Guide de l'utilisateur

AWS CloudHSM

AWS CloudHSM: Guide de l’utilisateur

Table of Contents

Qu'est-ce que c'est AWS CloudHSM ?

AWS CloudHSM associe les avantages du AWS cloud à la sécurité des modules de sécurité matériels (HSM). Un module de sécurité matériel (HSM, Hardware Security Module) est un périphérique informatique qui traite des opérations de chiffrement et fournit un stockage sécurisé pour les clés cryptographiques. Vous disposez ainsi d' AWS CloudHSM un contrôle total sur les HSM à haute disponibilité présents dans le cloud AWS, d'un accès à faible latence et d'une racine de confiance sécurisée qui automatise la gestion des HSM (y compris les sauvegardes, le provisionnement, la configuration et la maintenance).

AWS CloudHSM offre aux clients de nombreux avantages :

Les HSM sont validés par la norme FIPS 140-2 de niveau 3

AWS CloudHSM utilise des HSM à usage général conformes aux normes, à locataire unique et validés par la norme FIPS 140-2 de niveau 3. Ils offrent une plus grande flexibilité par rapport aux services AWS entièrement gérés dotés d'algorithmes et de longueurs de clé prédéterminés pour votre application.

Le chiffrement E2E n'est pas visible pour AWS

Votre plan de données étant chiffré end-to-end (E2E) et invisible pour AWS, vous contrôlez votre propre gestion des utilisateurs (en dehors des rôles IAM). L'inconvénient de ce contrôle est que vous êtes plus responsable que si vous utilisiez un service AWS géré.

Contrôle total de vos clés, de vos algorithmes et du développement de vos applications

AWS CloudHSM vous donne le contrôle total des algorithmes et des clés que vous utilisez. Vous pouvez générer, stocker, importer, exporter, gérer et utiliser des clés cryptographiques (y compris les clés de sessions, les clés de jetons, les clés symétriques et les paires de clés asymétriques. En outre, AWS CloudHSM les SDK vous permettent de contrôler totalement le développement des applications, le langage des applications, le threading et l'emplacement physique de vos applications.

Migrez vos charges de travail cryptographiques vers le cloud

Les clients qui migrent une infrastructure à clé publique utilisant les normes de cryptographie à clé publique #11 (PKCS #11), Java Cryptographic Extension (JCE), Cryptography API : Next Generation (CNG) ou un fournisseur de stockage de clés (KSP) peuvent effectuer la migration en apportant moins de modifications à leur application. AWS CloudHSM

Accès aux clusters FIPS et non FIPS

Pour en savoir plus sur ce que vous pouvez faire avec AWS CloudHSM, consultez les rubriques suivantes. Lorsque vous serez prêt à commencer AWS CloudHSM, voyezPremiers pas.

> **Note**
>
> Si vous voulez un service géré pour la création et le contrôle de vos clés de chiffrement, mais si vous ne voulez pas ou n'avez pas besoin d'exploiter votre propre HSM, pensez à utiliser AWS Key Management Service.
> Si vous recherchez un service souple qui gère les HSM de paiement et les clés pour les applications de traitement des paiements dans le cloud, pensez à utiliser AWS Payment Cryptography.

Table des matières

AWS CloudHSM cas d'utilisation

AWS CloudHSM peut être utilisé pour atteindre divers objectifs. Le contenu de cette rubrique donne un aperçu de ce que vous pouvez en faire AWS CloudHSM.

Conformité aux réglementations

Les entreprises qui doivent se conformer aux normes de sécurité d'entreprise peuvent utiliser AWS CloudHSM pour gérer des clés privées qui protègent des données hautement confidentielles. Les HSM fournis par AWS CloudHSM sont certifiés FIPS 140-2 niveau 3 et sont conformes à la norme PCI DSS. De plus, AWS CloudHSM il est conforme aux normes PCI PIN et PCI-3DS. Pour plus d'informations, consultez Conformité d'.

Chiffrer et déchiffrer des données

AWS CloudHSM Utilisez-le pour gérer les clés privées qui protègent les données hautement confidentielles, le chiffrement en transit et le chiffrement au repos. En outre, AWS CloudHSM offre une intégration conforme aux normes avec plusieurs SDK cryptographiques.

Signer et vérifier des documents à l'aide de clés privées et publiques

En cryptographie, l'utilisation d'une clé privée pour signer un document permet aux destinataires d'utiliser une clé publique pour vérifier que vous (et non quelqu'un d'autre) avez bien envoyé le document. AWS CloudHSM À utiliser pour créer des paires de clés publiques et privées asymétriques spécialement conçues à cet effet.

Authentifier les messages à l'aide des HMAC et des CMAC

En cryptographie, les codes d'authentification de messages chiffrés (CMAC) et les codes d'authentification de messages basés sur le hachage (HMAC) sont utilisés pour authentifier et garantir l'intégrité des messages envoyés sur des réseaux non sécurisés. Avec AWS CloudHSM, vous pouvez créer et gérer en toute sécurité des clés symétriques compatibles avec les HMAC et les CMAC.

Tirez parti des avantages de AWS CloudHSM et AWS Key Management Service

Les clients peuvent combiner AWS CloudHSM et AWS KMSstocker des informations clés dans un environnement à locataire unique certifié FIPS 140-2 niveau 3, tout en bénéficiant des principaux avantages de gestion, de mise à l'échelle et d'intégration dans le cloud. AWS KMS Pour plus de détails sur la procédure à suivre, consultez les AWS CloudHSM magasins de clés dans le AWS Key Management Service Guide du développeur.

Déchargement du traitement SSL/TLS pour les serveurs Web

Pour envoyer des données en toute sécurité sur Internet, les serveurs Web utilisent des paires de clés publiques-privées et un certificat de clé publique SSL/TLS pour établir des sessions HTTPS. Ce processus implique beaucoup de calculs pour les serveurs Web, mais vous pouvez réduire la charge de calcul tout en renforçant la sécurité en déchargeant une partie de cette charge vers votre cluster. AWS CloudHSM Pour plus d'informations sur la configuration du déchargement SSL/TLS avec, consultez. AWS CloudHSMDéchargement SSL/TLS

Activer le chiffrement transparent des données (TDE)

Le chiffrement TDE (Transparent Data Encryption) est utilisé pour chiffrer les fichiers de base de données. Avec le TDE, le logiciel de base de données chiffre les données avant de les stocker sur le disque. Vous pouvez améliorer la sécurité en stockant la clé de chiffrement principale TDE dans les HSM de votre AWS CloudHSM. Pour plus d'informations sur la configuration d'Oracle TDE avec AWS CloudHSM, consultezChiffrement des bases de données Oracle.

Protéger les clés privées d'une autorité de certification émettrice (CA)

Une autorité de certification (CA) est une entité de confiance qui émet des certificats numériques qui lient une clé publique à une identité (une personne ou une organisation). Pour exploiter une autorité de certification, vous devez garantir la confiance en protégeant les clés privées qui signent les certificats délivrés par votre autorité de certification. Vous pouvez stocker ces clés privées dans votre AWS CloudHSM cluster, puis utiliser vos HSM pour effectuer des opérations de signature cryptographique.

Générer des nombres aléatoires

La génération de nombres aléatoires pour créer des clés de chiffrement est essentielle à la sécurité en ligne. AWS CloudHSM peuvent être utilisés pour générer en toute sécurité des nombres aléatoires dans les HSM que vous contrôlez et qui ne sont visibles que par vous.

Comment AWS CloudHSM fonctionne

Cette rubrique fournit une vue d'ensemble des concepts de base et de l'architecture que vous utilisez pour chiffrer des données en toute sécurité et effectuer des opérations cryptographiques dans les HSM. AWS CloudHSM fonctionne dans votre propre Amazon Virtual Private Cloud (VPC). Avant de pouvoir l'utiliser AWS CloudHSM, vous devez d'abord créer un cluster, y ajouter des HSM, créer des utilisateurs et des clés, puis utiliser les SDK clients pour intégrer vos HSM à votre application. Une fois cela fait, vous utilisez les journaux du SDK client AWS CloudTrail, les journaux d'audit et Amazon CloudWatch pour effectuer le suivi AWS CloudHSM.

Découvrez AWS CloudHSM les concepts de base et la façon dont ils fonctionnent ensemble pour protéger vos données.

Rubriques

- AWS CloudHSM clusters
- Utilisateurs HSM
- Clés HSM
- SDK clients
- AWS CloudHSM sauvegardes en cluster
- Régions

AWS CloudHSM clusters

Il peut être difficile de faire fonctionner des HSM individuels ensemble au sein d'un cluster synchronisé, redondant et hautement disponible, mais AWS CloudHSM cela vous permet de faire le plus gros du travail en fournissant des modules de sécurité matériels (HSM) dans des clusters. Un cluster est un ensemble de HSM individuels qui AWS CloudHSM restent synchronisés. Lorsque vous effectuez une tâche ou une opération sur un HSM dans un cluster, les autres HSM de ce cluster sont automatiquement tenus à jour. Pour atteindre vos objectifs de disponibilité, de durabilité et de capacité de mise à l'échelle, vous définissez le nombre de HSM de votre cluster dans plusieurs zones de disponibilité.

Vous pouvez créer un cluster comportant de 1 à 28 HSM (la limite par défaut est de 6 HSM par AWS compte et par AWS région). Vous pouvez placer les HSM dans différentes zones de disponibilité d'une AWS région. L'ajout de plusieurs HSM à un cluster offre de meilleures performances. La répartition des clusters entre les zones de disponibilité fournit redondance et haute disponibilité.

Pour de plus amples informations sur les clusters, consultez Gestion des AWS CloudHSM clusters.

Pour créer un cluster, consultez Premiers pas.

Utilisateurs HSM

Contrairement à la plupart des AWS services et ressources, vous n'utilisez pas d'utilisateurs AWS Identity and Access Management (IAM) ni de politiques IAM pour accéder aux ressources de votre cluster. Au lieu de cela, vous utilisez les utilisateurs HSM directement sur les HSM de votre AWS CloudHSM cluster.

