web 服务器设置 Web 服务器的 SSL/TLS 分载，则必须满足这些先决条件。

> **Note**
>
> 不同的平台需要不同的先决条件。请始终按照适用于您的平台的正确安装步骤进行操作。

先决条件

- 运行安装了 tomcat Web 服务器的 Linux 操作系统的 Amazon EC2 实例。
- 一个加密用户（CU），该用户拥有和管理 HSM 上的 Web 服务器的私有密钥。
- 具有至少两个硬件安全模块 (HSM) 的活动 AWS CloudHSM 集群，且安装并配置了 JCE for Client SDK 5。

> **Note**
>
> 您可以使用单个 HSM 集群，但您必须首先禁用客户端密钥持久性。有关更多信息，请参阅管理客户端密钥持久性设置和客户端软件开发工具包 5 配置工具。

如何满足先决条件

1. 在至少有两个硬件安全模块 (HSM) 的活动 AWS CloudHSM 集群 AWS CloudHSM 上安装和配置 JCE。有关安装的更多信息，请参阅适用于客户端软件开发工具包 5 的 JCE。
2. 在可以访问您的 AWS CloudHSM 集群的 EC2 Linux 实例上，按照 Apache Tomcat 的说明下载并安装 Tomcat Web 服务器。
3. 使用 CloudHSM CLI 创建加密用户（CU）。有关管理 HSM 用户的更多信息，请参阅使用 CloudHSM CLI 管理 HSM 用户。

   > **Tip**
   >
   > 跟踪 CU 用户名和密码。您稍后为 Web 服务器生成或导入 HTTPS 私有密钥和证书时需要它们。

4. 要使用 Java Keytool 设置 JCE，请按照 使用客户端软件开发工具包 5 与 Java Keytool 和 Jarsigner 集成 中的说明进行操作。

完成这些步骤后，请转到 步骤 2：生成或导入私有密钥和 SSL/TLS 证书。

注意

- 要使用安全增强型 Linux (SELinux) 和 Web 服务器，必须允许端口 2223 上的出站 TCP 连接，这是客户端软件开发工具包 5 用于与 HSM 通信的端口。
- 要创建和激活集群并授予 EC2 实例访问该集群的权限，请完成入门 AWS CloudHSM的步骤。本节提供 step-by-step 有关创建包含一个 HSM 和一个 Amazon EC2 客户端实例的活动集群的说明。您可使用此客户端实例作为您的 Web 服务器。
- 为避免禁用客户端密钥持久性，请向集群添加多个 HSM。有关更多信息，请参阅 添加 HSM。
- 要连接到客户端实例，可以使用 SSH 或 PuTTY。有关更多信息，请参阅 Amazon EC2 文档中的使用 SSH 连接到您的 Linux 实例或使用 PuTTY 从 Windows 连接到您的 Linux 实例。

步骤 2：生成或导入私有密钥和 SSL/TLS 证书

要启用 HTTPS，您的 Tomcat Web 服务器应用程需要私有密钥和相应的 SSL/TLS 证书。要将 Web 服务器 SSL/TLS 卸载与一起使用 AWS CloudHSM，必须将私钥存储在集群的 HSM 中。 AWS CloudHSM

> **Note**
>
> 如果您没有私有密钥及相应的证书，则可以在 HSM 中生成私有密钥。使用私有密钥创建证书签名请求（CSR），并用它来创建 SSL/TLS 证书。

您可以创建一个本地 AWS CloudHSM KeyStore 文件，其中包含对您在 HSM 上的私钥的引用和关联的证书。在 SSL/TLS 卸载期间，您的 Web 服务器使用该 AWS CloudHSM KeyStore 文件来识别 HSM 上的私钥。

主题

- 生成私有密钥
- 生成自签名证书

生成私有密钥

本节介绍如何使用来 KeyTool 自 JDK 的生成密钥对。在 HSM 中生成密钥对后，可以将其导出为 KeyStore 文件，然后生成相应的证书。

根据您的使用案例，您可以生成 RSA 或 EC 密钥对。以下步骤将显示如何生成 RSA 密钥对。

使用中的**genkeypair** KeyTool 命令生成 RSA key pair

1. 用您的特定数据替换 *<VARIABLES>* 以下内容后，使用以下命令生成名为`jsse_keystore.keystore`的密钥库文件，该文件将引用您在 HSM 上的私有密钥。

```
$ 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>***：待生成密钥库文件的路径。
   - ***<UNIQUE ALIAS FOR KEYS>***：这是您在 HSM 上的密钥的唯一标识。此别名将被设置为密钥的 LABEL 属性。
   - ***<KEY PASSWORD>***：我们将对您的密钥的引用存储在本地密钥库文件中，此密码可以保护该本地引用。
   - ***<KEYSTORE PASSWORD>***：这是您的本地密钥库文件的密码。
   - ***<JSSE KEYSTORE NAME>***：密钥库文件的名称。
   - ***<CERT DOMAIN NAME>***：X.500 可分辨名称。
   - ***<KEY ALGORITHM>***：生成密钥对的密钥算法（例如，RSA 和 EC）。
   - ***<KEY SIZE>***：生成密钥对的密钥大小（例如，2048、3072 和 4096）。
   - ***<SIGN ALGORITHM>***：生成密钥对的密钥大小（例如 SHA1withRSA、SHA224withRSA、SHA256withRSA、SHA384withRSA 和 SHA512withRSA）。
2. 要确认命令是否成功，请输入以下命令并验证您是否成功生成了 RSA 密钥对。

```
$ ls <PATH>/<JSSE KEYSTORE NAME>.keystore
```

生成自签名证书

生成私有密钥和密钥库文件后，即可使用此文件生成证书签名请求（CSR）和证书。

在生产环境中，您通常使用证书颁发机构 (CA) 通过 CSR 创建证书。测试环境无需 CA。如果您使用 CA，则请将 CSR 文件发送给他们，然后使用他们在您的适用于 HTTPS的 Web 服务器中为您提供的签名 SSL/TLS 证书。

除了使用 CA 之外，您还可以使用创建自签名证书。 KeyTool 自签名证书不受浏览器的信任，不应在生产环境中使用。它们可在测试环境中使用。

> **Warning**
>
> 自签名证书只应在测试环境中使用。对于生产环境，请使用更安全的方法 (如证书颁发机构) 来创建证书。

生成证书

1. 获取前面步骤中生成的密钥库文件的副本。
2. 运行以下命令以使用创建证书签名请求 (CSR)。 KeyTool

```
$ keytool -certreq -keyalg RSA -alias unique_alias_for_key -file certreq.csr \
        -keystore <JSSE KEYSTORE NAME>.keystore -storetype CLOUDHSM \
        -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \
        -keypass <KEY PASSWORD> -storepass <KEYSTORE PASSWORD>
```

> **Note**
>
> 证书签名请求的输出文件是`certreq.csr`。

签署证书

- 用您的特定数据替换 *<VARIABLES>* 以下内容后，运行以下命令在 HSM 上使用私有密钥签署您的 CSR。这会创建自签名证书。

```
$ 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是使用别名私有密钥的已签名证书。

在密钥库中导入证书

- 用您的特定数据替换 *<VARIABLES>* 以下内容后，运行以下命令将签名证书作为可信证书导入。此步骤会将证书存储在由别名标识的密钥库条目中。

```
$ 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>
```

将证书转换为 PEM

- 运行以下命令将签名证书文件（.crt）转换为 PEM。PEM 文件将用于发送来自于 http 客户端的请求。

```
$ openssl x509 -inform der -in certificate.crt -out certificate.pem
```

完成这些步骤后，转至步骤 3：配置 Web 服务器。

步骤 3：配置 Tomcat Web 服务器

更新您的 Web 服务器软件的配置，以使用 HTTPS 证书和在上一步中创建的相应 PEM 文件。切记在开始前备份您现有的证书。这将完成 Linux网络服务器软件的设置，以便借助 AWS CloudHSM进行 SSL/TLS 分载。有关更多信息，请参考 Apache Tomcat 9 配置参考。

停止服务器

- 用您的特定数据替换 *<VARIABLES>* 以下内容后，请在更新配置之前运行以下命令停止 Tomcat Server

```
$ /<TOMCAT DIRECTORY>/bin/shutdown.sh
```

- ***<TOMCAT DIRECTORY>***：您的 Tomcat 安装目录。

更新 Tomcat 的类路径

1. 连接到您的客户端实例。
2. 找到 Tomcat 安装文件夹。
3. 用您的特定数据替换 *<VARIABLES>* 后，使用以下命令在 Tomcat 类路径中 Tomcat/bin/catalina.sh 文件下添加 Java 库和 Cloudhsm Java 路径。

```
$ 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 库的位置。
- ***<TOMCAT PATH>***：Tomcat 安装文件夹。

在服务器配置中添加 HTTPS 连接器。

1. 前往 Tomcat 安装文件夹。
2. 用您的特定数据替换 *<VARIABLES>* 以下内容后，使用以下命令添加 HTTPS 连接器以使用先决条件生成的证书：

```
$ 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>***：密钥库文件所在的目录。
- ***<JSSE KEYSTORE NAME>***：密钥库文件的名称。
- ***<KEYSTORE PASSWORD>***：这是您的本地密钥库文件的密码。

- ***<KEY PASSWORD>***：我们将对您的密钥的引用存储在本地密钥库文件中，此密码可以保护该本地引用。
- ***<UNIQUE ALIAS FOR KEYS>***：这是您在 HSM 上的密钥的唯一标识。此别名将被设置为密钥的 LABEL 属性。
- ***<TOMCAT PATH>***：您的 Tomcat 文件夹的路径。

启动服务器

- 用您的特定数据替换 *<VARIABLES>* 以下内容后，请运行以下命令启动 Tomcat Server

```
$ /<TOMCAT DIRECTORY>/bin/startup.sh
```

> **Note**
>
> ***<TOMCAT DIRECTORY>***是您的 Tomcat 安装目录的名称。

在更新您的网络服务器配置后，转到 步骤 4：启用 HTTPS 流量并验证证书。

步骤 4：启用 HTTPS 流量并验证证书

使用 SSL/TLS 卸载配置 Web 服务器后 AWS CloudHSM，将您的 Web 服务器实例添加到允许入站 HTTPS 流量的安全组中。Web 浏览器等客户端可通过 Web 服务器创建 HTTPS 连接。然后与您的 Web 服务器建立 HTTPS 连接，并确认它使用的是您为 SSL/TLS 卸载配置的证书。 AWS CloudHSM

主题

- 启用入站 HTTPS 连接
- 验证 HTTPS 使用的是您已配置的证书

启用入站 HTTPS 连接

要从客户端 (如 Web 浏览器) 连接到您的 Web 服务器，请创建一个允许入站 HTTPS 连接的安全组。具体来说，它应允许端口 443 上的入站 TCP 连接。将此安全组分配给您的网络服务器。

为 HTTPS 创建安全组并将其分配给您的网络服务器

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。

2. 在导航窗格中，选择安全组。
3. 选择创建安全组。
4. 对于创建安全组，执行以下操作：
   a. 对于安全组名称，键入您要创建的安全组的名称。
   b. (可选) 键入对您要创建的安全组的描述。
   c. 对于 VPC，选择包含您的 Web 服务器 Amazon EC2 实例的 VPC。
   d. 选择添加规则。
   e. 对于类型，从下拉窗口中选择 HTTPS。
   f. 对于来源，输入来源位置。
   g. 选择创建安全组。
5. 在导航窗格中，选择实例。
6. 选中您的 Web 服务器实例旁边的复选框。
7. 在页面顶部选择操作下拉菜单。选择安全，然后选择更改安全组。
8. 对于关联安全组，请选择搜索框，然后选择您为 HTTPS 创建的安全组。然后选择添加安全组。
9. 选择保存。

验证 HTTPS 使用的是您已配置的证书

将 Web 服务器添加到安全组后，就可以验证 SSL/TLS 分载是否使用了自签名证书。您可以使用网络浏览器或使用工具 (如 OpenSSL s_client) 执行此操作。

使用网络浏览器验证 SSL/TLS 分载

1. 使用 Web 浏览器连接到采用服务器的公共 DNS 名称或 IP 地址的 Web 服务器。确保地址栏中的 URL 以 https:// 开头。例如，**`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**。

   > **Tip**
   >
   > 您可以使用 DNS 服务 (如 Amazon Route 53) 将网站域名 (例如，https://www.example.com/) 路由到您的 Web 服务器。有关更多信息，请参阅 《Amazon Route 53 开发人员指南》或 DNS 服务文档中的将流量路由到 Amazon EC2 实例。

2. 使用您的 Web 浏览器查看 Web 服务器证书。有关更多信息，请参阅下列内容：

- 对于 Mozilla Firefox，请参阅 Mozilla Support 网站上的查看证书。
- 关于 Google Chrome 浏览器，请参阅 Google Tools for Web Developers 网站上的了解安全问题。

其他网络浏览器可能具有相似的功能，可使用这些功能来查看网络服务器证书。

3. 确保 SSL/TLS 证书是您配置 Web 服务器使用的证书。

使用 OpenSSL s_client 验证 SSL/TLS 分载

1. 运行以下 OpenSSL 命令以通过 HTTPS 连接到您的 Web 服务器。将 *<server name>* 替换为您的 Web 服务器的公有 DNS 名称或 IP 地址。

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> 您可以使用 DNS 服务 (如 Amazon Route 53) 将网站域名 (例如，https://www.example.com/) 路由到您的 Web 服务器。有关更多信息，请参阅 《Amazon Route 53 开发人员指南》或 DNS 服务文档中的将流量路由到 Amazon EC2 实例。

2. 确保 SSL/TLS 证书是您配置 Web 服务器使用的证书。

您现在有一个通过 HTTPS 保护的网站。Web 服务器的私钥存储在 AWS CloudHSM 集群的 HSM 中。

若要添加负载均衡器，请参阅 使用 Elastic Load Balancing 添加负载均衡器（可选）。

在 Windows 上使用借助 CNG 的 IIS 进行 SSL/TLS 分载

本教程提供了在 Windows Web 服务器 AWS CloudHSM 上使用设置 SSL/TLS 卸载的 step-by-step 说明。

主题

- 步骤 3：配置 Web 服务器
- 步骤 4：启用 HTTPS 流量并验证证书

概述

在 Windows 上，Internet Information Services (IIS) for Windows Server Web 服务器应用程序本机支持 HTTPS。适用于 Microsoft 的“加密 API：下一代”(CNG) 的AWS CloudHSM 密钥存储提供程序 (KSP) 提供了一个接口，该接口允许 IIS 使用您的群集中的 HSM 进行加密分载和密钥存储。 AWS CloudHSM KSP 是将 IIS 连接到您的 AWS CloudHSM 集群的桥梁。

本教程介绍如何执行以下操作：

- 在 Amazon EC2 实例上安装 Web 服务器软件。
- 使用存储在您的 AWS CloudHSM 集群中的私有密钥将 Web 服务器软件配置为支持 HTTPS。
- （可选）使用 Amazon EC2 创建第二个 Web 服务器实例，并使用 Elastic Load Balancing 创建负载均衡器。使用负载均衡器可以在多台服务器中分配负载，从而提高性能。它还能在一台或多台服务器发生故障的情况下提供冗余和更高的可用性。

在您准备好开始使用后，请转到 步骤 1：设置先决条件。

步骤 1：设置先决条件

要使用设置 Web 服务器 SSL/TLS 卸载 AWS CloudHSM，您需要满足以下条件：

- 至少有一个 HSM 的活动 AWS CloudHSM 集群。
- 一个运行已安装以下软件的 Windows 操作系统的 Amazon EC2 实例：
  - 适用于 Windows 的 AWS CloudHSM 客户端软件。
  - Internet Information Services (IIS) for Windows Server。
- 一个加密用户（CU），该用户拥有和管理 HSM 上的 Web 服务器的私有密钥。

> **Note**
>
> 本教程使用的是 Microsoft Windows Server 2016。还支持 Microsoft Windows Server 2012，但不支持 Microsoft Windows Server 2012 R2。

在 HSM 上设置 Windows Server 实例并创建 CU

1. 完成开始使用中的步骤。启动 Amazon EC2 客户端时，选择 Windows Server 2016 或 Windows Server 2012 AMI。完成这些步骤后，您有一个至少包含一个 HSM 的活动集群。你还有一个运行 Windows 服务器的 Amazon EC2 AWS CloudHSM 客户端实例，其中安装了 Windows 客户端软件。
2. (可选) 向您的集群添加更多 HSM。有关更多信息，请参阅 添加 HSM。
3. 连接到您的 Windows Server。有关更多信息，请参阅 Amazon EC2 用户指南中的 “连接到您的实例”。
4. 使用 CloudHSM CLI 创建加密用户（CU）。跟踪 CU 用户名和密码。您需要这些信息才能完成下一步。

   > **Note**
   >
   > 有关创建用户的信息，请参阅使用 CloudHSM CLI 管理 HSM 用户。

5. 使用您在上一步中创建的 CU 用户名和密码设置 HSM 的登录凭证。
6. 在步骤 5 中，如果您使用 Windows 凭据管理器设置 HSM 凭据，请psexec.exe从下载并以 NT Auth SysInternals ority\ SYST EM 的身份运行以下命令：

   ```
   psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
    --username <USERNAME> --password <PASSWORD>
   ```

   将 *<USERNAME>* 和 *<PASSWORD>* 替换为 HSM 凭证。

在您的 Windows Server 上安装 IIS

1. 如果您尚未完成此操作，请连接到您的 Windows 服务器。有关更多信息，请参阅 Amazon EC2 用户指南中的 “连接到您的实例”。
2. 在 Windows 服务器上，启动服务器管理器。
3. 在服务器管理器控制面板中，选择添加角色和功能。
4. 阅读开始之前信息，然后选择下一步。
5. 对于安装类型，选择基于角色或基于功能的安装。然后选择下一步。
6. 对于服务器选择，选择从服务器池中选择服务器。然后选择下一步。
7. 对于服务器角色，请执行以下操作：

a. 选择 Web 服务器(IIS)。
b. 对于添加 Web Server (IIS) 所需的功能，选择添加功能。
c. 选择下一步完成选择服务器角色。

8. 对于功能，接受默认值。然后选择下一步。
9. 阅读 Web 服务器角色(IIS) 信息。然后选择下一步。
10. 对于选择角色服务，接受默认值或根据偏好更改设置。然后选择下一步。
11. 对于确认，阅读确认信息。然后选择安装。
12. 安装完成后，选择关闭。

完成这些步骤后，请转到 步骤 2：创建证书签名请求 (CSR) 和证书。

步骤 2：创建证书签名请求 (CSR) 和证书

要启用 HTTPS，您的 Web 服务器和 SSL/TLS 证书需要一个相应的私有密钥。要将 SSL/TLS 卸载与一起使用 AWS CloudHSM，您需要将私钥存储在集群的 HSM 中。 AWS CloudHSM 为此，您可使用适用于 Microsoft 的加密 API：下一代 (CNG) 的AWS CloudHSM 密钥存储提供程序 (KSP) 来创建证书签名请求 (CSR)。然后，您向证书颁发机构 (CA) 提供 CSR，该机构负责签署 CSR 以生成证书。

主题

- 创建 CSR
- 获取签名证书并导入

创建 CSR

使用 Windows 服务器上的 AWS CloudHSM KSP 创建 CSR。

创建 CSR

1. 如果您尚未完成此操作，请连接到您的 Windows 服务器。有关更多信息，请参阅 Amazon EC2 用户指南中的 “连接到您的实例”。
2. 使用以下命令启动 AWS CloudHSM 客户端守护程序。

   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

- 对于 Windows 客户端 1.1.2 以上版本：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 对于 Windows 客户端 1.1.1 及更低版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

3. 在您的 Windows Server 上，使用文本编辑器创建一个名为 IISCertRequest.inf 的证书请求文件。以下显示了示例 IISCertRequest.inf 文件的内容。有关可在文件中指定的各部分、键和值，请参阅 Microsoft 的文档。请勿更改 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 命令来从您在上一步中创建的 IISCertRequest.inf 文件创建 CSR。以下示例将 CSR 保存至名为 IISCertRequest.csr 的文件。如果您为证书请求文件使用了不同的文件名，请将 *IIS CertRequest .inf* 替换为相应的文件名。您可以选择将 *IIS CertRequest .cs* r 替换为 CSR 文件的不同文件名。

```
C:\>certreq -new IISCertRequest.inf IISCertRequest.csr
        SDK Version: 2.03

CertReq: Request Created
```

   IISCertRequest.csr 文件包含您的 CSR。您需要此 CSR 才能获取签名证书。

获取签名证书并导入

在生产环境中，您通常使用证书颁发机构 (CA) 通过 CSR 创建证书。测试环境无需 CA。如果您确实使用了一个 CA，请向它发送 CSR 文件 (IISCertRequest.csr) 并使用该 CA 创建已签名的 SSL/TLS 证书。

作为使用 CA 的替代方案，您可以使用 OpenSSL 等工具创建自签名证书。

> **Warning**
>
> 自签名证书不受浏览器的信任，不应在生产环境中使用。它们可在测试环境中使用。

以下过程演示了如何创建自签名证书并使用该证书来签署您的 Web 服务器的 CSR。

创建自签名证书

1. 使用以下 OpenSSL 命令创建私有密钥。您可以选择将 *SelfSignedCA.key* 替换为包含私钥的文件名。

```
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，通过您在上一步中创建的私有密钥创建自签名证书。这是一个交互式命令。阅读屏幕上的说明，并按照提示操作。将 *SelfSignedCA.key* 替换为包含您的私钥的文件名（如果不同）。您可以选择将 *SelfSignedCA.crt* 替换为包含自签名证书的文件名。

```
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 []:
```

使用您的自签名证书签署 Web 服务器的 CSR

- 使用以下 OpenSSL 命令，通过您的私有密钥和自签名证书签署 CSR。将以下文件替换为包含相应数据的文件的名称（如果不同）。

- *IIS CertRequest .csr-#### Web #### CSR* 的文件的名称
- *SelfSignedca.crt*-包含您的自签名证书的文件的名称
- *SelfSignedCA.key*-包含您的私钥的文件的名称
- *IISCert.crt* – 包含您的 Web 服务器签名证书的文件的名称

```
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:
```

在完成上一步后，您会有一个用于 Web 服务器的签名证书 (IISCert.crt) 和一个自签名证书 (SelfSignedCA.crt)。在具有这些文件后，请转到步骤 3：配置 Web 服务器。

步骤 3：配置 Web 服务器

将您的 IIS 网站的配置更新为使用在上一步结束时创建的 HTTPS 证书。这将完成 Windows Web 服务器软件 (IIS) 的设置，以便借助 AWS CloudHSM进行 SSL/TLS 分载。

如果您使用了自签名证书签署 CSR，则必须首先将自签名证书导入 Windows 受信任的根证书颁发机构。

将您的自签名证书导入 Windows 受信任的根证书颁发机构

1. 如果您尚未完成此操作，请连接到您的 Windows 服务器。有关更多信息，请参阅 Amazon EC2 用户指南中的 “连接到您的实例”。
2. 将您的自签名证书复制到 Windows Server。
3. 在 Windows Server 上，打开 Control Panel (控制面板)。
4. 对于搜索控制面板，键入 **certificates**。然后选择管理计算机证书。
5. 在证书 - 本地计算机窗口中，双击受信任的根证书颁发机构。
6. 右键单击证书，然后选择所有任务、导入。
7. 在证书导入向导 中，选择下一步。

8. 选择浏览，然后找到并选择您的自签名证书。如果按照本教程的上一步中的说明创建了您的自签名证书，则您的自签名证书将命名为 `SelfSignedCA.crt`。选择打开。
9. 选择下一步。
10. 对于证书存储，选择将所有证书置于以下存储中。然后，确保针对证书存储选中受信任的根证书颁发机构。
11. 选择下一步，然后选择完成。

更新 IIS 网站的配置

1. 如果您尚未完成此操作，请连接到您的 Windows 服务器。有关更多信息，请参阅 Amazon EC2 用户指南中的 “连接到您的实例”。
2. 启动 AWS CloudHSM 客户端守护程序。
3. 将您在本教程的上一步中创建的 Web 服务器的签名证书复制到您的 Windows 服务器。
4. 在您的 Windows Server 上，使用 Windows certreq 命令接受此签名证书，如以下示例所示。将 *IISCert.crt* 替换为包含您的 Web 服务器的签名证书的文件的名称。

```
C:\>certreq -accept IISCert.crt
        SDK Version: 2.03
```

5. 在 Windows 服务器上，启动服务器管理器。
6. 在服务器管理器控制面板的右上角，选择工具、Internet Information Services (IIS)管理器。
7. 在 Internet Information Services (IIS)管理器窗口中，双击您的服务器名称。然后双击网站。选择您的网站。
8. 选择 SSL 设置。然后，在窗口左侧选择绑定。
9. 在网站绑定 窗口中，选择添加。
10. 对于类型，选择 https。对于 SSL 证书，选择您在本教程的上一步结束时创建的 HTTPS 证书。

> **Note**
>
> 如果您在该绑定证书的过程中遇到错误，请重新启动服务器并重试此步骤。

11. 选择确定。

在更新您的网站配置后，请转到步骤 4：启用 HTTPS 流量并验证证书。

步骤 4：启用 HTTPS 流量并验证证书

使用 SSL/TLS 卸载配置 Web 服务器后 AWS CloudHSM，将您的 Web 服务器实例添加到允许入站 HTTPS 流量的安全组中。Web 浏览器等客户端可通过 Web 服务器创建 HTTPS 连接。然后与您的 Web 服务器建立 HTTPS 连接，并确认它使用的是您为 SSL/TLS 卸载配置的证书。 AWS CloudHSM

主题

启用入站 HTTPS 连接

要从客户端 (如 Web 浏览器) 连接到您的 Web 服务器，请创建一个允许入站 HTTPS 连接的安全组。具体来说，它应允许端口 443 上的入站 TCP 连接。将此安全组分配给您的网络服务器。

为 HTTPS 创建安全组并将其分配给您的网络服务器

1. 通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在导航窗格中，选择安全组。
3. 选择创建安全组。
4. 对于创建安全组，执行以下操作：
    a. 对于安全组名称，键入您要创建的安全组的名称。
    b. (可选) 键入对您要创建的安全组的描述。
    c. 对于 VPC，选择包含您的 Web 服务器 Amazon EC2 实例的 VPC。
    d. 选择添加规则。
    e. 对于类型，从下拉窗口中选择 HTTPS。
    f. 对于来源，输入来源位置。
    g. 选择创建安全组。
5. 在导航窗格中，选择实例。
6. 选中您的 Web 服务器实例旁边的复选框。
7. 在页面顶部选择操作下拉菜单。选择安全，然后选择更改安全组。
8. 对于关联安全组，请选择搜索框，然后选择您为 HTTPS 创建的安全组。然后选择添加安全组。
9. 选择保存。

验证 HTTPS 使用的是您已配置的证书

将 Web 服务器添加到安全组后，就可以验证 SSL/TLS 分载是否使用了自签名证书。您可以使用网络浏览器或使用工具 (如 OpenSSL s_client) 执行此操作。

使用网络浏览器验证 SSL/TLS 分载

1. 使用 Web 浏览器连接到采用服务器的公共 DNS 名称或 IP 地址的 Web 服务器。确保地址栏中的 URL 以 https:// 开头。例如，**`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**。

   > **Tip**
   >
   > 您可以使用 DNS 服务 (如 Amazon Route 53) 将网站域名 (例如，https://www.example.com/) 路由到您的 Web 服务器。有关更多信息，请参阅 《Amazon Route 53 开发人员指南》或 DNS 服务文档中的将流量路由到 Amazon EC2 实例。

2. 使用您的 Web 浏览器查看 Web 服务器证书。有关更多信息，请参阅下列内容：

   - 对于 Mozilla Firefox，请参阅 Mozilla Support 网站上的查看证书。
   - 关于 Google Chrome 浏览器，请参阅 Google Tools for Web Developers 网站上的了解安全问题。

   其他网络浏览器可能具有相似的功能，可使用这些功能来查看网络服务器证书。

3. 确保 SSL/TLS 证书是您配置 Web 服务器使用的证书。

使用 OpenSSL s_client 验证 SSL/TLS 分载

1. 运行以下 OpenSSL 命令以通过 HTTPS 连接到您的 Web 服务器。将 *`<server name>`* 替换为您的 Web 服务器的公有 DNS 名称或 IP 地址。

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> 您可以使用 DNS 服务 (如 Amazon Route 53) 将网站域名 (例如，https://www.example.com/) 路由到您的 Web 服务器。有关更多信息，请参阅 《Amazon Route 53 开发人员指南》或 DNS 服务文档中的将流量路由到 Amazon EC2 实例。

2. 确保 SSL/TLS 证书是您配置 Web 服务器使用的证书。

您现在有一个通过 HTTPS 保护的网站。Web 服务器的私钥存储在 AWS CloudHSM 集群的 HSM 中。

若要添加负载均衡器，请参阅 使用 Elastic Load Balancing 添加负载均衡器（可选）。

使用 Elastic Load Balancing 添加负载均衡器（可选）

对一个 web 服务器设置 SSL/TLS 分载后，您可以创建更多 Web 服务器和一个将 HTTPS 流量路由到这些 web 服务器的 Elastic Load Balancing 负载均衡器。负载均衡器可通过使两台或更多 web 服务器上的流量达到均衡，来减少您的单个 web 服务器上的负载。它还可以提高您的网站的可用性，因为负载均衡器可监视您的 Web 服务器的运行状况并且仅将流量路由到运行状况良好的服务器。如果某个 Web 服务器出现故障，负载均衡器会自动停止将流量路由到该服务器。

主题

- 为第二个 Web 服务器创建子网
- 创建第二个 Web 服务器
- 创建负载均衡器

为第二个 Web 服务器创建子网

在创建另一个 Web 服务器之前，您需要在包含现有 Web 服务器和 AWS CloudHSM 集群的同一 VPC 中创建一个新子网。

创建新子网

1. 打开 Amazon VPC 控制台的子网部分。
2. 选择创建子网。
3. 在创建子网对话框中，执行以下操作：

    a. 对于名称标签，键入您的子网的名称。

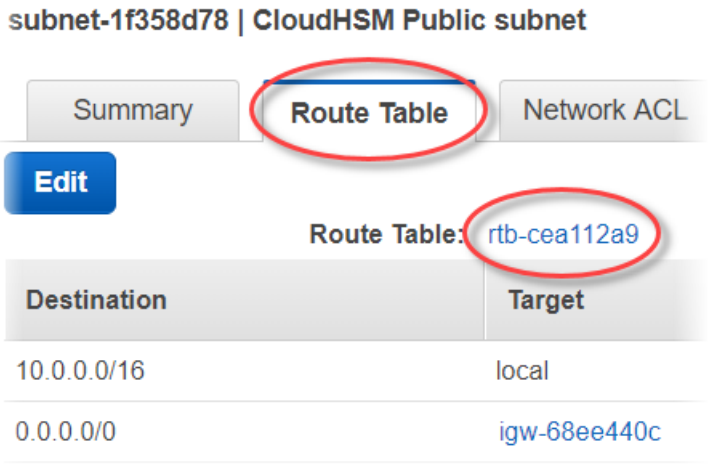
b. 对于 VPC，请选择包含您的现有 Web 服务器和 AWS CloudHSM 集群的 AWS CloudHSM VPC。

c. 对于可用区，选择与包含现有 Web 服务器的可用区不同的可用区。

d. 对于 IPv4 CIDR block，键入要用于子网的 CIDR 块。例如，键入 **`10.0.10.0/24`**。

e. 选择是，创建。

4. 选中包含现有 Web 服务器的公有子网旁边的复选框。这与您在上一步中创建的公有子网不同。
5. 在内容窗格中，选择路由表选项卡。然后选择路由表的链接。



6. 选中路由表旁边的复选框。
7. 选择子网关联选项卡。然后选择编辑。
8. 选中您在此过程的前面部分创建的公有子网旁的复选框。然后选择保存。

创建第二个 Web 服务器

完成以下步骤以创建与现有 Web 服务器配置相同的另一个 Web 服务器。

创建另一个 Web 服务器

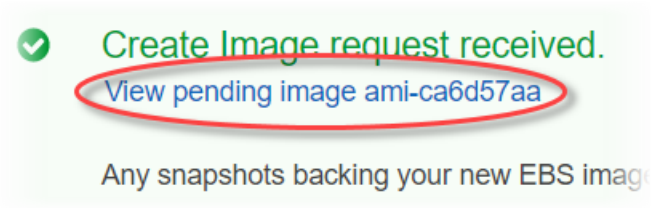
1. 打开 Amazon EC2 控制台的实例章节，其位于。
2. 选中现有 Web 服务器实例旁边的复选框。
3. 依次选择操作、映像和创建映像。
4. 在创建映像 对话框中，执行以下操作：

a. 对于映像名称 ，请键入映像的唯一名称。

b. 对于映像描述 ，请键入映像的描述。

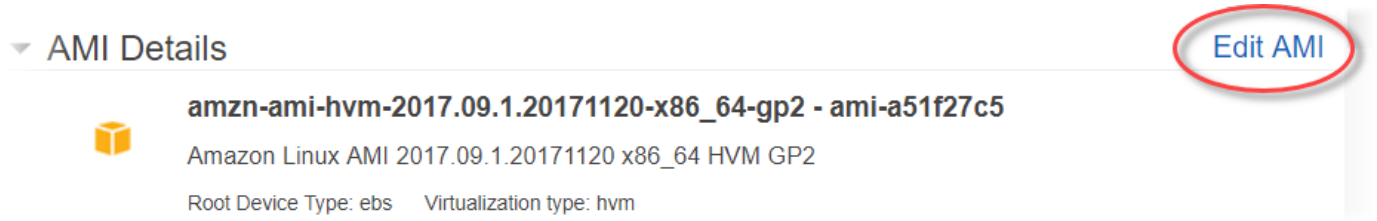
c. 选择创建映像。此操作将重启现有 Web 服务器。

d. 选择查看待处理的影像 ami-*<AMI ID>* 链接。



在状态列中，记下您的影像状态。当您的映像状态为可用时 (这可能需要几分钟)，请转到下一步。

5. 在导航窗格中，选择实例。
6. 选中现有 Web 服务器旁边的复选框。
7. 选择操作 ，然后选择启动更多类似项。
8. 选择编辑 AMI。



9. 在左侧导航窗格中，选择我的 AMI。然后，清除搜索框中的文本。
10. 在您的 Web 服务器映像旁，选择选择。
11. 选择是的，我希望继续使用此 AMI (*<image name>* - ami-*<AMI ID>*)。
12. 选择下一步。
13. 选择实例类型，然后选择下一步：配置实例详细信息。
14. 对于步骤 3：配置实例详细信息，执行以下操作：

    a. 对于网络，选择包含现有 Web 服务器的 VPC。

    b. 对于子网，选择您为第二个 Web 服务器创建的公有子网。

    c. 对于自动分配公有 IP，选择启用。

    d. 根据偏好更改其余实例的详细信息。然后选择下一步：添加存储。

15. 根据偏好更改存储设置。然后，选择下一步：添加标签。
16. 根据偏好添加或编辑标签。然后选择下一步：配置安全组。
17. 对于步骤 6：配置安全组，执行以下操作：

a. 对于分配安全组，选择选择现有安全组。

b. 选中名为 cloudhsm-*<cluster ID>*-sg 的安全组旁边的复选框。 AWS CloudHSM 在您创建集群时代表您创建了此安全组。您必须选择此安全组以允许 Web 服务器实例连接到集群中的 HSM。

c. 选中允许入站 HTTPS 流量的安全组旁边的复选框。您之前创建了此安全组。

d. (可选) 选中允许来自网络的入站 SSH（适用 Linux）或 RDP（适用 Windows）流量的安全组旁边的复选框。也就是说，安全组必须允许端口 22（Linux 上的 SSH）或端口 3389（Windows 上的 RDP）上的入站 TCP 流量。否则，您将无法连接到客户端实例。如果您没有此类安全组，则必须创建一个安全组并稍后将它分配给客户端实例。

选择审核并启动。

18. 查看实例详细信息，然后选择 启动。

19. 选择是使用现有密钥对启动您的实例、创建新的密钥对还是在没有密钥对的情况下启动您的实例。

- 要使用现有密钥对，请执行以下操作：
  1. 选择选择现有密钥对。
  2. 对于选择密钥对，选择要使用的密钥对。
  3. 选中我认可我可以访问所选的私有密钥文件 (*<private key file name>*.pem)，如果没有该文件，我将无法登录我的实例。旁边的复选框。
- 要创建新密钥对，请执行以下操作：
  1. 选择 Create a new key pair。
  2. 对于密钥对名称 ，请键入一个密钥对名称。
  3. 选择下载密钥对并将私有密钥文件保存在一个安全且易于访问的位置。

  > **Warning**
  >
  > 此后您无法再次下载私有密钥文件。如果您现在不下载私有密钥文件，您将无法访问客户端实例。

- 要在没有密钥对的情况下启动您的实例，请执行以下操作：
  1. 选择继续操作但不提供密钥对。
  2. 选中 I acknowledge that I will not be able to connect to this instance unless I already know the password built into this AMI 旁边的复选框。

选择启动新实例。

创建负载均衡器

完成以下步骤以创建将 HTTPS 流量路由到您的 Web 服务器的 Elastic Load Balancing 负载均衡器。

创建负载均衡器

1. 打开 Amazon EC2 控制台的负载均衡器章节。
2. 选择创建负载均衡器。
3. 在网络负载均衡器部分中，选择创建。
4. 对于步骤 1：配置负载均衡器，执行以下操作：
   a. 对于名称，键入您创建的负载均衡器的名称。
   b. 在侦听器章节中，对于负载均衡器端口，将值更改为 **443**。
   c. 在可用区部分中，对于 VPC，选择包含您的 Web 服务器的 VPC。
   d. 在可用区部分中，选择包含您的 Web 服务器的子网。
   e. 选择下一步：配置路由。
5. 对于步骤 2：配置路由，执行以下操作：
   a. 对于名称，键入您创建的目标组的名称。
   b. 对于端口，将值更改为 **443**。
   c. 选择下一步：注册目标。
6. 对于步骤 3：注册目标，请执行以下操作：
   a. 在实例部分，选中您的 web 服务器实例旁边的复选框。然后选择添加到已注册。
   b. 选择下一步：审核。
7. 审核您的负载均衡器详细信息，然后选择创建。
8. 在成功创建负载均衡器后，选择关闭。

完成前面的步骤后，Amazon EC2 控制台将显示您的 Elastic Load Balancing 负载均衡器。

当负载均衡器处于活动状态时，您可以验证负载均衡器是否正常工作。也就是说，您可以验证它是否正在使用 AWS CloudHSM将 HTTPS 流量发送到带 SSL/TLS 卸载的 Web 服务器。您可以使用 Web 浏览器或使用工具（如 OpenSSL s_client）执行此操作。

验证您的负载均衡器是否正在与 Web 浏览器结合使用

1. 在 Amazon EC2 控制台中，找到您刚刚创建的负载均衡器的 DNS name。然后选择 DNS 名称并复制该名称。
2. 使用 Web 浏览器 (例如 Mozilla Firefox 或 Google Chrome) 通过负载均衡器的 DNS 名称连接到负载均衡器。确保地址栏中的 URL 以 https:// 开头。

   > **Tip**
   >
   > 您可以使用 DNS 服务 (如 Amazon Route 53) 将网站域名 (例如，https://www.example.com/) 路由到您的 Web 服务器。有关更多信息，请参阅 《Amazon Route 53 开发人员指南》或 DNS 服务文档中的将流量路由到 Amazon EC2 实例。

3. 使用您的 Web 浏览器查看 Web 服务器证书。有关更多信息，请参阅下列内容：

   - 对于 Mozilla Firefox，请参阅 Mozilla Support 网站上的查看证书。
   - 关于 Google Chrome 浏览器，请参阅 Google Tools for Web Developers 网站上的了解安全问题。

   其他网络浏览器可能具有相似的功能，可使用这些功能来查看网络服务器证书。
4. 确保该证书是您配置 web 服务器使用的证书。

验证您的负载均衡器是否正在与 OpenSSL s_client 结合使用

1. 使用以下 OpenSSL 命令通过 HTTPS 连接到您的负载均衡器。将 *<DNS name>* 替换为负载均衡器的 DNS 名称。

```
openssl s_client -connect <DNS name>:443
```

> **Tip**
>
> 您可以使用 DNS 服务 (如 Amazon Route 53) 将网站域名 (例如，https://www.example.com/) 路由到您的 Web 服务器。有关更多信息，请参阅 《Amazon Route 53 开发人员指南》或 DNS 服务文档中的将流量路由到 Amazon EC2 实例。

2. 确保该证书是您配置 web 服务器使用的证书。

您现在有一个使用 HTTPS 保护的网站，Web 服务器的私钥存储在 AWS CloudHSM 集群的 HSM 中。您的网站有两个 Web 服务器和一个负载均衡器，可帮助提高效率和可用性。

将 Windows Server 配置为具有 AWS CloudHSM的证书颁发机构 (CA)

在公有密钥基础设施 (PKI) 中，证书颁发机构 (CA) 是一个颁发数字证书的可信实体。这些数字证书通过公有密钥加密和数字签名的方式，将公有密钥与身份（用户或组织）绑定。要操作 CA，您必须通过保护签署由 CA 颁发的证书的私有密钥来维持信任。您可以将此私有密钥存储至 AWS CloudHSM 集群的 HSM，然后使用 HSM 执行加密签名操作。

在本教程中，您将使用 Windows 服务器和 AWS CloudHSM 来配置 CA。在 Windows Server 上安装适用于 Windows 的 AWS CloudHSM 客户端软件，然后将 Active Directory 证书服务 (AD CS) 角色添加到 Windows Server。配置此角色时，您可以使用 AWS CloudHSM 密钥存储提供程序 (KSP) 在 AWS CloudHSM 集群上创建和存储 CA 的私钥。KSP 是将 Windows 服务器连接到 AWS CloudHSM 集群的桥梁。在最后一个步骤中，您使用 Windows Server CA 签署证书签名请求 (CSR)。

有关更多信息，请参阅以下主题：

主题

- Windows Server CA 步骤 1：设置先决条件
- Windows Server CA 步骤 2：创建使用 AWS CloudHSM 的 Windows Server CA
- Windows Server CA 第 3 步：使用以下命令与你的 Windows 服务器 CA 签署证书签名请求 (CSR) AWS CloudHSM

Windows Server CA 步骤 1：设置先决条件

要将 Windows 服务器设置为证书颁发机构 (CA) AWS CloudHSM，你需要满足以下条件：

- 至少有一个 HSM 的活动 AWS CloudHSM 集群。
- 运行 Windows 服务器操作系统且安装了 Windows AWS CloudHSM 客户端软件的 Amazon EC2 实例。本教程使用的是 Microsoft Windows Server 2016。
- 一个加密用户 (CU)，该用户拥有和管理 HSM 上的 CA 私有密钥。

使用以下命令设置 Windows 服务器 CA 的先决条件 AWS CloudHSM

1. 完成开始使用中的步骤。启动 Amazon EC2 客户端时，选择 Windows Server AMI。本教程使用的是 Microsoft Windows Server 2016。完成这些步骤后，您有一个至少包含一个 HSM 的活动集群。你还有一个运行 Windows 服务器的 Amazon EC2 AWS CloudHSM 客户端实例，其中安装了 Windows 客户端软件。
2. (可选) 向您的集群添加更多 HSM。有关更多信息，请参阅 添加 HSM。
3. 连接到您的客户端实例。有关更多信息，请参阅 Amazon EC2 用户指南中的 “连接到您的实例”。
4. 通过使用 CloudHSM CLI 管理 HSM 用户或使用 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility) 管理 HSM 用户来创建加密用户（CU）跟踪 CU 用户名和密码。您需要这些信息才能完成下一步。
5. 使用您在上一步中创建的 CU 用户名和密码设置 HSM 的登录凭证。
6. 在步骤 5 中，如果您使用 Windows 凭据管理器设置 HSM 凭据，请psexec.exe从下载并以 NT Auth SysInternals ority\ SYST EM 的身份运行以下命令：

   ```
   psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
    --username <USERNAME> --password <PASSWORD>
   ```

   将 *<USERNAME>* 和 *<PASSWORD>* 替换为 HSM 凭证。

要使用创建 Windows 服务器 CA AWS CloudHSM，请转至创建 Windows Server CA。

Windows Server CA 步骤 2：创建使用 AWS CloudHSM 的 Windows Server CA

要创建 Windows Server CA，您将向 Windows Server 添加 Active Directory 证书服务 (AD CS) 角色。添加此角色时，您可以使用 AWS CloudHSM 密钥存储提供程序 (KSP) 在 AWS CloudHSM 集群上创建和存储 CA 的私钥。

> **Note**
>
> 创建 Windows Server CA 时，您可以选择创建根 CA 或从属 CA。您通常根据组织的公有密钥基础设施和安全策略的设计制定此决策。本教程介绍了如何创建根 CA 以简化操作。

向 Windows Server 添加 AD CS 角色和创建 CA 的私有密钥

1. 如果您尚未完成此操作，请连接到您的 Windows 服务器。有关更多信息，请参阅 Amazon EC2 用户指南中的 “连接到您的实例”。
2. 在 Windows 服务器上，启动服务器管理器。
3. 在服务器管理器控制面板中，选择添加角色和功能。
4. 阅读开始之前信息，然后选择下一步。
5. 对于安装类型，选择基于角色或基于功能的安装。然后选择下一步。
6. 对于服务器选择，选择从服务器池中选择服务器。然后选择下一步。
7. 对于服务器角色，请执行以下操作：
   a. 选择 Active Directory 证书服务。
   b. 对于添加 Active Directory 证书服务所需的功能，选择添加功能。
   c. 选择下一步完成选择服务器角色。
8. 对于功能，接受默认值，然后选择下一步。
9. 对于 AD CS，请执行以下操作：
   a. 选择下一步。
   b. 选择证书颁发机构，然后选择下一步。
10. 对于确认，阅读确认信息，然后选择安装。不要关闭该窗口。
11. 选择突出显示的配置目标服务器上的 Active Directory 证书服务链接。
12. 对于凭据，请验证或更改显示的凭证。然后选择下一步。
13. 对于角色服务，选择证书颁发机构。然后选择下一步。
14. 对于设置类型，选择独立 CA。然后选择下一步。
15. 对于 CA 类型，选择根 CA。然后选择下一步。

> **Note**
>
> 您可以选择根据公有密钥基础设施和您组织的安全策略的设计创建根 CA 或从属 CA。本教程介绍了如何创建根 CA 以简化操作。

16. 对于私有密钥，选择新建私有密钥。然后选择下一步。
17. 对于加密，请执行以下操作：

    a. 对于选择加密提供程序，从菜单中选择一个 Cavium 密钥存储提供程序。这些是 AWS CloudHSM 密钥存储提供程序。例如，您可以选择 RSA # Cavium 密钥存储提供程序。
    b. 对于密钥长度，从中选择一个密钥长度选项。
    c. 对于选择用于签署此 CA 颁发的证书的哈希算法，请选择其中一个哈希算法选项。

    选择下一步。
18. 对于 CA 名称，请执行以下操作：

    a. （可选）编辑公用名。
    b. （可选）键入可分辨名称后缀。

    选择下一步。
19. 对于有效期，指定时间段（以年、月、周或天为单位）。然后选择下一步。
20. 对于证书数据库，您可以接受默认值，也可以选择更改数据库和数据库日志的位置。然后选择下一步。
21. 对于确认，请查看有关 CA 的信息；然后选择配置。
22. 选择关闭，然后再次选择关闭。

你现在有了带的 Windows 服务器 CA AWS CloudHSM。要了解如何使用 CA 签署证书签名请求 (CSR) 的信息，请转到签署 CSR。

Windows Server CA 第 3 步：使用以下命令与你的 Windows 服务器 CA 签署证书签名请求 (CSR) AWS CloudHSM

你可以使用你的 Windows 服务器 CA AWS CloudHSM 来签署证书签名请求 (CSR)。要完成这些步骤，您需要一个有效的 CSR。可以通过以下方法来创建 CSR：

- 使用 OpenSSL
- 使用 Windows Server Internet Information Services (IIS) 管理器
- 使用 Microsoft 管理控制台中的证书单元
- 在 Windows 上使用 certreq 命令行实用程序

创建 CSR 的步骤不在本教程的介绍范围之内。如果您有 CSR，则可使用您的 Windows Server CA 登录。

使用 Windows Server CA 登录 CSR

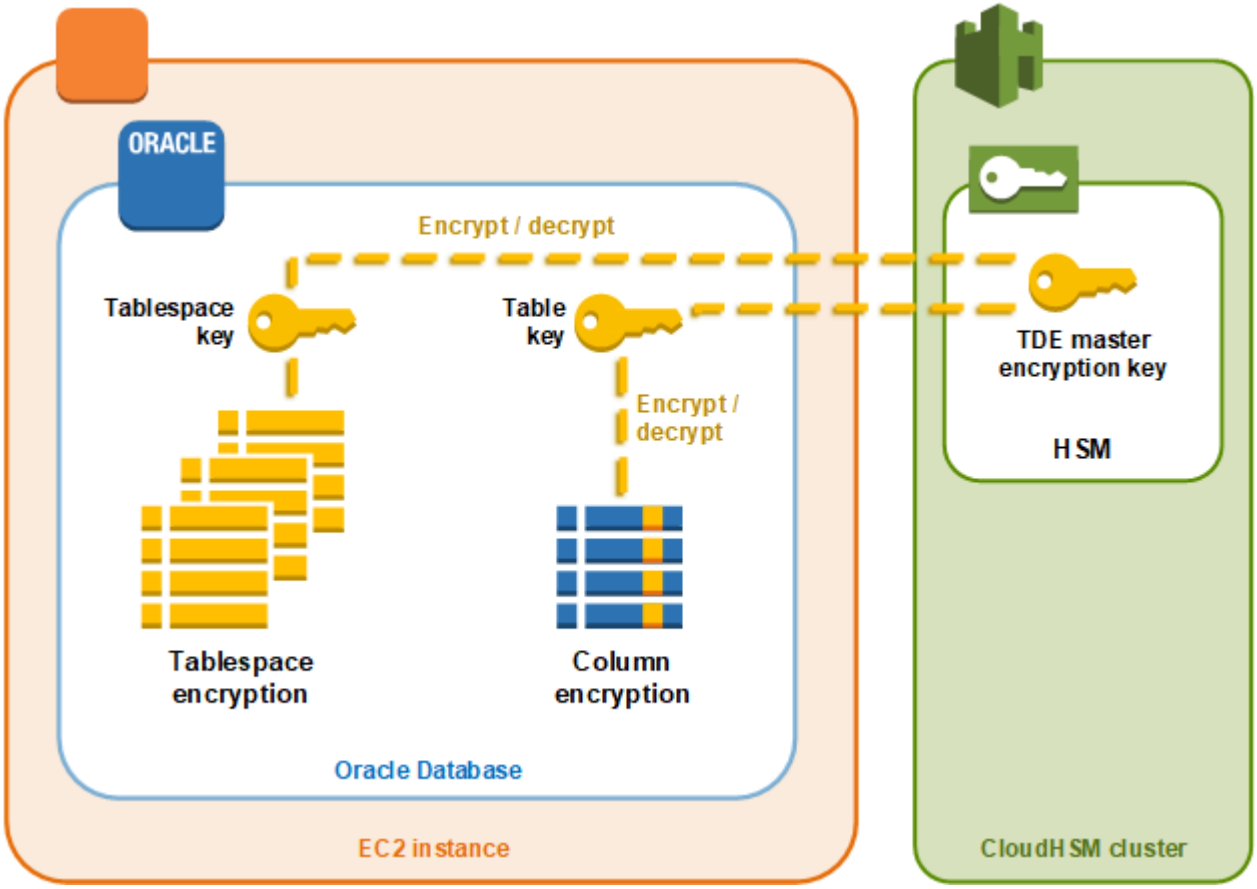
1. 如果您尚未完成此操作，请连接到您的 Windows 服务器。有关更多信息，请参阅 Amazon EC2 用户指南中的 “连接到您的实例”。
2. 在 Windows 服务器上，启动服务器管理器。
3. 在服务器管理器控制面板的右上角，选择工具、证书颁发机构。
4. 在证书颁发机构窗口中，选择您的计算机名称。
5. 在操作菜单上，选择所有任务、提交新请求。
6. 选择 CSR 文件，然后选择打开。
7. 在证书颁发机构窗口中，双击 待定的请求。
8. 选择待定的请求。然后在操作菜单上，选择所有任务、颁发。
9. 在证书颁发机构窗口中，双击 已发出的请求以查看签署的证书。
10. （可选）要将签署的证书导出到文件中，请完成以下步骤：
    a. 在证书颁发机构窗口中，双击证书。
    b. 选择详细信息选项卡，然后选择复制到文件。
    c. 按照证书导出向导的说明操作。

现在，你有一个 Windows 服务器 CA 和一个由 Windows Server CA 签署的有效证书。 AWS CloudHSM

使用 AWS CloudHSM的 Oracle Database 透明数据加密 (TDE)

透明数据加密（TDE）用于数据库文件加密。使用 TDE，数据库软件可以先对数据进行加密，然后再将其存储在磁盘上。数据库的表列或表空间中的数据通过表密钥或表空间密钥进行加密。Oracle 数据

库软件的某些版本提供 TDE。在 Oracle TDE 中，这些密钥通过 TDE 主加密密钥进行加密。通过将 TDE 主加密密钥存储在 AWS CloudHSM 集群的 HSM 中，可以提高安全性。



在此解决方案中，您将使用安装在 Amazon EC2 实例上的 Oracle Database。Oracle Database 将与适用于 PKCS #11 的AWS CloudHSM 软件库集成以将 TDE 主密钥存储在群集的 HSM 中。

> **Important**
>
> - 我们建议在 Amazon EC2 实例上安装 Oracle Database。

完成以下步骤以完成 Oracle TDE 与 AWS CloudHSM的集成。

配置 Oracle TDE 集成 AWS CloudHSM

1. 按照设置先决条件中的步骤准备您的环境。
2. 按照中的配置数据库步骤配置 Oracle 数据库以与您的 AWS CloudHSM 集群集成。

Oracle TDE 使用 AWS CloudHSM：设置先决条件

要完成 Oracle TDE 与的集成 AWS CloudHSM，您需要满足以下条件：

- 至少有一个 HSM 的活动 AWS CloudHSM 集群。
- 运行 Amazon Linux 操作系统并安装了以下软件的 Amazon EC2 实例：
  - AWS CloudHSM 客户端和命令行工具。
  - PKCS #11 的 AWS CloudHSM 软件库。
  - 甲骨文数据库。 AWS CloudHSM 支持 Oracle TDE 集成。客户端软件开发工具包 5.6 及更高版本支持适用于 Oracle Database 19c 的 Oracle TDE。客户端软件开发工具包 3 支持适用于 Oracle Database 11g 和 12c 的 Oracle TDE。
- 一个加密用户 (CU)，该用户拥有和管理集群中的 HSM 上的 TDE 主加密密钥。

完成以下步骤可设置所有先决组件。

设置与 Oracle TDE 集成的先决条件 AWS CloudHSM

1. 完成开始使用中的步骤。完成后，您将拥有一个带一个 HSM 的活动集群。您还将拥有一个运行 Amazon Linux 操作系统的 Amazon EC2 实例。还将安装和配置 AWS CloudHSM 客户端和命令行工具。
2. (可选) 向您的集群添加更多 HSM。有关更多信息，请参阅 添加 HSM。
3. 连接到您的 Amazon C2 客户端实例并执行以下操作：
   a. 安装 PKCS #11 的 AWS CloudHSM 软件库。
   b. 安装 Oracle Database。有关更多信息，请参阅 Oracle Database 文档。客户端软件开发工具包 5.6 及更高版本支持适用于 Oracle Database 19c 的 Oracle TDE。客户端软件开发工具包 3 支持适用于 Oracle Database 11g 和 12c 的 Oracle TDE。
   c. 使用 cloudhsm_mgmt_util 命令行工具在您的集群上创建加密用户 (CU)。有关创建 CU 的更多信息，请参阅如何使用 CMU 管理 HSM 用户和管理 HSM 用户。

完成这些步骤后，您可以配置数据库。

使用 Oracle TDE AWS CloudHSM：配置数据库并生成主加密密钥

要将 Oracle TDE 与您的 AWS CloudHSM 集群集成，请参阅以下主题：

1. 更新 Oracle Database 配置以使用您的群集中的 HSM 作为外部安全模块。有关外部安全模块的信息，请参阅《Oracle Database 高级安全指南》中的透明数据加密简介。
2. 在您的集群中的 HSM 上生成 Oracle TDE 主加密密钥。

更新 Oracle Database 配置

要将 Oracle Database 配置更新为使用您的集群中的 HSM 作为外部安全模块，请完成以下步骤。有关外部安全模块的信息，请参阅《Oracle Database 高级安全指南》中的透明数据加密简介。

更新 Oracle 配置

1. 连接到您的 Amazon EC2 客户端实例。这是在其中安装 Oracle Database 的实例。
2. 为名为 `sqlnet.ora` 的文件创建备份副本。有关此文件的位置，请参阅 Oracle 文档。
3. 使用文本编辑器编辑名为 `sqlnet.ora` 的文件。添加以下行。如果文件中的现有行以 `encryption_wallet_location` 开头，请将现有行替换为以下行。

   ```
   encryption_wallet_location=(source=(method=hsm))
   ```

   保存该文件。
4. 运行以下命令创建 Oracle 数据库期望在其中查找 AWS CloudHSM PKCS #11 软件库库的库文件的目录。

   ```
   sudo mkdir -p /opt/oracle/extapi/64/hsm
   ```
5. 运行以下命令将 PKCS #11 文件的 AWS CloudHSM 软件库复制到您在上一步中创建的目录中。

   ```
   sudo cp /opt/cloudhsm/lib/libcloudhsm_pkcs11.so /opt/oracle/extapi/64/hsm/
   ```

   > **Note**
   >
   > `/opt/oracle/extapi/64/hsm` 目录只能包含一个库文件。删除该目录中存在的所有其他文件。
6. 运行以下命令更改 `/opt/oracle` 目录的所有权及其包含的所有内容。

   ```
   sudo chown -R oracle:dba /opt/oracle
   ```

7. 启动 Oracle Database。

生成 Oracle TDE 主加密密钥

要在您的集群中的 HSM 上生成 Oracle TDE 主密钥，请完成以下过程中的步骤。

生成主密钥

1. 使用以下命令打开 Oracle SQL*Plus。在系统提示时，键入安装 Oracle Database 时设置的系统密码。

```
sqlplus / as sysdba
```

> **Note**
>
> 使用客户端软件开发工具包 3 生成主密钥时，每次都必须设置 CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE 环境变量。仅在生成主密钥时需要此变量。有关更多信息，请参阅“问题：Oracle 在主密钥生成过程中设置了 PKCS #11 属性CKA_MODIFIABLE”，但在有关集成第三方应用程序的已知问题中 HSM 并不支持该属性。

2. 运行 SQL 语句来创建主加密密钥，如以下示例所示。使用与您的 Oracle Database 的版本对应的语句。将 *<CU user name>* 替换为加密用户 (CU) 的用户名。将 *<password>* 替换为 CU 密码。

> **Important**
>
> 仅运行以下命令一次。每次运行此命令时，它都会创建一个新的主加密密钥。

- 对于 Oracle Database 版本 11，运行以下 SQL 语句。

```
SQL> alter system set encryption key identified by "<CU user name>:<password>";
```

- 对于 Oracle Database 版本 12 和版本 19c，运行以下 SQL 语句。

```
SQL> administer key management set key identified by "<CU user
 name>:<password>";
```

如果响应是 `System altered` 或 `keystore altered`，则您已成功生成并设置 Oracle TDE 的主密钥。

3. (可选) 运行以下命令来验证 Oracle wallet 的状态。

```
SQL> select * from v$encryption_wallet;
```

如果 wallet 未打开，请使用下列命令之一将其打开。将 *<CU user name>* 替换为加密用户 (CU) 的名称。将 *<password>* 替换为 CU 密码。

- 对于 Oracle 11，运行以下命令以打开 wallet。

```
SQL> alter system set encryption wallet open identified by "<CU user
 name>:<password>";
```

要手动关闭 wallet，请运行以下命令。

```
SQL> alter system set encryption wallet close identified by "<CU user
 name>:<password>";
```

- 对于 Oracle 12 和 Oracle 19c，运行以下命令以打开 wallet。

```
SQL> administer key management set keystore open identified by "<CU user
 name>:<password>";
```

要手动关闭 wallet，请运行以下命令。

```
SQL> administer key management set keystore close identified by "<CU user
 name>:<password>";
```

使用 Microsof AWS CloudHSM t SignTool 与签名文件

在加密和公有密钥基础结构 (PKI) 中，数字签名用于确认数据已由可信实体发送。签名还表明数据在传输过程中未被篡改。签名是使用发送方的私有密钥生成的加密哈希。接收方可以通过使用发送方的公有密钥解密其哈希签名来验证数据的完整性。反过来，发送方有责任维护数字证书。数字证书显示发送方对私有密钥的所有权，并向接收方提供解密所需的公有密钥。只要私钥归发件人所有，签名就可以被信任。 AWS CloudHSM 提供安全的 FIPS 140-2 3 级验证硬件，让您可以通过独家单租户访问来保护这些密钥。

许多组织使用 Microsoft SignTool（一种命令行工具）来签名、验证和给文件加盖时间戳来简化代码签名过程。您可以使用安全 AWS CloudHSM 地存储密钥对，直到需要它们为止 SignTool，从而创建易于自动化的签名数据工作流程。

以下主题概述了如何 SignTool 与配合使用 AWS CloudHSM：

主题

Micro SignTool soft AWS CloudHSM 第 1 步：设置先决条件

要将 Microsoft SignTool 与配合使用 AWS CloudHSM，你需要满足以下条件：

- 一个运行 Windows 操作系统的 Amazon EC2 客户端实例。
- 一个证书颁发机构 (CA)，自我维护或由第三方提供商建立。
- 与您的 EC2 AWS CloudHSM 实例位于同一个虚拟公有云 (VPC) 中的活动集群。该集群必须包含至少一个 HSM。
- 在 AWS CloudHSM 集群中拥有和管理密钥的加密用户 (CU)。
- 未签名文件或可执行文件。
- Microsoft Windows 软件开发工具包 (SDK)。

设置在 Windows 中使用的 AWS CloudHSM 先决条件 SignTool

1. 按照本指南的入门部分中的说明进行操作，启用 Windows EC2 实例和 AWS CloudHSM 集群。
2. 如果你想托管自己的 Windows Server CA，请按照将 Windows 服务器配置为证书颁发机构中的步骤 1 和 2 进行操作 AWS CloudHSM。否则，请继续使用您的公开可信的第三方 CA。
3. 在 Windows EC2 实例上下载并安装下列版本的 Microsoft Windows 开发工具包之一：
   - Microsoft Windows 开发工具包 10
   - Microsoft Windows 开发工具包 8.1
   - Microsoft Windows 开发工具包 7

SignTool 可执行文件是 Windows SDK Signing Tools for Desktop Apps 安装功能的一部分。您可以省略要安装的其他功能（如果您不需要它们）。默认安装位置是：

```
C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\<CPU
 architecture>\signtool.exe
```

现在，你可以使用 Microsoft Windows SDK、你的 AWS CloudHSM 集群和你的 CA 来创建签名证书。

微软 SignTool 的 AWS CloudHSM 第 2 步：创建签名证书

现在您已将 Windows 开发工具包下载到 EC2 实例，您可以使用它来生成证书签名请求 (CSR)。CSR 是未签名的证书，该证书最终将传递给您的 CA 进行签名。在此示例中，我们使用 Windows 开发工具包中包含的 certreq 可执行文件来生成 CSR。

使用 **certreq** 可执行文件生成 CSR

1. 如果您尚未完成此操作，请连接到您的 Windows EC2 实例。有关更多信息，请参阅 Amazon EC2 用户指南中的 “连接到您的实例”。
2. 创建一个名为 request.inf 的文件，其中包含下列行。将 Subject 信息替换为该组织的相应信息。有关每个参数的说明，请参阅 Microsoft 的文档。

```
[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. 运行 certreq.exe。在此示例中，我们将 CSR 保存为 request.csr。

```
certreq.exe -new request.inf request.csr
```

在内部，将在您的 AWS CloudHSM 集群上生成一个新的密钥对，该密钥对的私钥用于创建 CSR。

4. 将 CSR 提交给 CA。如果您使用的是 Windows Server CA，请执行以下步骤：

   a. 输入以下命令以打开 CA 工具：

   ```
   certsrv.msc
   ```

   b. 在新窗口中，右键单击 CA 服务器的名称。选择所有任务，然后选择提交新请求。

   c. 导航到 `request.csr` 的位置，然后选择打开。

   d. 展开服务器 CA 菜单，导航至待处理请求文件夹。右键单击您刚刚创建的请求，并在所有任务下，选择 Issue (发布)。

   e. 现在，导航到颁发的证书文件夹（位于待处理请求文件夹上）。

   f. 选择打开以查看证书，然后选择详细信息选项卡。

   g. 选择复制到文件以启动证书导出向导。将经 DER 编码的 X.509 文件作为 `signedCertificate.cer` 保存到安全位置。

   h. 退出 CA 工具并使用以下命令，该命令将证书文件移动到 Windows 中的个人证书存储。然后，它可由其他应用程序使用。

   ```
   certreq.exe -accept signedCertificate.cer
   ```

现在，您可以使用导入的证书来签署文件。

微 SignTool 软第三 AWS CloudHSM 步：签署文件

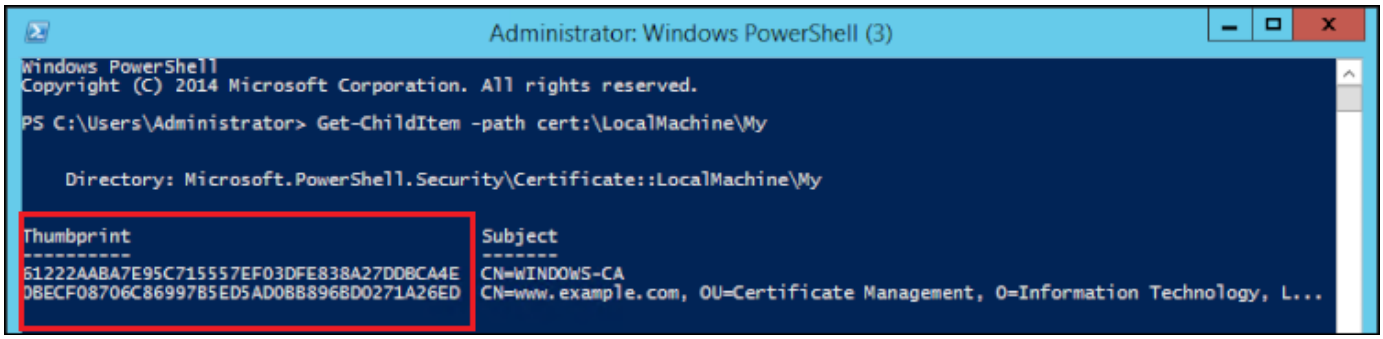
现在，您可以使用导入 SignTool 的证书来签署示例文件。为此，您需要知道证书的 SHA-1 哈希或指纹。指纹用于确保 SignTool仅使用经过验证的 AWS CloudHSM证书。在此示例中，我们使用 PowerShell 获取证书的哈希值。您还可以使用 CA 的 GUI 或 Windows 开发工具包的 `certutil` 可执行文件。

获取证书的指纹并使用它来对文件进行签名

1. 以管理员 PowerShell 身份打开并运行以下命令：

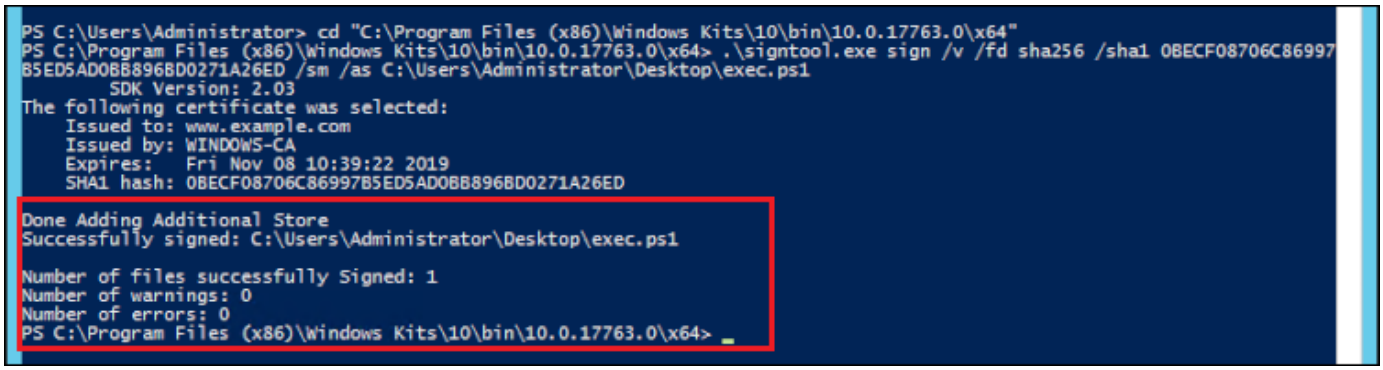
   ```
   Get-ChildItem -path cert:\LocalMachine\My
   ```

复制返回的 Thumbprint。



2. 导航到其中包含 PowerShell 的目录SignTool.exe。默认位置是 C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64。
3. 最后，通过运行以下命令来对文件签名。如果命令成功，则 PowerShell 返回成功消息。

```
signtool.exe sign /v /fd sha256 /sha1 <thumbprint> /sm C:\Users\Administrator
\Desktop\<test>.ps1
```



4. （可选）要验证文件上的签名，请使用以下命令：

```
signtool.exe verify /v /pa C:\Users\Administrator\Desktop\<test>.ps1
```

Java Keytool 和 Jarsigner

AWS CloudHSM 通过客户端 SDK 3 和客户端 SDK 5 提供与 Java Keytool 和 Jarsigner 实用程序的集成。根据您当前下载的客户端软件开发工具包 的版本，使用这些工具的步骤会有所不同：

- 使用客户端软件开发工具包 5 与 Java Keytool 和 Jarsigner 集成
- 使用客户端软件开发工具包 3 与 Java Keytool 和 Jarsigner 集成

使用客户端软件开发工具包 5 与 Java Keytool 和 Jarsigner 集成

AWS CloudHSM 密钥存储是一种特殊用途的 JCE 密钥存储区，它通过第三方工具（如和）使用与 HSM 上的密钥关联的证书。`keytool` `jarsigner` AWS CloudHSM 不在 HSM 上存储证书，因为证书是公开的非机密数据。 AWS CloudHSM 密钥库将证书存储在本地文件中，并将证书映射到您的 HSM 上的相应密钥。

当您使用密 AWS CloudHSM 钥库生成新密钥时，本地密钥存储文件中不会生成任何条目——密钥是在 HSM 上创建的。同样，当您使用 AWS CloudHSM 密钥库搜索密钥时，搜索将传递到 HSM。当您将证书存储在 AWS CloudHSM 密钥库中时，提供程序会验证 HSM 上是否存在具有相应别名的密钥对，然后将提供的证书与相应的密钥对相关联。