Les utilisateurs HSM sont distincts des utilisateurs IAM. Les utilisateurs IAM qui disposent des informations d'identification correctes peuvent créer des HSM en interagissant avec les ressources

via l'API AWS. Comme le chiffrement E2E n'est pas visible pour AWS, vous devez utiliser les informations d'identification utilisateur du HSM pour authentifier les opérations sur le HSM, car les informations d'identification sont effectuées directement sur le HSM. Le HSM authentifie chaque utilisateur HSM au moyen d'informations d'identification que vous définissez et gérez. Chaque utilisateur HSM est d'un type qui détermine les opérations que celui-ci est autorisé à effectuer sur le HSM. Chaque HSM authentifie chaque utilisateur HSM au moyen d'informations d'identification que vous définissez à l'aide de la CLI CloudHSM.

Si vous utilisez la série de versions de SDK précédentes, vous utiliserez l'Utilitaire de gestion CloudHSM??? (CMU).

Clés HSM

AWS CloudHSM vous permet de générer, de stocker et de gérer en toute sécurité vos clés de chiffrement dans des HSM à locataire unique qui se trouvent dans votre cluster AWS CloudHSM. Les clés peuvent être symétriques ou asymétriques, peuvent être des clés de session (clés éphémères) pour des sessions uniques, des clés de jeton (clés persistantes) pour une utilisation à long terme, et peuvent être exportées et importées dans AWS CloudHSM. Les clés peuvent également être utilisées pour effectuer des tâches et des fonctions cryptographiques courantes :

- Effectuez la signature des données cryptographiques et la vérification des signatures à l'aide d'algorithmes de chiffrement symétriques et asymétriques.
- Travaillez avec des fonctions de hachage pour calculer la synthèse des messages et les codes d'authentification de message basés sur le hachage (HMAC).
- Enveloppez et protégez les autres clés.
- Accédez à des données aléatoires sécurisées par cryptographie.

En outre, AWS CloudHSM suit quelques principes fondamentaux pour l'utilisation et la gestion des clés :

De nombreux Types de clé et algorithmes parmi lesquels choisir

> Pour vous permettre de personnaliser vos propres solutions, AWS CloudHSM propose de nombreux types de clés et algorithmes parmi lesquels choisir. Les algorithmes prennent en charge différentes tailles de clés. Pour plus d'informations, reportez-vous aux pages d'attributs et de mécanismes de chaque AWS CloudHSM Kits de développement logiciel pour clients.

Comment gérer les clés

AWS CloudHSM les clés sont gérées via des kits de développement logiciel (SDK) et des outils de ligne de commande. Pour plus d'informations sur l'utilisation de ces outils pour gérer les clés, reportez-vous aux sections Gestion des clés dans AWS CloudHSM et Les meilleures pratiques pour AWS CloudHSM.

À qui appartiennent les clés

Dans AWS CloudHSM, l'utilisateur cryptographique (CU) qui crée la clé en est propriétaire. Le propriétaire peut utiliser les commandes key share et key unshare pour partager ou annuler le partage de la clé avec d'autres CU. Pour plus d'informations, consultez Utilisation de la CLI CloudHSM pour partager et annuler le partage de clés.

L'accès et l'utilisation peuvent être contrôlés grâce au chiffrement basé sur les attributs

AWS CloudHSM vous permet d'utiliser le chiffrement basé sur les attributs, une forme de chiffrement qui vous permet d'utiliser des attributs clés pour contrôler les personnes autorisées à déchiffrer les données en fonction des politiques.

SDK clients

Lors de l'utilisation AWS CloudHSM, vous effectuez des opérations cryptographiques avec des kits de développement logiciel AWS CloudHSM client (SDK). AWS CloudHSM Les SDK clients incluent :

- Public Key Cryptography Standards #11 (PKCS) #11
- Fournisseur JCE
- OpenSSL Dynamic Engine
- API de cryptographie : fournisseur de nouvelle génération (CNG) et de stockage de clés (KSP) pour Microsoft Windows

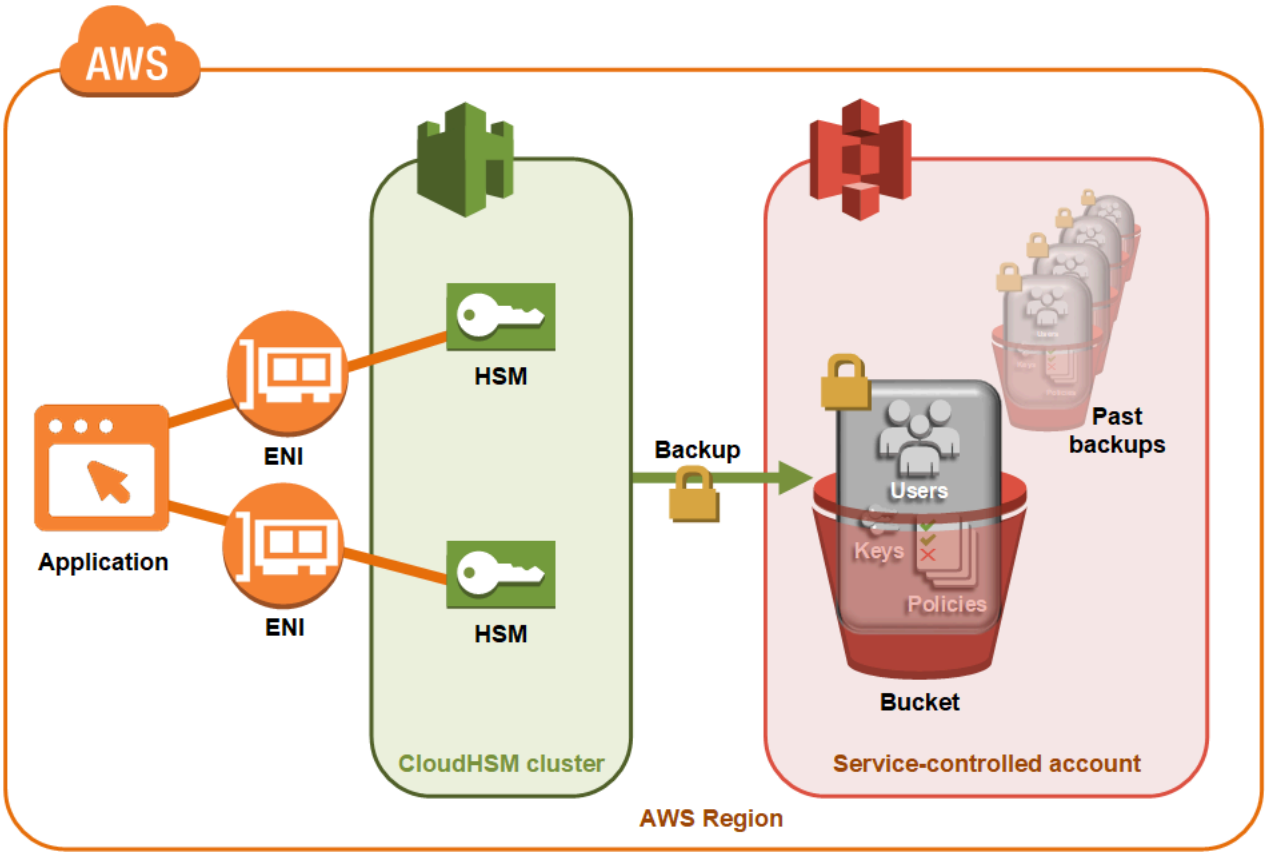
Vous pouvez utiliser l'un ou l'ensemble de ces SDK dans votre AWS CloudHSM cluster. Écrivez le code de votre application pour utiliser ces SDK afin d'effectuer des opérations cryptographiques dans vos HSM.

Les outils utilitaires et de ligne de commande sont nécessaires non seulement pour utiliser les SDK, mais également pour configurer les informations d'identification, les politiques et les paramètres de votre application. Pour plus d'informations, consultez AWS CloudHSM outils de ligne de commande.

Pour plus d'informations sur l'installation et l'utilisation du SDK client ou sur la sécurité de la connexion client, consultez SDK clients et nd-to-end Chiffrement électronique.

AWS CloudHSM sauvegardes en cluster

AWS CloudHSM effectue des sauvegardes périodiques des utilisateurs, des clés et des politiques du cluster. Les sauvegardes sont sécurisées, durables et mises à jour selon un calendrier prévisible. L'illustration suivante montre la relation entre vos sauvegardes et le cluster.



Pour plus d'informations sur le fonctionnement des sauvegardes, veuillez consulter Gestion des sauvegardes.

Sécurité

Lorsqu'il AWS CloudHSM effectue une sauvegarde à partir du HSM, celui-ci chiffre toutes ses données avant de les envoyer à. AWS CloudHSM Les données ne quittent jamais le HSM dans

le formulaire en texte brut. De plus, les sauvegardes ne peuvent pas être déchiffrées AWS car il AWS n'a pas accès à la clé utilisée pour déchiffrer les sauvegardes. Pour plus d'informations, consultez Sécurité des sauvegardes de clusters.

Durabilité

AWS CloudHSM stocke les sauvegardes dans un bucket Amazon Simple Storage Service (Amazon S3) contrôlé par le service dans la même région que votre cluster. Les sauvegardes ont un niveau de durabilité de 99,999999999 %, identique à celui de n'importe quel objet stocké dans Amazon S3.

Régions

Pour plus d'informations sur les régions prises en charge pour AWS CloudHSM, consultez la section AWS CloudHSM Régions et points de terminaison dans le Références générales AWSTableau des régions ou dans le tableau des régions.

AWS CloudHSM peut ne pas être disponible dans toutes les zones de disponibilité d'une région donnée. Toutefois, cela ne devrait pas affecter les performances, car les charges AWS CloudHSM sont automatiquement équilibrées sur tous les HSM d'un cluster.

Comme la plupart AWS des ressources, les clusters et les HSM sont des ressources régionales. Vous ne pouvez pas réutiliser ou étendre un cluster d'une région à une autre. Vous devez effectuer toutes les étapes requises répertoriées dans Commencer avec AWS CloudHSM pour créer un cluster dans une nouvelle région.

À des fins de reprise après sinistre, vous AWS CloudHSM permet de copier des sauvegardes de votre AWS CloudHSM cluster d'une région à l'autre. Pour plus d'informations, voir AWS CloudHSM sauvegardes en cluster.

Tarification

Avec AWS CloudHSM, vous payez à l'heure sans engagement à long terme ni paiement initial. Pour plus d'informations, consultez la section AWS CloudHSM Tarification sur le AWS site Web.

Commencer avec AWS CloudHSM

Les rubriques suivantes vous aident à créer, initialiser et activer un AWS CloudHSM cluster. Une fois que vous aurez suivi ces instructions, vous serez prêt à gérer des utilisateurs et des clusters, et à effectuer des opérations de chiffrement à l'aide des bibliothèques de logiciels incluses.

Table des matières

Création de groupes administratifs IAM

Il est recommandé de ne pas utiliser votre Utilisateur racine d'un compte AWS pour interagir avec AWS, y compris AWS CloudHSM. Utilisez plutôt AWS Identity and Access Management (IAM) pour créer un utilisateur IAM, un rôle IAM ou un utilisateur fédéré. Suivez les étapes décrites dans la section Création d'un utilisateur IAM et d'un groupe d'administrateurs IAM pour créer un groupe d'administrateurs et y associer la AdministratorAccesspolitique. Ensuite, créez un nouveau groupe d'administrateurs, puis ajoutez l'utilisateur au groupe. Si nécessaire, ajoutez des utilisateurs supplémentaires au groupe. Chaque utilisateur que vous ajoutez hérite de la AdministratorAccesspolitique du groupe.

Une autre bonne pratique consiste à créer un groupe d' AWS CloudHSM administrateurs disposant uniquement des autorisations nécessaires pour s'exécuter AWS CloudHSM. Si nécessaire, ajoutez

des utilisateurs individuels à ce groupe. Chaque utilisateur hérite des autorisations attachées au groupe plutôt que de l'accès complet à AWS . La Politiques gérées par le client pour AWS CloudHSM section qui suit contient la politique que vous devez associer à votre groupe AWS CloudHSM d'administrateurs.

AWS CloudHSM définit un rôle lié à un service pour votre AWS compte. Le rôle lié au service définit actuellement les autorisations qui permettent à votre compte de consigner AWS CloudHSM les événements. Le rôle peut être créé automatiquement par vous AWS CloudHSM ou manuellement. Vous ne pouvez pas modifier le rôle, mais vous pouvez le supprimer. Pour plus d'informations, consultez Rôles liés à un service pour AWS CloudHSM.

Création d'un utilisateur IAM et d'un groupe d'administrateurs IAM

Commencez par créer un utilisateur IAM ainsi qu'un groupe d'administrateurs pour cet utilisateur.

Inscrivez-vous pour un Compte AWS

Si vous n'en avez pas Compte AWS, procédez comme suit pour en créer un.

Pour vous inscrire à un Compte AWS

1. Ouvrez https://portal.aws.amazon.com/billing/signup.
2. Suivez les instructions en ligne.

   Dans le cadre de la procédure d'inscription, vous recevrez un appel téléphonique et vous saisirez un code de vérification en utilisant le clavier numérique du téléphone.

   Lorsque vous vous inscrivez à un Compte AWS, un Utilisateur racine d'un compte AWSest créé. Par défaut, seul l'utilisateur racine a accès à l'ensemble des Services AWS et des ressources de ce compte. Pour des raisons de sécurité, attribuez un accès administratif à un utilisateur et utilisez uniquement l'utilisateur root pour effectuer les tâches nécessitant un accès utilisateur root.

AWS vous envoie un e-mail de confirmation une fois le processus d'inscription terminé. Vous pouvez afficher l'activité en cours de votre compte et gérer votre compte à tout moment en accédant à https://aws.amazon.com/ et en choisissant Mon compte.

Création d'un utilisateur doté d'un accès administratif

Après vous être inscrit à un Compte AWS, sécurisez Utilisateur racine d'un compte AWS AWS IAM Identity Center, activez et créez un utilisateur administratif afin de ne pas utiliser l'utilisateur root pour les tâches quotidiennes.

Sécurisez votre Utilisateur racine d'un compte AWS

1. Connectez-vous en AWS Management Consoletant que propriétaire du compte en choisissant Utilisateur root et en saisissant votre adresse Compte AWS e-mail. Sur la page suivante, saisissez votre mot de passe.

   Pour obtenir de l'aide pour vous connecter en utilisant l'utilisateur racine, consultez Connexion en tant qu'utilisateur racine dans le Guide de l'utilisateur Connexion à AWS .
2. Activez l'authentification multifactorielle (MFA) pour votre utilisateur racine.

   Pour obtenir des instructions, voir Activer un périphérique MFA virtuel pour votre utilisateur Compte AWS root (console) dans le guide de l'utilisateur IAM.

Création d'un utilisateur doté d'un accès administratif

1. Activez IAM Identity Center.

   Pour obtenir des instructions, consultez Activation d' AWS IAM Identity Center dans le Guide de l'utilisateur AWS IAM Identity Center .
2. Dans IAM Identity Center, accordez un accès administratif à un utilisateur.

   Pour un didacticiel sur l'utilisation du Répertoire IAM Identity Center comme source d'identité, voir Configurer l'accès utilisateur par défaut Répertoire IAM Identity Center dans le Guide de AWS IAM Identity Center l'utilisateur.

Connectez-vous en tant qu'utilisateur disposant d'un accès administratif

- Pour vous connecter avec votre utilisateur IAM Identity Center, utilisez l'URL de connexion qui a été envoyée à votre adresse e-mail lorsque vous avez créé l'utilisateur IAM Identity Center.

  Pour obtenir de l'aide pour vous connecter en utilisant un utilisateur d'IAM Identity Center, consultez la section Connexion au portail AWS d'accès dans le guide de l'Connexion à AWS utilisateur.

Attribuer l'accès à des utilisateurs supplémentaires

1. Dans IAM Identity Center, créez un ensemble d'autorisations conforme aux meilleures pratiques en matière d'application des autorisations du moindre privilège.

   Pour obtenir des instructions, voir Création d'un ensemble d'autorisations dans le guide de AWS IAM Identity Center l'utilisateur.
2. Affectez des utilisateurs à un groupe, puis attribuez un accès d'authentification unique au groupe.

   Pour obtenir des instructions, voir Ajouter des groupes dans le guide de AWS IAM Identity Center l'utilisateur.

Pour des exemples de politiques AWS CloudHSM que vous pouvez associer à votre groupe d'utilisateurs IAM, voirGestion des identités et des accès pour AWS CloudHSM.

Création d'un cloud privé virtuel (VPC)

Si vous n'avez pas déjà un cloud privé virtuel (VPC), suivez les étapes indiquées dans cette rubrique pour en créer un.

> **Note**
>
> Le respect de ces étapes entraînera la création de sous-réseaux publics et privés.

Pour créer un VPC

1. Ouvrez la console Amazon VPC à l'adresse https://console.aws.amazon.com/vpc/.
2. Dans la barre de navigation, utilisez le sélecteur de région pour choisir l'une des AWS régions AWS CloudHSM actuellement prises en charge.
3. Sélectionnez le bouton Créer un VPC.
4. Sous Ressources à créer, choisissez VPC et plus encore.
5. Pour Génération automatique de balise de nom, saisissez un nom identifiable tel que **`CloudHSM`**.
6. Conservez les valeurs par défaut de toutes les autres options.
7. Sélectionnez Create VPC (Créer un VPC).

8. Une fois le VPC créé, sélectionnez Afficher le VPC pour afficher le VPC que vous venez de créer.

Créer un cluster

Un cluster est un ensemble de HSM individuels. AWS CloudHSM synchronise les HSM de chaque cluster afin qu'ils fonctionnent comme une unité logique.

Lorsque vous créez un cluster, il AWS CloudHSM crée un groupe de sécurité pour le cluster en votre nom. Ce groupe de sécurité contrôle l'accès réseau aux HSM dans le cluster. Il autorise les connexions entrantes uniquement à partir d'instances Amazon Elastic Compute Cloud (Amazon EC2) qui se trouvent dans le groupe de sécurité. Par défaut, le groupe de sécurité ne contient aucune instance. Par la suite, vous lancez une instance client et configurez le groupe de sécurité du cluster pour autoriser les communications et les connexions avec le HSM.

> **Important**
>
> Lorsque vous créez un cluster, il AWS CloudHSM crée un rôle lié à un service nommé. AWSServiceRoleForCloudHSM Si vous AWS CloudHSM ne pouvez pas créer le rôle ou s'il n'existe pas déjà, vous ne pourrez peut-être pas créer de cluster. Pour plus d'informations, consultez Résolution des échecs de création de cluster. Pour plus d'informations sur les rôles liés à un service, consultez Rôles liés à un service pour AWS CloudHSM.

Vous pouvez créer un cluster à partir de la AWS CloudHSM console, de la AWS Command Line Interface (CLI) ou de l' AWS CloudHSM API.

Pour créer un cluster (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Dans la barre de navigation, utilisez le sélecteur de région pour choisir l'une des AWS régions AWS CloudHSM actuellement prises en charge.
3. Choisissez Créer un cluster.
4. Dans la section Configuration de cluster, effectuez les opérations suivantes :

    a. Pour VPC, sélectionnez le VPC que vous avez créé dans Création d'un cloud privé virtuel (VPC).

b. Pour Zone(s) de disponibilité, en regard de chaque zone de disponibilité, choisissez le sous-réseau privé que vous avez créé.

> **Note**
>
> Même s'il n' AWS CloudHSM est pas pris en charge dans une zone de disponibilité donnée, les performances ne devraient pas être affectées, car les charges AWS CloudHSM sont automatiquement équilibrées sur tous les HSM d'un cluster. Voir AWS CloudHSM Régions et points de terminaison dans le Références générales AWSpour voir la prise en charge des zones de disponibilité pour AWS CloudHSM.