主题

先决条件

要使用 AWS CloudHSM 密钥库，必须先初始化并配置 AWS CloudHSM JCE SDK。

步骤 1：安装 JCE

要安装 JCE，包括 AWS CloudHSM 客户端先决条件，请按照安装 Java 库的步骤进行操作。

步骤 2：将 HSM 登录凭证添加到环境变量

设置环境变量以包含 HSM 登录凭证。

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 提供各种登录选项。要将 AWS CloudHSM 密钥存储用于第三方应用程序，必须使用带环境变量的隐式登录。如果要通过应用程序代码使用显式登录，则必须使用 AWS CloudHSM 密钥库构建自己的应用程序。有关更多信息，请参阅有关使用 AWS CloudHSM 密钥库的文章。

步骤 3：注册 JCE 提供程序

要在 Java CloudProvider 配置中注册 JCE 提供程序，请执行以下步骤：

1. 在 Java 安装中打开 `java.security` 配置文件进行编辑。
2. 在 `java.security` 配置文件中，添加 `com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider` 作为最后一个提供程序。例如，如果 `java.security` 文件中有 9 个提供程序，则将以下提供程序添加为本节中的最后一个提供程序：

   ```
   security.provider.10=com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider
   ```

> **Note**
>
> 将 AWS CloudHSM 提供商添加为更高的优先级可能会对系统的性能产生负面影响，因为对于可以安全地卸载到软件的操作，将优先考虑 AWS CloudHSM 提供商。作为最佳实践，请务必指定要用于操作的提供商，无论是基于软件的提供商还是基于软件的提供商。 AWS CloudHSM

> **Note**
>
> 在使用带有 AWS CloudHSM 密钥库的 keytool 生成密钥时指定 `-providerName`、`-providerclass` 和 `-providerpath` 命令行选项可能会导致错误。

使用带有 AWS CloudHSM keytool 的密钥库

Keytool 是一个常见的命令行实用程序，用于常见密钥和证书任务。有关 keytool 的完整教程不在 AWS CloudHSM 文档的讨论范围内。本文介绍了在通过密钥库用作信任根时，应与各种 AWS CloudHSM 密钥工具函数一起使用的特定参数。 AWS CloudHSM

在密钥库中 AWS CloudHSM 使用 keytool 时，请为任何 keytool 命令指定以下参数：

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 AWS CloudHSM 密钥库创建新的密钥库文件，请参阅使用 AWS CloudHSM KeyStore。要使用现有密钥库，请使用 keytool 的 –keystore 参数指定密钥库的名称（包括路径）。如果您在 keytool 命令中指定了不存在的密钥存储文件，则 AWS CloudHSM 密钥库会创建一个新的密钥存储文件。

使用 keytool 创建新密钥

您可以使用 keytool 生成 AWS CloudHSM的 JCE SDK 支持的 RSA、AES 和 DESede 类型的密钥。

> **Important**
>
> 通过 keytool 生成的密钥在软件中生成，然后 AWS CloudHSM 作为可提取的永久密钥导入到。

我们强烈建议在 keytool 之外生成不可导出的密钥，然后将相应的证书导入密钥库。如果您通过 keytool 和 Jarsigner 使用可提取的 RSA 或 EC 密钥，则提供程序会从中导出密钥， AWS CloudHSM 然后在本地使用该密钥进行签名操作。

如果您有多个客户端实例连接到您的 AWS CloudHSM 集群，请注意，在一个客户端实例的密钥库中导入证书不会自动使证书在其他客户端实例上可用。要在每个客户端实例上注册密钥和关联证书，您需要运行 Java 应用程序，如 the section called “使用 Keytool 生成 CSR” 中所述。或者，您可以在一个客户端上进行必要的更改，并将生成的密钥库文件复制到其他每个客户端实例。

示例 1：生成对称 AES-256 密钥，并将其保存在工作目录中名为“my_keystore.store”的密钥库文件中。将 *<secret label>* 替换为唯一标签。

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\*"'
```

示例 2：生成 RSA 2048 密钥对，并将其保存在工作目录中名为“my_keystore.store”的密钥库文件中。将 *<RSA key pair label>* 替换为唯一标签。

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 库中找到支持的签名算法列表。

使用 Keytool 删除密钥

密 AWS CloudHSM 钥库不支持删除密钥。您可以使用 Destroyable 接口的销毁方法删除密钥。

```
((Destroyable) key).destroy();
```

使用 Keytool 生成 CSR

如果使用 OpenSSL 动态引擎，您可以在生成证书签名请求 (CSR) 时获得最大的灵活性。以下命令使用 keytool 为具有别名 `my-key-pair` 的密钥对生成 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**
>
> 要从 keytool 使用密钥对，该密钥对必须在指定的密钥库文件中包含一个条目。如果要使用在 keytool 之外生成的密钥对，则必须将密钥和证书元数据导入密钥库中。有关导入密钥库数据的说明，请参阅 the section called “使用 keytool 将中间证书和根证书导入 AWS CloudHSM 密钥存储库 ”。

使用 keytool 将中间证书和根证书导入 AWS CloudHSM 密钥存储库

要导入 CA 证书，您必须在新导入的证书上启用完整证书链的验证。以下命令是一个示例。

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\*"'
```

如果您将多个客户端实例连接到 AWS CloudHSM 集群，则在一个客户端实例的密钥库中导入证书不会自动使该证书在其他客户端实例上可用。您必须在每个客户端实例上导入证书。

使用 keytool 从 AWS CloudHSM 密钥库中删除证书

以下命令举例说明了如何从 Java keytool 密钥库中删除证书。

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\*"'
```

如果您将多个客户端实例连接到 AWS CloudHSM 集群，则删除一个客户端实例密钥存储中的证书不会自动从其他客户端实例中删除该证书。您必须删除每个客户端实例上的证书。

使用 keytool 将有效的证书导入 AWS CloudHSM 密钥库

签署证书签名请求 (CSR) 后，您可以将其导入 AWS CloudHSM 密钥库并将其与相应的密钥对关联。以下命令是一个示例。

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\*"'
```

别名应该是密钥库中具有关联证书的密钥对。如果密钥是在 keytool 之外生成的，或者在其他客户端实例上生成的，则必须先将密钥和证书元数据导入密钥库中。

证书链必须是可验证的。如果无法验证证书，则可能需要将签名（证书颁发机构）证书导入密钥库，以便验证证书链。

使用 Keytool 导出证书

以下示例生成二进制 X.509 格式的证书。要导出人类可读的证书，请将 `-rfc` 添加到 `-exportcert` 命令中。

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\*"'
```

在 Jarsigner 中使用 AWS CloudHSM 密钥库

Jarsigner 是一种流行的命令行实用程序，用于使用安全存储在 HSM 上的密钥对 JAR 文件进行签名。有关 Jarsigner 的完整教程不在 AWS CloudHSM 文档的讨论范围内。本节介绍了 Jarsigner 参数，您应该使用这些参数通过 AWS CloudHSM 密钥存储区 AWS CloudHSM 作为信任根进行签名和验证签名。

设置密钥和证书

在使用 Jarsigner 签署 JAR 文件之前，请确保您已设置或完成以下步骤：

1. 按照 AWS CloudHSM 密钥库先决条件中的指导操作。
2. 设置您的签名密钥以及相关的证书和证书链，这些证书和证书链应存储在当前服务器或客户端实例的 AWS CloudHSM 密钥存储中。在上创建密钥， AWS CloudHSM 然后将关联的元数据导入您的 AWS CloudHSM 密钥存储区。如果要使用 keytool 设置密钥和证书，请参阅 the section called “使用 keytool 创建新密钥”。如果您使用多个客户端实例签署 JAR，请创建密钥并导入证书链。然后将生成的密钥库文件复制到每个客户端实例。如果您经常生成新密钥，您可能会发现将证书单独导入到每个客户端实例更容易。
3. 整个证书链应该是可验证的。要使证书链可验证，您可能需要将 CA 证书和中间证书添加到 AWS CloudHSM 密钥库中。请参阅 the section called “使用 AWS CloudHSM 和 Jarsigner 签署 JAR 文件” 中的代码片段，了解有关使用 Java 代码验证证书链的说明。如果您愿意，可以使用 keytool 导入证书。有关使用 keytool 的说明，请参阅 the section called “使用 keytool 将中间证书和根证书导入 AWS CloudHSM 密钥存储库 ”。

使用 AWS CloudHSM 和 Jarsigner 签署 JAR 文件

使用以下命令签署 JAR 文件：

Linux;

适用于 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>
```

适用于 OpenJDK 11、openJDK 17 和 OpenJDK 21

```
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

For 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>
```

适用于 OpenJDK 11、openJDK 17 和 OpenJDK 21

```
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>
```

使用以下命令验证已签名的 JAR：

Linux

For 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>
```

适用于 OpenJDK 11、openJDK 17 和 OpenJDK 21

```
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

适用于 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>
```

适用于 OpenJDK 11、openJDK 17 和 OpenJDK 21

```
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>
```

已知问题

1. 我们不支持 Keytool 和 Jarsigner 的 EC 密钥。

使用客户端软件开发工具包 3 与 Java Keytool 和 Jarsigner 集成

AWS CloudHSM 密钥存储是一种特殊用途的 JCE 密钥存储区，它通过第三方工具（如和）使用与 HSM 上的密钥关联的证书。`keytool` `jarsigner` AWS CloudHSM 不在 HSM 上存储证书，因为证书是公开的非机密数据。 AWS CloudHSM 密钥库将证书存储在本地文件中，并将证书映射到您的 HSM 上的相应密钥。

当您使用密 AWS CloudHSM 钥库生成新密钥时，本地密钥存储文件中不会生成任何条目——密钥是在 HSM 上创建的。同样，当您使用 AWS CloudHSM 密钥库搜索密钥时，搜索将传递到 HSM。当您将证书存储在 AWS CloudHSM 密钥库中时，提供程序会验证 HSM 上是否存在具有相应别名的密钥对，然后将提供的证书与相应的密钥对相关联。

主题

先决条件

要使用 AWS CloudHSM 密钥库，必须先初始化并配置 AWS CloudHSM JCE SDK。

步骤 1：安装 JCE

要安装 JCE，包括 AWS CloudHSM 客户端先决条件，请按照安装 Java 库的步骤进行操作。

步骤 2：将 HSM 登录凭证添加到环境变量

设置环境变量以包含 HSM 登录凭证。

```
export HSM_PARTITION=PARTITION_1
export HSM_USER=<HSM user name>
export HSM_PASSWORD=<HSM password>
```

> **Note**
>
> CloudHSM JCE 提供各种登录选项。要将 AWS CloudHSM 密钥存储用于第三方应用程序，必须使用带环境变量的隐式登录。如果要通过应用程序代码使用显式登录，则必须使用 AWS CloudHSM 密钥库构建自己的应用程序。有关更多信息，请参阅有关使用 AWS CloudHSM 密钥库的文章。

步骤 3：注册 JCE 提供程序

要注册 JCE 提供程序，请在 Java CloudProvider 配置中进行注册。

1. 在 Java 安装中打开 java.security 配置文件进行编辑。
2. 在 java.security 配置文件中，添加 `com.cavium.provider.CaviumProvider` 作为最后一个提供程序。例如，如果 java.security 文件中有 9 个提供程序，则将以下提供程序添加为本部分中的最后一个提供程序。将 Cavium 提供程序添加为一个更高的优先级可能会对您的系统性能产生负面影响。

   ```
   security.provider.10=com.cavium.provider.CaviumProvider
   ```

> **Note**
>
> 使用 keytool 时，高级用户可能习惯于指定 `-providerName`、`-providerclass` 和 `-providerpath` 命令行选项，而不是更新安全配置文件。如果您在使用密钥存储库生成密钥时尝试指定命令行选项，则会导致错误。 AWS CloudHSM

使用带有 AWS CloudHSM keytool 的密钥库

Keytool 是一个常见的命令行实用程序，用于 Linux 系统上的常见密钥和证书任务。有关 keytool 的完整教程不在 AWS CloudHSM 文档的讨论范围内。本文介绍了在通过密钥库用作信任根时，应与各种 AWS CloudHSM 密钥工具函数一起使用的特定参数。 AWS CloudHSM

在密钥库中 AWS CloudHSM 使用 keytool 时，请为任何 keytool 命令指定以下参数：

```
-storetype CLOUDHSM \
  -J-classpath '-J/opt/cloudhsm/java/*' \
  -J-Djava.library.path=/opt/cloudhsm/lib
```

如果要使用 AWS CloudHSM 密钥库创建新的密钥库文件，请参阅使用 AWS CloudHSM KeyStore。要使用现有密钥库，请使用 keytool 的 –keystore 参数指定密钥库的名称（包括路径）。如果您在 keytool 命令中指定了不存在的密钥存储文件，则 AWS CloudHSM 密钥库会创建一个新的密钥存储文件。

使用 keytool 创建新密钥

你可以使用 keytool 生成的 JCE SDK 支持的任何 AWS CloudHSM类型的密钥。请参阅 Java 库中支持的密钥文章中的密钥和长度的完整列表。

> **Important**
>
> 通过 keytool 生成的密钥在软件中生成，然后 AWS CloudHSM 作为可提取的永久密钥导入到。

直接在 HSM 上创建不可提取的密钥，然后将其与 keytool 或 Jarsigner 一起使用的说明，如向密钥库注册已存在的密钥中的代码示例所示。 AWS CloudHSM 我们强烈建议在 keytool 之外生成不可导出的密钥，然后将相应的证书导入密钥库。如果您通过 keytool 和 jarsigner 使用可提取的 RSA 或 EC 密钥，则提供程序会从中导出密钥， AWS CloudHSM 然后在本地使用该密钥进行签名操作。

如果您有多个客户端实例连接到 CloudHSM 集群，请注意，在一个客户端实例的密钥库上导入证书不会自动使证书在其他客户端实例上可用。要在每个客户端实例上注册密钥和关联证书，您需要运行 Java 应用程序，如使用 Keytool 生成 CSR 中所述。或者，您可以在一个客户端上进行必要的更改，并将生成的密钥库文件复制到其他每个客户端实例。

示例 1：生成对称 AES-256 密钥，并将其保存在工作目录中名为“my_keystore.store”的密钥库文件中。将 *<secret label>* 替换为唯一标签。

```
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/
```

示例 2：生成 RSA 2048 密钥对，并将其保存在工作目录中名为“my_keystore.store”的密钥库文件中。将 *<RSA key pair label>* 替换为唯一标签。

```
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/
```

示例 3：生成 p256 ED 密钥对，并将其保存在工作目录中名为“my_keystore.store”的密钥库文件中。将 *<ec key pair label>* 替换为唯一标签。

```
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 库中找到支持的签名算法列表。

使用 Keytool 删除密钥

密 AWS CloudHSM 钥库不支持删除密钥。要删除密钥，必须使用 AWS CloudHSM的命令行工具的`deleteKey`功能deleteKey。

使用 Keytool 生成 CSR

如果使用 OpenSSL 动态引擎，您可以在生成证书签名请求 (CSR) 时获得最大的灵活性。以下命令使用 keytool 为具有别名 my-key-pair 的密钥对生成 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**
>
> 要从 keytool 使用密钥对，该密钥对必须在指定的密钥库文件中包含一个条目。如果要使用在 keytool 之外生成的密钥对，则必须将密钥和证书元数据导入密钥库中。有关导入密钥库数据的说明，请参阅使用 Keytool 将中间证书和根证书导入 AWS CloudHSM 密钥库。

使用 keytool 将中间证书和根证书导入 AWS CloudHSM 密钥存储库

要导入 CA 证书，您必须在新导入的证书上启用完整证书链的验证。以下命令是一个示例。

```
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/
```

如果您将多个客户端实例连接到 AWS CloudHSM 集群，则在一个客户端实例的密钥库中导入证书不会自动使该证书在其他客户端实例上可用。您必须在每个客户端实例上导入证书。

使用 keytool 从 AWS CloudHSM 密钥库中删除证书

以下命令举例说明了如何从 Java keytool 密钥库中删除证书。

```
keytool -delete -alias mydomain -keystore \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
```

```
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

如果您将多个客户端实例连接到 AWS CloudHSM 集群，则删除一个客户端实例密钥存储中的证书不会自动从其他客户端实例中删除该证书。您必须删除每个客户端实例上的证书。

使用 keytool 将有效的证书导入 AWS CloudHSM 密钥库

签署证书签名请求 (CSR) 后，您可以将其导入 AWS CloudHSM 密钥库并将其与相应的密钥对关联。以下命令是一个示例。

```
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/
```

别名应该是密钥库中具有关联证书的密钥对。如果密钥是在 keytool 之外生成的，或者在其他客户端实例上生成的，则必须先将密钥和证书元数据导入密钥库中。有关导入证书元数据的说明，请参阅向密钥库注册已存在的密钥中的代码示例。 AWS CloudHSM

证书链必须是可验证的。如果无法验证证书，则可能需要将签名（证书颁发机构）证书导入密钥库，以便验证证书链。

使用 Keytool 导出证书

以下示例生成二进制 X.509 格式的证书。要导出人类可读的证书，请将 `-rfc` 添加到 `-exportcert` 命令中。

```
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/
```

在 jarsigner 中使用 AWS CloudHSM 密钥库

Jarsigner 是一种流行的命令行实用程序，用于使用安全存储在 HSM 上的密钥对 JAR 文件进行签名。有关 Jarsigner 的完整教程不在 AWS CloudHSM 文档的讨论范围内。本节介绍了 Jarsigner 参数，您

应该使用这些参数通过 AWS CloudHSM 密钥存储区 AWS CloudHSM 作为信任根进行签名和验证签名。

设置密钥和证书

在使用 Jarsigner 签署 JAR 文件之前，请确保您已设置或完成以下步骤：

1. 按照 AWS CloudHSM 密钥库先决条件中的指导操作。
2. 设置您的签名密钥以及相关的证书和证书链，这些证书和证书链应存储在当前服务器或客户端实例的 AWS CloudHSM 密钥存储中。在上创建密钥， AWS CloudHSM 然后将关联的元数据导入您的 AWS CloudHSM 密钥存储区。使用向密钥库注册预先存在的 AWS CloudHSM 密钥中的代码示例将元数据导入密钥库。如果要使用 keytool 设置密钥和证书，请参阅 使用 keytool 创建新密钥。如果您使用多个客户端实例签署 JAR，请创建密钥并导入证书链。然后将生成的密钥库文件复制到每个客户端实例。如果您经常生成新密钥，您可能会发现将证书单独导入到每个客户端实例更容易。
3. 整个证书链应该是可验证的。要使证书链可验证，您可能需要将 CA 证书和中间证书添加到 AWS CloudHSM 密钥库中。有关使用 Java 代码验证证书链的说明，请参阅使用对 Jarsigner 文件进行签名中的代码片段， AWS CloudHSM 以及 Jarsigner。如果您愿意，可以使用 keytool 导入证书。有关使用 keytool 的说明，请参阅使用 Keytool 将中间证书和根证书导入 AWS CloudHSM 密钥库。

使用 AWS CloudHSM 和 jarsigner 签署 JAR 文件

使用以下命令签署 JAR 文件：

```
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>
```

使用以下命令验证已签名的 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>
```

已知问题

以下列表提供了已知问题的当前列表。

- 使用 keytool 生成密钥时，提供程序配置中的第一个提供程序不能是。 CaviumProvider
- 使用 keytool 生成密钥时，安全配置文件中的第一个（受支持的）提供程序将用于生成密钥。这通常是一个软件提供程序。然后，生成的密钥会被赋予一个别名，并在密钥添加过程中作为永久（令牌）密钥导入 AWS CloudHSM HSM。
- 将 keytool 与 AWS CloudHSM 密钥存储一起使用时`-providerName`，请勿在命令行中指定`-providerclass`、或`-providerpath`选项。在安全提供程序文件中指定这些选项，如密钥库先决条件中所述。
- 当通过 keytool 和 Jarsigner 使用不可提取的 EC 密钥时，SunEC 提供程序需要从 java.security 文件中的提供程序列表中删除/禁用。如果您通过 keytool 和 Jarsigner 使用可提取的 EC 密钥，则提供程序会从 AWS CloudHSM HSM 中导出密钥位并在本地使用该密钥进行签名操作。我们不建议您将可导出的密钥与 keytool 或 Jarsigner 一起使用。

向密钥库注册先前存在的密 AWS CloudHSM 钥

为了实现属性和标签的最大安全性和灵活性，我们建议您使用 key_mgmt_util 生成签名密钥。您还可以使用 Java 应用程序在 AWS CloudHSM中生成密钥。

以下部分提供了一个代码示例，演示如何在 HSM 上生成新的密钥对，并使用导入到密钥库中的 AWS CloudHSM 现有密钥对其进行注册。导入的密钥可与第三方工具（如 keytool 和 Jarsigner）一起使用。

要使用预先存在的密钥，请修改代码示例以通过标签查找密钥，而不是生成新密钥。上的 KeyUtilitiesRunner.java 示例中提供了按标签查找密钥的 GitHub示例代码。

> **Important**
>
> 在本地密钥存储库中注册存储 AWS CloudHSM 的密钥不会导出该密钥。注册密钥后，密钥库会注册密钥的别名（或标签），并将本地存储证书对象与 AWS CloudHSM上的密钥对关联。只要密钥对创建为不可导出的对象，密钥位就不会离开 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());
        }
    }

}
```

其他第三方供应商集成

一些第三方供应商支持 AWS CloudHSM 将其作为信任根源。这意味着您可以使用自己选择的软件解决方案，同时在 CloudHSM 集群中创建并存储底层密钥。因此，您的工作负载 AWS 可以依赖 CloudHSM 的延迟、可用性、可靠性和弹性优势。以下列表包括了支持 CloudHSM 的第三方供应商。

> **Note**
>
> AWS 不为任何第三方供应商背书或担保。

- Hashicorp Vault 是一个密钥管理工具，旨在实现跨组织的协作和管理。它支持 AWS Key Management Service 并 AWS CloudHSM 作为额外保护的信任根源。
- Thycotic Secrets Server 可帮助客户跨特权账户管理敏感凭证。它 AWS CloudHSM 作为信任根提供支持。
- P6R 的 KMIP 适配器允许您通过标准 KMIP 接口使用您的 AWS CloudHSM 实例。
- PrimeKey EJBCA 是 PKI 的流行开源解决方案。它允许您使用安全地创建和存储密钥对 AWS CloudHSM。

- Box KeySafe 为许多具有严格安全、隐私和监管合规要求的组织提供云内容的加密密钥管理。客户可以直接 AWS Key Management Service 或 AWS CloudHSM 通过 AWS KMS 自定义 KeySafe 密钥存储区间接保护密钥。
- Insyde Software 支持 AWS CloudHSM 作为固件签名的信任根。
- F5 BIG-IP LTM 支持 AWS CloudHSM 作为信任根。
- Cloudera Navigator Key HSM 允许您使用 CloudHSM 集群创建和存储 Cloudera Navigator Key Trustee Server 的密钥。
- Venafi 信任保护平台通过 AWS CloudHSM 密钥生成和保护，为 TLS、SSH 以及代码签名提供全面的计算机身份管理。

监控 AWS CloudHSM

除了客户端 SDK 中内置的日志功能外，您还可以使用 AWS CloudTrail Amazon CloudWatch Logs 和 Amazon CloudWatch 进行监控 AWS CloudHSM。

客户端软件开发工具包日志

使用客户端软件开发工具包日志记录来监控您创建的应用程序中的诊断和故障排除信息。

CloudTrail

CloudTrail 用于监控您 AWS 账户中的所有 API 调用，包括您为创建和删除集群、硬件安全模块 (HSM) 和资源标签而进行的调用。

CloudWatch 日志

使用 CloudWatch 日志监控 HSM 实例的日志，其中包括创建和删除 HSM 用户、更改用户密码、创建和删除密钥等事件。

CloudWatch

CloudWatch 用于实时监控集群的运行状况。

主题

使用客户端软件开发工具包日志

您可以检索客户端 SDK 生成的日志。 AWS CloudHSM 提供了使用客户端 SDK 3 和客户端 SDK 5 进行日志记录的实现。

主题

客户端软件开发工具包 5 日志记录

客户端软件开发工具包 5 日志包含以组件命名的文件中的每个组件的信息。您可以使用适用于客户端软件开发工具包 5 的配置工具为每个组件配置日志记录。

如果您并未为文件指定位置，则系统会将日志写入默认位置：

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
```

有关如何为客户端软件开发工具包 5 配置日志记录的信息，请参阅客户端软件开发工具包 5 配置工具

客户端软件开发工具包 3 日志记录

客户端 SDK 3 日志包含来自 AWS CloudHSM 客户端守护程序的详细信息。日志的位置取决于您在其中运行客户端进程守护程序的 Amazon EC2 客户端实例的操作系统。

Amazon Linux

在 Amazon Linux 中，AWS CloudHSM 客户端日志会写入名为的文件中`/opt/cloudhsm/run/cloudhsm_client.log`。您可以使用 logrotate 或类似工具来轮换和管理这些日志。

Amazon Linux 2

在 Amazon Linux 2 中，AWS CloudHSM 客户日志被收集并存储在日志中。您可以使用 journalctl 查看和管理这些日志。例如，使用以下命令查看 AWS CloudHSM 客户机日志。

```
journalctl -f -u cloudhsm-client
```

CentOS 7

在 CentOS 7 中，AWS CloudHSM 客户日志被收集并存储在日志中。您可以使用 journalctl 查看和管理这些日志。例如，使用以下命令查看 AWS CloudHSM 客户机日志。

```
journalctl -f -u cloudhsm-client
```

CentOS 8

在 CentOS 8 中，AWS CloudHSM 客户日志被收集并存储在日志中。您可以使用 journalctl 查看和管理这些日志。例如，使用以下命令查看 AWS CloudHSM 客户机日志。

```
journalctl -f -u cloudhsm-client
```

RHEL 7

在红帽企业 Linux 7 中，收集 AWS CloudHSM 客户端日志并将其存储在日志中。您可以使用 journalctl 查看和管理这些日志。例如，使用以下命令查看 AWS CloudHSM 客户机日志。

```
journalctl -f -u cloudhsm-client
```

RHEL 8

在红帽企业 Linux 8 中，收集 AWS CloudHSM 客户端日志并将其存储在日志中。您可以使用 journalctl 查看和管理这些日志。例如，使用以下命令查看 AWS CloudHSM 客户机日志。

```
journalctl -f -u cloudhsm-client
```

Ubuntu 16.04

在 Ubuntu 16.04 中，收集 AWS CloudHSM 客户端日志并将其存储在日志中。您可以使用 journalctl 查看和管理这些日志。例如，使用以下命令查看 AWS CloudHSM 客户机日志。

```
journalctl -f -u cloudhsm-client
```

Ubuntu 18.04

在 Ubuntu 18.04 中，收集 AWS CloudHSM 客户端日志并将其存储在日志中。您可以使用 journalctl 查看和管理这些日志。例如，使用以下命令查看 AWS CloudHSM 客户机日志。

```
journalctl -f -u cloudhsm-client
```

Windows

- 对于 Windows 客户端 1.1.2 以上版本：

  AWS CloudHSM 客户机日志写入 AWS CloudHSM 程序`cloudhsm.log`文件文件夹 (`C:\Program Files\Amazon\CloudHSM\`) 中的一个文件。每个日志文件名的后缀都有一个时间戳，指示 AWS CloudHSM 客户端的启动时间。

- 对于 Windows 客户端 1.1.1 及更低版本：

  客户端的日志不会写入文件。日志显示在命令提示符处或启动 AWS CloudHSM 客户端的 PowerShell 窗口中。

使用 AWS CloudTrail 和 AWS CloudHSM

AWS CloudHSM 与 AWS CloudTrail一项服务集成，该服务提供用户、角色或 AWS 服务在中执行的操作的记录 AWS CloudHSM。 CloudTrail 将所有 API 调用捕获 AWS CloudHSM 为事件。捕获的调用包括来自 AWS CloudHSM 控制台的调用和对 AWS CloudHSM API 操作的代码调用。如果您创建了跟踪，则可以允许将 CloudTrail 事件持续传输到 Amazon S3 存储桶，包括的事件 AWS CloudHSM。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。使用收集的信息 CloudTrail，您可以确定向哪个请求发出 AWS CloudHSM、发出请求的 IP 地址、谁发出了请求、何时发出请求以及其他详细信息。

要了解更多信息 CloudTrail，请参阅《AWS CloudTrail 用户指南》。有关 AWS CloudHSM API 操作的完整列表，请参阅 AWS CloudHSM API 参考中的操作。

AWS CloudHSM 信息在 CloudTrail

CloudTrail 在您创建 AWS 账户时已在您的账户上启用。当活动发生在中时 AWS CloudHSM，该活动会与其他 AWS 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用事件历史记录查看 CloudTrail 事件。

要持续记录您 AWS 账户中的事件，包括的事件 AWS CloudHSM，请创建跟踪。跟踪允许 CloudTrail 将日志文件传输到 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 AWS 区域。跟踪记录 AWS 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 AWS 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅下列内容：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 配置 Amazon SNS 通知 CloudTrail
- 接收来自多个区域的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail日志文件

CloudTrail 记录所有 AWS CloudHSM 操作，包括只读操作（例如`DescribeClusters`和`ListTags`）以及管理操作（例如`InitializeClusterCreatHsm`、和）`DeleteBackup`。

每个事件或日记账条目都包含有关生成请求的人员信息。身份信息有助于您确定以下内容：

- 请求是使用根证书还是 AWS Identity and Access Management (IAM) 用户凭证发出。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅CloudTrail 用户身份元素。