5. Choisissez Suivant.
6. Spécifiez la durée pendant laquelle le service doit conserver les sauvegardes.

> **Note**
>
> Acceptez la période de conservation par défaut de 90 jours ou saisissez une nouvelle valeur comprise entre 7 et 379 jours. Le service supprimera automatiquement les sauvegardes de ce cluster plus anciennes que la valeur que vous spécifiez ici. Cette valeur peut être modifiée par la suite. Pour plus d'informations, consultez Configuration de la rétention des sauvegardes.

7. Choisissez Suivant.
8. (Facultatif) Saisissez une clé de balise et une valeur de balise facultative. Pour ajouter plusieurs balises au cluster, sélectionnez Ajouter une balise.
9. Choisissez Examiner.
10. Vérifiez la configuration de votre cluster, puis choisissez Create cluster (Créer un cluster).

Pour créer un cluster (CLI)

- À partir d'une invite de commande, exécutez la commande create-cluster. Spécifiez le type d'instance HSM, la période de conservation des sauvegardes et les ID de sous-réseau des sous-réseaux dans lesquels vous envisagez de créer des HSM. Utilisez les ID de sous-réseau des sous-réseaux privés que vous avez créés. Spécifiez un seul sous-réseau par zone de disponibilité.

```
$ 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",
        "CreateTimestamp": 1502423370.069
    }
}
```

Pour créer un cluster (AWS CloudHSM API)

- Envoyez une demande CreateCluster. Spécifiez le type d'instance HSM, la stratégie de conservation des sauvegardes et les ID de sous-réseau des sous-réseaux dans lesquels vous envisagez de créer des HSM. Utilisez les ID de sous-réseau des sous-réseaux privés que vous avez créés. Spécifiez un seul sous-réseau par zone de disponibilité.

Si vos tentatives pour créer un cluster échouent, cela peut être lié à des problèmes avec les rôles liés au service AWS CloudHSM . Pour obtenir de l'aide sur la résolution du problème, consultez Résolution des échecs de création de cluster.

Vérification d'un groupe de sécurité de cluster

Lorsque vous créez un cluster, il AWS CloudHSM crée un groupe de sécurité portant le nom`cloudhsm-cluster-`*`clusterID`*`-sg`. Ce groupe de sécurité contient une règle TCP préconfigurée qui autorise les communications entrantes et sortantes au sein du groupe de sécurité du cluster sur les ports 2223-2225. Il permet à vos instances EC2 d'utiliser votre VPC pour communiquer avec les HSM de votre cluster.

⚠ Warning

- Ne supprimez pas et ne modifiez pas la règle TCP préconfigurée qui est renseignée dans le groupe de sécurité du cluster. Cette règle peut éviter des problèmes de connectivité et tout accès non autorisé à vos HSM.
- Le groupe de sécurité du cluster empêche tout accès non autorisé à vos HSM. Toute personne ayant accès aux instances du groupe de sécurité peut accéder à vos HSM. La plupart des opérations nécessitent qu'un utilisateur se connecte au HSM. Toutefois, il est possible de remettre à zéro les HSM sans authentification, ce qui détruit les clés, les certificats et d'autres données. Si cela se produit, les données créées ou modifiées après la dernière sauvegarde sont perdues et irrécupérables. Pour empêcher tout accès non autorisé, assurez-vous que seuls les administrateurs de confiance peuvent modifier ou accéder aux instances dans le groupe de sécurité par défaut.

Dans l'étape suivante, vous pouvez démarrer une instance Amazon EC2 et la connecter à vos HSM en y joignant le groupe de sécurité du cluster.

Lancement d'une instance client Amazon EC2.

Pour interagir avec votre AWS CloudHSM cluster et vos instances HSM et les gérer, vous devez être en mesure de communiquer avec les interfaces réseau élastiques de vos HSM. La manière la plus simple de le faire est d'utiliser une instance EC2 dans le même VPC que votre cluster. Vous pouvez également utiliser les ressources AWS suivantes pour vous connecter à votre cluster :

- Appairage de VPC Amazon
- AWS Direct Connect
- Connexions VPN

> **Note**
>
> Ce guide fournit un exemple simplifié de connexion d'une instance EC2 à votre AWS CloudHSM cluster. Pour connaître les meilleures pratiques en matière de configurations réseau sécurisées, reportez-vous àAccès sécurisé à votre cluster.

La AWS CloudHSM documentation suppose généralement que vous utilisez une instance EC2 dans le même VPC et la même zone de disponibilité (AZ) dans lesquels vous créez votre cluster.

Pour créer une instance EC2

1. Ouvrez le tableau de bord EC2 à l'adresse https://console.aws.amazon.com/ec2/.
2. Sélectionnez Démarrer l'instance. Dans le menu déroulant, choisissez Démarrer l'instance.
3. Dans le champ Nom, entrez un nom pour votre instance EC2.
4. Dans la section Images d'application et de système d'exploitation (AMI), choisissez une Amazon Machine Image (AMI) qui correspond à une plate-forme prise en charge par CloudHSM. Pour de plus amples informations, consultez Plateformes prises en charge par le SDK client 5.
5. Dans la section Type d'instance, sélectionnez un type d'instance.
6. Dans la section Paire de clés, utilisez une paire de clés existante ou sélectionnez Créer une nouvelle paire de clés et effectuez les étapes suivantes :

    a. Pour Nom de paire de clés, saisissez un nom pour la nouvelle paire de clés.

    b. Pour Type de paire de clés, choisissez un type de paire de clés.

    c. Pour Format de fichier de clé privée, sélectionnez le format de fichier de clé privée.

    d. Sélectionnez Créer une paire de clés.

    e. Téléchargez le fichier de clé privée et enregistrez-le.

> **Important**
>
> C'est votre seule occasion d'enregistrer le fichier de clé privée. Téléchargez et stockez le fichier en lieu sûr. Vous fournissez le nom de votre paire de clés quand vous lancez une instance. De plus, vous devez fournir la clé privée correspondante chaque fois que vous vous connectez à l'instance, puis choisir la paire de clés que vous avez créé lors de la configuration.

7. Pour Paramètres réseau, sélectionnez Modifier.
8. Pour VPC, choisissez le VPC créé précédemment pour votre cluster.
9. Pour Sous-réseau, choisissez le sous-réseau public que vous avez créé pour le VPC.
10. Pour Auto-assign Public IP (Attribuer automatiquement l'adresse IP publique), choisissez Enable (Activer).
11. Choisissez Select an existing security group.
12. Dans Groupes de sécurité communs, sélectionnez le groupe de sécurité par défaut dans le menu déroulant.
13. Dans Configurer le stockage, utilisez les menus déroulants pour choisir une configuration de stockage.
14. Dans la fenêtre Résumé, sélectionnez Démarrer l'instance.

> **Note**
>
> Cette étape lancera le processus de création de votre instance EC2.

Pour plus d'informations sur la création d'un client Linux Amazon EC2, consultez Mise en route sur les instances Linux Amazon EC2. Pour plus d'informations sur la connexion au client en cours d'exécution, consultez les rubriques suivantes :

- Connexion à votre instance Linux à l'aide de SSH
- Connexion à votre instance Linux à partir de Windows à l'aide de PuTTY

Le guide de l'utilisateur Amazon EC2 contient des instructions détaillées pour la configuration et l'utilisation de vos instances Amazon EC2. La liste suivante fournit une vue d'ensemble de la documentation disponible pour les clients sous Linux et Windows Amazon EC2 :

- Pour créer un client Amazon EC2 sous Linux, consultez Mise en route avec les instances Amazon EC2 sous Linux.

  Pour plus d'informations sur la connexion au client en cours d'exécution, consultez les rubriques suivantes :

  - Connexion à votre instance Linux à l'aide de SSH
  - Connexion à votre instance Linux à partir de Windows à l'aide de PuTTY

- Pour créer un client Amazon EC2 sous Windows, consultez Mise en route avec les instances Amazon EC2 sous Windows. Pour plus d'informations sur la connexion à votre client Windows, consultez Connexion à votre instance Windows.

> **Note**
>
> Votre instance EC2 peut exécuter toutes les commandes CLI contenues dans ce guide. Si la CLI n'est pas installée, vous pouvez la télécharger depuis AWS Command Line Interface. Si vous utilisez Windows, vous pouvez télécharger et exécuter un programme d'installation Windows 64 bits ou 32 bits. Si vous utilisez Linux ou macOS, vous pouvez installer l'interface de ligne de commande avec pip.

Configuration des groupes de sécurité de l'instance Amazon EC2 cliente

Lorsque vous avez lancé une instance Amazon EC2, vous l'avez associée à un groupe de sécurité Amazon VPC par défaut. Cette rubrique explique comment associer le groupe de sécurité du cluster avec l'instance EC2. Cette association permet au AWS CloudHSM client exécuté sur votre instance EC2 de communiquer avec vos HSM. Pour connecter votre instance EC2 à votre AWS CloudHSM cluster, vous devez configurer correctement le groupe de sécurité par défaut du VPC et associer le groupe de sécurité du cluster à l'instance.

Modifier le groupe de sécurité par défaut

Vous devez modifier le groupe de sécurité par défaut pour autoriser la connexion SSH ou RDP afin que vous puissiez télécharger et installer le logiciel client et interagir avec votre HSM.

Pour modifier le groupe de sécurité par défaut

1. Ouvrez le tableau de bord EC2 à l'adresse https://console.aws.amazon.com/ec2/.
2. Sélectionnez Instances (en cours d'exécution), puis cochez la case à côté de l'instance EC2 sur laquelle vous souhaitez installer le AWS CloudHSM client.
3. Dans l'onglet Sécurité, choisissez le groupe de sécurité intitulé Par défaut.
4. En haut de la page, choisissez Actions, puis Modifier les règles entrantes.
5. Sélectionnez Ajouter une règle.

6. Pour Type, effectuez l'une des actions suivantes :
   - Pour une instance Amazon EC2 sous Windows Server, choisissez RDP. Le port 3389 est automatiquement renseigné.
   - Pour une instance Amazon EC2 sous Linux, choisissez SSH. La plage de ports 22 est automatiquement renseignée.
7. Quelle que soit l'option, définissez Source sur Mon IP pour pouvoir communiquer avec votre instance Amazon EC2.

   > **Important**
   >
   > Ne spécifiez pas 0.0.0.0/0 comme plage CIDR pour éviter d'autoriser n'importe qui à accéder à votre instance.

8. Choisissez Enregistrer.

Connectez l'instance Amazon EC2 au cluster AWS CloudHSM

Vous devez attacher le groupe de sécurité du cluster à l'instance EC2 afin que l'instance EC2 puisse communiquer avec les HSM de votre cluster. Le groupe de sécurité du cluster contient une règle préconfigurée qui autorise la communication entrante sur les ports 2223-2225.

Pour connecter l'instance EC2 au cluster AWS CloudHSM

1. Ouvrez le tableau de bord EC2 à l'adresse https://console.aws.amazon.com/ec2/.
2. Sélectionnez Instances (en cours d'exécution), puis cochez la case correspondant à l'instance EC2 sur laquelle vous souhaitez installer le AWS CloudHSM client.
3. En haut de la page, choisissez Actions, Sécurité, puis Modifier les groupes de sécurité.
4. Sélectionnez le groupe de sécurité avec le nom de groupe qui correspond à l'ID de votre cluster, tel que `cloudhsm-cluster-`*`clusterID`*`-sg`.
5. Choisissez Appliquer les groupes de sécurité.
6. Sélectionnez Save.

> **Note**
>
> Vous pouvez attribuer un maximum de cinq groupes de sécurité à une instance Amazon EC2. Si vous avez atteint la limite maximale, vous devez modifier le groupe de sécurité par défaut de l'instance Amazon EC2 et le groupe de sécurité du cluster :
>
> Dans le groupe de sécurité par défaut, procédez comme suit :
>
> - Ajoutez une règle entrante pour autoriser le trafic à l'aide du protocole TCP sur les ports `2223-2225` depuis le groupe de sécurité du cluster.
>
> Dans le groupe de sécurité du cluster, procédez comme suit :
>
> - Ajoutez une règle entrante pour autoriser le trafic à l'aide du protocole TCP sur les ports `2223-2225` depuis le groupe de sécurité par défaut.

Création d'un HSM

Une fois que vous avez créé un cluster, vous pouvez créer un HSM. Toutefois, avant de pouvoir créer un HSM dans votre cluster, ce dernier ne doit pas être initialisé. Pour déterminer l'état du cluster, consultez la page des clusters dans la AWS CloudHSM console, utilisez la CLI pour exécuter la describe-clusters commande ou envoyez une DescribeClustersdemande dans l' AWS CloudHSM API. Vous pouvez créer un HSM à partir de la AWS CloudHSM console, de la CLI ou de l' AWS CloudHSM API.

Pour créer un HSM (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Sélectionnez le bouton radio en regard de l'ID du cluster pour lequel vous souhaitez créer un HSM.
3. Sélectionnez Actions. Dans le menu déroulant, choisissez Initialiser.
4. Choisissez une zone de disponibilité (AZ) pour le HSM que vous créez.
5. Sélectionnez Créer.

Pour créer un HSM (CLI)

- À partir d'une invite de commande, exécutez la commande create-hsm. Spécifiez l'ID du cluster que vous avez créé précédemment et une zone de disponibilité pour le HSM. Spécifiez la zone de disponibilité dans le format suivant : `us-west-2a`, `us-west-2b`, etc.

```
$ 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"
    }
}
```

Pour créer un HSM (AWS CloudHSM API)

- Envoyez une demande CreateHsm. Spécifiez l'ID du cluster que vous avez créé précédemment et une zone de disponibilité pour le HSM.

Une fois que vous avez créé un cluster et le HSM, vous pouvez vérifier l'identité du HSM ou passer directement à Initialiser le cluster.

Vérification de l'identité et de l'authenticité du HSM de votre cluster (facultatif)

Pour initialiser votre cluster, vous devez signer une demande de signature de certificat (CSR) générée par le premier HSM du cluster. Avant cela, il se peut que vous vouliez vérifier l'identité et l'authenticité du HSM.

> **Note**
>
> Cette procédure est facultative. Cependant, elle fonctionne uniquement jusqu'à ce qu'un cluster soit initialisé. Une fois que le cluster a été initialisé, vous ne pouvez pas utiliser ce processus pour obtenir les certificats ou vérifier les HSM.

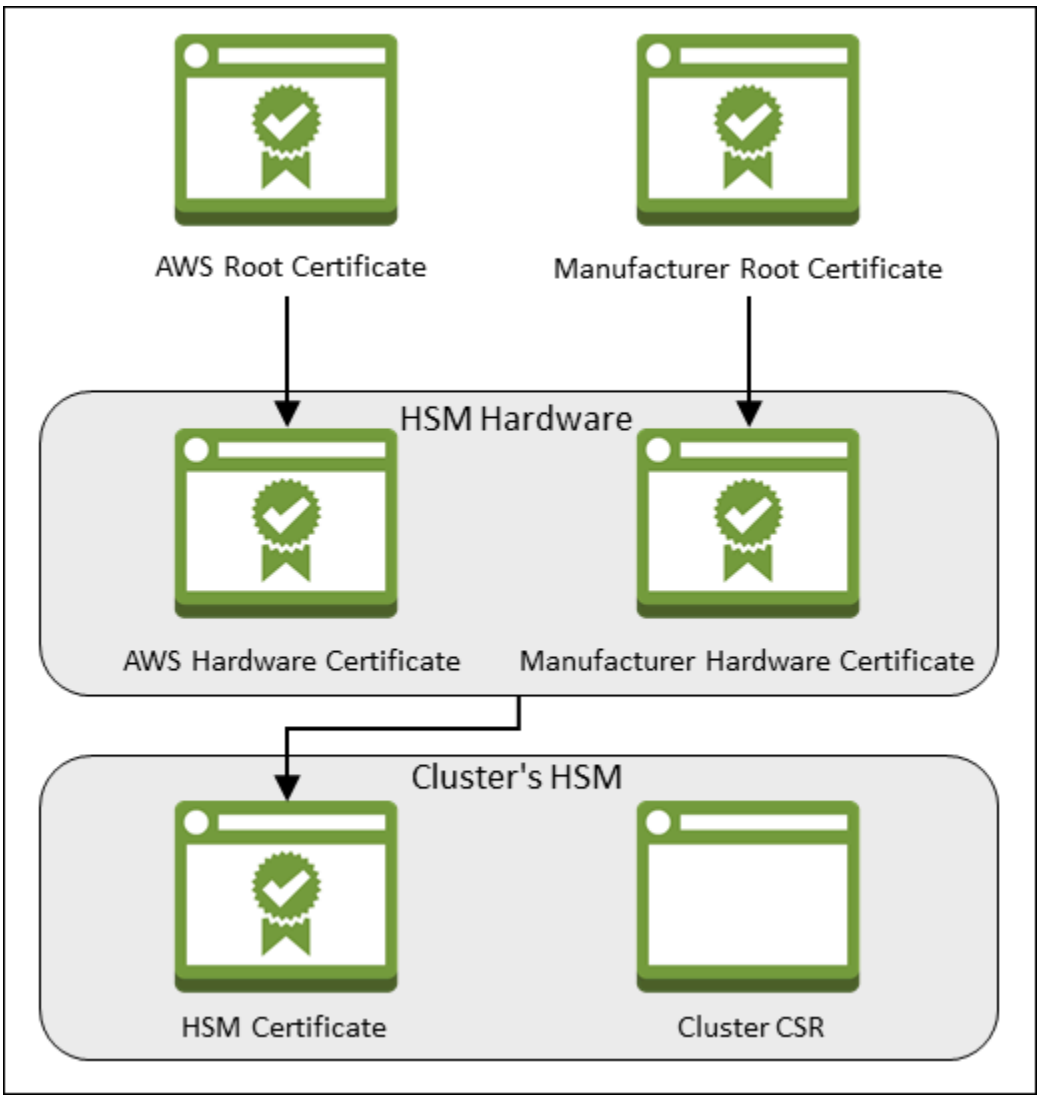
Rubriques

Présentation

Pour vérifier l'identité du premier HSM de votre cluster, effectuez les étapes suivantes :

1. Obtenir les certificats et la demande CSR Au cours de cette étape, vous obtenez trois certificats et une demande CSR auprès du HSM. Vous obtenez également deux certificats racine, l'un du fabricant du matériel HSM AWS CloudHSM et l'autre du fabricant.
2. Vérifier les chaînes de certificats : au cours de cette étape, vous créez deux chaînes de certificats, l'une vers le certificat AWS CloudHSM racine et l'autre vers le certificat racine du fabricant. Ensuite, vous vérifiez le certificat HSM à l'aide de ces chaînes de certificats pour le déterminer AWS CloudHSM et le fabricant du matériel atteste de l'identité et de l'authenticité du HSM.
3. Comparer les clés publiques Au cours de cette étape, vous allez extraire et comparer les clés publiques incluses dans le certificat HSM et la demande CSR du cluster pour vérifier qu'elles sont identiques. Vous pourrez ainsi être certain que la demande CSR a été générée par un HSM authentique et de confiance.

Le schéma suivant montre la demande CSR, les certificats et les relations qui les unissent. La liste suivante définit chaque certificat.



AWS Certificat racine

C'est AWS CloudHSM le certificat racine.

Certificat racine du fabricant

Il s'agit du certificat racine du fabricant du matériel.

AWS Certificat de matériel

AWS CloudHSM a créé ce certificat lorsque le matériel HSM a été ajouté à la flotte. Ce certificat affirme que le matériel AWS CloudHSM est propriétaire.

Certificat du fabricant du matériel

Le fabricant du matériel HSM a créé ce certificat lorsqu'il a fabriqué le matériel HSM. Ce certificat atteste que le fabricant a créé le matériel.

Certificat HSM

Le certificat HSM est généré par le matériel validé FIPS lorsque vous créez le premier HSM du cluster. Ce certificat atteste que le matériel HSM a créé le HSM.