了解 AWS CloudHSM 日志文件条目

跟踪是一种配置，允许将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定的顺序出现。

以下示例显示了演示该 AWS CloudHSM `CreateHsm`操作的 CloudTrail 日志条目。


```
{
    "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 日志和 AWS CloudHSM 审核日志

当您账户中的 HSM 收到来自命令行工具或软件库的 AWS CloudHSM 命令时，它会以审计日志的形式记录该命令的执行情况。HSM 审计日志包含所有客户端启动的管理命令，包括用于创建和删除 HSM、登录和注销 HSM 以及管理用户和密钥的命令。这些日志提供了已更改 HSM 的状态的操作的可靠记录。

AWS CloudHSM 收集您的 HSM 审计日志，并代表您将其发送到 Amazon CloudWatch Logs。您可以使用 CloudWatch 日志的功能来管理您的 AWS CloudHSM 审核日志，包括搜索和筛选日志，以及将日志数据导出到 Amazon S3。您可以在 Amazon CloudWatch 控制台中使用 HSM 审核日志，也可以使用AWS CLI和 CloudWatch 日志软件开发工具包中的CloudWatch 日志命令。

主题

- HSM 审核日志记录的工作原理
- 在 CloudWatch 日志中查看 HSM 审核日志
- 解释 HSM 审核日志
- HSM 审核日志参考

HSM 审核日志记录的工作原理

在所有 AWS CloudHSM 集群中自动启用审核日志。它不能被禁用或关闭，也没有任何设置可以 AWS CloudHSM 阻止将日志导出到 CloudWatch 日志。每个日志事件均有一个时间戳和序列号，它们指示事件的顺序并帮助您检测任何日志篡改。

每个 HSM 实例均生成自己的日志。各种 HSM 的审计日志（甚至是同一群集中的审计日志）可能有所不同。例如，每个群集中只有第一个 HSM 记录 HSM 的初始化。初始化事件不会出现在从备份克隆的 HSM 日志中。类似地，当您创建密钥时，生成密钥的 HSM 会记录一个密钥生成事件。群集中的其他 HSM 在通过同步接收密钥时将记录事件。

AWS CloudHSM 收集日志并将其发布到您账户中的 CloudWatch 日志。要代表您与 CloudWatch 日志服务通信，请 AWS CloudHSM 使用服务相关角色。与该角色关联的 IAM 策略仅 AWS CloudHSM 允许执行将审计日志发送到 Logs 所需的任务。 CloudWatch

> **Important**
>
> 如果您在 2018 年 1 月 20 日前创建了一个群集，并且还没有创建附加的服务相关角色，则必须手动创建一个。这是从您的 AWS CloudHSM 集群 CloudWatch 接收审计日志所必需的。有关如何在 HSM 集群上启用服务相关角色的说明，请参阅 《IAM 用户指南》中的了解服务相关角色和创建服务相关角色。有关创建服务相关角色的更多信息，请参阅了解服务相关角色以及《IAM 用户指南》中的《创建服务相关角色》。

在 CloudWatch 日志中查看 HSM 审核日志

Amazon Logs 将审核 CloudWatch 日志组织到日志组中，并在日志组内组织到日志流中。每个日志条目都是一个事件。 AWS CloudHSM 为每个集群创建一个日志组，为集群中的每个 HSM 创建一个日志流。您无需创建任何 CloudWatch Logs 组件或更改任何设置。

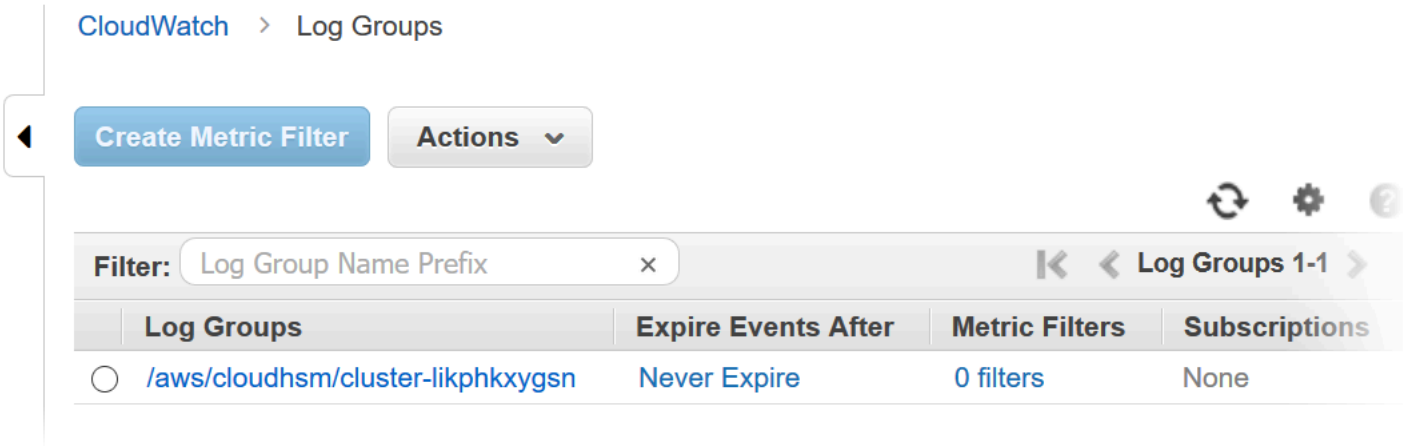
- 日志组名称为 `/aws/cloudhsm/<cluster ID>`；例如 `/aws/cloudhsm/cluster-likphkxygsn`。在 AWS CLI 或 PowerShell 命令中使用日志组名称时，请务必用双引号将其括起来。
- 日志流名称为 HSM ID；例如 `hsm-nwbbiqbj4jk`。

  一般来说，每个 HSM 均有一个日志流。不过，任何更改 HSM ID 的操作（例如，当 HSM 发生故障并被替换时）都会创建新的日志流。

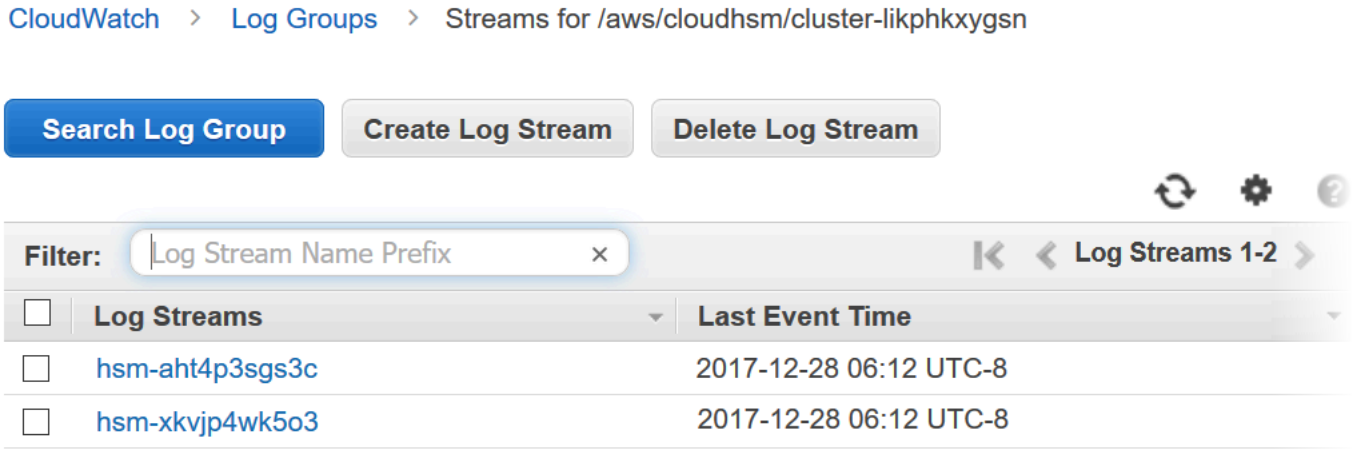
有关 CloudWatch 日志概念的更多信息，请参阅 Amazon CloudWatch 日志用户指南中的概念。

您可以从中的日志页面、中的日志命令 AWS Management Console、 CloudWatch 日志 PowerShell cmdlet 或CloudWatch 日志SDK 中查看 HSM 的审核CloudWatch 日志。 AWS CLI CloudWatch 有关说明，请参阅 Amazon 日志用户指南中的查看 CloudWatch 日志数据。

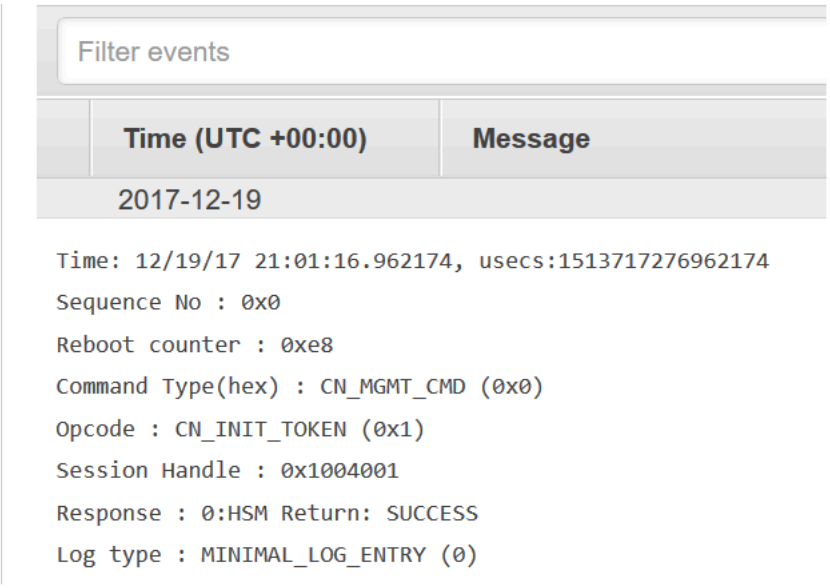
例如，下图显示 AWS Management Console中的 `cluster-likphkxygsn` 群集的日志组。



当您选择群集日志组名称时，可以查看群集中每个 HSM 的日志流。下图显示 `cluster-likphkxygsn` 群集中的 HSM 的日志流。

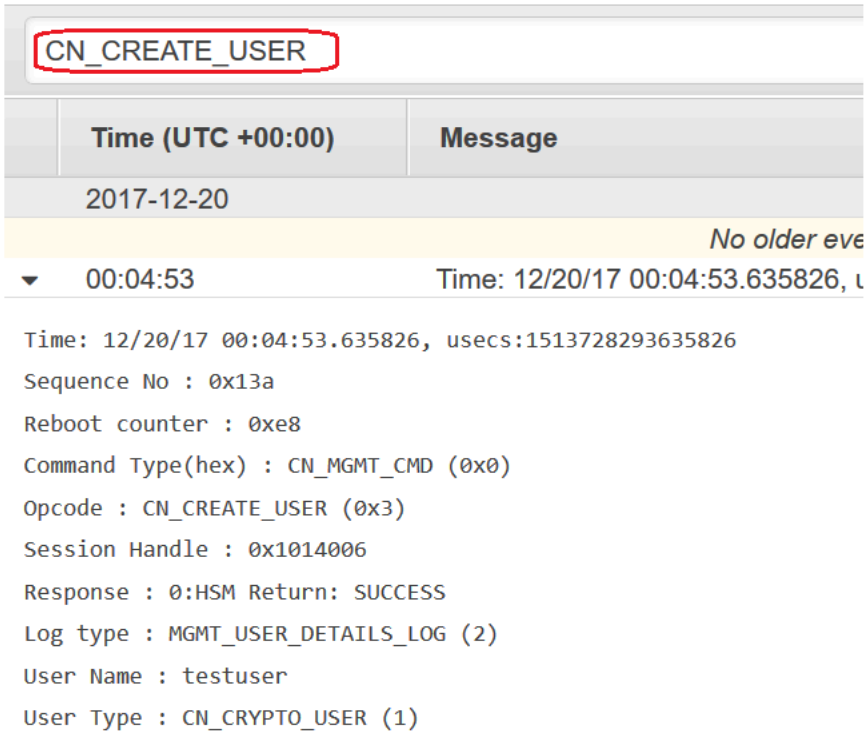


当您选择 HSM 日志流名称时，可以查看审核日志中的事件。例如，此事件（其序列号为 0x0 且 `CN_INIT_TOKEN` 为 `Opcode`）通常是每个群集中的第一个 HSM 的第一个事件。它记录群集中 HSM 的初始化。



您可以使用 CloudWatch 日志中的所有功能来管理审核日志。例如，您可以使用筛选事件功能来查找事件中的特定文本，例如 CN_CREATE_USER Opcode。

要查找所有不包含指定文本的事件，请在文本前添加减号 (-)。例如，要查找不包含 CN_CREATE_USER 的事件，请输入 -CN_CREATE_USER。



解释 HSM 审核日志

HSM 审核日志中的事件具有标准字段。一些事件类型具有附加字段，这些字段将捕获有关事件的有用信息。例如，用户登录和用户管理事件包括用户的名称和类型。密钥管理命令包括密钥句柄。

有几个字段提供了特别重要的信息。Opcode 标识正在记录的管理命令。Sequence No 标识日志流中的事件并指示事件的记录顺序。

例如，以下示例事件是 HSM 的日志流中的第二个事件 (Sequence No: 0x1)。它显示了生成密码加密密钥的 HSM，该密钥是启动例程的一部分。

```
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)
```

以下字段是审计日志中每个 AWS CloudHSM 事件的通用字段。

时间

事件的发生时间（用 UTC 时区表示）。时间显示为人类可读的时间和 Unix 时间（以微秒为单位）。

重启计数器

在重新启动 HSM 硬件时递增的 32 位持续序数计数器。

日志流中的所有事件都具有相同的重启计数器值。但是，重启计数器对于日志流可能不是唯一的，因为它在同一集群中的不同 HSM 实例之间可能是不同的。

序列号

为每个日志事件递增的 64 位序数计数器。每个日志流中的第一个事件都有一个序列号 0x0。Sequence No 值中不应该有空白。序列号仅在日志流中是唯一的。

命令类型

一个十六进制值，它表示命令的类别。 AWS CloudHSM 日志流中的命令的命令类型为 CN_MGMT_CMD (0x0) 或 CN_CERT_AUTH_CMD (0x9)。

Opcode

标识已执行的管理命令。有关 AWS CloudHSM 审计日志中的0pcode值列表，请参阅HSM 审核日志参考。

会话句柄

标识已在其中运行命令并记录事件的会话。

响应

记录对管理命令的响应。您可以在 Response 字段中搜索 SUCCESS 和 ERROR 值。

日志类型

表示记录命令的 AWS CloudHSM 日志的日志类型。

- MINIMAL_LOG_ENTRY (0)
- MGMT_KEY_DETAILS_LOG (1)
- MGMT_USER_DETAILS_LOG (2)
- GENERIC_LOG

审核日志事件的示例

日志流中的事件将记录 HSM 的历史记录（从其创建到删除）。您可以使用日志来审查 HSM 的生命周期并深入了解其操作。在解释事件时，请注意 Opcode（它表示管理命令或操作）和 Sequence No（它表示事件的顺序）。

主题

示例：初始化集群中的第一个 HSM

每个集群中的第一个 HSM 的审核日志流与集群中的其他 HSM 的日志流有显著区别。每个集群中的第一个 HSM 的审核日志记录了其创建和初始化。集群中通过备份生成的附加 HSM 的日志的开头为登录事件。

> **Important**
>
> 以下初始化条目不会出现在在 CloudHSM 审计 CloudWatch 日志功能发布（2018 年 8 月 30 日）之前初始化的集群的日志中。有关更多信息，请参阅文档历史记录。

以下示例事件显示在集群中的第一个 HSM 的日志流中。日志中的第一个事件 —— 带 Sequence No 0x0 的事件 —— 表示用于初始化 HSM (CN_INIT_TOKEN) 的命令。此响应表示该命令已成功 (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)
```

此示例日志流 (Sequence No 0x1) 中的第二个事件记录了用于创建 HSM 使用的密码加密密钥的命令 (CN_GEN_PSWD_ENC_KEY)。

这是每个集群中的第一个 HSM 的典型启动顺序。由于同一集群中的后续 HSM 是第一个 HSM 的克隆，因此，它们将使用相同的密码加密密钥。

```
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)
```

此示例日志流 (Sequence No 0x2) 中的第三个事件是创建设备用户 (AU)（即 AWS CloudHSM 服务）。涉及 HSM 用户的事件包括用户名和用户类型的额外字段。

```
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)
```

此示例日志流 (Sequence No 0x3) 中的第四个事件记录了 CN_INIT_DONE 事件，该事件完成了 HSM 的初始化。

```
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)
```

您可以跟踪启动序列中的其余事件。这些事件可能包括几个登录和注销事件，以及密钥加密密钥 (KEK) 的生成。以下事件记录了用于更改准加密员 (PRECO) 的密码的命令。此命令将激活集群。

```
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)
```

登录和注销事件

在解释审核日志时，请注意记录用户登录和注销 HSM 的事件。这些事件可帮助您确定哪个用户负责在登录命令和注销命令之间按顺序出现的管理命令。

例如，此日志条目记录名为 admin 的加密员的登录。序列号 0x0 表示这是此日志流中的第一个事件。

当用户登录一个 HSM 时，集群中的其他 HSM 也会记录该用户的登录事件。在初始登录事件后不久，您可以在集群中的其他 HSM 的日志流中找到相应的登录事件。

```
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)
```

以下示例事件记录 admin 加密员注销。序列号 0x2 表示这是日志流中的第三个事件。

如果已登录的用户关闭会话而不注销，则日志流会包含一个 CN_APP_FINALIZE 或关闭会话事件 (CN_SESSION_CLOSE)，而不是 CN_LOGOUT 事件。与登录事件不同，此注销事件通常只由执行命令的 HSM 记录。

```
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)
```

如果登录尝试因用户名称无效而导致失败，HSM 会记录一个 CN_LOGIN 事件（具有登录命令中提供的用户名和类型）。响应显示错误消息 157，这说明用户名不存在。

```
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)
```

如果登录尝试因密码无效而导致失败，HSM 会记录一个 CN_LOGIN 事件（具有登录命令中提供的用户名和类型）。响应显示了错误消息，并返回 RET_USER_LOGIN_FAILURE 错误代码。

```
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)
```

示例：创建和删除用户

此示例显示了在加密员 (CO) 创建和删除用户时记录的日志事件。

第一个事件记录登录到 HSM 的 CO admin。序列号 0x0 表示这是日志流中的第一个事件。已登录的用户的名称和类型包含在事件中。

```
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)
```

日志流中的下一个事件（序列 0x1）记录创建新的加密用户 (CU) 的 CO。新用户的名称和类型包含在事件中。

```
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)
```

然后，CO 会创建另一个加密员 alice。序列号表明此操作紧跟上一个操作，而没有干预操作。

```
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)
```

稍后，名为 admin 的 CO 将登录并删除名为 alice 的加密员。HSM 会记录一个 CN_DELETE_USER 事件。已删除用户的名称和类型包含在事件中。

```
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)
```

示例：创建和删除密钥对

此示例显示了在您创建和删除密钥对时，HSM 审核日志中记录的事件。

以下事件记录名为 crypto_user 的加密用户 (CU) 登录到 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)
```

接下来，CU 会生成一个密钥对 (CN_GENERATE_KEY_PAIR)。私有密钥具有密钥句柄 131079。公有密钥具有密钥句柄 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
```

该 CU 会立即删除密钥对。CN_DESTROY_OBJECT 事件记录公有密钥的删除 (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
```

然后，另一个 CN_DESTROY_OBJECT 事件记录私有密钥的删除 (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
```

最后，CU 将注销。

```
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)
```

示例：生成和同步密钥

此示例显示了在具有多个 HSM 的集群中创建密钥的效果。密钥在一个 HSM 上生成，从 HSM 中提取作为遮蔽对象，并作为遮蔽对象插入到其他 HSM 中。

> **Note**
>
> 客户端工具可能无法同步密钥。或者，命令可能包含 min_srv 参数，该参数仅将密钥同步到指定数量的 HSM。无论哪种情况， AWS CloudHSM 服务都会将密钥同步到集群中的其他 HSM。由于 HSM 仅在其日志中记录客户端管理命令，因此，不会将服务器端同步记录在 HSM 日志中。

首先，考虑接收并执行命令的 HSM 的日志流。日志流以 HSM ID hsm-abcde123456 命名，但 HSM ID 不会出现在日志事件中。

首先，testuser 加密用户 (CU) 登录到 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 运行exSymKey命令来生成对称密钥。hsm-abcde123456 HSM 生成一个具有密钥句柄 262152 的对称密钥。HSM 在其日志中记录一个 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 的日志流中的下一个事件记录密钥同步过程中的第一步。新密钥（密钥句柄 262152）是从 HSM 中作为遮蔽对象提取的。

```
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
```

现在考虑 HSM hsm-zyxwv987654（同一集群中的另一个 HSM）的日志流。此日志流还包括 testuser CU 的登录事件。时间值显示在用户登录到 hsm-abcde123456 HSM 后不久发生。

```
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 的日志流没有 CN_GENERATE_KEY 事件。但它有一个记录将密钥同步到此 HSM 的事件。CN_INSERT_MASKED_OBJECT_USER 事件记录将密钥 262152 接收为遮蔽对象。现在，密钥 262152 存在于集群中的两个 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
```

当 CU 用户注销时，此 CN_LOGOUT 事件仅在接收了命令的 HSM 的日志流中出现。

示例：导出密钥

此示例显示了在加密用户（CU）从具有多个 HSM 的集群中导出密钥时，记录的审核日志事件。

以下事件记录了登录 key_mgmt_util 的 CU (testuser)。

```
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 运行导出密钥的exSymKey命令7，即 256 位 AES 密钥。此命令使用密钥 6（HSM 上的 256 位 AES 密钥）作为包装密钥。

接收命令的 HSM 记录密钥 7（正在导出的密钥）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
```

然后，HSM 记录包装密钥（密钥 6）的 CN_NIST_AES_WRAP 事件。先包装密钥，然后立即被解开包装，而 HSM 仅记录一个事件。

```
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
```

该 exSymKey 命令将导出的密钥写入一个文件中，但不更改 HSM 上的密钥。因此，集群中其他 HSM 的日志中没有相应的事件。

示例：导入密钥

此示例显示了在您将密钥导入集群的 HSM 中时，记录的审核日志事件。在此示例中，加密用户 (CU) 使用imSymKey命令将 AES 密钥导入 HSM。该命令使用密钥 6 作为包装密钥。

接收命令的 HSM 首先记录密钥 6（包装密钥）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
```

然后，HSM 记录一个 CN_UNWRAP_KEY 事件，该事件表示导入操作。将为导入的密钥分配密钥句柄 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
```

在生成或导入新密钥时，客户端工具会自动尝试将新密钥同步到集群中的其他 HSM。在此情况下，HSM 会在从 HSM 中将密钥 11 作为遮蔽对象提取时，记录一个 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
```

集群中的其他 HSM 的日志流反映了新导入的密钥已到达。

例如，已将此事件记录在同一个集群中的其他 HSM 的日志流中。此 CN_INSERT_MASKED_OBJECT_USER 事件记录表示密钥 11 的遮蔽对象已到达。

```
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
```

示例：共享和取消共享密钥

此示例显示当加密用户 (CU) 与其他加密用户共享或取消共享 ECC 私有密钥时记录的审计日志事件。CU 使用 shareKey 命令并提供密钥句柄、用户 ID 和值 1 来共享密钥（或值 0 来取消共享密钥）。

在以下示例中，接收命令的 HSM 将记录一个表示共享操作的 CM_SHARE_OBJECT 事件。

```
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 审核日志参考

AWS CloudHSM 在审核日志事件中记录 HSM 管理命令。每个事件均有一个操作代码 (0pcode) 值，该值标识已发生的操作及其响应。您可以使用 0pcode 值来搜索、排序和筛选日志。

下表定义了 AWS CloudHSM 审计日志中的0pcode值。

操作代码 (Opcode)	描述
用户登录：这些事件包含用户名和用户类型	
CN_LOGIN (0xd)	用户登录
CN_LOGOUT (0xe)	用户注销
CN_APP_FINALIZE	与 HSM 的连接已关闭。此连接中的所有会话密钥或法定令牌均已删除。
CN_CLOSE_SESSION	与 HSM 的会话已经结束。此会话中的所有会话密钥或法定令牌均已删除。
用户管理：这些事件包含用户名和用户类型。	
CN_CREATE_USER (0x3)	创建加密用户 (CU)

操作代码 (Opcode)	描述
CN_CREATE_CO	创建加密员 (CO)
CN_DELETE_USER	删除用户
CN_CHANGE_PSWD	更改用户密码
CN_SET_M_VALUE	为用户操作设置法定身份验证（M of N）
CN_APPROVE_TOKEN	批准用户操作的法定身份验证令牌
CN_DELETE_TOKEN	删除一个或多个法定代币
CN_GET_TOKEN	请求签名令牌以启动法定人数操作
密钥管理：这些事件包括密钥处理程序	
CN_GENERATE_KEY	生成对称密钥
CN_GENERATE_KEY_PAIR (0x19)	生成非对称密钥对
CN_CREATE_OBJECT	导入公有密钥（无包装）
CN_MODIFY_OBJECT	设置关键属性
CN_DESTROY_OBJECT (0x11)	删除会话密钥
CN_TOMBSTONE_OBJECT	删除令牌密钥
CN_SHARE_OBJECT	共享或取消共享密钥
CN_WRAP_KEY	导出密钥的加密副本 (wrapKey)
CN_UNWRAP_KEY	导入密钥的加密副本 (unwrapKey)
CN_DERIVE_KEY	从现有密钥派生对称密钥
CN_NIST_AES_WRAP	使用 AES 密钥加密或解密密钥
CN_INSERT_MASKED_OBJECT_USER	在集群中插入带有其他 HSM 属性的加密密钥。

操作代码 (Opcode)	描述
CN_EXTRACT_MASKED_OBJECT_USER	使用来自 HSM 的属性对密钥进行封装/加密，以发送到集群中的另一个 HSM。
Back up HSMs	
CN_BACKUP_BEGIN	开始备份过程
CN_BACKUP_END	已完成备份过程
CN_RESTORE_BEGIN	开始从备份中恢复
CN_RESTORE_END	已完成从备份中恢复的过程
Certificate-Based Authentication	
CN_CERT_AUTH_STORE_CERT	存储集群证书
HSM Instance Commands	
CN_INIT_TOKEN (0x1)	启动 HSM 初始化过程
CN_INIT_DONE	HSM 初始化过程已完成
CN_GEN_KEY_ENC_KEY	生成密钥加密密钥 (KEK)
CN_GEN_PSWD_ENC_KEY (0x1d)	生成密码加密密钥 (PEK)
HSM crypto commands	
CN_FIPS_RAND	生成符合 FIPS 标准的随机数

正在获取 CloudWatch 以下各项的指标 AWS CloudHSM

CloudWatch 用于实时监控您的 AWS CloudHSM 集群。指标可按区域、集群 ID 或集群 ID 和 HSM ID 分组。

AWS/CloudHSM 命名空间包括以下指标：

指标	描述
HsmUnhealthy	HSM 实例运行不正常。 AWS CloudHSM 自动为您替换运行状况不佳的实例。当我们替换 HSM 时，您可以选择主动地扩展集群大小以降低性能影响。
HsmTemperature [1]	硬件处理器的联接温度。如果温度达到 110 摄氏度，系统将关闭。
HsmKeysSessionOccupied	正由 HSM 实例使用的会话密钥的数量。
HsmKeysTokenOccupied	正由 HSM 实例和集群使用的令牌密钥的数量。
HsmSslCtxsOccupied [1]	当前为 HSM 实例建立的 end-to-end 加密通道数量。最多允许 2,048 个通道。
HsmSessionCount	与 HSM 实例的打开连接的数量。最多允许 2,048 个。默认情况下，客户端守护程序配置为打开两个会话，每个 HSM 实例都位于一个 end-to-end 加密通道下。 AWS CloudHSM 还可以打开最多 2 个与 HSM 的连接，以监控 HSM 的运行状况。
HsmUsersAvailable	可创建的额外用户的数量。这等于最大用户数（列在中 HsmUsersMax）减去迄今为止创建的用户数。
HsmUsersMax [1]	可在 HSM 实例上创建的最大用户数。目前，此数量为 1,024。
InterfaceEth2OctetsInput [1]	至今传入到 HSM 的累积流量总和。
InterfaceEth2OctetsOutput [1]	至今流向 HSM 的流量的累积和。

- [1] 此指标不适用于 hsm2m.medium。

AWS CloudHSM 性能

对于生产集群，您应跨一个区域中的不同可用区分布至少两个 HSM 实例。我们建议您对集群执行负载测试，以确定您的峰值负载预期，然后添加 HSM 以确保高可用性。对于需要新生成密钥持久性的应用程序，我们建议跨一个区域中的不同可用区分布至少三个 HSM 实例。

性能数据

AWS CloudHSM 集群的性能因具体工作负载而异。要提高性能，您可以向集群中添加更多 HSM 实例。性能可能因配置、数据大小以及 EC2 实例上的额外应用程序负载而异。我们鼓励对应用程序进行负载测试，以确定扩展需求。

下表显示了在带有 hsm1.medium 实例的 EC2 实例上运行的常见加密算法的近似性能。

hsm1.medium 的性能数据

操作	双 HSM 集群 [1]	三 HSM 集群 [2]	六 HSM 集群 [3]
RSA 2048 位签名	每秒 2000 次操作	每秒 3000 次操作	每秒 5000 次操作
EC P256 签名	每秒 500 次操作	每秒 750 次操作	每秒 1500 次操作

- [1] 带有 Java 多线程应用程序的双 HSM 集群在一个 c4.large EC2 实例上运行，其中一个 HSM 与 EC2 实例位于同一个可用区。
- [2] 带有 Java 多线程应用程序的三 HSM 集群在一个 c4.large EC2 实例上运行，其中一个 HSM 与 EC2 实例位于同一个可用区。
- [3] 带有 Java 多线程应用程序的六 HSM 集群在一个 c4.large EC2 实例上运行，其中两个 HSM 与 EC2 实例位于同一个可用区。

HSM 节流

当您的工作负载超过集群的 HSM 容量时，您将收到报错消息，指出 HSM 忙碌或已被节流。关于此情况的应对操作，请参阅 HSM 节流

安全性 AWS CloudHSM

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于的合规计划 AWS CloudHSM，请参阅按合规计划划分的 AWS 范围内服务 AWS 按合规计划。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用时如何应用分担责任模型 AWS CloudHSM。以下主题向您介绍如何进行配置 AWS CloudHSM 以满足您的安全和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 AWS CloudHSM 资源。

内容

中的数据保护 AWS CloudHSM

分 AWS 担责任模型适用于中的数据保护 AWS CloudHSM。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础架构上的内容的控制。您还负责您所使用的 AWS 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与资源通信。 AWS 我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API AWS CloudHSM 或 SDK 或以其他 AWS 服务 方式使用控制台 AWS CLI、API 或 AWS SDK 的情况。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供网址，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

静态加密

从 HSM AWS CloudHSM 进行备份时，HSM 会先对其数据进行加密，然后再将其发送到。 AWS CloudHSM使用唯一的临时加密密钥对数据进行加密。有关更多信息，请参阅 AWS CloudHSM 集群备份。

传输中加密

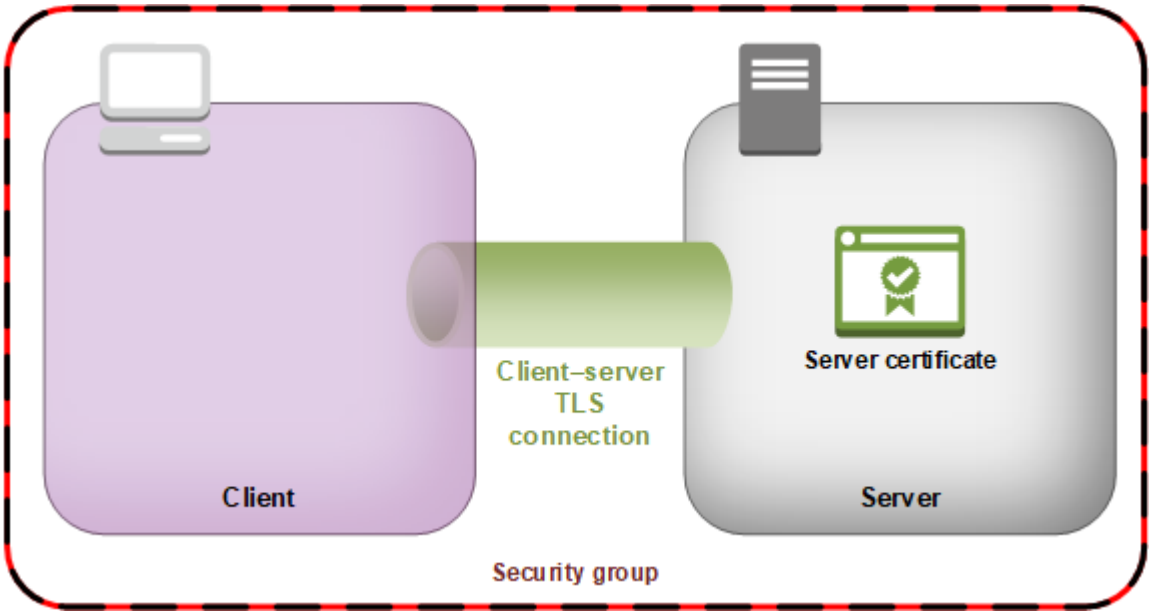
AWS CloudHSM 客户端与集群中的 HSM 之间的通信是端到端加密的。此通信只能由您的客户端和 HSM 进行解密。有关更多信息，请参阅 E nd-to-end 加密。

AWS CloudHSM 客户端 end-to-end 加密

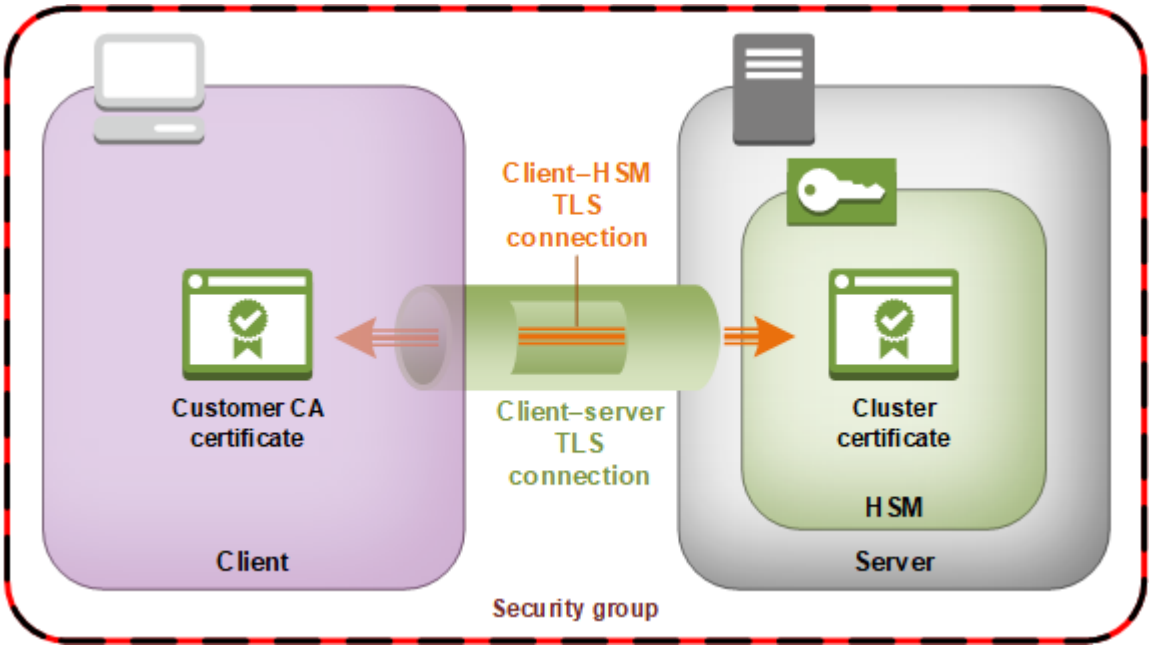
客户端实例和集群中 HSM 之间的通信已接受端到端加密。只有您的客户端和 HSM 可以对通信进行解密。

以下过程说明了客户端如何与 HSM 建立 end-to-end 加密通信。

1. 您的客户端与托管 HSM 硬件的服务器建立传输层安全性 (TLS) 连接。您的集群的安全组仅允许从该安全组中的客户端实例发往服务器的入站流量。该客户端还会检查服务器的证书以确保服务器可信。



2. 接下来，该客户端会与 HSM 硬件建立加密连接。HSM 具有您使用自己的证书颁发机构 (CA) 签名的集群证书，客户端具有 CA 的根证书。在建立客户端与 HSM 之间的加密连接之前，客户端会根据 HSM 的根证书验证其集群证书。只有当客户端成功验证 HSM 可信时，才会建立此连接。



集群备份安全性

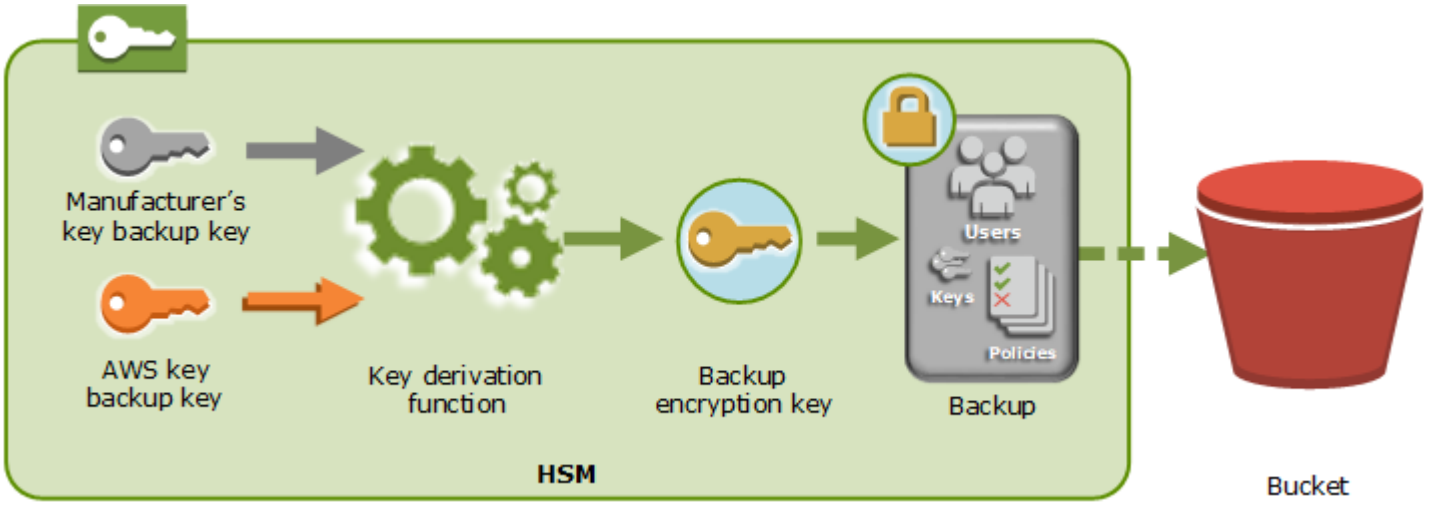
从 HSM AWS CloudHSM 进行备份时，HSM 会先对其所有数据进行加密，然后再将其发送到。 AWS CloudHSM数据绝不会以明文形式离开 HSM。此外，无法通过解密备份， AWS 因为 AWS 无法访问用于解密备份的密钥。

为了对其数据进行加密，HSM 将使用称为“临时备份密钥 (EBK)”的唯一的临时加密密钥。EBK 是进行备份时 AWS CloudHSM 在 HSM 内部生成的 AES 256 位加密密钥。HSM 将生成 EBK，然后借助经 FIPS 批准的 AES 密钥包装方法 (此方法符合 NIST 特刊 800-38F) 通过 EBK 加密 HSM 的数据。然后 HSM 将加密的数据提供给。 AWS CloudHSM加密的数据包括 EBK 的加密副本。

为了对 EBK 进行加密，HSM 使用另一个名为“永久备份密钥”(PBK) 的加密密钥。PBK 也是 AES 256 位加密密钥。为了生成 PBK，HSM 将在符合 NIST 特刊 800-108 的计数器模式下使用经 FIPS 批准的密钥派生函数 (KDF)。此 KDF 的输入包括：

- 一个制造商密钥备份密钥 (MKBK)，此密钥永久嵌入在硬件制造商提供的 HSM 硬件中。
- AWS 密钥备份密钥 (AKBK)，最初由配置时安全地安装在 HSM 中。 AWS CloudHSM

下图中总结了加密流程。备份加密密钥表示永久备份密钥 (PBK) 和临时备份密钥 (EBK)。



AWS CloudHSM 可以将备份还原到同一制造商制造的仅 AWS属于自己的 HSM 上。由于每个备份包含原始 HSM 中的所有用户、密钥和配置，因此，还原的 HSM 包含与原始 HSM 相同的保护和访问控制。还原的数据将覆盖在还原之前可能已位于 HSM 上的所有其他数据。

备份只包含加密数据。在该服务将备份存储在 Amazon S3 中之前，该服务会使用 AWS Key Management Service (AWS KMS) 再次加密备份。

的身份和访问管理 AWS CloudHSM

AWS 使用安全凭证来识别您的身份并向您授予对 AWS 资源的访问权。您可以使用 AWS Identity and Access Management (IAM) 的功能来允许其他用户、服务和应用程序完全或以有限的方式使用您的 AWS 资源。您可以在不共享安全凭证的情况下执行此操作。

默认情况下，IAM 用户没有创建、查看或修改 AWS 资源的权限。要允许 IAM 用户访问资源（如负载均衡器）并执行任务，您可以：

1. 创建授予 IAM 用户使用所需特定资源和 API 操作的权限的 IAM 策略。
2. 将该策略附加到 IAM 用户或 IAM 用户所属的组。

在将策略附加到一个用户或一组用户时，它会授权或拒绝用户使用指定资源执行指定任务。

例如，您可以使用 IAM 在您的 Amazon Web Services account 下创建用户和组。IAM 用户可以是人员、系统或应用程序。然后，使用 IAM 策略向用户和组授予对指定资源执行特定操作的权限。