CSR de cluster

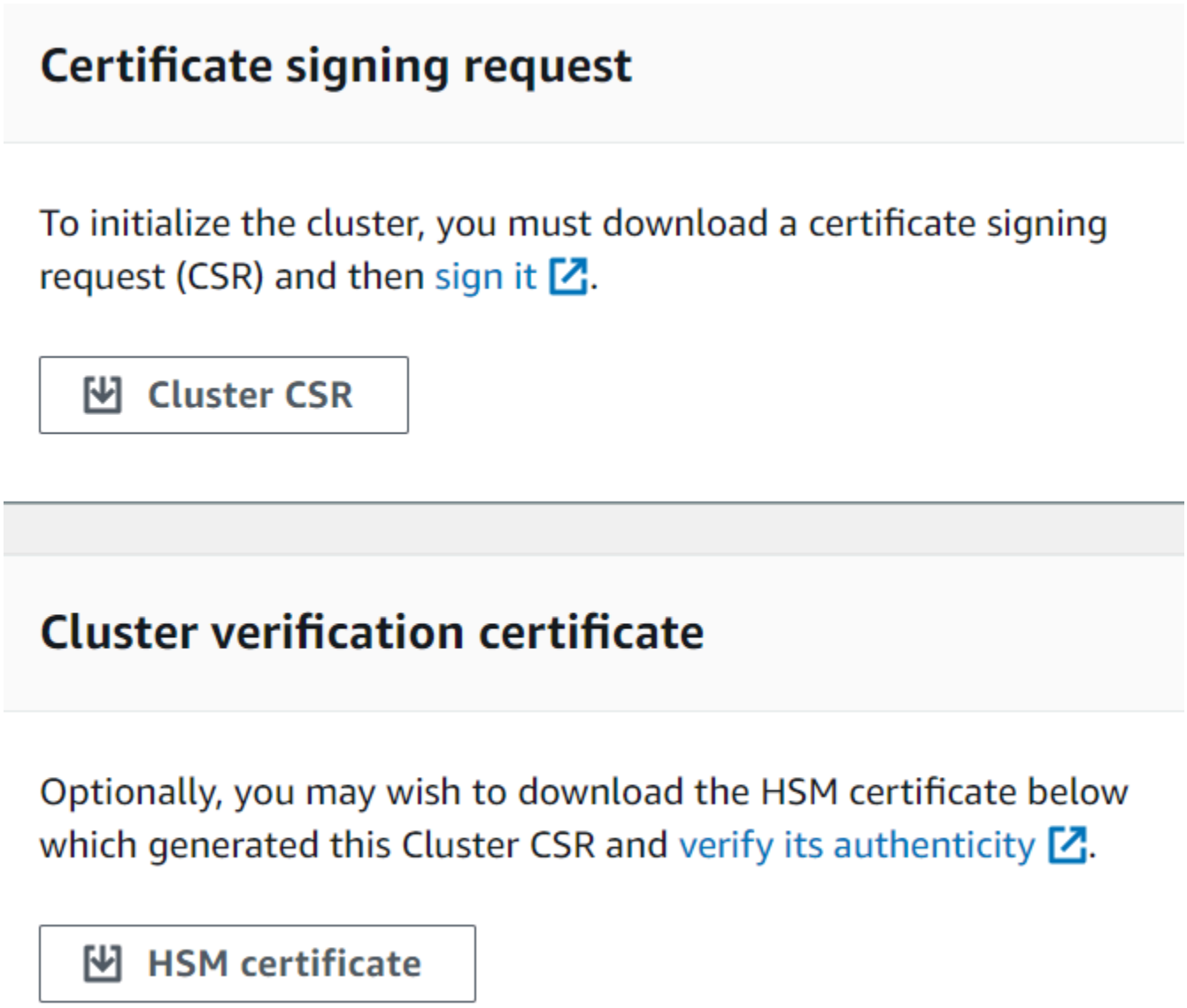
Le premier HSM crée le cluster CSR. Lorsque vous vous connectez au cluster CSR, vous demandez le cluster. Ensuite, vous pouvez utiliser la CSR signée pour initialiser le cluster.

Obtention des certificats auprès du HSM

Pour vérifier l'identité et l'authenticité de votre HSM, commencez par obtenir une demande de signature de certificat et cinq certificats. Vous obtenez trois des certificats du HSM, ce que vous pouvez faire à l'aide de la AWS CloudHSM console, de la AWS Command Line Interface (CLI) ou de l' AWS CloudHSM API.

Pour obtenir les certificats HSM et la demande de signature de certificat (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Sélectionnez le bouton radio en regard de l'ID du cluster avec le HSM que vous souhaitez vérifier.
3. Sélectionnez Actions. Dans le menu déroulant, choisissez Initialiser.
4. Si vous n'avez pas effectué l'étape précédente pour créer un HSM, choisissez une zone de disponibilité (AZ) pour le HSM que vous créez. Sélectionnez ensuite Créer.
5. Lorsque les certificats et la demande de signature de certificat sont prêts, des liens de téléchargement s'affichent.



6. Cliquez sur chaque lien pour télécharger et enregistrer la demande de signature de certificat et les certificats. Pour simplifier les étapes ultérieures, enregistrez tous les fichiers dans le même répertoire et utilisez les noms de fichier par défaut.

Pour obtenir les certificats CSR et HSM (CLI)

- À l'invite de commande, exécutez la commande describe-clusters quatre fois, en extrayant la CSR et les différents certificats à chaque fois, et en les enregistrant dans des fichiers.

  a. Entrez la commande suivante pour extraire la CSR du cluster. Remplacez *<ID cluster>* par l'ID du cluster que vous avez créé.

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

b. Entrez la commande suivante pour extraire le certificat HSM. Remplacez *<ID cluster>* par l'ID du cluster que vous avez créé.

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

c. Émettez la commande suivante pour extraire le certificat AWS matériel. Remplacez *<ID cluster>* par l'ID du cluster que vous avez créé.

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

d. Entrez la commande suivante pour extraire le certificat matériel du fabricant. Remplacez *<ID cluster>* par l'ID du cluster que vous avez créé.

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

Pour obtenir les certificats CSR et HSM (AWS CloudHSM API)

- Envoyez une demande DescribeClusters, puis extrayez ces éléments dans la réponse et enregistrez-les.

Obtention des certificats racine

Procédez comme suit pour obtenir les certificats racines pour AWS CloudHSM et le fabricant. Enregistrez les fichiers du certificat racine dans le répertoire contenant les fichiers de certificat CSR et HSM.

Pour obtenir les certificats racine AWS CloudHSM et ceux du fabricant

1. Téléchargez le certificat AWS CloudHSM racine : AWS_CloudHSM_Root-G1.zip
2. Téléchargez le certificat racine du fabricant : liquid_security_certificate.zip

   Pour télécharger le certificat depuis la page d'accueil, https://www.marvell.com/products/security-solutions/ liquid-security-hsm-adapters -and-appliances/liquidsecurity-certificate.html, puis choisissez Télécharger le certificat.

   Vous devrez peut-être cliquer avec le bouton droit de la souris sur le lien Télécharger le certificat, puis choisir Enregistrer le lien sous... pour enregistrer le fichier de certificat.
3. Après avoir téléchargé le fichier, extrayez (décompressez) son contenu.

Vérification des chaînes de certificat

Au cours de cette étape, vous créez deux chaînes de certificats, l'une vers le certificat AWS CloudHSM racine et l'autre vers le certificat racine du fabricant. Utilisez ensuite OpenSSL pour vérifier le certificat HSM par rapport à chaque chaîne de certificats.

Pour créer les chaînes de certificats, ouvrez un shell Linux. Vous avez besoin d'OpenSSL, qui est disponible dans la plupart des shells Linux, et vous avez besoin du certificat racine et des fichiers de certificat HSM que vous avez téléchargés. Cependant, vous n'avez pas besoin de la CLI pour cette étape, et le shell n'a pas besoin d'être associé à votre AWS compte.

Pour vérifier le certificat HSM avec le certificat AWS CloudHSM racine

1. Naviguez vers le répertoire où vous avez enregistré le certificat racine et les fichiers de certificats HSM que vous avez téléchargés. Les commandes suivantes supposent que tous les certificats sont dans le répertoire actuel et qu'ils utilisent les noms de fichier par défaut.

   Utilisez la commande suivante pour créer une chaîne de certificats qui inclut le certificat AWS matériel et le certificat AWS CloudHSM racine, dans cet ordre. Remplacez `<ID cluster>` par l'ID du cluster que vous avez créé.

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

2. Utilisez la commande OpenSSL suivante pour vérifier le certificat HSM avec la chaîne de certificat AWS . Remplacez *<ID cluster>* par l'ID du cluster que vous avez créé.

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

Pour vérifier le certificat HSM avec le certificat racine du fabricant

1. Utilisez la commande suivante pour créer une chaîne de certificat qui inclut le certificat du matériel du fabricant, puis le certificat racine du fabricant. Remplacez *<ID cluster>* par l'ID du cluster que vous avez créé.

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

2. Utilisez la commande OpenSSL suivante pour vérifier le certificat HSM avec la chaîne de certificat du fabricant. Remplacez *<ID cluster>* par l'ID du cluster que vous avez créé.

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

Extraction et comparaison des clés publiques

Utilisez OpenSSL pour extraire et comparer les clés publiques incluses dans le certificat HSM et la demande de signature de certificat du cluster, et vérifier qu'elles sont identiques.

Pour comparer les clés publiques, utilisez votre shell Linux. Vous avez besoin d'OpenSSL, qui est disponible dans la plupart des shells Linux, mais vous n'avez pas besoin de la CLI pour cette étape. Il n'est pas nécessaire d'associer le shell à votre AWS compte.

Pour extraire et comparer les clés publiques

1. Utilisez la commande suivante pour extraire la clé publique du certificat HSM.

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

2. Utilisez la commande suivante pour extraire la clé publique de la demande de signature de certificat du cluster.

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

3. Utilisez la commande suivante pour comparer les clés publiques. Si les clés publiques sont identiques, la commande suivante ne donne aucun résultat.

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

Une fois que vous avez vérifié l'identité et l'authenticité du HSM, passez à Initialiser le cluster.

Initialiser le cluster

Suivez les étapes décrites dans les rubriques suivantes pour initialiser votre AWS CloudHSM cluster.

> **Note**
>
> Avant d'initialiser le cluster, vérifiez le processus par lequel vous pouvez vérifier l'identité et l'authenticité des modules HSM. Ce processus est facultatif et fonctionne uniquement jusqu'à ce qu'un cluster soit initialisé. Une fois que le cluster a été initialisé, vous ne pouvez pas utiliser ce processus pour obtenir vos certificats ou vérifier les HSM.

Rubriques

- Obtenir la CSR du cluster
- Signez la CSR
- Initialiser le cluster

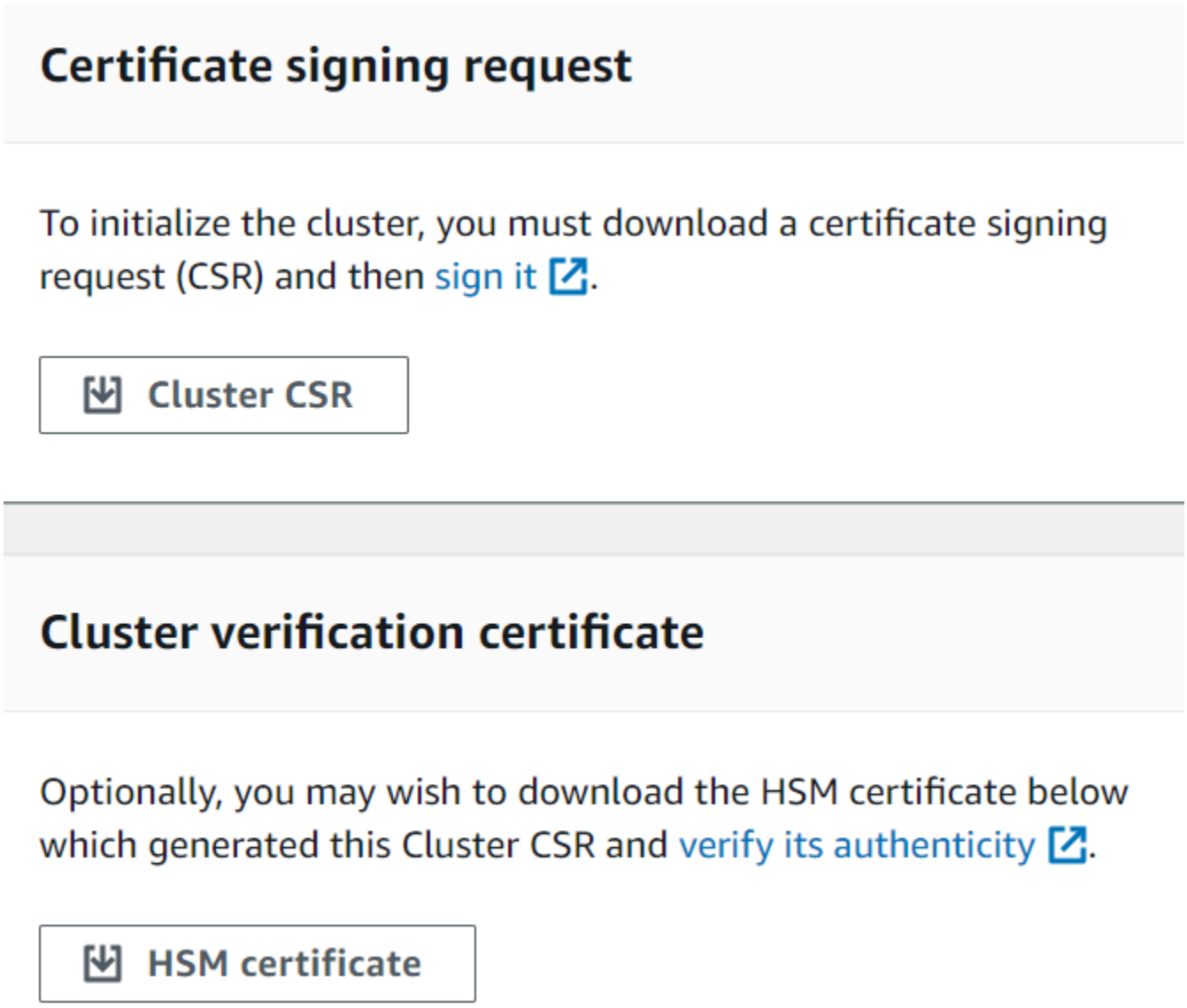
Obtenir la CSR du cluster

Avant de pouvoir initialiser le cluster, vous devez télécharger et signer une demande de signature de certificat (CSR) émise par le premier HSM du cluster. Si vous avez suivi la procédure de vérification de l'identité du HSM de votre cluster, vous disposez déjà de la CSR et vous pouvez la signer. Sinon, obtenez le CSR maintenant à l'aide de la AWS CloudHSM console, de la AWS Command Line Interface (CLI) ou de l' AWS CloudHSM API.

> **Important**
>
> Pour initialiser votre cluster, votre point de confiance doit être conforme à la RFC 5280 et répondre aux exigences suivantes :
>
> - Si vous utilisez les extensions X509v3, l'extension X509v3 Basic Constraints doit être présente.
> - Le point de confiance doit être un certificat auto-signé.
> - Les valeurs d'extension ne doivent pas entrer en conflit les unes avec les autres.

Pour obtenir la demande de signature de certificat (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Sélectionnez le bouton radio en regard de l'ID du cluster avec le HSM que vous souhaitez vérifier.
3. Sélectionnez Actions. Dans le menu déroulant, choisissez Initialiser.
4. Si vous n'avez pas effectué l'étape précédente pour créer un HSM, choisissez une zone de disponibilité (AZ) pour le HSM que vous créez. Sélectionnez ensuite Créer.
5. Lorsque la demande CSR est prête, un lien de téléchargement s'affiche.



6. Choisissez Cluster CSR pour télécharger et enregistrer cette demande.

Pour obtenir le CSR (CLI)

- À l'invite de commande, exécutez la commande describe-clusters suivante, qui extrait la CSR et l'enregistre dans un fichier. Remplacez *<ID cluster>* par l'ID du cluster que vous avez créé.

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

Pour obtenir le CSR (AWS CloudHSM API)

1. Envoyez une demande DescribeClusters.
2. Extrayez la demande CSR de la réponse et enregistrez-la.

Signez la CSR

Actuellement, vous devez créer un certificat de signature auto-signé et l'utiliser pour signer la CSR de votre cluster. Vous n'avez pas besoin de la CLI pour cette étape, et le shell n'a pas besoin d'être associé à votre AWS compte. Pour signer la CSR, vous devez effectuer les opérations suivantes :

1. Complétez la section précédente (voir Obtenir la CSR du cluster).
2. Créer une clé privée.
3. Utiliser la clé privée pour créer un certificat de signature.
4. Connecter votre cluster CSR.

Créer une clé privée

> **Note**
>
> Pour un cluster de production, la clé doit être créée de manière sécurisée à l'aide d'une source fiable de caractère aléatoire. Nous vous recommandons d'utiliser un HSM hors site et hors ligne sécurisé, ou équivalent. Stockez la clé en toute sécurité. La clé établit l'identité du cluster et votre seul contrôle sur les HSM qu'il contient.
> Au cours de la phase de développement et de test, vous pouvez utiliser n'importe quel outil pratique (par exemple, OpenSSL) pour créer et signer le certificat de cluster. L'exemple suivant montre comment créer une clé. Une fois que vous avez utilisé la clé pour créer un certificat auto-signé (voir ci-dessous), vous devez la stocker en toute sécurité. Pour vous connecter à votre AWS CloudHSM instance, le certificat doit être présent, mais pas la clé privée.

Utilisez la commande suivante pour créer une clé privée. Lors de l'initialisation d'un AWS CloudHSM cluster, vous devez utiliser le certificat RSA 2048 ou le certificat RSA 4096.

```
$ 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:
```

Utilisation de la clé privée pour créer un certificat auto-signé

Le matériel de confiance que vous utilisez pour créer la clé privée pour votre cluster de production doit également fournir un logiciel pour générer un certificat auto-signé à l'aide de cette clé. L'exemple suivant utilise OpenSSL et la clé privée que vous avez créée à l'étape précédente pour créer un certificat de signature. Le certificat est valable pendant 10 ans (3652 jours). Lisez les instructions à l'écran et suivez les invites.

```
$ 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 []:
```

Cette commande crée un fichier de certificat nommé `customerCA.crt`. Placez ce certificat sur chaque hôte à partir duquel vous allez vous connecter à votre AWS CloudHSM cluster. Si vous accordez un autre nom au fichier ou le stocker dans un emplacement autre que la racine de votre hôte, vous devez modifier votre fichier de configuration client en conséquence. Utilisez le certificat et la clé privée que vous venez de créer pour signer la demande de signature du certificat (CSR) de cluster à l'étape suivante.

Signature de la CSR du cluster

Le matériel approuvé que vous utilisez pour créer la clé privée pour votre cluster de production doit également fournir un logiciel pour signer la CSR à l'aide de cette clé. L'exemple suivant utilise OpenSSL pour signer la CSR du cluster. L'exemple utilise votre clé privée et le certificat auto-signé créé à l'étape précédente.

```
$ 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:
```

Cette commande crée un fichier nommé *<cluster ID>*`_CustomerHsmCertificate.crt`. Utilisez ce fichier comme certificat signé lorsque vous initialiserez le cluster.

Initialiser le cluster

Utilisez votre certificat HSM signé et votre certificat de signature pour initialiser votre cluster. Vous pouvez utiliser la AWS CloudHSM console, la CLI ou l' AWS CloudHSM API.