使用 IAM policy 授予权限

在将策略附加到一个用户或一组用户时，它会授权或拒绝用户使用指定资源执行指定任务。

IAM 策略是包含一个或多个语句的 JSON 文档。每个语句的结构如下例所示。

```
{
  "Version": "2012-10-17",
  "Statement":[{
    "Effect": "effect",
    "Action": "action",
    "Resource": "resource-arn",
    "Condition": {
      "condition": {
        "key":"value"
      }
    }
  }]
}
```

- 效果 - 效果可以是 `Allow` 或 `Deny`。在默认情况下，IAM 用户没有使用资源和 API 操作的许可，因此，所有请求均会被拒绝。显式允许将覆盖默认规则。显式拒绝将覆盖任何允许。

- 操作：操作是对其授予或拒绝权限的特定 API 操作。有关指定 操作的更多信息，请参阅适用于 API 的操作 AWS CloudHSM。
- 资源-受操作影响的资源。 AWS CloudHSM 不支持资源级权限。必须使用* 通配符来指定所有 AWS CloudHSM 资源。
- 条件—— 您可以选择性地使用条件来控制策略的生效时间。有关更多信息，请参阅 的条件键 AWS CloudHSM。

有关更多信息，请参阅 IAM 用户指南。

适用于 API 的操作 AWS CloudHSM

在 IAM 策略声明的操作元素中，您可以指定 AWS CloudHSM 提供的任何 API 操作。如以下示例所示，您必须使用小写形式的字符串 `cloudhsm:` 作为操作名称的前缀。

```
"Action": "cloudhsm:DescribeClusters"
```

要在单个语句中指定多项操作，请使用方括号将操作括起来并以逗号分隔，如以下示例所示。

```
"Action": [
    "cloudhsm:DescribeClusters",
    "cloudhsm:DescribeHsm"
]
```

您也可以使用 * 通配符指定多项操作。以下示例指定了以开头 AWS CloudHSM 的所有 API 操作名称`List`。

```
"Action": "cloudhsm:List*"
```

要为指定所有 API 操作 AWS CloudHSM，请使用* 通配符，如以下示例所示。

```
"Action": "cloudhsm:*"
```

有关的 API 操作列表 AWS CloudHSM，请参阅AWS CloudHSM 操作。

的条件键 AWS CloudHSM

在创建策略时，您可指定控制策略生效时间的条件。每个条件都包含一个或多个键值对。有全局条件键和特定于服务的条件键。

AWS CloudHSM 没有特定于服务的上下文密钥。

有关全局条件键的更多信息，请参阅 IAM 用户指南中的 AWS 全球条件上下文键。

预定义的 AWS 托管策略 AWS CloudHSM

AWS 创建的托管策略将授予针对常用案例的必要权限。您可以根据您的 IAM 用户对 AWS CloudHSM 所需的访问权限将这些策略附加到这些用户：

- AWSCloudHSMFullAccess— 授予使用 AWS CloudHSM 功能所需的完全访问权限。
- AWSCloudHSMReadOnlyAccess— 授予对 AWS CloudHSM 功能的只读访问权限。

客户管理的政策 AWS CloudHSM

我们建议您为其创建仅 AWS CloudHSM 包含运行所需权限的 IAM 管理员组 AWS CloudHSM。将具有适当权限的策略附加到此组。根据需要向组添加 IAM 用户。添加的每个用户将继承管理员组的策略。

此外，我们建议您根据用户所需的权限来创建其他用户组。这将确保仅受信任的用户能够访问关键 API 操作。例如，您可以创建用于授予对集群和 HSM 的只读访问权限的用户组。由于此组不允许用户删除集群或 HSM，因此，不受信任的用户不能影响生产工作负载的可用性。

随着时间的推移，新的 AWS CloudHSM 管理功能不断增加，您可以确保只有受信任的用户才能立即获得访问权限。通过在创建时将有限权限分配给策略，您可以在以后手动为它们分配新的功能权限。

以下是的策略示例 AWS CloudHSM。有关如何创建策略并将其附在 IAM 用户组上的信息，请参阅 IAM 用户指南中的在 JSON 选项卡上创建策略。

示例

- 只读权限
- 高级用户权限
- 管理员权限

Example 示例：只读权限

此策略允许访问 `DescribeClusters` 和 `DescribeBackups` API 操作。它还包括特定 Amazon EC2 API 操作的其他权限。它不允许用户删除集群或 HSM。

```
{
```

```
   "Version": "2012-10-17",
   "Statement": {
      "Effect": "Allow",
      "Action": [
         "cloudhsm:DescribeClusters",
         "cloudhsm:DescribeBackups",
         "cloudhsm:ListTags"
      ],
      "Resource": "*"
   }
}
```

Example 示例：高级用户权限

此策略允许访问一部分 AWS CloudHSM API 操作。它还包括特定 Amazon EC2 操作的其他权限。它不允许用户删除集群或 HSM。您必须包括 AWS CloudHSM 允许在您的账户中自动创建AWSServiceRoleForCloudHSM服务相关角色的`iam:CreateServiceLinkedRole`操作。此角色 AWS CloudHSM 允许记录事件。有关更多信息，请参阅 的服务相关角色 AWS CloudHSM。

> **Note**
>
> 有关每种 API 的特定权限，请参阅服务授权参考中的AWS CloudHSM的操作、资源和条件键表。

```
{
   "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 示例：管理员权限

此策略允许访问所有 AWS CloudHSM API 操作，包括删除 HSM 和集群的操作。它还包括特定 Amazon EC2 操作的其他权限。您必须包括 AWS CloudHSM 允许在您的账户中自动创建AWSServiceRoleForCloudHSM服务相关角色的`iam:CreateServiceLinkedRole`操作。此角色 AWS CloudHSM 允许记录事件。有关更多信息，请参阅 的服务相关角色 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":"*"
    }
}
```

的服务相关角色 AWS CloudHSM

您之前创建的 IAM 策略客户管理的政策 AWS CloudHSM包含
该iam:CreateServiceLinkedRole操作。 AWS CloudHSM 定义
名AWSServiceRoleForCloudHSM为的服务相关角色。该角色由预定义 AWS CloudHSM ，包括代表您调用其他 AWS 服务 AWS CloudHSM 所需的权限。通过该角色可以更轻松地设置您的服务，因为您无需手动添加角色策略和信任策略权限。

角色策略 AWS CloudHSM 允许创建 Amazon Log CloudWatch s 日志组和日志流，并代表您写入日志事件。您可以在下面以及 IAM 控制台中查看该策略。

```
{
    "Version": "2018-06-12",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

该AWSServiceRoleForCloudHSM角色的信任策略 AWS CloudHSM 允许代入该角色。

```
{
  "Version": "2018-06-12",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudhsm.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

创建服务相关角色 (自动)

AWS CloudHSM 如果您在创建 AWS CloudHSM 管理员组时定义的权限中包含该`iam:CreateServiceLinkedRole`操作，则会在创建集群时创建该AWSServiceRoleForCloudHSM角色。请参阅 客户管理的政策 AWS CloudHSM。

如果您已经有一个或多个集群并且只想添加AWSServiceRoleForCloudHSM角色，则可以使用控制台、create-cluster 命令或 CreateClusterAPI 操作来创建集群。然后使用控制台、delete-cluster 命令或DeleteClusterAPI 操作将其删除。创建新集群时，会创建服务相关角色并将其应用到您的账户中的所有集群。或者，您可以手动创建该角色。有关更多信息，请参阅以下部分。

> **Note**
>
> 如果您只是为了添加AWSServiceRoleForCloudHSM角色而创建集群，则无需执行中入门AWS CloudHSM概述的所有步骤来创建集群。

创建服务相关角色 (手动)

您可以使用 IAM 控制台或 API 来创建AWSServiceRoleForCloudHSM角色。 AWS CLI有关更多信息，请参阅《IAM 用户指南》中的创建服务相关角色。

编辑服务相关角色

AWS CloudHSM 不允许您编辑AWSServiceRoleForCloudHSM角色。例如，在创建该角色后，您无法更改其名称，因为可能有不同的实体使用名称来引用该角色。此外，您无法更改角色策略。不过，您可以使用 IAM 编辑角色描述。有关更多信息，请参见 IAM 用户指南中的编辑服务相关角色。

删除 服务相关角色

只要服务相关角色应用于的集群仍然存在，您就无法删除该角色。要删除该角色，必须先删除集群中的每个 HSM，然后删除集群。必须删除您的账户中的每个集群。然后，您可以使用 IAM 控制台或 API 删除该角色。 AWS CLI有关删除集群的更多信息，请参阅删除集 AWS CloudHSM 群。有关更多信息，请参见《IAM 用户指南》中的删除服务相关角色。

合规

对于处于 FIPS 模式的集群， AWS CloudHSM 提供符合 PCI-PIN、PCI-3DS 和 SOC2 合规性要求的 FIPS 批准的 HSM。 AWS CloudHSM 还为客户提供了选择非 FIPS 模式集群的选项。有关适用于每种认证和合规要求的详细信息，请参阅AWS CloudHSM 集群模式和 HSM 类型。

依靠 FIPS 验证的 HSM 可以帮助您满足云端数据安全的企业、合同和监管合规要求。 AWS

FIPS 140-2 合规性

联邦信息处理标准 (FIPS) 第 140-2 版是美国政府安全标准，其中规定了对保护敏感信息的加密模块的安全要求。提供的 hsm1.medium 类型 HSM AWS CloudHSM 已通过 FIPS 140-2 3 级认证（证书 #4218）。有关更多信息，请参阅硬件 FIPS 验证。

PCI DSS 合规性

支付卡行业数据信息安全标准 (PCI DSS) 是一种由 PCI 安全标准协会管理的专有信息安全标准。提供的 HSM AWS CloudHSM 符合 PCI DSS。

PCI PIN 合规

PCI PIN 为传输、处理和管理个人识别码 (PIN) 数据提供了安全要求和评估标准，这些数据用于在 ATM 和 point-of-sale (POS) 终端进行交易。自 2023 年 1 月以来，提供的 hsm1.medium 密码机 AWS CloudHSM 一直符合 PCI PIN 标准。有关更多信息，请参阅文章 AWS CloudHSM 现已获得 PCI PIN 认证。

PCI-3DS 合规性

PCI 3DS (或 Three Domain Secure、3-D Secure) 保护 EMV 3D 安全电子商务支付的数据安全。PCI 3DS 为在线购物提供了另一层安全保护。提供的 hsm1.medium 类型 HSM 符合 PCI-3DS 标准。 AWS CloudHSM

SOC2

SOC2 框架旨在帮助服务组织展示其云和数据中心安全控制措施。AWS CloudHSM 已在关键领域实施 SOC2 控制措施，以遵守信任服务原则。有关更多信息，请参阅 AWS SOC 常见问题页面。

AWS CloudHSM PCI-PIN 合规性常见问题解答

PCI PIN 为传输、处理和管理个人识别码 (PIN) 数据提供了安全要求和评估标准，这些数据用于在 ATM 和 point-of-sale (POS) 终端进行交易。

客户可通过 AWS Artifact（按需获得 AWS 合规报告的自助门户）获得 PCI-PIN 合规证明 (AOC) 和责任摘要。若要了解更多信息，请在 Amazon Web Services Management Console中登录 AWS Artifact，或参见 AWS Artifact 入门以了解更多信息。

常见问题解答

问：什么是《合规证明和责任摘要》？

合规证明 (AOC) 由合格的 PIN 评估员 (QPA) 制作，证明 AWS CloudHSM 符合 PCI-PIN 标准中的适用控制措施。责任摘要矩阵描述了控制措施，这些控制措施是客户 AWS CloudHSM 及其客户的各自责任。

问：如何获得合规性 AWS CloudHSM 证明？

客户可通过 AWS Artifact（按需获得 AWS 合规报告的自助门户）获得 PCI-PIN 合规证明 (AOC)。若要了解更多信息，请在 Amazon Web Services Management Console中登录 AWS Artifact，或参见 AWS Artifact 入门以了解更多信息。

问：我怎样才能知道我负责哪些 PCI PIN 控制措施？

有关详细信息，请参阅 AWS AWS CloudHSM PCI PIN 合规包中的 “PCI PIN 责任摘要”，该文件可通过自助门户 AWS Artifact 向客户提供，AWS Artifact 是一个按需访问 AWS 合规报告的自助门户。若要了解更多信息，请在 Amazon Web Services Management Console中登录 AWS Artifact，或参见 AWS Artifact 入门以了解更多信息。

问：作为 AWS CloudHSM 客户，我能否信赖 PCI-PIN 合规认证 (AOC)？

客户必须负责管理自己的 PCI-PIN 合规。您需要由符合资质的 PIN 评估员 (QPA) 通过正式的 PCI-PIN 认证流程，以验证您的支付工作负载是否满足所有 PCI-PIN 控制/要求。但是，对于由 AWS 负责的控制措施，您的 QPA 无需进一步 AWS CloudHSM 测试即可依赖合规性证明 (AOC)。

问：是否 AWS CloudHSM 负责与密钥管理生命周期相关的 PCI-PIN 要求？

AWS CloudHSM 负责 HSM 的物理设备生命周期。客户负责使密钥管理生命周期要求符合 PCI-PIN 标准。

问：哪些 AWS CloudHSM 控件符合 PCI-PIN？

AOC 总结了由 QPA 评估的 AWS CloudHSM 控制措施。客户可通过 AWS Artifact（按需获得 AWS 合规报告的自助门户）获得 PCI-PIN 责任摘要。

问：是否 AWS CloudHSM 支持 PIN 翻译和 DUKPT 等支付功能？

不是， AWS CloudHSM 提供通用的 HSM。随着时间的推移，我们可能会提供付款功能。尽管该服务不直接执行支付功能，但 AWS CloudHSM PCI PIN合规性证明使客户能够为其正在运行的服务实现自己的PCI合规性。 AWS CloudHSM如果您有兴趣使用 AWS Payment Cryptography 服务处理您的工作负载，请参阅博客“使用 AWS 支付加密将付款处理迁移至云端”。

弃用通知

为了保持符合 FIPS 140、PCI-DSS、PCI-PIN、PCI-3DS 和 SOC2 的要求， AWS CloudHSM 可能会不时弃用功能。此页面列出了当前适用的更改项。

FIPS 140 合规：2024 年机制弃用

美国国家标准与技术研究所 (NIST)[1]建议，在 2023 年 12 月 31 日之后，不允许支持 Triple DES (deSede、3DES、DES3) 加密，亦不允许使用 PKCS #1 v1.5 填充进行 RSA 密钥包装和解包。因此，我们的联邦信息处理标准 (FIPS) 模式集群对这些集群的支持将于 2024 年 1 月 1 日结束。对于非 FIPS 模式下的集群，仍然支持这些集群。

本指南适用于以下形式加密操作：

- 生成三重 DES 密钥
  - 适用于 PKCS#11 库的 `CKM_DES3_KEY_GEN`
  - 适用于 JCE 提供程序的 `DESede Keygen`
  - 带 `-t=21` 的 KMU `genSymKey`
- 通过三重 DES 密钥加密（注意：允许解密操作）
  - 对于 PKCS #11 库：`CKM_DES3_CBC` 加密、`CKM_DES3_CBC_PAD` 加密和`CKM_DES3_ECB` 加密
  - 对于 JCE 提供程序：`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 填充对 RSA 密钥进行包装、解包、加密和解密
  - `CKM_RSA_PKCS` 对 PKCS #11 SDK 进行包装、解包、加密和解密
  - `RSA/ECB/PKCS1Padding` 对 JCE SDK 进行包装、解包、加密和解密

- 带 `-m 12` 的 `wrapKey` 和 `unWrapKey` KMU 加上（注释 12 是机制 `RSA_PKCS` 的值）

[1] 有关此更改的详细信息，请参阅过渡使用加密算法和密钥长度中的表 1 和表 5。

韧性在 AWS CloudHSM

AWS 全球基础设施是围绕 AWS 区域和可用区构建的。 AWS 区域提供多个物理隔离和隔离的可用区，这些可用区通过低延迟、高吞吐量和高度冗余的网络相连。利用可用区，您可以设计和操作在可用区之间无中断地自动实现失效转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅AWS 全球基础设施。有关用于支持弹性的 AWS CloudHSM 功能的更多信息，请参阅集群高可用性和负载均衡。

中的基础设施安全 AWS CloudHSM

作为一项托管服务， AWS CloudHSM 受到《Amazon Web Services：安全流程概述》白皮书中描述的 AWS 全球网络安全程序的保护。

您可以使用 AWS 已发布的 API 调用 AWS CloudHSM 通过网络进行访问。此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

网络隔离

虚拟私有云（VPC）是 Amazon Web Services Cloud 内您自己的逻辑隔离区域中的虚拟网络。可以在 VPC 的私有子网中创建集群。创建 VPC 时，您可创建私有子网。有关更多信息，请参阅 创建虚拟私有云（VPC）。

创建 HSM 时，请在子网中 AWS CloudHSM 放置一个弹性网络接口 (ENI)，这样您就可以与 HSM 进行交互。有关更多信息，请参阅 集群架构。

AWS CloudHSM 创建安全组，允许集群中 HSM 之间的入站和出站通信。可以使用此安全组来允许 EC2 实例与集群中的 HSM 进行通信。有关更多信息，请参阅 配置客户端 Amazon EC2 实例安全组。

用户授权

使用 AWS CloudHSM，在 HSM 上执行的操作需要经过身份验证的 HSM 用户的证书。有关更多信息，请参阅 the section called “了解 HSM 用户”。

AWS CloudHSM 和 VPC 终端节点

您可以通过创建接口 VPC 终端节点在您 AWS CloudHSM 的 VPC 之间建立私有连接。接口终端节点由一项技术提供支持 AWS PrivateLink，该技术使您无需互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接即可私密访问 AWS CloudHSM API。您的 VPC 中的实例不需要公有 IP 地址即可与 AWS CloudHSM API 通信。VPC 和 AWS CloudHSM 之间的流量不会脱离 Amazon 网络。

每个接口端点均由子网中的一个或多个弹性网络接口表示。

有关更多信息，请参阅 A mazon VPC 用户指南中的接口 VPC 终端节点 (AWS PrivateLink)。

AWS CloudHSM VPC 终端节点的注意事项

在为设置接口 VPC 终端节点之前 AWS CloudHSM，请务必查看 Amazon VPC 用户指南中的接口终端节点属性和限制。

- AWS CloudHSM 支持从您的 VPC 调用其所有 API 操作。

为 AWS CloudHSM创建接口 VPC 端点

您可以使用 Amazon VPC 控制台或 AWS Command Line Interface (AWS CLI) 为 AWS CloudHSM 服务创建 VPC 终端节点。有关更多信息，请参阅《Amazon VPC 用户指南》中的创建接口端点。

要为创建 VPC 终端节点 AWS CloudHSM，请使用以下服务名称：

```
com.amazonaws.region.cloudhsmv2
```

例如，在美国西部（俄勒冈）区域 (us-west-2)，服务名称为：

```
com.amazonaws.us-west-2.cloudhsmv2
```

为了更轻松地使用 VPC 终端节点，您可以为 VPC 终端节点启用私有 DNS 主机名。如果您选择启用私有 DNS 名称选项，则标准 DN AWS CloudHSM S 主机名 (`https://cloudhsmv2.<region>.amazonaws.com`) 将解析到您的 VPC 终端节点。

此选项可让您更轻松地使用 VPC 终端节点。 AWS 软件开发工具包和默认 AWS CLI 使用标准 AWS CloudHSM DNS 主机名，因此您无需在应用程序和命令中指定 VPC 终端节点 URL。

有关更多信息，请参阅《Amazon VPC 用户指南》中的通过接口端点访问服务。

为创建 VPC 终端节点策略 AWS CloudHSM

您可以为 VPC 端点附加控制对 AWS CloudHSM的访问的端点策略。该策略指定以下信息：

- 可执行操作的主体。
- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 端点控制对服务的访问。

示例：用于 AWS CloudHSM 操作的 VPC 终端节点策略

以下是的终端节点策略示例 AWS CloudHSM。当连接到终端节点时，此策略授予所有委托人对所有资源 AWS CloudHSM 执行所列操作的访问权限。的身份和访问管理 AWS CloudHSM有关其他 AWS CloudHSM 操作及其相应的 IAM 权限，请参阅。

```
{
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "cloudhsm:DescribeBackups",
            "cloudhsm:DescribeClusters",
            "cloudhsm:ListTags",
         ],
         "Resource":"*"
      }
   ]
}
```

中的更新管理 AWS CloudHSM

AWS 会管理固件。固件由第三方维护，并且必须由 NIST 进行 FIPS 140-2 第 3 级合规性评估。只有由 FIPS 密钥加密签署的固件（AWS 对其无访问权限）才符合安装条件。

故障排除 AWS CloudHSM

如果您遇到问题 AWS CloudHSM，以下主题可以帮助您解决这些问题。

主题

已知问题

AWS CloudHSM 存在以下已知问题。选择一个主题以了解更多信息。

主题

所有 HSM 实例的已知问题

以下问题会影响所有 AWS CloudHSM 用户，无论他们使用的是 key_mgmt_util 命令行工具、PKCS #11 SDK、JCE SDK 还是 OpenSSL SDK。

主题

问题：AES 密钥包装使用 PKCS #5 填充，而不是提供与标准兼容的使用零填充的密钥包装

此外，不支持不使用填充和使用零填充的密钥包装。

- 影响：如果您在其中使用此算法进行 AWS CloudHSM封装和解包，则不会产生任何影响。但是，使用封装的密钥 AWS CloudHSM 不能在要求符合无填充规范的其他 HSM 或软件中解包。这是因为，在执行与标准兼容的解开包装操作期，您的密钥数据后面可能会添加 8 字节的填充数据。外部封装的密钥无法正确解封到实例中。 AWS CloudHSM
- 变通办法：要在外部对某个密钥解开包装，而该密钥是在 AWS CloudHSM 实例上使用 AES 密钥包装和 PKCS #5 填充进行包装的，则您必须先去除额外的填充，然后才能尝试使用该密钥。您可以通过以下方式执行此操作：在文件编辑器中修剪额外的字节，或者只将密钥字节复制到您代码中的新缓冲区。
- 解决状态：对于 3.1.0 客户端和软件版本， AWS CloudHSM 为 AES 密钥包装提供符合标准的选项。有关更多信息，请参阅 AES 密钥包装。

问题：客户端进程守护程序要求其配置文件中至少有一个有效的 IP 地址才能成功连接到集群

- 影响：如果您删除了集群中的所有 HSM，然后添加了获得新 IP 地址的另一个 HSM，客户端守护程序将继续在您的 HSM 的原始 IP 地址搜索它们。
- 解决办法：如果您运行间歇性工作负载，我们建议您使用CreateHsm函数中的`IpAddress`参数将 elastic network interface (ENI) 设置为其原始值。请注意，ENI 是特定于可用区 (AZ) 的。替代方法是删除 `/opt/cloudhsm/daemon/1/cluster.info` 文件，然后将客户端配置重置为新 HSM 的 IP 地址。您可以使用 `client -a <IP address>` 命令。有关更多信息，请参阅安装和配置 AWS CloudHSM 客户端 (Linux) 或安装和配置 AWS CloudHSM 客户端 (Windows)。

问题： AWS CloudHSM 使用客户端 SDK 3 可以进行哈希处理和签名的数据上限为 16 KB

- 解决状态：大小小于 16KB 的数据继续发送至 HSM 进行哈希处理。我们添加了在软件中对于大小介于 16KB 和 64KB 之间的数据本地进行哈希处理的功能。如果数据缓冲区大于 64KB，客户端 SDK 5 将明显失败。您必须将您的客户端和 SDK 更新到高于 5.0.0 或更高版本才能从此修复中受益。

问题：无法将导入的密钥指定为不可导出

- 解决状态：此问题已修复。您无需执行任何操作，即可从修复获益。

问题：wrapKey 的默认机制和 key_mgmt_util 中的 unWrapKey命令已被删除

- 解决方案：使用 WrapKey 或 unWrapKey 命令时，必须使用`-m`选项来指定机制。有关更多信息，请参阅 WrapKey 或unWrapKey文章中的示例。

问题：如果您在集群中有一个 HSM，HSM 失效转移将无法正常工作

- 影响: 如果集群中的单个 HSM 实例失去连接，客户端将不会与其重新连接，即使 HSM 实例稍后恢复。
- 解决方法:我们建议任意生产集群中至少有两个 HSM 实例。如果您使用此配置，将不会受此问题影响。对于单个 HSM 集群，退回客户端守护程序以恢复连接。
- 解决状态： 此问题已经在 AWS CloudHSM 客户端 1.1.2 发布版中解决。您必须升级到此客户端，才能从修复中获益。

问题：如果在短时间内超过集群中 HSM 的密钥容量，客户端会输入未处理的错误状态

- 影响: 当客户端遇到未处理的错误状态时会冻结，必须重新启动。
- Workaround: (解决方法) 测试您的吞吐量，以确保您未以客户端无法处理的速率创建会话密钥。您可以通过将 HSM 添加到集群降低您的速率，或减慢会话密钥创建过程。
- 解决状态： 此问题已经在 AWS CloudHSM 客户端 1.1.2 发布版中解决。您必须升级到此客户端，才能从修复中获益。

问题：大小大于 800 字节的 HMAC 密钥的摘要操作不受支持

- 影响: 大于 800 字节的 HMAC 密钥可以在 HSM 上生成或导入。但是，如果通过 JCE 或 key_mgmt_util 在摘要操作中使用这个较大的密钥，操作将失败。请注意，如果您使用的是 PKCS11，HMAC 密钥的大小限制为 64 个字节。
- 解决方法: 如果要使用 HSM 上摘要操作的 HMAC 密钥，请确保大小小于 800 字节。
- 解决状态：此时为无。

问题：与客户端软件开发工具包 3 一起分发的 client_info 工具会删除由可选输出参数指定的路径的内容

- 影响：指定输出路径下的所有现有文件和子目录可能会永远丢失。
- 变通办法：使用该 `client_info` 工具时请勿使用可选参数 `-output` *`path`*。
- 解决状态：此问题已在客户端软件开发工具包 3.3.2 发布版中解决。您必须升级到此客户端，才能从修复中获益。

问题：在容器化环境中使用 `--cluster-id` 参数运行 SDK 5 配置工具时收到错误

在配置工具中使用 --cluster-id 参数时，您会收到以下错误：

```
No credentials in the property bag
```

此错误由 Instance Metadata Service 版本 2 (IMDSv2, Instance Metadata Service Version 2) 的更新引起。有关更多信息，请参阅 IMDSv2 文档。

- 影响：此问题将影响在容器化环境中在 SDK 版本 5.5.0 及更高版本上运行配置工具以及利用 EC2 实例元数据提供凭证的用户。

- 变通办法：将 PUT 响应跃点限制设置为至少两个。有关执行此操作的指导，请参阅配置实例元数据选项。

问题：您收到错误 “无法从提供的 pfx 文件创建证书/密钥。错误： NotPkcs8 英寸

- 影响：使用证书和私钥重新配置 SSL 的 SDK 5.11.0 用户如果私钥不是 PKCS8 格式，则其私钥将失败。
- 解决方法：您可以使用 openssl 命令将自定义 SSL 私钥转换为 PKCS8 格式：`openssl pkcs8 -topk8 -inform PEM -outform PEM -in ssl_private_key -out ssl_private_key_pkcs8`
- 解决状态：此问题已在客户端 SDK 5.12.0 版本中得到解决。您必须升级到此客户端版本或更高版本才能从此修复中受益。

hsm2m.medium 实例的已知问题

以下问题会影响所有 hsm2m.medium 实例。

主题

- 问题：由于 PBKDF2 迭代次数增加，登录延迟会增加
- 问题：使用 Client SDK 5.12.0 及更早版本时，使用尝试设置密钥的可信属性的 CO 将失败

问题：由于 PBKDF2 迭代次数增加，登录延迟会增加

- 影响：为了提高安全性，hsm2m.medium 在登录请求期间对基于密码的密钥派生函数 2 (PBKDF2) 执行了 60,000 次迭代，而 hsm1.medium 中为 1,000 次。这种增加可能会导致每次登录请求的延迟增加多达 2 秒 (2 秒)。

  AWS CloudHSM 客户端 SDK 的默认超时时间为 20 秒。登录请求可能会超时并导致错误。
- 解决办法：如果可能，在同一个应用程序中序列化登录请求，以避免在登录期间出现延迟。
- 解析状态：未来版本的 Client SDK 将增加登录请求的默认超时时间，以解决延迟增加的原因。

问题：使用 Client SDK 5.12.0 及更早版本时，使用尝试设置密钥的可信属性的 CO 将失败

- 影响：任何试图设置密钥的可信属性的 CO 用户都将收到一条错误消息，指出这一点`User type should be CO or CU`。
- 解决方案：客户端 SDK 的未来版本将解决此问题。更新将在我们的用户指南中公布。文档历史记录

PKCS#11 库的已知问题

主题

问题：PKCS #11 库 3.0.0 版中的 AES 密钥包装无法在使用前验证 IV

如果您指定长度小于 8 字节的 IV，则在使用之前会填充不可预知的字节。

> **Note**
>
> 这只会对使用 `CKM_AES_KEY_WRAP` 的 `C_WrapKey` 机制产生影响。

- 影响：如果您在 PKCS #11 库 3.0.0 版本中提供的 IV 小于 8 字节，则可能无法对密钥进行解包。
- 解决方法：

  - 我们强烈建议您升级到 PKCS #11 库 3.0.1 版或更高版本，以便在 AES 密钥包装期间正确地强制执行 IV 长度。修改包装代码以便传递 NULL IV，或者指定默认 IV `0xA6A6A6A6A6A6A6A6`。有关更多信息，请参阅长度不符合 AES 密钥包装的自定义 IV。
  - 如果您使用短于 8 字节的 IV，通过 PKCS #11 库 3.0.0 版来包装任何密钥，请联系我们以获取支持。
- 解决状态：此问题已在 PKCS #11 库 3.0.1 版中解决。要使用 AES 密钥包装来包装密钥，请指定 NULL 或长度为 8 字节的 IV。

问题：PKCS #11 SDK 2.0.4 以及更早版本始终使用 **0xA6A6A6A6A6A6A6A6** 的默认 IV 来执行 AES 密钥包装和解包

用户提供的 IV 被静默地忽略。

> **Note**
>
> 这只会对使用 `CKM_AES_KEY_WRAP` 的 `C_WrapKey` 机制产生影响。

- 影响：
  - 如果您使用 PKCS #11 开发工具包 2.0.4 或早期版本以及用户提供的 IV，则您的密钥将使用默认 IV `0xA6A6A6A6A6A6A6A6` 进行包装。
  - 如果您使用 PKCS #11 开发工具包 3.0.0 或更高版本以及用户提供的 IV，则您的密钥将使用用户提供的 IV 进行包装。
- 解决方法：
  - 要解包使用 PKCS #11 开发工具包 2.0.4 或更早版本包装的密钥，请使用默认 IV `0xA6A6A6A6A6A6A6A6`。
  - 要解包使用 PKCS #11 开发工具包 3.0.0 或更高版本包装的密钥，请使用用户提供的 IV。
- 解决状态：我们强烈建议您修改包装和解包代码，以便传递 NULL IV，或者指定默认 IV `0xA6A6A6A6A6A6A6A6`。

问题：不支持且不处理 **CKA_DERIVE** 属性

- 解决状态：我们已经实施修复，以便接受 `CKA_DERIVE`（如果它设置为 `FALSE`）。在我们开始为 AWS CloudHSM增加密钥派生函数支持之前，不支持将 `CKA_DERIVE` 设置为 `TRUE` 您必须将客户端和开发工具包更新至版本 1.1.1 或更高版本，才能从修复获益。

问题：不支持且不处理 **CKA_SENSITIVE** 属性

- 解决状态：我们已实施修改，以便接受并正确处理 CKA_SENSITIVE 属性。您必须将客户端和开发工具包更新至版本 1.1.1 或更高版本，才能从修复获益。

问题：不支持多部分哈希和签名

- 影响：C_DigestUpdate 和 C_DigestFinal 不会实施。C_SignFinal 也不会实施，并且对于非 NULL 缓冲区将会失败并显示 CKR_ARGUMENTS_BAD。
- 解决方法：在应用程序中对数据进行哈希处理， AWS CloudHSM 仅用于对哈希进行签名。
- 解决状态：我们正在修复客户端和开发工具包，以正确实施多部分哈希。将在 AWS CloudHSM 论坛和版本历史记录页面中公布更新。

问题：**C_GenerateKeyPair** 不以符合标准的方式处理私有模板中的 **CKA_MODULUS_BITS** 或 **CKA_PUBLIC_EXPONENT**

- 影响：C_GenerateKeyPair 应在私有模板包含 CKA_MODULUS_BITS 或 CKA_PUBLIC_EXPONENT 时返回 CKA_TEMPLATE_INCONSISTENT。相反，它生成了一个所有用法字段都设置为 FALSE 的私有密钥。无法使用该密钥。
- 解决方法：建议您的应用程序检查用法字段值以及错误代码。
- 解析状态：我们正在实施修复，以在使用了不正确的私有密钥模板时返回正确的错误消息。将在版本历史记录页面中公布更新后的 PKCS#11 库。

问题：在使用 **CKM_AES_GCM** 机制时，用于执行 **C_Encrypt** 和 **C_Decrypt** API 操作的缓冲区不能超过 16 KB

AWS CloudHSM 不支持多部分 AES-GCM 加密。

- 影响：您不能使用 CKM_AES_GCM 机制加密大于 16 KB 的数据。
- 变通办法：您可以使用一个替代机制 (如 CKM_AES_CBC、CKM_AES_CBC_PAD)，也可以将数据拆分为多个部分并用 AES_GCM 为各个部分分别加密。如果您正在使用AES_GCM，则必须管理数据的划分和随后的加密。 AWS CloudHSM 不会为您执行多部分 AES-GCM 加密。请注意，FIPS 要求在 HSM 上生成 AES-GCM 的初始化向量 (IV)。因此，您的 AES-GCM 加密数据的每个部分的 IV 将不同。

- 解决状态：我们正在修复开发工具包，以在数据缓冲区过大时显式失败。我们将为 `C_EncryptUpdate` 和 `C_DecryptUpdate` API 操作返回 `CKR_MECHANISM_INVALID`。我们正在评估替代方法来支持较大的缓冲区，而不依靠多部分加密。更新将在 AWS CloudHSM 论坛和版本历史记录页面上公布。

问题：在 HSM 内部分执行椭圆曲线迪菲-赫尔曼 (ECDH, Elliptic-curve Diffie-Hellman) 密钥派生

您的 EC 私有密钥始终保留在 HSM 中，但密钥派生过程分多步执行。因此，客户端上可以提供每个步骤的中间结果。

- 影响：在 Client SDK 3 中，使用该`CKM_ECDH1_DERIVE`机制派生的密钥首先在客户端上可用，然后导入到 HSM 中。密钥句柄随后会返回到您的应用程序。
- 解决办法：如果要在 AWS CloudHSM中实施 SSL/TLS 分载，此限制可能不是问题。如果您的应用程序需要将您的密钥始终保持在 FIPS 边界内，请考虑使用不依赖 ECDH 密钥派生的替代协议。
- 解决状态：我们正在开发完全在 HSM 内部执行 ECDH 密钥派生的选项。更新的实施将在版本历史记录页面中公布。

问题：在 Centos6 和 RHEL 6 等 EL6 平台上验证 secp256k1 签名失败

这是因为 CloudHSM PKCS#11 库通过使用 OpenSSL 来验证 EC 曲线数据，在初始化验证操作的过程中避免网络调用。由于在 EL6 平台上默认的 OpenSSL 包不支持 Secp256k1，因此初始化失败。

- 影响：Secp256k1 签名验证将会在 EL6 平台上失败。验证调用失败，并显示 `CKR_HOST_MEMORY` 错误。
- 解决方法：如果您的 PKCS#11 应用需要验证 secp256k1 签名，我们建议您使用 Amazon Linux 1 或任何 EL7 平台。或者，升级到支持 secp256k1 曲线的 OpenSSL 软件包版本。
- 解决状态：我们正在实施修复，如果本地曲线验证不可用，则回退到 HSM。将在版本历史记录页面中公布更新后的 PKCS#11 库。

问题：错误的函数调用时序会给出未定义的结果而不是导致失败

- 影响：如果您函数调用的顺序不正确，即使单个函数调用返回成功，最终结果也是不正确的。例如，解密后的数据可能与原始明文不匹配，或者签名可能无法验证。此问题会影响单部件和多部分操作。

  不正确的函数顺序示例：

- C_EncryptInit/C_EncryptUpdate 之后是 C_Encrypt
- C_DecryptInit/C_DecryptUpdate 之后是 C_Decrypt
- C_SignInit/C_SignUpdate 之后是 C_Sign
- C_VerifyInit/C_VerifyUpdate 之后是 C_Verify
- C_FindObjectsInit 之后是 C_FindObjectsInit

- 变通办法：您的应用程序应按照 PKCS #11 规范，对单部分和多部分操作使用正确的函数调用顺序。在这种情况下，您的应用程序不应依赖 CloudHSM PKCS #11 库返回错误。

问题：SDK 5 不支持只读会话

- 问题：SDK 5 不支持使用 C_OpenSession 打开只读会话。
- 影响：如果您在未提供 CKF_RW_SESSION 的情况下尝试调用 C_OpenSession，则调用将失败并显示错误 CKR_FUNCTION_FAILED。
- 解决办法：打开会话时，必须将 CKF_SERIAL_SESSION | CKF_RW_SESSION 标记传递给 C_OpenSession 函数调用。

问题：**cryptoki.h** 标头文件仅适用于 Windows

- 问题：在 Linux 上的 AWS CloudHSM Client SDK 5 版本 5.0.0 到 5.4.0 中，头文件/opt/cloudhsm/include/pkcs11/cryptoki.h仅与 Windows 操作系统兼容。
- 影响：当您在基于 Linux 的操作系统的应用程序中尝试包含此标头文件时，您可能会遇到问题。
- 解析状态：升级到 AWS CloudHSM Client SDK 5 版本 5.4.1 或更高版本，其中包括此头文件的 Linux 兼容版本。

JCE 开发工具包的已知问题

主题

- 问题：客户端软件开发工具包 5 抛出警告“发生了非法的反射访问操作”
- 问题：JCE 会话池已用尽
- 问题：GetKey 操作导致客户端 SDK 5 内存泄漏

问题：当使用非对称密钥对时，即使未显式创建或导入密钥，也会看到占用密钥容量

- 影响：此问题可能导致 HSM 意外地耗尽密钥空间，并且当您的应用程序使用标准 JCE 密钥对象进行加密操作而不是 CaviumKey 对象时，会发生此问题。当您使用标准 JCE 密钥对象时，CaviumProvider 会将该密钥作为会话密钥隐式导入 HSM 中，并且在应用程序退出之前不会删除此密钥。因此，密钥会在应用程序运行时累积，并可能导致 HSM 耗尽空闲密钥空间，从而导致应用程序不响应。
- 解决方法：当使用 CaviumSignature 类、 CaviumCipher 类、 CaviumMac 类或 CaviumKeyAgreement 类时，您应提供 CaviumKey 密钥而不是标准 JCE 密钥作为对象。

  您可以使用 ImportKey 类将普通密钥手动转换为 CaviumKey，并在之后可在操作完成后手动删除该密钥。
- 解决状态：我们正在更新 CaviumProvider 以正确管理隐式导入。当该修复可用时，将在版本历史记录页面中公布。

问题：JCE KeyStore 是只读的

- 影响：您目前无法将 HSM 不支持的对象类型存储在 JCE 密钥库中。具体来说，您不能在密钥存储中存储证书。这会阻止与 jarsigner 之类的工具的互操作性，这些工具预期在密钥库中查找证书。
- 解决方法：您可以修改您的代码，以从本地文件或 S3 存储桶位置中加载证书，而不是从密钥存储中加载。
- 解决状态：我们正在增加在密钥存储中存储证书的支持。当该功能可用时，将在版本历史记录页面中公布。

问题：要进行 AES-GCM 加密的缓冲区不能超过 16,000 字节

此外，多部分 AES-GCM 加密不受支持。

- 影响：您不能使用 AES-GCM 加密大于 16,000 字节的数据。

- 解决方法：您可以使用一个替代机制（如 AES-CBC），也可以将数据拆分为多个部分并为各个部分分别加密。如果您拆分了数据，则必须管理已拆分的密文及其解密内容。由于 FIPS 要求在 HSM 上生成 AES-GCM 的初始化向量 (IV)，因此每个 AES-GCM 加密的数据片段的 IV 将有所不同。
- 解决状态：我们正在修复开发工具包，以在数据缓冲区过大时显式失败。我们正在评估支持较大的缓冲区而不依靠多部分加密的替代方法。将在 AWS CloudHSM 论坛和版本历史记录页面中公布更新。

问题：在 HSM 内部分执行椭圆曲线迪菲-赫尔曼 (ECDH, Elliptic-curve Diffie-Hellman) 密钥派生

您的 EC 私有密钥始终保留在 HSM 中，但密钥派生过程分多步执行。因此，客户端上可以提供每个步骤的中间结果。Java 代码示例中提供了 ECDH 密钥派生示例。

- 影响：客户端 SDK 3 为 JCE 增加了 ECDH 功能。使用该`KeyAgreement`类派生 a 时 SecretKey，它首先在客户端上可用，然后导入到 HSM 中。密钥句柄随后会返回到您的应用程序。
- 解决办法：如果您在中实现 SSL/TLS 卸载 AWS CloudHSM，则此限制可能不是问题。如果您的应用程序需要将您的密钥始终保持在 FIPS 边界内，请考虑使用不依赖 ECDH 密钥派生的替代协议。
- 解决状态：我们正在开发完全在 HSM 内部执行 ECDH 密钥派生的选项。如果可用，我们将在版本历史记录页面上公布更新的实现。

问题： KeyGenerator 并且 KeyAttribute 错误地将密钥大小参数解释为字节数而不是位

当使用KeyGenerator 类的`init`函数或AWS CloudHSM KeyAttribute 枚举的`SIZE`属性生成密钥时，API 错误地期望参数为密钥字节数，而应改为密钥位数。

- 影响： 客户端软件开发工具包版本 5.4.0 至 5.4.2 错误地预期以字节形式提供给指定 API 的密钥大小。
- 解决办法：如果使用客户端 SDK 版本 5.4.0 到 5.4.2，则在使用 KeyGenerator 类或 KeyAttribute 枚举使用 AWS CloudHSM JCE 提供程序生成密钥之前，请将密钥大小从位转换为字节。
- 解析状态：将您的客户端 SDK 版本升级到 5.5.0 或更高版本，其中包括在使用 KeyGenerator 类或 KeyAttribute 枚举生成密钥时正确预期以位为单位的密钥大小的修复程序。

问题：客户端软件开发工具包 5 抛出警告“发生了非法的反射访问操作”

在 Java 11 中使用客户端软件开发工具包 5 时，CloudHSM 会抛出以下 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
```

这些警告没有任何影响。我们已经意识到这个问题，并正在努力解决它。无需任何解决方案或变通办法。

问题：JCE 会话池已用尽

影响：看到以下消息后，您可能无法在 JCE 中执行操作：

```
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
```

解决方法：

- 如果您受到影响，请重新启动 JCE 应用程序。
- 执行操作时，可能需要先完成 JCE 操作，然后才能丢失对该操作的引用。

> **Note**
>
> 根据操作的不同，可能需要一种完成方法。

操作	完成方法
密码	在加密或解密模式下的 `doFinal()` 在包装模式下的 `wrap()` 在解包模式下的 `unwrap()`

操作	完成方法
KeyAgreement	`generateSecret()` 或 `generateSecret(String)`
KeyPairGenerator	`generateKeyPair()` 、`genKeyPair()` 或 `reset()`
KeyStore	无需任何方法
MAC	`doFinal()` 或 `reset()`
MessageDigest	`digest()` 或 `reset()`
SecretKeyFactory	无需任何方法
SecureRandom	无需任何方法
签名	在签名模式下的 `sign()` 在验证模式下的 `verify()`

解决状态：我们已经在客户端软件开发工具包 5.9.0 及更高版本中解决了此问题。要修复此问题，请将您的客户端软件开发工具包升级到其中一个版本。

问题：GetKey 操作导致客户端 SDK 5 内存泄漏

- 影响：在客户端 SDK 版本 5.10.0 及更早版本中，API `getKey` 操作在 JCE 中存在内存泄漏。如果您在应用程序中多次使用 `getKey` API，则会导致内存增长增加，从而增加应用程序中的内存占用量。随着时间的推移，这可能会导致限制错误或需要重新启动应用程序。
- 解决办法：我们建议升级到客户端 SDK 5.11.0。如果无法做到这一点，我们建议不要在您的应用程序中多次调用 `getKey` API。相反，应尽可能重复使用先前`getKey`操作中返回的密钥。
- 解决状态：将您的客户端 SDK 版本升级到 5.11.0 或更高版本，其中包括此问题的修复程序。

OpenSSL Dynamic Engine 的已知问题

这些为 OpenSSL Dynamic Engine 的已知问题

主题

问题：你无法在 RHEL 6 和 C AWS CloudHSM entOS6 上安装 OpenSSL 动态引擎

- 影响：OpenSSL 动态引擎仅支持 OpenSSL 1.0.2[f+]。默认情况下，RHEL 6 和 CentOS 6 都附带了 OpenSSL 1.0.1。
- 解决方法：将 RHEL 6 和 CentOS 6 上的 OpenSSL 库升级到版本 1.0.2[f+]。

问题：默认情况下，仅支持 RSA 分流到 HSM。

- 影响：为了最大限度地提高性能，开发工具包未配置为卸载其他函数，例如随机数生成或 EC-DH 操作。
- 解决方法：如果您需要卸载其他操作，请建立一个支持案例来与我们联系。
- 解决状态：我们正在增加开发工具包支持，来通过配置文件配置卸载选项。当该更新可用时，将在版本历史记录页面中公布。

问题：不支持使用 HSM 上的密钥实现具有 OAEP 填充的 RSA 加密和解密。

- 影响：任何使用 OAEP 填充的 RSA 加密和解密调用都将失败并显示错误。 divide-by-zero 出现这种情况的原因是，OpenSSL 动态引擎使用伪造的 PEM 文件在本地调用该操作，而不是将该操作分流到 HSM。
- 解决方法：您可以使用PKCS #11 库或JCE 提供程序执行此过程。
- 解决状态：我们正在添加对开发工具包的支持以正确分载此操作。当该更新可用时，将在版本历史记录页面中公布。

问题：只有 RSA 和 ECC 密钥的私有密钥生成任务才会分流到 HSM。

对于任何其他密钥类型，OpenSSL AWS CloudHSM 引擎不用于呼叫处理。相反会使用本地 OpenSSL 引擎。这将在本地生成一个软件密钥。

- 影响：由于故障转移没有提示，因此，没有迹象表明您尚未收到在 HSM 上安全生成的密钥。如果 OpenSSL 在本地生成了软件密钥，则您会看到含有字符串 "`.....++++++`" 的输出跟踪。将操作分流到 HSM 时，此跟踪将不复存在。由于 HSM 上没有生成或存储密钥，因此，以后将无法使用该密钥。
- 解决方法：仅为 OpenSSL 引擎支持的密钥类型使用此类引擎。对于所有其他密钥类型，请在应用程序中使用 PKCS #11 或 JCE，或者在 CLI `key_mgmt_util` 中使用。

问题：您无法在 RHEL 8、CentOS 8 或 Ubuntu 18.04 LTS 上安装适用于客户端软件开发工具包 3 的 OpenSSL Dynamic Engine

- 影响：默认情况下，RHEL 8、CentOS 8 和 Ubuntu 18.04 LTS 发布的 OpenSSL 版本与客户端软件开发工具包 3 的 OpenSSL Dynamic Engine 不兼容。
- 变通办法：使用支持 OpenSSL Dynamic Engine 的 Linux 平台。有关支持的平台的更多信息，请参阅支持的平台。
- 解析状态：使用适用于客户端 SDK 5 的 OpenSSL 动态引擎 AWS CloudHSM 支持这些平台。有关更多信息，请参阅支持的平台和 OpenSSL Dynamic Engine。

问题：SHA-1 在 RHEL 9 (9.2+) 上签署并验证弃用情况

- 影响：RHEL 9 (9.2+) 中已不建议出于加密目的使用 SHA-1 消息摘要。因此，使用 OpenSSL 动态引擎对 SHA-1 进行签名和验证操作将失败。
- 解决办法：如果您的场景需要使用 SHA-1 来签署/验证现有或第三方加密签名，请参阅增强 RHEL 安全性：了解 RHEL 9 (9.2+) 和 RHEL 9 (9.2+) 发行说明中已弃用 SHA-1 以了解更多详细信息。

问题： AWS CloudHSM OpenSSL 动态引擎与 OpenSSL v3.x 的 FIPS 提供程序不兼容

- 影响：在 OpenSSL 版本 3.x 中启用 FIPS 提供程序时，如果您尝试使用 AWS CloudHSM OpenSSL 动态引擎，则会收到错误消息。
- 解决办法：要在 AWS CloudHSM OpenSSL 3.x 版本中使用 OpenSSL 动态引擎，请确保配置了 “默认” 提供程序。在 OpenSSL 网站上阅读有关默认提供商的更多信息。

运行 Amazon Linux 2 的 Amazon EC2 实例的已知问题

问题：亚马逊 Linux 2 版本 2018.07 使用的更新后的**ncurses**软件包（版本 6）目前与软件开发工具包不兼容 AWS CloudHSM

运行 c AWS CloudHSMloudhsm_mgmt_util 或 key_mgmt_ util 时，你会看到返回以下错误：

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util: error while loading shared libraries:
 libncurses.so.5: cannot open shared object file: No such file or directory
```

- 影响：在 Amazon Linux 2 版本 2018.07 上运行的实例将无法使用所有 AWS CloudHSM 实用程序。
- 变通办法：在 Amazon Linux 2 EC2 实例上发布以下命令以安装支持的 ncurses 软件包（版本 5）：

  ```
  sudo yum update && yum install ncurses-compat-libs
  ```

- 解决状态： 此问题已经在 AWS CloudHSM 客户端 1.1.2 发布版中解决。您必须升级到此客户端，才能从修复中获益。

有关集成第三方应用程序的已知问题

问题：客户端软件开发工具包 3 不支持在主密钥生成过程中的 Oracle 设置 PKCS #11 的 属性 **CKA_MODIFIABLE**。

此限制是在 PKCS #11 库中定义的。有关更多信息，请参阅支持的 PKCS #11 属性上的注释 1。

- 影响：Oracle 主密钥创建失败。
- 解决方法：在创建新的主密钥时将特殊环境变量 CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE 设置为 TRUE。仅在生成主密钥时需要此环境变量，您无需将此环境变量用于任何其他项目。例如，可以将此变量用于您创建的第一个主密钥，之后，如果您要轮换主密钥版本，则只能再次使用此环境变量。有关更多信息，请参阅生成 Oracle TDE 主加密密钥。
- 解决状态：我们正在改进 HSM 固件以实现对 CKA_MODIFIABLE 属性的完全支持。更新将在 AWS CloudHSM 论坛和版本历史记录页面上公布

客户端软件开发工具包 3 密钥同步失败

在 Client SDK 3 中，如果客户端同步失败，则 AWS CloudHSM 会尽最大努力清理可能已创建（现在不需要的）的所有不需要的密钥。此过程包括立即移除不需要的密钥材料或标记不需要的材料以备日后移除。在这两种情况下，解决方案都不需要您采取任何操作。在极少数情况下，如果 AWS CloudHSM 无法移除也无法标记不需要的密钥材料，则必须删除密钥材料。

问题：您尝试令牌密钥生成、导入或解包操作，但看到指定墓碑失败的错误。

```
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： AWS CloudHSM 移除和标记不需要的密钥材料失败。

解决方案：您集群中的 HSM 包含未标记为不需要的不需要的密钥材料。您必须手动移除密钥材料。要手动删除不需要的密钥材料，请使用 key_mgmt_util (KMU) 或 PKCS #11 库或 JCE 提供程序中的 API。有关更多信息，请参阅 deleteKey 或 客户端软件开发工具包。

为了使令牌密钥更耐用，在客户端同步设置中指定的最小数量的 HSM 上无法成功创建密钥的操作 AWS CloudHSM 失败。有关更多信息，请参阅 AWS CloudHSM 中的密钥同步功能。

客户端软件开发工具包 3：使用 pkpspeed 工具验证 HSM 性能

本主题介绍如何使用客户端软件开发工具包 3 验证 HSM 性能。

要验证 AWS CloudHSM 集群中 HSM 的性能，你可以使用 Client SDK 3 附带的 pkpspeed (Linux) 或 pkpspeed_blocking (Windows) 工具。pkpspeed 工具在理想条件下执行操作，无需通过 PKCS11 等 SDK 即可直接调用 HSM 执行操作。我们建议对您的应用程序进行独立负载测试，以确定自己的扩展需求。我们不建议运行以下测试：随机 (I)、 ModExp (R) 和 EC point mul (Y)。

有关在 Linux EC2 实例上安装此客户端的更多信息，请参见安装和配置 AWS CloudHSM 客户端 (Linux)。有关在 Windows 实例上安装此客户端的更多信息，请参见安装和配置 AWS CloudHSM 客户端 (Windows)。

安装和配置 AWS CloudHSM 客户端后，运行以下命令将其启动。

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

- 对于 Windows 客户端 1.1.2 以上版本：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 对于 Windows 客户端 1.1.1 及更低版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

如果您已安装此客户端软件，则可能需要下载并安装最新版本才能获得 pkpspeed。您可以在 Linux 中的 `/opt/cloudhsm/bin/pkpspeed` 或 Windows 中的 `C:\Program Files\Amazon\CloudHSM\` 中找到 pkpspeed 工具。

要使用 pkpspeed，请运行 pkpspeed 命令或 pkpspeed_blocking.exe，并在 HSM 上指定加密用户 (CU) 的用户名和密码。然后设置要使用的选项，并考虑以下建议。

测试建议

- 要测试 RSA 签名和验证操作的性能，请在 Linux 中选择 RSA_CRT 密码或在 Windows 中选择选项 B。请勿选择 RSA（Windows 中为选项 A）。虽然密码是等效的，但 RSA_CRT 已针对性能进行优化。
- 从少量线程开始。对于 AES 性能测试，一个线程通常足以显示最高性能。对于 RSA 性能测试 (RSA_CRT)，三个或四个线程通常已足够。

pkpspeed 工具的配置选项

- FIPS 模式：始终 AWS CloudHSM 处于 FIPS 模式（详情请参阅AWS CloudHSM 常见问题解答）。这可以通过使用 AWS CloudHSM 用户指南中记录的 CLI 工具并运行指示 FIPS 模式状态的 getHSMInfo 命令来验证。
- 测试类型（分块与非分块）：这指定了如何以线程方式执行操作。非分块方式的数值结果更佳。原因是它们利用线程和并发性。
- 线程数：用于测试运行的线程数。
- 运行测试时间（以秒为单位）(最大 = 600)：pkpspeed 生成以 “操作/秒” 为单位测量的结果，并在测试运行的每秒钟报告此值。例如，如果测试运行了 5 秒，则可能输出类似以下示例值：
  - `OPERATIONS/second 821/1`
  - `OPERATIONS/second 833/1`
  - `OPERATIONS/second 845/1`
  - `OPERATIONS/second 835/1`
  - `OPERATIONS/second 837/1`

可通过 pkpspeed 工具运行的测试

- AES GCM：测试 AES GCM 模式加密。
- 基本 3DES CBC：测试 3DES CBC 模式加密。有关即将发生的更改，请参阅下面的注释 1。

- 基本 AES：测试 AES CBC/ECB 加密。
- 摘要：测试哈希摘要。
- ECDSA 签名：测试 ECDSA 签名。
- ECDSA 验证：测试 ECDSA 验证。
- FIPS Random：测试符合 FIPS 的随机数的生成（注意：这只能在分块模式下使用）。
- HMAC：测试 HMAC。
- Random：此测试无关紧要，因为我们使用的是 FIPS 140-2 HSM。
- RSA 非 CRT 与 RSA_CRT：测试 RSA 签名并验证操作。
- RSA OAEP Enc：测试 RSA OAEP 加密。
- RSA OAEP 12 月：测试 RSA OAEP 解密。
- RSA 私匙 dec 非 CRT：测试 RSA 私钥加密（未优化）。
- RSA 私匙 dec CRT：测试 RSA 私钥加密（已优化）。
- RSA PSS 签名：测试 RSA PSS 签名。
- RSA PSS 验证：测试 RSA PSS 验证。
- RSA 公钥 enc：测试 RSA 公钥加密。

RSA 公钥加密、RSA 私钥解密非 CRT 和 RSA 私钥解密 CRT，也将提示用户回答以下问题：

```
Do you want to use static key [y/n]
```

如果输入 y，则会将预先计算的密钥导入 HSM。

如果输入 n，则会生成新密钥。

[1] 根据 NIST 的指导方针，2023 年之后处于 FIPS 模式的集群不允许这样做。对于处于非 FIPS 模式的集群，2023 年之后仍允许这样做。有关详细信息，请参阅 FIPS 140 合规：2024 年机制弃用。

示例

以下示例显示了可选择与 pkpspeed (Linux) 或 pkpspeed_blocking (Windows) 一起用于测试 RSA 和 AES 操作的 HSM 性能的选项。

Example – 使用 pkpspeed 测试 RSA 性能

您可以在 Windows、Linux 和兼容的操作系统上运行此示例。

Linux

对 Linux 和兼容的操作系统使用这些说明。

```
/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 测试 AES 性能

Linux

对 Linux 和兼容的操作系统使用这些说明。

```
/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/...
```

客户端软件开发工具包 5 用户包含不一致的值

该 `user list` 命令返回集群中所有用户和用户属性的列表。如果用户的任何属性的值为“不一致”，则该用户不会在您的集群中同步。这意味着该用户在集群中的不同 HSM 上具有不同的属性。根据不一致的属性，可以采取不同的修复步骤。

下表包括解决单个用户不一致问题的步骤。如果单个用户存在多个不一致之处，请从上到下按照以下步骤解决这些问题。如果有多个用户存在不一致之处，请为每个用户仔细阅读此列表，完全解决一位用户的不一致之处，然后再继续下一步。

> **Note**
>
> 要执行这些步骤，理想情况下，您应该以管理员身份登录。如果您的管理员帐户不一致，请以管理员身份登录执行以下步骤，然后重复这些步骤，直到所有属性都一致为止。管理员帐户保持一致后，您可以继续使用该管理员帐户来同步集群中的其他用户。

不一致的属性	用户列表的示例输出	影响	恢复方法

用户 "role" 为 "inconsistent"	<pre>{
	"username":
	"test_user",
	"role":
	"**inconsistent** ",
	
	"locked":
	"false",
	"mfa": [],
	"cluster-
	coverage":
	"full"
	}</pre>
		此用户 CryptoUser 在某些 HSM 上是管理员，在其他 HSM 上是管理员。如果两个 SDK 尝试同时创建具有不同角色的同一个用户，则会发生这种情况。您必须移除此用户，然后使用所需的角色重新创建该用户。	1. 以管理员身份登录。
			2. 删除所有 HSM 上的用户：
			
			user delete --username **<user's name>** --role admin
			
			user delete --username **<user's name>** --role crypto-user
			3. 创建具有所需角色的用户：
			
			user create --username

不一致的属性	用户列表的示例输出	影响	恢复方法
			<user's name> --role **<desired role>**

不一致的属性	用户列表的示例输出	影响	恢复方法

用户 "cluster-coverage" 为 "inconsistent"	<pre>{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [],
	"cluster-coverage":
	"**inconsistent** "
	}</pre>
		此用户存在于集群中 HSM 的子集上。如果 user create 部分成功或 user delete 部分成功，则可能发生这种情况。
		
		您必须完成之前的操作，即在集群中创建或删除此用户。
			如果用户不应该存在，请按照以下步骤操作：
			
			1. 以管理员身份登录。
			2. 运行以下命令：
			
			user delete --username**<user's name>** --role admin
			3. 现在请运行以下命令：
			
			user delete --username**<user's name>** --role crypto-user
			
			如果用户应该存在，请按照以下步骤操作：
			
			1. 以管理员身份登录。
			2. 运行以下命令：
			
			user create --username **<user's name>** --role **<desired role>**

不一致的属性	用户列表的示例输出	影响	恢复方法

用户 "locked" 参数为 "inconsistent" 或 "true"	{
	"username": "test_user",
	"role": "crypto-user",
	"locked": **inconsistent**,
	"mfa": [],
	"cluster-coverage": "full"
	}
		该用户在 HSM 的某个子集中被锁定。
		
		如果用户使用了错误的密码并且只连接到集群中 HSM 的某个子集，则可能会发生这种情况。
		
		您必须更改用户的凭证，使这些凭证在整个集群中保持一致。
			如果用户已激活 MFA，请按照以下步骤操作：
			
			1. 以管理员身份登录。
			2. 运行以下命令以暂时停用 MFA：
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			3. 更改用户的密码，以便他们可以登录所有 HSM：
			
			user change-password --username **<user's name>** --role **<desired role>**
			
			如果用户应激活 MFA，请按照以下步骤操作：
			
			1. 让用户登录并重新启用 MFA（这将要求他们签署令牌并在 PEM 文件中提供其公有密钥）：

不一致的属性	用户列表的示例输出	影响	恢复方法
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** —token <File>

不一致的属性	用户列表的示例输出	影响	恢复方法

MFA 状态为 "inconsistent"	{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [
	{
	"strategy":
	"token-sign",
	"status":
	"**inconsistent** "
	}
	],
	"cluster-coverage":
	"full"
	}
		此用户在集群中的不同 HSM 上具有不同的 MFA 标记。
		
		如果 MFA 操作仅在 HSM 的某个子集中完成，则可能会发生这种情况。
		
		您必须重置用户密码，并允许他们重新启用 MFA。
			如果用户已激活 MFA，请按照以下步骤操作：
			
			1. 以管理员身份登录。
			2. 运行以下命令以暂时停用 MFA：
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			3. 然后，您还需更改用户密码，以便他们可以登录所有 HSM：
			
			user change-password --username **<user's name>** --role **<desired role>**
			
			如果用户应激活 MFA，请按照以下步骤操作：
			
			1. 让用户登录并重新启用 MFA（这将要求他们签署令牌并在 PEM 文件中提供其公有密钥）：

不一致的属性	用户列表的示例输出	影响	恢复方法
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** —token <File>

检查密钥可用性时出现错误

问题：HSM 返回以下错误：

```
Key <KEY HANDLE> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

原因：密钥可用性检查会查找在可能的罕见情况下可能丢失的密钥。该错误通常发生在只有一个 HSM 的集群或有两个 HSM 的集群（其中一个正在更换）中。在这些情况下，以下客户操作可能导致了上述错误：

- 使用key generate-symmetric 或之类的命令生成了新密钥钥匙 generate-asymmetric-pair 。
- key list 操作已开始。
- SDK 的新实例已启动。

> **Note**
>
> OpenSSL 经常分叉软件开发工具包的新实例。

解决方案/建议：从以下操作中进行选择，以防止此错误发生：

- 在配置工具的配置文件中，使用 --disable-key-availability-check 参数将密钥可用性设置为“false”。有关更多信息，请参阅配置工具的 参数 节。
- 如果使用带有两个 HSM 的集群，请避免进行触发错误的操作，初始化代码期间除外。
- 将您集群中的 HSM 数量增加到为至少三个。

使用 JCE 提取密钥

getencoded getPrivateExponent、或 GET 返回空值

getEncoded、getPrivateExponent 和 getS 将返回“Null”，因为它们在默认情况下处于禁用状态。要启用它们，请参阅 使用 JCE 提取密钥。

如果 getEncoded、getPrivateExponent、和 getS 在启用后返回“Null”，则说明您的密钥不符合正确的先决条件。有关更多信息，请参阅使用 JCE 提取密钥。

getencoded getPrivateExponent、或 GET 会返回 HSM 之外的密钥字节

您或有权访问您系统的人员已启用“清除密钥提取”。有关更多信息，包括如何将此配置重置为默认禁用状态，请参阅以下页面。

- 使用 JCE 提取密钥
- 保护和提取 HSM 中的密钥

HSM 节流

当您的工作负载超过集群的 HSM 容量时，您将收到报错消息，指出 HSM 忙碌或已被节流。发生这种情况时，您可能会看到 HSM 的吞吐量降低或拒绝请求的比率上升。此外，HSM 可能会发送以下繁忙错误。

适用于客户端软件开发工具包 5

- 在 PKCS11 中，繁忙错误映射到 CKR_FUNCTION_FAILED。出现此错误可能有多种原因，但如果 HSM 节流导致此错误，则您的日志中将显示以下日志行：
  - `[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，繁忙错误映射到 `com.amazonaws.cloudhsm.jce.jni.exception.InternalException: Unexpected`

`error with the Provider: The HSM could not queue the request for processing.`

- 其他 SDK 的繁忙错误会打印以下消息：`Received error response code from Server. Response Code: 187`

适用于客户端软件开发工具包 3

- 在 PKCS11 中，繁忙错误映射到 `CKR_OPERATION_ACTIVE` 错误。
- 在 JCE 中，繁忙错误映射到 `CFM2Exception`，状态为 `0xBB (187)`。应用程序可以使用 `CFM2Exception` 上的 `getStatus()` 功能检查 HSM 返回的状态。
- 其他 SDK 的繁忙错误会打印以下消息：`HSM Error: HSM is already busy generating the keys(or random bytes) for another request.`

解决方案

您可以通过完成以下一项或多项操作来解决这些问题：

- 为您的应用程序层中被拒绝的 HSM 操作添加重试命令。在启用重试命令之前，请确保您的集群大小足以满足峰值负载。

> **Note**
>
> 对于客户端软件开发工具包 5.8.0 及更高版本，重试命令默认处于启用状态。有关每个 SDK 的重试命令配置的详细信息，请参阅 客户端软件开发工具包 5 配置工具的高级配置。

- 按照 在集群中添加或移除 HSM AWS CloudHSM 中的说明向您的集群添加更多 HSM。

> **Important**
>
> 我们建议您对集群执行负载测试，以确定您的峰值负载预期，然后添加 HSM 以确保高可用性。

让 HSM 用户在集群中的 HSM 之间保持同步

要管理 HSM 的用户，您可以使用名为 cloudhsm_mgmt_util 的 AWS CloudHSM 命令行工具。它仅与位于该工具的配置文件中的 HSM 通信。它无法感知集群中不在此配置文件中的其他 HSM。

AWS CloudHSM 会将 HSM 上的密钥与集群中的所有其他 HSM 同步，但不会同步 HSM 的用户或策略。当您使用 cloudhsm_mgmt_util 管理 HSM 用户时，这些用户的更改可能只会影响集群中位于 cloudhsm_mgmt_util 配置文件中的那些 HSM。这可能会在群集中的所有 HSM 上 AWS CloudHSM 同步密钥时出现问题，因为集群中的所有 HSM 上可能并不存在拥有密钥的用户。

要避免这些问题，请先编辑 cloudhsm_mgmt_util 配置文件，然后再管理用户。有关更多信息，请参阅 ???。

与集群的连接已丢失

配置 AWS CloudHSM 客户端时，您提供了集群中第一个 HSM 的 IP 地址。此 IP 地址保存在 AWS CloudHSM 客户端的配置文件中。当客户端启动时，它会尝试连接到此 IP 地址。如果无法执行此操作（例如因为 HSM 失败或您已将其删除），您可能会看到与以下内容类似的错误：

```
LIQUIDSECURITY: Daemon socket connection error
```

```
LIQUIDSECURITY: Invalid Operation
```

要解决这些错误，请使用群集中可访问的活动 HSM 的 IP 地址更新配置文件。

更新 AWS CloudHSM 客户机的配置文件

1. 使用以下方式之一查找群集中活动 HSM 的 IP 地址。

   - 查看 控制台的集群详细信息页面上的AWS CloudHSM HSM 选项卡。
   - 使用 AWS Command Line Interface (AWS CLI) 发出describe-clusters命令。

   在后续步骤中，您将需要此 IP 地址。
2. 使用以下命令停止客户端。

   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

- 对于 Windows 客户端 1.1.2 以上版本：

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- 对于 Windows 客户端 1.1.1 及更低版本：

  在启动 AWS CloudHSM 客户端的命令窗口中使用 C trl + C。

3. 使用以下命令可更新客户端的配置文件，并提供您在上一步中找到的 IP 地址。

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

4. 使用以下命令启动 客户端。

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

- 对于 Windows 客户端 1.1.2 以上版本：

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- 对于 Windows 客户端 1.1.1 及更低版本：

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

缺少 AWS CloudHSM 审核日志 CloudWatch

如果您在 2018 年 1 月 20 日前创建了一个集群，则需要手动配置服务相关角色以启用该集群审查日志的传递。有关如何在 HSM 集群上启用服务相关角色的说明，请参阅 《IAM 用户指南》中的了解服务相关角色和创建服务相关角色。

AES 密钥包装长度不合规的自定义 IV

本故障排除主题可帮助您确定应用程序是否会生成无法恢复的包装好的密钥。如果您受到此问题的影响，请使用此主题来解决该问题。

主题

- 确定您的代码是否生成无法恢复的包装好的密钥
- 当您的代码生成了不可恢复的包装好的密钥时必须采取的操作

确定您的代码是否生成无法恢复的包装好的密钥

仅当您满足以下所有条件时，您才会受到影响：

状况	我怎样才能知道？
您的应用程序使用 PKCS #11 库	PKCS #11 库作为 `libpkcs11.so` 文件安装在您的 `/opt/cloudhsm/lib` 文件夹中。用 C 语言编写的应用程序通常直接使用 PKCS #11 库，而用 Java 编写的应用程序可能通过 Java 抽象层间接使用该库。如果您使用的是 Windows，则不会受到影响，因为 PKCS #11 库目前不适用于 Windows。

状况	我怎样才能知道？
您的应用程序专门使用 PKCS #11 库的 3.0.0 版	如果你收到了 AWS CloudHSM 团队的电子邮件，那么你可能使用的是 PKCS #11 库的 3.0.0 版。 要检查应用程序实例上的软件版本，请使用以下命令： `rpm -qa | grep ^cloudhsm`
您可以使用 AES 密钥包装来包装密钥	AES 密钥包装是指使用 AES 密钥包装其他密钥。相应的机制名称是 CKM_AES_KEY_WRAP 。它与函数 C_WrapKey 一起使用。其他使用初始化向量 (IV, initialization vector) 的基于 AES 的包装机制，例如 CKM_AES_GCM 和 CKM_CLOUDHSM_AES_GCM ，不受此问题的影响。[了解有关函数和机制的更多信息](#)。
您在调用 AES 密钥包装时指定自定义 IV，并且此 IV 的长度小于 8	AES 密钥包装通常使用如下 CK_MECHANISM 结构进行初始化： `CK_MECHANISM mech = {CKM_AES_KEY_WRAP, IV_POINTER, IV_LENGTH};` 只有在下列情况下，本问题才适用于您： • IV_POINTER 不为 NULL • IV_LENGTH 小于 8 字节

如果您不符合上述所有条件，则可立即停止阅读。您可以正确解开包装好的密钥，此问题不会影响您。否则，请参阅the section called “当您的代码生成了不可恢复的包装好的密钥时必须采取的操作”。

当您的代码生成了不可恢复的包装好的密钥时必须采取的操作

您应该采取以下三个步骤：

1. 立即将您的 PKCS #11 库升级到更新的版本
   - 适用于 Amazon Linux、CentOS 6 和 RHEL 6 的最新 PKCS #11 库
   - 适用于 Amazon Linux 2、CentOS 7 和 RHEL 7 的最新 PKCS #11 库
   - 适用于 Ubuntu 16.04 LTS 的最新 PKCS #11 库
2. 更新您的软件以使用符合标准的 IV

   我们强烈建议您遵循我们的代码示例，只需指定 NULL IV 即可，这能使 HSM 使用符合标准的默认 IV。或者，您可以明确指定 IV 为 0xA6A6A6A6A6A6A6A6，相应的 IV 长度为 8。我们不建议使用任何其他 IV 进行 AES 密钥包装，并且将在未来版本的 PKCS #11 库中明确禁用 AES 密钥包装的自定义 IV。

   正确指定 IV 的示例代码显示在 aes_wrapping.c 中。 GitHub
3. 标识并恢复现有的包装好的密钥

   您应标识出使用 PKCS #11 库的 3.0.0 版包装的所有密钥，然后联系客户服务寻求帮助 (https://aws.amazon.com/support)，以恢复这些密钥。

> **Important**
>
> 此问题仅影响使用 PKCS #11 库 3.0.0 版包装的密钥。您可以使用 PKCS #11 库的早期版本（2.0.4 及更低版本的软件包）或后续版本（3.0.1 及更高版本的软件包）来包装密钥。

解决集群创建失败问题

创建集群时，如果 AWSServiceRoleForCloudHSM 服务相关角色尚不存在，则会 AWS CloudHSM 创建该角色。如果 AWS CloudHSM 无法创建服务相关角色，则创建集群的尝试可能会失败。

本主题介绍了如何解决最常见的问题以便您可以成功创建集群。您只需创建此角色一次。在您的账户中创建该服务相关角色后，您即可使用任何支持的方法来创建其他集群并管理它们。

以下部分提供了解决与该服务相关角色相关的集群创建失败问题的建议。如果您试用这些建议但仍无法创建群集，请联系 AWS Support。有关 AWSServiceRoleForCloudHSM 服务相关角色的更多信息，请参阅的服务相关角色 AWS CloudHSM。

主题

- 添加缺少的权限

- 手动创建服务相关角色
- 使用非联合身份用户

添加缺少的权限

要创建服务相关角色，用户必须具有 iam:CreateServiceLinkedRole 权限。如果创建集群的 IAM 用户没有此权限，则当它尝试在您的 AWS 账户中创建服务相关角色时，集群创建过程将失败。

当缺少权限导致失败时，错误消息将包含以下文本。

```
This operation requires that the caller have permission to call
 iam:CreateServiceLinkedRole to create the CloudHSM Service Linked Role.
```

要解决此错误，请向要创建集群的 IAM 用户提供 AdministratorAccess 权限或将 iam:CreateServiceLinkedRole 权限添加到用户的 IAM 策略。有关说明，请参阅向新用户或现有用户添加权限。

然后，再次尝试创建集群。

手动创建服务相关角色

您可以使用 IAM 控制台、CLI 或 API 来创建 AWSServiceRoleForCloudHSM服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的创建服务相关角色。

使用非联合身份用户

凭证来自外部的 AWS联合用户可以执行非联合用户的许多任务。但是，AWS 不允许用户进行 API 调用来通过联合端点创建服务相关角色。

要解决此问题，请创建非联合身份用户 (具有 iam:CreateServiceLinkedRole 权限)，或者向现有非联合身份用户提供 iam:CreateServiceLinkedRole 权限。然后，让该用户通过 ???创建集群 AWS CLI。这将在您的账户中创建该服务相关角色。

创建该服务相关角色后，如果您愿意，即可删除非联合身份用户创建的集群。删除该集群不会影响角色。此后，任何具有所需权限的用户（包括联合用户）都可以在您的账户中创建 AWS CloudHSM 集群。

要验证角色是否已创建，请打开在 https://console.aws.amazon.com/iam/ 上的 IAM 控制台并选择角色。或者，在 AWS CLI中使用 IAM get-role 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"
    }
}
```

检索客户端配置日志

AWS CloudHSM 为客户端 SDK 3 和客户端 SDK 5 提供工具，用于收集有关您的环境的信息，以便 Su AWS pport 解决问题。

主题

- 客户端软件开发工具包 5 支持工具
- 客户端软件开发工具包 3 支持工具

客户端软件开发工具包 5 支持工具

该脚本提取以下信息：

- 客户端软件开发工具包 5 组件的配置文件
- 可用日志文件

- 操作系统的当前版本
- 软件包信息

运行客户端软件开发工具包 5 的信息工具

客户端软件开发工具包 5 包括每个组件的客户端支持工具，但所有工具的功能都相同。运行工具以创建包含所有收集信息的输出文件。

这些工具使用类似于以下的语法：

```
[ pkcs11 | dyn | jce ]_info
```

例如，要从运行 PKCS #11 库的 Linux 主机收集支持信息并让系统写入默认目录，您可以运行以下命令：

```
/opt/cloudhsm/bin/pkcs11_info
```

该工具在 /tmp 目录内创建输出文件。

PKCS #11 library

要收集 Linux 上的 PKCS #11 库的支持数据

- 使用支持工具收集数据。

  ```
  /opt/cloudhsm/bin/pkcs11_info
  ```

要收集 Windows 上的 PKCS #11 库的支持数据

- 使用支持工具收集数据。

  ```
  C:\Program Files\Amazon\CloudHSM\bin\pkcs11_info.exe
  ```

OpenSSL Dynamic Engine

收集 Linux 上的 OpenSSL 动态引擎的支持数据

- 使用支持工具收集数据。

```
/opt/cloudhsm/bin/dyn_info
```

JCE provider

为 Linux 上的 JCE 提供程序收集支持数据

- 使用支持工具收集数据。

```
/opt/cloudhsm/bin/jce_info
```

为 Windows 上的 JCE 提供程序收集支持数据

- 使用支持工具收集数据。

```
C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe
```

自无服务器环境检索日志

要针对无服务器环境（例如 Fargate 或 Lambda）进行配置，我们建议您将日志类型配置为 AWS CloudHSM 。`term`配置为后`term`，无服务器环境将能够输出到。 CloudWatch

要从中获取客户端日志 CloudWatch，请参阅 Amazon L ogs 用户指南中的使用 CloudWatch 日志组和日志流。

客户端软件开发工具包 3 支持工具

该脚本提取以下信息：

- 操作系统及其当前版本
- 来自 `cloudhsm_client.cfg`、`cloudhsm_mgmt_util.cfg` 和 `application.cfg` 文件的客户端配置信息
- 来自平台特定位置的客户端日志
- 通过使用 cloudhsm_mgmt_util 收集的集群和 HSM 信息
- OpenSSL 信息
- 当前客户端和构建版本

- 安装程序版本

运行客户端软件开发工具包 3 的信息工具

该脚本使用收集的所有信息创建一个输出文件。该脚本在 /tmp 目录内创建输出文件。

Linux：/opt/cloudhsm/bin/client_info

Windows：C:\Program Files\Amazon\CloudHSM\client_info

> **Warning**
>
> 对于客户端软件开发工具包 3 版本（3.1.0 至 3.3.1），此脚本存在一个已知问题。我们强烈建议您升级到 3.3.2 版本，其中包含此问题的修复程序。在使用此工具之前，请参阅已知问题页面以获取更多信息。

AWS CloudHSM 配额

配额（以前称为限制）是为 AWS 资源分配的值。以下配额适用于每个 AWS 地区和每个 AWS 账户的 AWS CloudHSM 资源。默认配额是应用的初始值 AWS，下表列出了这些值。可以将可调配额增加到默认配额以上。

服务限额

资源	默认配额	是否可调整？
集群	4	是
HSM	6	是
每个集群的 HSM 数	28	否

请求增加配额的建议方式是打开服务配额控制台。在控制台中，选择您的服务和配额，然后提交请求。有关更多信息，请参阅服务配额文档。

以下系统配额表中的配额不可调整。

系统配额

资源	hsm1.medium 配额	hsm2m.medium 配额
每个集群的最大密钥数量	3,300	总共有 16,666 个密钥，其中非对称密钥的最大值为 3,333 个
每个集群的最大用户数量	1024	1024
用户名的最大长度	31 个字符	31 个字符
必需的密码长度	8 到 32 个字符	8 到 32 个字符
每个集群最大并发客户端连接数 [1]	900	900
每个应用程序 PKCS #11 会话的最大数量	1024	1024

[1]客户端软件开发工具包 3 的客户端连接是客户端进程守护程序。对于客户端软件开发工具包 5，客户端连接就是一款应用程序。

有关更多信息，请参阅 系统资源。

系统资源

系统资源配额是 AWS CloudHSM 客户机运行时允许使用的配额。

文件描述符是一种操作系统的机制，用于在每个进程的基础上识别和管理打开的文件。

CloudHSM 客户端守护程序利用文件描述符来管理应用程序与客户端之间以及客户端与服务器之间的连接。

默认情况下，CloudHSM 客户端配置将分配 3000 个文件描述符。此默认值旨在在客户端守护程序和 HSM 之间产生最佳会话和线程容量。

在极少数情况下，如果您在资源受限的环境中运行客户端，则可能需要更改这些默认值。

> **Note**
>
> 更改这些值后，您的 CloudHSM 客户端性能可能会受到影响，并且/或者您的应用程序可能无法运行。

1. 编辑 `/etc/security/limits.d/cloudhsm.conf` 文件。

```
#
# DO NOT EDIT THIS FILE
#
hsmuser soft nofile 3000
hsmuser hard nofile 3000
```

2. 根据需要编辑数值。

> **Note**
>
> `soft` 配额必须小于或等于 `hard` 配额。

3. 重启您的 CloudHSM 客户端守护程序进程。

> **Note**
>
> 此配置选项在 Microsoft Windows 平台上不可用。

AWS CloudHSM 客户端 SDK 的下载内容

Downloads

2021 年 3 月， AWS CloudHSM 发布了 Client SDK 版本 5.0.0，该版本引入了具有不同要求、功能和平台支持的全新客户端 SDK。

客户端 SDK 5 完全支持生产环境，其组件和支持级别与客户端 SDK 3 相同，但对 CNG 和 KSP 提供商的支持除外。有关更多信息，请参阅 客户端软件开发工具包 组件对比。

> **Note**
>
> 有关每个客户端 SDK 支持哪些平台的信息，请参阅客户端软件开发工具包 5 支持的平台和客户端软件开发工具包 3 支持的平台。

最新版本

本部分包含最新版本的客户端 SDK。

客户端 SDK 5 版本：版本 5.12.0

Amazon Linux 2

在 x86_64 架构上下载适用于亚马逊 Linux 2 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL 动态引擎 (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE 提供程序 (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

在 ARM64 架构上下载适用于亚马逊 Linux 2 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum c28a1f27e23e6ab1550dab6a353c6c9338a391a84d57f4ac99a1a3a9810c753f)
- OpenSSL 动态引擎 (SHA256 checksum 7d2e864c31c13f55443c1b1d04589fbdd4558fe103954de4384691e2c429a872)

- JCE 提供程序 (SHA256 checksum e9a35eb87b2f257c47fb083d286deb835da45858b2d89759ca7d5bb4ef747b4b)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 28b6f918912b5c63bf10018824b642a805b309c21947a1d0ebbdc44647e80554)

Amazon Linux 2023

在 x86_64 架构上下载适用于亚马逊 Linux 2023 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum 02801365cba449c5238a4e5ad3df1ddf7edd00ade976f47e956e885286503f3f)
- OpenSSL 动态引擎 (SHA256 checksum 0abed69a7c6acaafdaabdcc5fab7d56611ffd94f5480cade6f8beace9aeae056)
- JCE 提供程序 (SHA256 checksum 3d5d9a903d3a216eca40f92dbb0b4030b7a86ad7ceee8d62241c97a6e1881e25)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum f96671d882b862033bba0b3633448dc6a26e45a25063e29b79a5cd4b7fc4945c)

在 ARM64 架构上下载适用于亚马逊 Linux 2023 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum 53d05006b46bda8e9c1dd76e8307a780bfe0a67b10a9a87723c97f94e29f5b8e)
- OpenSSL 动态引擎 (SHA256 checksum ec1cca8e01b3303ff9473eeef6b33dc85b6affac7a47387b098905f9f2fc85ba)
- JCE 提供程序 (SHA256 checksum c828ae56f46233215b9f35798b5859ebdac962af442acbc457081c3baaa44f11)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum ddd5dcd68d01f4fafaf13dc0b4ddcf98e3731ed51bdd51f85535b29353644a9f)

CentOS 7 (7.8+)

在 x86_64 架构上下载适用于 CentOS 7 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL 动态引擎 (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE 提供程序 (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 7 (7.8+)

在 x86_64 架构上下载适用于 RHEL 7 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL 动态引擎 (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- JCE 提供程序 (SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 8 (8.3+)

在 x86_64 架构上下载适用于 RHEL 8 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum 6e51e95122fd0991278888287f0c408808b26fb5f1196c46168477b9090fc478)
- OpenSSL 动态引擎 (SHA256 checksum 1f1d52ff7af6c537d8cfeb5973c691a9d90a518accd685ff9b66cd78daf98928)
- JCE 提供程序 (SHA256 checksum 156944607de987d6b39bd8a2d21ccd294c01377a9e35f9f15f8b0f4c8bb90033)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum 351e802f79dd2d0b5f7d23bb74c146be05e5169b603c9aace24189094a45a35d)

RHEL 9 (9.2+)

在 x86_64 架构上下载适用于 RHEL 9 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum d1b2f4ac7e6e0c18e788512e7726bc68b571d99a1442ce2f2e80f4b0f9956266)
- OpenSSL 动态引擎 (SHA256 checksum cf86a3f17cd6c51969d4ce80c1e3ea6513b995611be7e2e72e5e5233c71d6add)
- JCE 提供程序 (SHA256 checksum ae89e256eb89ec6b4fa0f001e7a4e1d8f1c08530423e81aa74d69a17b25d9a99)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum dfe6fe5d890c33b2f5d38f906ade113b06c8c05f3427a327744c454e7302f1a5)

在 ARM64 架构上下载适用于 RHEL 9 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum cad72a6ab2232b4c38b90d7c62147520b975d646773dd90d7be897fa0a537d2d)

- OpenSSL 动态引擎 (SHA256 checksum ad751f756530a2317c3c64380ea3a07865b13e1874fab0e61ac530b21487c7fb)
- JCE 提供程序 (SHA256 checksum d204e69acfbb90996fb08ae3573607b65630b1124fb379e078c002d55ac07766)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum c0f412cc59bafd235e046cdc1a0c5d330f2d72f7d6434672e9522f86bc945090)

Ubuntu 20.04 LTS

在 x86_64 架构上下载适用于 Ubuntu 20.04 LTS 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum d37b1f872eb2b1ab34303d5b8b803daa925902b645c57c6e15a28bb6321e0f42)
- OpenSSL 动态引擎 (SHA256 checksum cdc6e737652556b57d26d8816b2bc9820128cb3919360660b6f7fe65f9d39e3f)
- JCE 提供程序 (SHA256 checksum f567a08344414a4776e1c5a9715657476925ca32695c4c2dd84a4f3fc5dc1615)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum f2ee5ad01c5018fc3670f602228fd71087228cd3923bf5b9bc73e4d7084dac6c)

Ubuntu 22.04 LTS

在 x86_64 架构上下载适用于 Ubuntu 22.04 LTS 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum 0e78928acd7a1662e4b07b15d5c3ccb88714ff89e47b991c8ab6e4c2229ee5aa)
- OpenSSL 动态引擎 (SHA256 checksum 4f3168745edc5592234891a7b1d82b179a4947e87c72fade1be3bad58b7ed1a3)
- JCE 提供程序 (SHA256 checksum d4c3655cdc2b00d1ab5ceafac94dfbc5c5244ed20e10fdd9db9f4e741e013733)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum d00bbacb6f2e57bd92d832a2bd11cadede972f8e82cc402ec0684b9c6b23123c)

在 ARM64 架构上下载适用于 Ubuntu 22.04 LTS 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum 0c1121535c523acb864215338292bab32acee438357878b5fc0b6d268713b86f)
- OpenSSL 动态引擎 (SHA256 checksum dc7a219302021570bc8c36674d2bd33165557bb2f9a0af8fdf114f1b85a70d84)
- JCE 提供程序 (SHA256 checksum af3834a10081f1e4e7894275c8b9c7b7649b8de3b6f0aeb0781a3358183a9046)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)

- CloudHSM CLI (SHA256 checksum baa253ac62c2fbcc5712561e0fb0feb25461efc3ce68cf86d4c7bf0af0f14a34)

Windows Server 2016

在 x86_64 架构上下载适用于 Windows Server 2016 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- JCE 提供程序 (SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

Windows Server 2019

在 x86_64 架构上下载适用于 Windows Server 2019 的 5.12.0 版软件：

- PKCS #11 库 (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- JCE 提供程序 (SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- CloudHSM CLI (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

客户端 SDK 5.12.0 为多个平台添加了 ARM 支持，并改进了所有 SDK 的性能。CloudHSM CLI 和 JCE 提供程序中增加了新功能。

平台支持

- 在 ARM64 架构上为所有软件开发工具包增加了对亚马逊 Linux 2023 的支持。
- 在 ARM64 架构上为所有 SDK 添加了对红帽企业 Linux 9 (9.2+) 的支持。
- 在 ARM64 架构上为所有 SDK 添加了对 Ubuntu 22.04 LTS 的支持。

CloudHSM CLI

- 添加了以下命令：
  - 密钥复制

- 增加了对连接到多个集群的支持。有关更多信息，请参阅 使用 CloudHSM CLI 连接到多个集群。

JCE 提供程序

- 添加了KeyReferenceSpec用于使用KeyStoreWithAttributes检索密钥的功能。
- 添加了getKeys用于使用KeyStoreWithAttributes一次检索多个密钥的功能。

性能改进

- 改进了所有 SDK 的 AES CBC NoPadding 操作的性能。

之前的客户端 SDK 版本

本节列出了之前发布的客户端 SDK。

版本 5.11.0

Amazon Linux 2

在 x86_64 架构上下载适用于亚马逊 Linux 2 的 5.11.0 版软件：

- PKCS #11 库 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL 动态引擎 (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE 提供程序 (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

在 ARM64 架构上下载适用于亚马逊 Linux 2 的 5.11.0 版软件：

- PKCS #11 库 (SHA256 checksum 5ac16449ec149c9b5e7776865803245ab17d0f1ad56df80173840c5e8d257b19)
- OpenSSL 动态引擎 (SHA256 checksum 28c2eb7f3f60172b0186e5c25f71bb7341537058a71f288673936766048083c1)
- JCE 提供程序 (SHA256 checksum 06c9d9d281c12b1d2bd9a7b601d6317e46cedf175706bbfa3e4dcaed6ba05448)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 218982bb17aa751969a7866b0a9ff27e7aa5007a07817627d9cc1f7d60a78160)

Amazon Linux 2023

在 x86_64 架构上下载适用于亚马逊 Linux 2023 的 5.11.0 版软件：

- PKCS #11 库 (SHA256 checksum 55310ab333d18bcfabdc4b74115b040386b4508934bdff93e1d054c4c4a6f9ea)
- OpenSSL 动态引擎 (SHA256 checksum f3d4934dc872a9b5212a180b9814ca2af3eca01ee228a8725563f1770add0dce)
- JCE 提供程序 (SHA256 checksum 757d3abb515aeb08f4b1c83970ee0979399efee00ee78c9a9dbec05f4ed9768d)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 22af8f0501ff9a45a9e0683a408a63771c2c06c66abf5478d310d6d32e013555)

CentOS 7 (7.8+)

在 x86_64 架构上下载适用于 CentOS 7 的 5.11.0 版软件：

- PKCS #11 库 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL 动态引擎 (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE 提供程序 (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 7 (7.8+)

在 x86_64 架构上下载适用于 RHEL 7 的 5.11.0 版软件：

- PKCS #11 库 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL 动态引擎 (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- JCE 提供程序 (SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 8 (8.3+)

在 x86_64 架构上下载适用于 RHEL 8 的 5.11.0 版软件：

- PKCS #11 库 (SHA256 checksum b95b9f588656fb14fd08bb66ce0e0da807b96daa38348dec07a508c9bef7403a)
- OpenSSL 动态引擎 (SHA256 checksum 7bb437b91a52e863b2b00ff7f427ce22522026daf757be873ee031ec6ffffd88)
- JCE 提供程序 (SHA256 checksum e0db887e05eb535314f4d99f21da12d87d35ebb8baf9726f4ce8f01d9df0ea01)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 8485b5a6d679767ca9b4f611718159a643cf3e85090a8e4d20fe53c3707e25c3)

RHEL 9 (9.2+)

在 x86_64 架构上下载适用于 RHEL 9 的 5.11.0 版软件：

- PKCS #11 库 (SHA256 checksum 87b56a20accf67df53a203b7f115655b2acfaec4516682d4976d9475b10bec8e)
- OpenSSL 动态引擎 (SHA256 checksum 83a6b58572e985df937beede4b10e867b0ac6050ace8010dc8d535be365d2747)
- JCE 提供程序 (SHA256 checksum ee95213d02d913250478d0793d6dd578e5c54d765e635c7468a49bdff4c2a6f3)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 7e168ed3bef8e9c5110645e9960680e9a57f7b94e16aec71422e3c67ebc58fb5)

Ubuntu 20.04 LTS

在 x86_64 架构上下载适用于 Ubuntu 20.04 LTS 的 5.11.0 版软件：

- PKCS #11 库 (SHA256 checksum abc3a339d1fe5850db65620804e9a910f8b4f913624ef9b7189f2f0df1825c01)
- OpenSSL 动态引擎 (SHA256 checksum 075fc3f9974d552f27ad67fa92c8abff31b756b9add875b8cd4957e6801583a4)
- JCE 提供程序 (SHA256 checksum 5de45c519133a0dae8da3ac01809db7974be25c14c15eb773fc5c972c0178c13)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 83e0e4505a063792c19feb3d4cfd032b9089091916168d92b0f51a967a007734)

Ubuntu 22.04 LTS

在 x86_64 架构上下载适用于 Ubuntu 22.04 LTS 的 5.11.0 版软件：

- PKCS #11 库 (SHA256 checksum b8f20be125c8530b2a7bd945956e9c04296fba5634af408b40be4e03bdbad72a)
- OpenSSL 动态引擎 (SHA256 checksum d728c156eb4ee5c67159e57d6b092785800baa5fb61c14d64f460a8b8f53a778)

- JCE 提供程序 (SHA256 checksum 44e943b8cd1176ad666e249342687744a280c6222df58b5a9f084c932f628284)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 8ccf5389d459611be813e42d7f9d040090f94f3fe88f9d110bcfb25e9619e4a7)

Windows Server 2016

在 x86_64 架构上下载适用于 Windows Server 2016 的 5.11.0 版软件：

- PKCS #11 库 (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- JCE 提供程序 (SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

Windows Server 2019

在 x86_64 架构上下载适用于 Windows Server 2019 的 5.11.0 版软件：

- PKCS #11 库 (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- JCE 提供程序 (SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- CloudHSM CLI (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

客户端 SDK 5.11.0 增加了新功能，提高了稳定性，并修复了所有 SDK 的错误。

平台支持

- 为所有软件开发工具包增加了对亚马逊 Linux 2023 和 RHEL 9 (9.2+) 的支持。
- 由于Ubuntu 18.04 LTS最近已停产，因此删除了对Ubuntu 18.04 LTS的支持。
- 由于亚马逊 Linux 最近已停产，因此删除了对它的支持。

CloudHSM CLI

- 添加了以下命令：

- 加密标志
- 加密验证
- 密钥导入 pem
- 密钥解包
- 钥匙包装

- key generate-file现在支持导出公钥。

OpenSSL 动态引擎

- 安装了 AWS CloudHSM OpenSSL 库版本 3.x 的平台现在支持 OpenSSL 动态引擎。这包括亚马逊 Linux 2023、RHEL 9（9.2+）和 Ubuntu 22.04。

JCE

- 增加了对 JDK 17 和 JDK 21 的支持。
- 增加了对用于 HMAC 操作的 AES 密钥的支持。
- 添加了新的按键属性`ID`。
- 引入了用于密钥耗尽的新`DataExceptionCause`变体:`DataExceptionCause.KEY_EXHAUSTED`.

错误修复/改进

- 将该`label`属性的最大长度从 126 个字符增加到 127 个字符。
- 修复了无法使用该 RsaOaep 机制解开 EC 密钥的错误。
- 解决了 JCE 提供程序中 getKey 操作的已知问题。有关详细信息，请参阅问题：GetKey 操作导致客户端 SDK 5 内存泄漏。
- 根据 FIPS 140-2，改进了所有已达到最大加密区块限制的 Triple DES 密钥的 SDK 的日志记录。
- 添加了 OpenSSL 动态引擎的已知问题。有关详细信息，请参阅 OpenSSL Dynamic Engine 的已知问题。

版本 5.10.0

Amazon Linux

在 x86_64 架构上下载适用于 Amazon Linux 的 5.10.0 版软件：

- PKCS #11 库 (SHA256 checksum d63adf3e96c19c2d894b2defcbadd916dbb0398993050b1358bd93a36aa5acab)
- OpenSSL 动态引擎 (SHA256 checksum 4daa3e591ffd5f7ce8ef3759c41deaa38867f5e5d21f15927aea83afb1678ac5)
- JCE 提供程序 (SHA256 checksum 6c1ac94d3080f1c609d9dafbcb14480911beef3a488c4ed6f2b11b377da9b477)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum c12617fcd7990ba53e96f477979b410e3a5f17842ca7a912861b8b820809b5b5)

Amazon Linux 2

在 x86_64 架构上下载适用于 Amazon Linux 2 的 5.10.0 版软件：

- PKCS #11 库 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL 动态引擎 (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE 提供程序 (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

在 ARM64 架构上下载适用于 Amazon Linux 2 的 5.10.0 版软件：

- PKCS #11 库 (SHA256 checksum 5d8dfd835f1ed5a7f5a4fcc8ecf81cfa29883aca7e2985de69b5db723ab663db)
- OpenSSL 动态引擎 (SHA256 checksum 91fb8efe2646bf0dbd9087554baa09554714e9d56e9bfd5c0dc3023a9f485574)
- JCE 提供程序 (SHA256 checksum 99f6e55c37fdf00085a816d46835aeff54470797b3b71f4d28a70dc79c9caf44)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 4a88ba9b4cf0dd5573f3dd88ab9dc257e4c486069cb529c5d554979ee2dd83af)

CentOS 7 (7.8+)

在 x86_64 架构上下载适用于 CentOS 7 的 5.10.0 版软件：

- PKCS #11 库 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL 动态引擎 (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE 提供程序 (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)

  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 7 (7.8+)

在 x86_64 架构上下载适用于 RHEL 7 的 5.10.0 版软件：

- PKCS #11 库 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL 动态引擎 (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- JCE 提供程序 (SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 8 (8.3+)

在 x86_64 架构上下载适用于 RHEL 8 的 5.10.0 版软件：

- PKCS #11 库 (SHA256 checksum 96afb7042a148ddc7a60ab6235b49e176d0460d1c2957bd76ca3d8406ac1cb03)
- OpenSSL 动态引擎 (SHA256 checksum 2caad2bffea8aef73c91ad422d09772ef830fe7f80a7be19020e6a107eadfbe8)
- JCE 提供程序 (SHA256 checksum 3543551f08fbe3900821ea2d4ea148b4e86e2334bc94d7ffef6f3b831457cd71)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 812eccaadfc490f13bcd0b0a835ef58f3a3d4344ad7e0a237de476dd24509525)

Ubuntu 18.04 LTS

在 x86_64 架构上下载适用于 Ubuntu 18.04 LTS 的 5.10.0 版软件：

- PKCS #11 库 (SHA256 checksum be4c61766b8b46e1f6c14c3dcf90aaab9f38240fcd9c68b4009704276c5f6f4a)
- OpenSSL 动态引擎 (SHA256 checksum 64bd8af827b6dc3786e8ad28858cbc4ef6a0fd42164a0945f427eddcf5f02858)
- JCE 提供程序 (SHA256 checksum 9fcbdf08e93641468588b608173f26f18781bbc029ed95b2e086da29a968cc00)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)

- CloudHSM CLI (SHA256 checksum 13808bddddb7eedeb2b8486d23a9976c7fa8d9220149a6b9400626bcaff3b513)

> **Note**
>
> 由于Ubuntu 18.04 LTS的生命周期最近已结束，因此在下一个版本中 AWS CloudHSM 将无法再支持该平台。

Ubuntu 20.04 LTS

在 x86_64 架构上下载适用于 Ubuntu 20.04 LTS 的 5.10.0 版软件：

- PKCS #11 库 (SHA256 checksum 99ae96504580ff85ed4958a582903a847f666bdaafafbe887a5a76db58f24500)
- OpenSSL 动态引擎 (SHA256 checksum 13e3f6fe086acf9617b163f66e3941f973daa583fb9322d16c396aa29fc3611d)
- JCE 提供程序 (SHA256 checksum 44562cebd9af1aa965840cd9bcb237e518d24c715b3c8bca1405c9c1871835e2)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum ab71b4ec531c5e6d05c91539c7edc1c07e6c748052ebf6200f148cb6812538c5)

Ubuntu 22.04 LTS

在 x86_64 架构上下载适用于 Ubuntu 22.04 LTS 的 5.10.0 版软件：

- PKCS #11 库 (SHA256 checksum ee331a44fbe4936ec98a3ae55d58e67ed38e8bbff0a4f4ce8b1bd8239b75877b)
- 此平台尚不支持 OpenSSL 动态引擎。
- JCE 提供程序 (SHA256 checksum 9e44d14dd33624f6fe36711633013e47e4a93f4d4635e08900546113ded56e3d)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum 2df361546848cd3f8965b1007dca42a0c959eb10d9e3f4995e8e1c852406751d)

Windows Server 2016

在 x86_64 架构上下载适用于 Windows Server 2016 的 5.10.0 版软件：

- PKCS #11 库 (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- JCE 提供程序 (SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)

    - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

Windows Server 2019

在 x86_64 架构上下载适用于 Windows Server 2019 的 5.10.0 版软件：

- PKCS #11 库 (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- JCE 提供程序 (SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)
    - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- CloudHSM CLI (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

客户端软件开发工具包 5.10.0 提高了稳定性，并修复了所有软件开发套件的错误。

CloudHSM CLI

- 添加了允许客户通过 CloudHSM CLI 管理密钥的新命令，包括：
    - 创建对称密钥和非对称密钥对
    - 共享和取消共享密钥
    - 根据密钥属性列出和筛选密钥
    - 设置密钥属性
    - 生成密钥引用文件
    - 删除密钥
- 改进了错误日志记录。
- 增加支持交互模式下的多行 unicode 命令。

错误修复/改进

- 提高了所有软件开发工具包的在导入、解包、派生和创建会话密钥方面的表现。
- 修复了 JCE 提供程序中的错误，避免退出时移除临时文件。
- 更换了集群内的 HSM 后，修复了会在特定情况下导致连接故障的错误。
- 修改了 JCE `getVersion` 输出格式，以处理较大数量的次要版本，并纳入补丁编号。

平台支持

- 通过 JCE、PKCS #11 和 CloudHSM CLI，增加了对 Ubuntu 22.04 的支持（尚未支持 OpenSSL 动态引擎）。

版本 5.9.0

Amazon Linux

在 x86_64 架构上下载适用于 Amazon Linux 的 5.9.0 版软件：

- PKCS #11 库 (SHA256 checksum 4f368be41f006b751ac41b14e1435c27841f60bbde0f032ec02a359fea637dcf)
- OpenSSL 动态引擎 (SHA256 checksum 81af0d34683825cd6ff844ccacf9c8f4842a4ba76e3875a89121d09a286b4490)
- JCE 提供程序 (SHA256 checksum e8e5bc09d8e0b3cb24f30ab420fe08902a19073012335ac94382ec55fcc45abd)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 17284144b45043204ce012fe8b62b1973f10068950abedbd9c2c6172ed0979c6)

Amazon Linux 2

在 x86_64 架构上下载适用于 Amazon Linux 2 的 5.9.0 版软件：

- PKCS #11 库 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL 动态引擎 (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE 提供程序 (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

在 ARM64 架构上下载适用于 Amazon Linux 2 的 5.9.0 版软件：

- PKCS #11 库 (SHA256 checksum 4337dca5a08c5194b1118fa197bb4a4f7988df4e1b961e6f2e367295ba99d61d)
- OpenSSL 动态引擎 (SHA256 checksum 4f08689934e877662a7ce64554fb04eb4b2c213b936018609ff187d100e34a85)
- JCE 提供程序 (SHA256 checksum b337b80271a2d308949d5911971fe6ad35df4e34876a481fcac347f1d897fe39)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)

- CloudHSM CLI (SHA256 checksum a4d466e6b5f74dcd283ba32c9dd87441941d5e5a05936b7c2b4cc7ef85eb1071)

CentOS 7 (7.8+)

在 x86_64 架构上下载适用于 CentOS 7 的 5.9.0 版软件：

- PKCS #11 库 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL 动态引擎 (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE 提供程序 (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 7 (7.8+)

在 x86_64 架构上下载适用于 RHEL 7 的 5.9.0 版软件：

- PKCS #11 库 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL 动态引擎 (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- JCE 提供程序 (SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 8 (8.3+)

在 x86_64 架构上下载适用于 RHEL 8 的 5.9.0 版软件：

- PKCS #11 库 (SHA256 checksum 081887f6ea1d9df9d1e409b2b5bde83e965c42229acbeb1f950c8fe478361edc)
- OpenSSL 动态引擎 (SHA256 checksum 6b0500a42fd57c39f076f14e5079f80145b6ebd2c441395761eb04600c07bda5)
- JCE 提供程序 (SHA256 checksum 2bc7ac26b259af92a65fbd5a30d5eb2a92ce0e70efe41feb53bf82f168aa90bb)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 79ecbe9b4c5316ccf447d8c59b76b5ac2cc854bd79cd50c1f29197aa8cb080db)

Ubuntu 18.04 LTS

在 x86_64 架构上下载适用于 Ubuntu 18.04 LTS 的 5.9.0 版软件：

- PKCS #11 库 (SHA256 checksum bc6d2227edd7b5a83fed32741fbacbb1756d5df89ebb3435d96f0609a180db65)
- OpenSSL 动态引擎 (SHA256 checksum 2d6a26434fa6faf337f1dfb42de033220fa405a82d4540e279639a03b3ee6e9d)
- JCE 提供程序 (SHA256 checksum e12aef122f490e9026452ce31c25625b1accb9a5866b3d470488f10f047f1873)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum f0bcabe594db3e8ff86cc0f65c2a10858d34452eb6b9fc33d7aac05c0f5f4f30)

Ubuntu 20.04 LTS

在 x86_64 架构上下载适用于 Ubuntu 20.04 LTS 的 5.9.0 版软件：

- PKCS #11 库 (SHA256 checksum 15dde8182f432de9e7d369b05e384e1f2d80dcca85db3b16ecc26cdef1a34bb9)
- OpenSSL 动态引擎 (SHA256 checksum c8ba94a999038af87d4905b7c1feb4cc87e20d1776a32ef6f6d11ee000b5a896)
- JCE 提供程序 (SHA256 checksum de33cd3e8130a06d9da5207079533aac8276a1319ac435a3737b4f65bd8fb972)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum cfa31535ad9a99a5113496c06fbace38e9593491aca9bb031a18b51075973e68)

Windows Server 2016

在 x86_64 架构上下载适用于 Windows Server 2016 的 5.9.0 版软件：

- PKCS #11 库 (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- JCE 提供程序 (SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

Windows Server 2019

在 x86_64 架构上下载适用于 Windows Server 2019 的 5.9.0 版软件：

- PKCS #11 库 (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- JCE 提供程序 (SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)
  - 适用于 AWS CloudHSM的 Javadocs (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- CloudHSM CLI (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

客户端软件开发工具包 5.9.0 提高了稳定性，并修复了所有软件开发套件的错误。已对所有软件开发套件进行了优化，以在确定 HSM 不可用时立即告知应用程序此操作故障。此版本增强了 JCE 性能。

JCE 提供程序

- 增强性能
- 修复了关于会话池耗尽的已知问题

版本 3.4.4

要在 Linux 平台上升级客户端软件开发工具包 3，您必须使用批处理命令来同时升级客户端进程守护程序和所有库。若要获取有关升级的更多信息，请参阅客户端软件开发工具包 3 升级。

> **Note**
>
> 客户端 SDK 3 及其相关的命令行工具（密钥管理实用程序和 CloudHSM 管理实用程序）仅在 HSM 类型 hsm1.medium 中可用。有关详细信息，请参阅 AWS CloudHSM 集群模式和 HSM 类型。

要下载软件，请选择您的首选操作系统对应的选项卡，然后选择指向每个软件包的链接。

Amazon Linux

下载适用于 Amazon Linux 的版本 3.4.4 软件：

- AWS CloudHSM 客户 (SHA256 checksum 900de424d70f41e661aa636f256a6a79cc43bea6b0fe6eb95c2aaa63e5289505)
- PKCS #11 库 (SHA256 checksum a3f93f084d59fee5d7c859292bc02cb7e7f15fb06e971171ebf9b52bbd229c30)
- OpenSSL 动态引擎 (SHA256 checksum 8db07b9843d49016b0b6fec46d39881d94e426fcaae1cee2747be14af9313bb0)
- JCE 提供程序 (SHA256 checksum 360617c55bf4caa8e6e78ede079ca68cf9ef11473e7918154c22ba908a219843)

- AWS CloudHSM 管理实用程序 (SHA256 checksum c9961ffe38921131bd6f3702e10d73588e68b8ab10fbb241723e676f4fa8c4fa)

Amazon Linux 2

下载适用于 Amazon Linux 2 的版本 3.4.4 软件：

- AWS CloudHSM 客户 (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS #11 库 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL 动态引擎 (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE 提供程序 (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM 管理实用程序 (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 6

AWS CloudHSM 不支持 CentOS 6 和客户端 SDK 版本 3.4.4。

使用 CentOS 6 the section called “版本 3.2.1”，或选择支持的平台。

CentOS 7 (7.8+)

下载适用于 CentOS 7 的版本 3.4.4 软件：

- AWS CloudHSM 客户 (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS #11 库 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL 动态引擎 (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE 提供程序 (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM 管理实用程序 (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 8

下载适用于 CentOS 8 的版本 3.4.4 软件：

- AWS CloudHSM 客户 (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)
- PKCS #11 库 (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)
- JCE 提供程序 (SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM 管理实用程序 (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

> **Note**
>
> 由于近期 CentOS 8 终止服务，下一版本将不再支持此平台。

RHEL 6

AWS CloudHSM 不支持带有客户端 SDK 版本 3.4.4 的 E RedHat nterprise Linux 6。

the section called “版本 3.2.1”用于 RedHat 企业 Linux 6 或选择支持的平台。

RHEL 7 (7.8+)

下载适用于 RedHat 企业 Linux 7 的 3.4.4 版软件：

- AWS CloudHSM 客户 (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- PKCS #11 库 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL 动态引擎 (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- JCE 提供程序 (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM 管理实用程序 (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

RHEL 8 (8.3+)

下载适用于 RedHat 企业 Linux 8 的 3.4.4 版软件：

- AWS CloudHSM 客户 (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)
- PKCS #11 库 (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)

- JCE 提供程序 (SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM 管理实用程序 (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

Ubuntu 16.04 LTS

下载适用于 Ubuntu 16.04 LTS 的版本 3.4.4 软件：

- AWS CloudHSM 客户 (SHA256 checksum 317c92c2e0b5d60afab1beb947f053d13ddaacb994cccc2c2b898e997ece29b9)
- PKCS #11 库 (SHA256 checksum 91451c420c51488a022569fd32f052a3b988a2883ea4c2ac952acb61a2fea37c)
- OpenSSL 动态引擎 (SHA256 checksum 4098771ad0e38df9bf14d50520ca49b9395f819f0387e2bc3b0e61abb5888e66)
- JCE 提供程序 (SHA256 checksum e136ff183271c2f9590a9fccb8261a7eb809506686b070e3854df1b8686c6641)
- AWS CloudHSM 管理实用程序 (SHA256 checksum cbf24a4032f393a913a9898b1b27036392104e8e05d911cab84049b2bcca2541)

> **Note**
>
> 由于 Ubuntu 16.04 即将终止服务，下一版本将不再支持此平台。

Ubuntu 18.04 LTS

下载适用于 Ubuntu 18.04 LTS 的版本 3.4.4 软件：

- AWS CloudHSM 客户 (SHA256 checksum cf57d5e0e95efbf032aac8887aebd59ac8cc80e97c69e7c39fdad40873374fe8)
- PKCS #11 库 (SHA256 checksum 428f8bdad7925db5401112f707942ee8f3ca554f4ab53fa92237996e69144d2f)
- JCE 提供程序 (SHA256 checksum 1ff17b8f7688e84f7f0bfc96383564dca598a1cab2f2c52c888d0361682f2b9e)
- AWS CloudHSM 管理实用程序 (SHA256 checksum afe253046146ed6177c520b681efc680dac1048c4a95b3d8ad0f305e79bbe93e)

Windows Server

AWS CloudHSM 支持 64 位版本的 Windows Server 2012、Windows Server 2012 R2、Windows Server 2016 和 Windows Server 2019。适用于 Windows Server 的 AWS CloudHSM 3.4.4 客户端

软件包括所需的 CNG 和 KSP 提供商。有关详细信息，请参阅安装和配置 AWS CloudHSM 客户端 (Windows)。下载适用于 Windows Server 的最新版本 (3.4.4) 软件：

- AWS CloudHSM 适用于 Windows 服务器 (SHA256 checksum d51a7db588e9121d8f0b0351606bd986e1c4de6547f2c8235200dc8a5ffbe53e)
- AWS CloudHSM 管理实用程序 (SHA256 checksum 0c12d7da9086735cdf189535937a8e036163009c5018dcaf2ee9cdbb6bd4c06f)

3.4.4 版增加了 JCE 提供程序更新。

AWS CloudHSM 客户机软件

- 更新了版本以保持一致性。

PKCS #11 库

- 更新了版本以保持一致性。

OpenSSL 动态引擎

- 更新了版本以保持一致性。

JCE 提供程序

- 将 log4j 版本更新到 2.17.1。

适用于 Windows（CNG 和 KSP 提供程序）

- 更新了版本以保持一致性。

已弃用版本

5.8.0 及更早版本已弃用。我们建议不建议在生产工作负载中使用已弃用版本。我们不为已弃用的版本提供向后兼容更新，也不托管已弃用的版本供下载。如果弃用版本对您的生产造成影响，则您必须升级以修复软件。

已弃用的客户端 SDK 5 版本

本节列出了已弃用的客户端 SDK 5 版本。

版本 5.8.0

版本 5.8.0 引入了 CloudHSM CLI 的仲裁身份验证、JSSE 的 SSL/TLS 分流、PKCS #11 多插槽支持、JCE 多集群/多用户支持、JCE 密钥提取、JCE 的 KeyFactory 支持、非终端返回代码的新重试配置，提高了稳定性，修复了所有软件开发套件错误。

PKCS #11 库

- 增加多插槽配置支持。

JCE 提供程序

- 添加了基于配置的密钥提取。
- 增加了对多集群和多用户配置的支持。
- 增加了对 SSL 和 TSL 分流的支持。
- 添加了对 A NoPadding ES/CBC/ 的解包支持。
- 添加了新类型的关键工厂： SecretKeyFactory 和 KeyFactory。

CloudHSM CLI

- 增加了对仲裁身份验证的支持

版本 5.7.0

5.7.0 版引入了 CloudHSM CLI，并纳入了一种新的、基于加密消息的身份验证代码 (CMAC) 算法。此次发布增加了 Amazon Linux 2 上的 ARM 架构。JCE 提供程序 Javadocs 现已可用于 AWS CloudHSM。

PKCS #11 库

- 改进了稳定性，进行了错误修复。
- 现已支持 Amazon Linux 2 的 ARM 架构。
- 算法
  - CKM_AES_CMAC（签名和验证）

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。
- 现已支持 Amazon Linux 2 的 ARM 架构。

JCE 提供程序

- 改进了稳定性，进行了错误修复。
- 算法
  - AESCMAC

版本 5.6.0

版本 5.6.0 包含对 PKCS #11 库和 JCE 提供程序的新机制支持。此外，版本 5.6 还支持 Ubuntu 20.04。

PKCS #11 库

- 改进了稳定性，进行了错误修复。
- 机制
  - CKM_RSA_X_509，适用于加密、解密、签名和验证模式

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。

JCE 提供程序

- 改进了稳定性，进行了错误修复。
- 密码
  - RSA/ECB/NoPadding，用于加密和解密模式

支持的密钥

- 带 secp224r1 和 secp521r1 曲线的 EC

平台支持

- 增加了对 Ubuntu 20.04 的支持

版本 5.5.0

5.5.0 版增加了对 OpenJDK 11、Keytool 和 Jarsigner 集成的支持，并向 JCE 提供程序添加了其他机制。解决了一个已知问题，该问题涉及 KeyGenerator 类错误地将密钥大小参数解释为字节数而不是位数。

PKCS #11 库

- 改进了稳定性，进行了错误修复。

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。

JCE 提供程序

- 支持 Keytool 和 Jarsigner 实用程序
- 所有平台上都支持 OpenJDK 11
- 密码
  - AES/CBC/ NoPadding 加密和解密模式
  - AES/ECB/PKCS5Padding 加密和解密模式
  - AES/CTR/ NoPadding 加密和解密模式
  - AES/GCM/ NoPadding Wrap and Unwrap 模式
  - DESede/ECB/PKCS5Padding 加密和解密模式
  - desede/cbc/ 加密和解NoPadding 密模式
  - aesWrap/ec NoPadding b/ Wrap and Unwrap 模式
  - AESWrap/ECB/PKCS5Padding 包装和解包模式
  - aesWrap/ec ZeroPadding b/ Wrap and Unwrap 模式
  - RSA/ECB/PKCS1Padding 包装和解包模式
  - RSA/ECB/OAEPPadding 包装和解包模式
  - RSA/ECB/OAEPWithSHA-1ANDMGF1Padding 包装和解包模式

- RSA/ECB/OAEPWithSHA-224ANDMGF1Padding 包装和解包模式
- RSA/ECB/OAEPWithSHA-256ANDMGF1Padding 包装和解包模式
- RSA/ECB/OAEPWithSHA-384ANDMGF1Padding 包装和解包模式
- RSA/ECB/OAEPWithSHA-512ANDMGF1Padding 包装和解包模式
- RSAAESWrap/ECB/OAEPPadding 包装和解包模式
- RSAAESWrap/ECB/OAEPWithSHA-1ANDMGF1Padding 包装和解包模式
- RSAAESWrap/ECB/OAEPWithSHA-224ANDMGF1Padding 包装和解包模式
- RSAAESWrap/ECB/OAEPWithSHA-256ANDMGF1Padding 包装和解包模式
- RSAAESWrap/ECB/OAEPWithSHA-384ANDMGF1Padding 包装和解包模式
- RSAAESWrap/ECB/OAEPWithSHA-512ANDMGF1Padding 包装和解包模式

- KeyFactory 和 SecretKeyFactory
  - RSA 2048 位到 4096 位 RSA 密钥，增量为 256 位
  - AES – 128、192 和 256 位 AES 密钥。
  - NIST 曲线 secp256r1 (P-256)、secp384r1 (P-384) 和 secp256k1的 EC 密钥。
  - DESede (3DES)
  - GenericSecret
  - HMAC — 支持 SHA1、SHA224、SHA256、SHA384、SHA512 哈希
- 签署/验证
  - RSASSA-PSS
  - SHA1withRSA/PSS
  - SHA224withRSA/PSS
  - SHA256withRSA/PSS
  - SHA384withRSA/PSS
  - SHA512withRSA/PSS
  - SHA1withRSAandMGF1
  - SHA224withRSAandMGF1
  - SHA256withRSAandMGF1
  - SHA384withRSAandMGF1
  - SHA512withRSAandMGF1

版本 5.4.2

5.4.2 版提高了稳定性，并修复了所有软件开发套件的错误。这也是 CentOS 8 平台的最新版本。有关更多信息，请参阅 CentOS 网站。

PKCS #11 库

- 改进了稳定性，进行了错误修复。

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。

JCE 提供程序

- 改进了稳定性，进行了错误修复。

版本 5.4.1

5.4.1 版解决了 PKCS #11 库的一个已知问题。这也是 CentOS 8 平台的最新版本。有关更多信息，请参阅 CentOS 网站。

PKCS #11 库

- 改进了稳定性，进行了错误修复。

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。

JCE 提供程序

- 改进了稳定性，进行了错误修复。

版本 5.4.0

5.4.0 版增加了对所有平台的 JCE 提供程序的初始支持。JCE 提供程序可与 OpenJDK 8 兼容。

PKCS #11 库

- 改进了稳定性，进行了错误修复。

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。

JCE 提供程序

- 密钥类型
  - RSA 2048 位到 4096 位 RSA 密钥，增量为 256 位。
  - AES – 128、192 和 256 位 AES 密钥。
  - NIST 曲线 secp256r1 (P-256)、secp384r1 (P-384) 和 secp256k1 的 ECC 密钥。
  - DESede (3DES)
  - HMAC — 支持 SHA1、SHA224、SHA256、SHA384、SHA512 哈希。
- 密码（仅限加密和解密）
  - AES/GCM/ NoPadding
  - AES/ECB/ NoPadding
  - AES/CBC/PKCS5Padding
  - desede/ECB/ NoPadding
  - DESede/CBC/PKCS5Padding
  - AES/CTR/ NoPadding
  - RSA/ECB/PKCS1Padding
  - RSA/ECB/OAEPPadding
  - RSA/ECB/OAEPWithSHA-1ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-224ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-256ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-384ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-512ANDMGF1Padding
- 摘要
  - SHA-1

  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- 签署/验证
  - NONEwithRSA
  - SHA1withRSA
  - SHA224withRSA
  - SHA256withRSA
  - SHA384withRSA
  - SHA512withRSA
  - NONEwithECDSA
  - SHA1withECDSA
  - SHA224withECDSA
  - SHA256withECDSA
  - SHA384withECDSA
  - SHA512withECDSA
- 与 Java 集成 KeyStore

版本 5.3.0

PKCS #11 库

- 改进了稳定性，进行了错误修复。

OpenSSL 动态引擎

- 添加了“通过 P-256、P-384 和 secp256k1 曲线执行 ECDSA 验证/签名”的支持。
- 增加对以下平台的支持：亚马逊 Linux、亚马逊 Linux 2、Centos 7.8+、RHEL 7（7.8+）。
- 增加了对 OpenSSL 1.0.2 版的支持。
- 改进了稳定性，进行了错误修复。

JCE 提供程序

- 密钥类型
  - RSA 2048 位到 4096 位 RSA 密钥，增量为 256 位。
  - AES – 128、192 和 256 位 AES 密钥。
  - NIST 曲线 secp256r1 (P-256)、secp384r1 (P-384) 和 secp256k1 的 ECC 密钥对。
  - DESede (3DES)
  - HMAC — 支持 SHA1、SHA224、SHA256、SHA384、SHA512 哈希。
- 密码（仅限加密和解密）
  - AES/GCM/ NoPadding
  - AES/ECB/ NoPadding
  - AES/CBC/PKCS5Padding
  - desede/ECB/ NoPadding
  - DESede/CBC/PKCS5Padding
  - AES/CTR/ NoPadding
  - RSA/ECB/PKCS1Padding
  - RSA/ECB/OAEPPadding
  - RSA/ECB/OAEPWithSHA-1ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-224ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-256ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-384ANDMGF1Padding
  - RSA/ECB/OAEPWithSHA-512ANDMGF1Padding
- 摘要
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- 签署/验证
  - NONEwithRSA

- SHA1withRSA
- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA
- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

- 与 Java 集成 KeyStore

版本 5.2.1

PKCS #11 库

- 改进了稳定性，进行了错误修复。

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。

版本 5.2.0

5.2.0 版对 PKCS #11 库增加了“对其他密钥类型和机制”支持。

PKCS #11 库

密钥类型

- ECDSA — P-224、P-256、P-384、P-521 和 secp256k1 曲线
- 三重 DES (3DES)

机制

- 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
- 用于加密/解密的 CKM_RSA_PKCS

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。

版本 5.1.0

5.1.0 版对 PKCS #11 库增加了“对其他机制”的支持。

PKCS #11 库

机制

- 用于包装/解包的 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_SHA224_HMAC
- CKM_SHA224_HMAC
- CKM_SHA512_HMAC
- CKM_RSA_PKCS_OAEP 仅限包装/解包
- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_KEY_WRAP_NO_PAD
- CKM_CLOUDHSM_AES_KEY_KEY_WRAP_PKCS5_PAD
- CKM_CLOUDHSM_AES_KEY_KEY_WRAP_ZERO_PAD

API 操作

- C_ CreateObject
- C_ DeriveKey
- C_ WrapKey
- C_ UnWrapKey

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。

版本 5.0.1

5.0.1 版增加了对 OpenSSL 动态引擎的初始支持。

PKCS #11 库

- 改进了稳定性，进行了错误修复。

OpenSSL 动态引擎

- OpenSSL 动态引擎的初始版本。
- 此版本为密钥类型和 OpenSSL API 提供入门支持：
  - 针对 2048、3072 和 4096 位密钥的 RSA 密钥生成
  - OpenSSL API：
    - 使用 RSA PKCS 与 SHA1/224/256/384/512 和 RSA PSS 签名
    - RSA 密钥生成

  有关更多信息，请参阅 OpenSSL 动态引擎。
- 支持的平台：CentOS 8.3+、红帽企业 Linux (RHEL) 8.3+ 和 Ubuntu 18.04 LTS
  - 需要：OpenSSL 1.1.1

  有关更多信息，请参阅支持的平台。
- 在 CentOS 8.3+、红帽企业 Linux (RHEL) 8.3 和 Ubuntu 18.04 LTS 上支持 SSL/TLS 卸载，包括 NGINX 1.19（适用于部分密码套件）。

  有关更多信息，请参阅使用 Linux 上的 SSL/TLS 卸载。

版本 5.0.0

5.0.0 是第一个发行版本。

PKCS #11 库

- 这是初始版本。

客户软件开发工具包 5.0.0 版中的 PKCS #11 库入门支持

本节详细介绍了客户端软件开发工具包 5.0.0 版的密钥类型、机制、API 操作和属性支持。

密钥类型：

- AES– 128、192 和 256 位 AES 密钥
- RSA– 2048 位到 4096 位 RSA 密钥，增量为 256 位

机制：

- 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

属性：

- GenerateKeyPair
  - 所有 RSA 密钥属性
- GenerateKey
  - 所有 AES 密钥属性
- GetAttributeValue
  - 所有 RSA 密钥属性
  - 所有 AES 密钥属性

示例：

- 生成密钥（AES、RSA、EC）
- 列出密钥属性
- 使用 AES GCM 加密和解密数据
- 使用 RSA 对数据进行签名和验证

已弃用的客户端 SDK 3 版本

本节列出了已弃用的客户端 SDK 3 版本。

版本 3.4.3

3.4.3 版增加了 JCE 提供程序更新。

AWS CloudHSM 客户机软件

- 更新了版本以保持一致性。

PKCS #11 库

- 更新了版本以保持一致性。

OpenSSL 动态引擎

- 更新了版本以保持一致性。

JCE 提供程序

- 将 log4j 版本更新到 2.17.0。

适用于 Windows（CNG 和 KSP 提供程序）

- 更新了版本以保持一致性。

版本 3.4.2

3.4.2 版增加了 JCE 提供程序更新。

AWS CloudHSM 客户机软件

- 更新了版本以保持一致性。

PKCS #11 库

- 更新了版本以保持一致性。

OpenSSL 动态引擎

- 更新了版本以保持一致性。

JCE 提供程序

- 将 log4j 版本更新到 2.16.0。

适用于 Windows（CNG 和 KSP 提供程序）

- 更新了版本以保持一致性。

版本 3.4.1

3.4.1 版增加了 JCE 提供程序更新。

AWS CloudHSM 客户机软件

- 更新了版本以保持一致性。

PKCS #11 库

- 更新了版本以保持一致性。

OpenSSL 动态引擎

- 更新了版本以保持一致性。

JCE 提供程序

- 将 log4j 版本更新到 2.15.0。

适用于 Windows（CNG 和 KSP 提供程序）

- 更新了版本以保持一致性。

版本 3.4.0

3.4.0 版本增加了对所有组件的更新。

AWS CloudHSM 客户机软件

- 改进了稳定性，进行了错误修复。

PKCS #11 库

- 改进了稳定性，进行了错误修复。

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。

JCE 提供程序

- 改进了稳定性，进行了错误修复。

适用于 Windows（CNG 和 KSP 提供程序）

- 改进了稳定性，进行了错误修复。

版本 3.3.2

3.3.2 版解决了一项 client_info script 问题。

AWS CloudHSM 客户机软件

- 更新了版本以保持一致性。

PKCS #11 库

- 更新了版本以保持一致性。

OpenSSL 动态引擎

- 更新了版本以保持一致性。

JCE 提供程序

- 更新了版本以保持一致性。

适用于 Windows（CNG 和 KSP 提供程序）

- 更新了版本以保持一致性。

版本 3.3.1

3.3.1 版本增加了对所有组件的更新。

AWS CloudHSM 客户机软件

- 改进了稳定性，进行了错误修复。

PKCS #11 库

- 改进了稳定性，进行了错误修复。

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。

JCE 提供程序

- 改进了稳定性，进行了错误修复。

适用于 Windows（CNG 和 KSP 提供程序）

- 改进了稳定性，进行了错误修复。

版本 3.3.0

3.3.0 版本增加了双因素身份验证 (2FA) 和其他改进。

AWS CloudHSM 客户机软件

- 为加密员（CO）添加了双重身份验证。有关更多信息，请参阅加密员（CO）双重身份验证管理。
- 移除了对 RedHat 企业 Linux 6 和 CentOS 6 的平台支持。有关详细信息，请参阅 Linux 支持。
- 添加了独立版本 CMU，可与客户端软件开发工具包 5 或客户端软件开发工具包 3 配合使用。这与 3.3.0 版客户端进程守护程序中包含的 CMU 版本相同，现在您不需要下载客户端守护程序，即可下载 CMU。

PKCS #11 库

- 改进了稳定性，进行了错误修复。
- 移除了对 RedHat 企业 Linux 6 和 CentOS 6 的平台支持。有关详细信息，请参阅 Linux 支持。

OpenSSL 动态引擎

- 更新了版本以保持一致性
- 移除了对 RedHat 企业 Linux 6 和 CentOS 6 的平台支持。有关详细信息，请参阅 Linux 支持。

JCE 提供程序

- 改进了稳定性，进行了错误修复。

- 移除了对 RedHat 企业 Linux 6 和 CentOS 6 的平台支持。有关详细信息，请参阅 Linux 支持。

适用于 Windows（CNG 和 KSP 提供程序）

- 更新了版本以保持一致性

版本 3.2.1

版本 3.2.1 添加了 PKCS #11 库和 PKCS #11 标准的 AWS CloudHSM 实现、新平台和其他改进之间的合规性分析。

AWS CloudHSM 客户机软件

- 增加了对 CentOS 8、RHEL 8 和 Ubuntu 18.04 LTS 的平台支持。有关更多信息，请参阅 ???。

PKCS #11 库

- PKCS #11 客户端软件开发工具包 3.2.1 的库合规性报告
- 增加了对 CentOS 8、RHEL 8 和 Ubuntu 18.04 LTS 的平台支持。有关更多信息，请参阅 ???。

OpenSSL 动态引擎

- 不支持 CentOS 8、RHEL 8 以及 Ubuntu 18.04 LTS。有关更多信息，请参阅 ???。

JCE 提供程序

- 增加了对 CentOS 8、RHEL 8 和 Ubuntu 18.04 LTS 的平台支持。有关更多信息，请参阅 ???。

适用于 Windows（CNG 和 KSP 提供程序）

- 改进了稳定性，进行了错误修复。

版本 3.2.0

3.2.0 版本增加了对替换式密码的支持和其他改进。

AWS CloudHSM 客户机软件

- 新增支持：使用命令行工具时隐藏密码。有关更多信息，请参阅 loginHSM 和 logoutHSM (cloudhsm_mgmt_util) 以及 loginHSM 和 logoutHSM (key_mgmt_util)。

PKCS #11 库

- 新增支持：对此前不支持的部分 PKCS #11 机制中的软件内的大数据执行哈希处理。有关更多信息，请参阅受支持的机制。

OpenSSL 动态引擎

- 改进了稳定性，进行了错误修复。

JCE 提供程序

- 更新了版本以保持一致性。

适用于 Windows（CNG 和 KSP 提供程序）

- 改进了稳定性，进行了错误修复。

版本 3.1.2

3.1.2 版增加了 JCE 提供程序更新。

AWS CloudHSM 客户机软件

- 更新了版本以保持一致性

PKCS #11 库

- 更新了版本以保持一致性

OpenSSL 动态引擎

- 更新了版本以保持一致性

JCE 提供程序

- 将 log4j 版本更新到 2.13.3。

适用于 Windows（CNG 和 KSP 提供程序）

- 更新了版本以保持一致性

版本 3.1.1

AWS CloudHSM 客户机软件

- 更新了版本以保持一致性。

PKCS #11 库

- 更新了版本以保持一致性。

OpenSSL 动态引擎

- 更新了版本以保持一致性。

JCE 提供程序

- 缺陷修复和性能改进。

Windows（CNG、KSP）

- 更新了版本以保持一致性。

版本 3.1.0

版本 3.1.0 添加了符合标准的 AES 密钥包装。

AWS CloudHSM 客户机软件

- 有关升级的新要求：客户端的版本必须与您正在使用的任意软件库的版本匹配。要进行升级，您必须使用批处理命令来同时升级客户端和所有库。有关更多信息，请参阅客户端软件开发工具包 3 升级。
- Key_mgmt_util (KMU) 包括以下更新：

- 添加了两种新的 AES 密钥包装方法 - 符合标准的 AES 密钥包装（零填充）和无填充的 AES 密钥包装。有关更多信息，请参阅 wrapKey 和 unwrapKey。
- 禁用了在使用 AES_KEY_WRAP_PAD_PKCS5 包装密钥时指定自定义 IV 的功能。有关更多信息，请参阅 AES 密钥包装。

PKCS #11 库

- 添加了两种新的 AES 密钥包装方法 - 符合标准的 AES 密钥包装（零填充）和无填充的 AES 密钥包装。有关更多信息，请参阅 AES 密钥包装。
- 您可以配置 RSA-PSS 签名的 salt 长度。要了解如何使用此功能，请参阅上的 RSA-PSS 签名的可配置盐长度。 GitHub

OpenSSL 动态引擎

- 重大更改：带 SHA1 的 TLS 1.0 和 1.2 密码套件不适用于 OpenSSL Engine 3.1.0。此问题将很快得到解决。
- 如果您打算在 RHEL 6 或 CentOS 6 上安装 OpenSSL 动态引擎库，请参阅有关这些操作系统上已安装的默认 OpenSSL 版本的已知问题。
- 改进了稳定性和错误修复

JCE 提供程序

- 重大更改：为了解决 Java Cryptography Extension (JCE) 的合规性问题，AES 包装和解开包装功能现在可以正确地使用 AESWrap 算法而不是 AES 算法。这意味着对于 AES/ECB 和 AES/CBC 机制，`Cipher.WRAP_MODE` 和 `Cipher.UNWRAP_MODE` 不会再取得成功。

  要升级到客户端版本 3.1.0，您必须更新您的代码。如果您有已包装的密钥，则必须特别注意用于解开包装的机制以及 IV 默认值的变化。如果您使用客户端版本 3.0.0 或更早版本包装了密钥，那么在 3.1.1 中，您必须使用 AESWrap/ECB/PKCS5Padding 来解开现有密钥的包装。有关更多信息，请参阅 AES 密钥包装。
- 您可以列出 JCE 提供程序中带相同标签的多个密钥。要了解如何遍历所有可用密钥，请参阅 “查找所有按键” GitHub。
- 可以在密钥创建过程中为属性设置更受限的值，包括为公有密钥和私有密钥指定不同的标签。有关更多信息，请参阅支持的 Java 属性。

Windows（CNG、KSP）

- 改进了稳定性，进行了错误修复。

E nd-of-life 版本

AWS CloudHSM 宣布不再与该服务兼容的版本的生命周期终止。为了保护您的应用程序的安全，我们保留主动拒绝 end-of-life 发布连接的权利。

- 目前，客户端 SDK 没有版本 end-of-life 发布。

文档历史记录

本主题介绍了对 AWS CloudHSM 用户指南 的重要更新。

主题

最近的更新

下表介绍了自 2018 年 4 月起对此文档的一些重要更改。除了此处列出的主要更改以外，我们还会经常更新文档，以改进说明和示例以及处理您发送给我们的反馈意见。要获得有关重要更改的通知，请使用右上角的链接来订阅 RSS 源。

有关新版本的详细信息，请参阅 AWS CloudHSM 客户端 SDK 的下载内容

变更	说明	日期
新的 HSM 类型和集群模式	启动了新的 HSM 类型 (hsm2m.med ium) 和新的集群模式（非 FIPS）。	2024 年 6 月 10 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.12.0。	2024 年 3 月 20 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.11.0。	2024 年 1 月 17 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.10.0。	2023 年 7 月 28 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.9.0。	2023 年 5 月 23 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.8.0。	2023 年 3 月 16 日

增加了新版本	已发布 AWS CloudHSM 客户端版本 5.7.0。	2022 年 11 月 16 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.6.0。	2022 年 9 月 1 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.5.0。	2022 年 5 月 13 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.4.2。	2022 年 3 月 18 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.4.1。	2022 年 2 月 10 日
增加了新版本	发布了适用于 Windows 平台的 AWS CloudHSM JCE 提供程序版本 5.4.0。	2022 年 2 月 1 日
增加了新版本	发布了 AWS CloudHSM 客户端版本 5.4.0，该版本增加了对所有 Linux 平台的 JCE 提供程序的初始支持。	2022 年 1 月 28 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.3.0。	2022 年 1 月 3 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.4.4。	2022 年 1 月 3 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.4.3。	2021 年 12 月 20 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.4.2。	2021 年 12 月 15 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.4.1。	2021 年 12 月 10 日

增加了新版本	已发布 AWS CloudHSM 客户端版本 5.2.1。	2021 年 10 月 4 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.4.0。	2021 年 8 月 25 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.2.0。	2021 年 8 月 3 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.3.2。	2021 年 7 月 2 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.1.0。	2021 年 6 月 1 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.3.1。	2021 年 4 月 26 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.0.1。	2021 年 4 月 8 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 5.0.0。	2021 年 3 月 12 日
新增内容	添加了接口 VPC 终端节点，这是一项 AWS 功能，允许您在 VPC 之间创建私有连接，AWS CloudHSM 无需通过互联网或 NAT 设备、VPN 连接或 AWS Direct Connect 连接进行访问。	2021 年 2 月 10 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.3.0。	2021 年 2 月 3 日
添加内容	添加了托管备份保留功能，该功能可自动删除旧备份。	2020 年 11 月 18 日

添加内容	添加了一份合规性报告，该报告分析了采用 PKCS #11 标准的 PKCS #11 库的 AWS CloudHSM 客户端 SDK 3.2.1 实现。	2020 年 10 月 29 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.2.1。	2020 年 10 月 8 日
新增内容	在 AWS CloudHSM中添加了描述密钥同步设置的文档。	2020 年 9 月 1 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.2.0。	2020 年 8 月 31 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.1.2。	2020 年 7 月 30 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.1.1。	2020 年 6 月 3 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.1.0。	2020 年 5 月 21 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 3.0.1。	2020 年 4 月 20 日
增加了新版本	已发布适用于 Windows 服务器平台的 AWS CloudHSM 客户端版本 3.0.0。	2019 年 10 月 30 日
增加了新版本	已发布适用于除 Windows 之外的所有平台的 AWS CloudHSM 客户端版本 3.0.0。	2019 年 10 月 22 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 2.0.4。	2019 年 8 月 26 日

增加了新版本	已发布 AWS CloudHSM 客户端版本 2.0.3。	2019 年 5 月 13 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 2.0.1。	2019 年 3 月 21 日
增加了新版本	已发布 AWS CloudHSM 客户端版本 2.0.0。	2019 年 2 月 6 日
增加了区域支持	增加了对欧洲（斯德哥尔摩）和 AWS GovCloud （美国东部）区域的 AWS CloudHSM 支持。	2018 年 12 月 19 日
增加了新版本	已发布适用于 Windows 的 AWS CloudHSM 客户端版本 1.1.2。	2018 年 11 月 20 日
更新的已知问题	在故障排除指南中增加了新内容。	2018 年 11 月 8 日
增加了新版本	已发布适用于 Linux 平台的 AWS CloudHSM 客户端版本 1.1.2。	2018 年 11 月 8 日
增加了区域支持	增加了对欧洲（巴黎）和亚太地区（首尔）地区的 AWS CloudHSM 支持。	2018 年 10 月 24 日
新增内容	增加了删除和恢复 AWS CloudHSM 备份的功能。	2018 年 10 月 10 日
新增内容	在 Amazon 日志中添加了自动审核 CloudWatch 日志传输。	2018 年 8 月 13 日
新增内容	增加了跨区域复制集 AWS CloudHSM 群备份的功能。	2018 年 7 月 30 日

增加了区域支持	增加了对欧洲（伦敦）地区的 AWS CloudHSM 支持。	2018 年 13 月 6 日
新增内容	增加了对亚马逊 Linux 2、红帽企业 Linux (RHEL) 6、红帽企业 Linux (RHEL) 7、CentOS 6、CentOS 7 和 Ubuntu 16.04 LTS 的 AWS CloudHSM 客户端和库支持。	2018 年 5 月 10 日
增加了新版本	添加了 Windows AWS CloudHSM 客户端。	2018 年 4 月 30 日

早期更新

下表描述了与 2018 年 AWS CloudHSM 之前版本相比的重要变化。

更改	描述	日期
新增内容	增加了面向加密员 (CO) 的仲裁身份验证（M of N 访问控制）。有关更多信息，请参阅 使用 CloudHSM 管理实用程序 (CMU, CloudHSM Management Utility) 管理仲裁身份验证（M of N 访问控制）。	2017 年 11 月 9 日
更新	增加了有关使用 key_mgmt_util 命令行工具的文档。有关更多信息，请参阅 key_mgmt_util 命令引用。	2017 年 11 月 9 日
新增内容	增加了 Oracle 透明数据加密。有关更多信息，请参阅 Oracle Database 加密。	2017 年 10 月 25 日

更改	描述	日期
新增内容	增加了 SSL 卸载。有关更多信息，请参阅 SSL/TLS 分载。	2017 年 10 月 12 日
新指南	此版本引入了 AWS CloudHSM	2017 年 8 月 14 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。