Pour initialiser un cluster (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Sélectionnez le bouton radio en regard de l'ID du cluster avec le HSM que vous souhaitez vérifier.
3. Sélectionnez Actions. Dans le menu déroulant, choisissez Initialiser.
4. Si vous n'avez pas effectué l'étape précédente pour créer un HSM, choisissez une zone de disponibilité (AZ) pour le HSM que vous créez. Sélectionnez ensuite Créer.
5. Sur la page Télécharger la demande de signature de certificat, cliquez sur Next. Si l'option Next n'est pas disponible, vous devez d'abord choisir l'un des liens renvoyant à la demande CSR ou au certificat. Ensuite, sélectionnez Suivant.

6. Sur la page Sign certificate signing request (CSR), cliquez sur Next.
7. Sur la page Upload the certificates, procédez comme suit :

   a. En regard de Cluster certificate (Certificat de cluster), choisissez Upload file (Charger le fichier). Ensuite, recherchez et sélectionnez le certificat HSM signé précédemment. Si vous avez suivi la procédure indiquée à la section précédente, sélectionnez le fichier *<cluster ID>*_CustomerHsmCertificate.crt.

   b. En regard de Émission du certificat, choisissez Charger le fichier. Ensuite, sélectionnez votre certificat de signature. Si vous avez suivi la procédure indiquée à la section précédente, sélectionnez le fichier `customerCA.crt`.

   c. Choisissez Upload and initialize.

Pour initialiser un cluster (CLI)

- À partir d'une invite de commande, exécutez la commande initialize-cluster. Fournissez les éléments suivants :
  - ID du cluster que vous avez créé précédemment.
  - Certificat HSM que vous avez signé précédemment. Si vous avez suivi la procédure indiquée à la section précédente, il est enregistré dans le fichier *<cluster ID>*_CustomerHsmCertificate.crt.
  - Votre certificat de signature. Si vous avez suivi la procédure indiquée à la section précédente, le certificat de signature est enregistré dans le fichier nommé `customerCA.crt`.

```
$ 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."
}
```

Pour initialiser un cluster (AWS CloudHSM API)

- Envoyez une demande InitializeCluster avec les éléments suivants :
  - ID du cluster que vous avez créé précédemment.
  - Certificat HSM que vous avez signé précédemment.
  - Votre certificat de signature.

Installation et configuration de l'interface de ligne de commande CloudHSM CLI

Pour interagir avec le HSM de votre AWS CloudHSM cluster, vous avez besoin de la CLI CloudHSM.

Tâches

- Installation des outils AWS CloudHSM de ligne de commande

Installation des outils AWS CloudHSM de ligne de commande

Connectez-vous à votre instance cliente et exécutez les commandes suivantes pour télécharger et installer les outils de ligne de AWS CloudHSM commande. Pour plus d'informations, consultez Lancement d'une instance client Amazon EC2..

Utilisez les commandes suivantes pour télécharger et installer CloudHSM CLI.

Amazon Linux 2

Amazon Linux 2 sur architecture x86_64 :

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

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

Amazon Linux 2 sur architecture ARM64 :

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

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

Amazon Linux 2023

Amazon Linux 2023 sur une architecture x86_64 :

```
$ 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
```

Amazon Linux 2023 sur architecture ARM64 :

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

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

CentOS 7 (7.8+)

CentOS 7 sur architecture x86_64 :

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

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

RHEL 7 (7.8+)

RHEL 7 sur architecture x86_64 :

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

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

RHEL 8 (8.3+)

RHEL 8 sur architecture x86_64 :

```
$ 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+)

RHEL 9 sur une architecture x86_64 :

```
$ 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
```

RHEL 9 sur l'architecture ARM64 :

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

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

Ubuntu 20.04 LTS

Ubuntu 20.04 LTS sur architecture x86_64 :

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

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

Ubuntu 22.04 LTS

Ubuntu 22.04 LTS sur architecture x86_64 :

```
$ 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
```

Ubuntu 22.04 LTS sur architecture ARM64 :

```
$ 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

Pour Windows Server 2016 sur une architecture x86_64, ouvrez PowerShell en tant qu'administrateur et exécutez la commande suivante :

```
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

Pour Windows Server 2019 sur une architecture x86_64, ouvrez PowerShell en tant qu'administrateur et exécutez la commande suivante :

```
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
```

Utilisez les commandes suivantes pour configurer CloudHSM CLI.

Pour démarrer une instance EC2 Linux pour un SDK client 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP du ou des HSM de votre cluster.

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

Pour démarrer une instance EC2 Windows pour le client SDK 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP du ou des HSM de votre cluster.

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

Activation du cluster

Lorsque vous activez un AWS CloudHSM cluster, son état passe d'initialisé à actif. Vous pouvez ensuite gérer les utilisateurs du HSM et utiliser le HSM.

Important

Avant de pouvoir activer le cluster, vous devez d'abord copier le certificat émetteur vers l'emplacement par défaut de la plate-forme sur chaque instance EC2 connectée au cluster (vous créez le certificat émetteur lorsque vous initialisez le cluster).
Linux

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

Windows

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

Après avoir placé le certificat émetteur, installez la CLI CloudHSM et exécutez la commande cluster activate sur votre premier HSM. Vous remarquerez que le compte administrateur du premier HSM de votre cluster possède le rôle unactivated-admin. Il s'agit d'un rôle temporaire qui n'existe qu'avant l'activation du cluster. Lorsque vous activez votre cluster, le rôle unactivated-admin bascule sur admin

Activation d'un cluster

1. Connectez-vous à l'instance client que vous avez lancée précédemment. Pour plus d'informations, consultez Lancement d'une instance client Amazon EC2.. Vous pouvez démarrer une instance Linux ou un serveur Windows Server.
2. Exécutez la CLI CloudHSM en mode interactif.

   Linux

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

   Windows

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

3. (Facultatif) Utilisez la commande user list pour afficher les utilisateurs existants.

   ```
   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. Utilisez la commande cluster activate pour définir le mot de passe administrateur d'origine.

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

Nous vous recommandons d'enregistrer le nouveau mot de passe dans une feuille de calcul réservée à cet effet. Ne perdez pas celle-ci. Nous vous recommandons d'imprimer une copie de la feuille des mots de passe, de l'utiliser pour enregistrer vos mots de passe HSM critiques et de la stocker dans un endroit sûr. Nous vous conseillons également de conserver une copie de cette feuille dans un espace de stockage sécurisé hors site.

5. (Facultatif) Utilisez la commande user list pour vérifier que le type de l'utilisateur est désormais administrateur/responsable du chiffrement (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. Utilisez la commande quit pour fermer l'outil de la CLI CloudHSM.

```
aws-cloudhsm > quit
```

Pour plus d'informations sur l'utilisation de la CLI CloudHSM ou de l'utilitaire CMU, consultez les sections Comprendre les utilisateurs HSM et Comprendre la gestion des utilisateurs HSM avec l'utilitaire CMU.

Reconfigurer SSL avec un nouveau certificat et une nouvelle clé privée (facultatif)

AWS CloudHSM utilise un certificat SSL pour établir une connexion à un HSM. Une clé et le certificat SSL par défaut sont inclus lorsque vous installez le client. Toutefois, vous pouvez créer et utiliser les vôtres. Notez que vous aurez besoin du certificat auto-signé (*customerCA.crt*) que vous avez créé lorsque vous avez initialisé votre cluster.

À un haut niveau, il s'agit d'un processus en deux étapes :

1. Vous devez d'abord créer une clé privée, puis l'utiliser pour créer une demande de signature de certificat (CSR). Utilisez le certificat émetteur, le certificat que vous avez créé lors de l'initialisation du cluster, pour signer la CSR.
2. Ensuite, utilisez l'outil de configuration pour copier la clé et le certificat dans les répertoires appropriés.

Créez une clé, une CSR, puis signez la CSR

Les étapes sont les mêmes pour le SDK client 3 ou le SDK client 5.

Pour reconfigurer SSL avec un nouveau certificat et une nouvelle clé privée

1. Créez une clé privée à l'aide de la commande OpenSSL suivante :

```
openssl genrsa -out ssl-client.key 2048
Generating RSA private key, 2048 bit long modulus
.........+++
```

```
.............+++
e is 65537 (0x10001)
```

2. Utilisez la commande OpenSSL suivante pour créer une demande de signature de certificat (CSR). Vous devrez répondre à une série de questions concernant votre certificat.

```
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. Signez la CSR avec le certificat *customerCA.crt* que vous avez créé lorsque vous avez initialisé votre cluster.

```
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
```

Activer le protocole SSL personnalisé pour AWS CloudHSM

Les étapes sont différentes pour le SDK client 3 ou le SDK client 5. Pour plus d'informations sur l'utilisation de l'outil de ligne de commande de configuration, consultez ???.

Rubriques

- SSL personnalisé pour le SDK client 3
- SSL personnalisé pour le SDK client 5

SSL personnalisé pour le SDK client 3

Utilisez l'outil de configuration du SDK client 3 pour activer le SSL personnalisé. Pour plus d'informations sur l'outil de configuration du SDK client 3, consultez ???.

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 3 sous Linux

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. Utilisez l'outil de configuration pour spécifier `ssl-client.crt` et `ssl-client.key`.

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

3. Ajoutez le certificat `customerCA.crt` au magasin d'approbations. Créez un hachage du nom d'objet du certificat. Cette opération crée un index pour autoriser la recherche du certificat par ce nom.

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

   Créez un répertoire.

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

Créez un fichier qui contient le certificat avec le nom de hachage.

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

SSL personnalisé pour le SDK client 5

Utilisez l'un des outils de configuration du SDK client 5 pour activer le protocole SSL personnalisé. Pour plus d'informations sur l'outil de configuration du SDK client 5, consultez ???.

PKCS #11 library

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Linux

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. Utilisez l'outil de configuration pour spécifier `ssl-client.crt` et `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
```

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Windows

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. À l'aide d'un PowerShell interpréteur, utilisez l'outil de configuration pour spécifier `ssl-client.crt` et`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

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Linux

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. Utilisez l'outil de configuration pour spécifier `ssl-client.crt` et `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

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Linux

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. Utilisez l'outil de configuration pour spécifier `ssl-client.crt` et `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
```

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Windows

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. À l'aide d'un PowerShell interpréteur, utilisez l'outil de configuration pour spécifier `ssl-client.crt` et`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

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Linux

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. Utilisez l'outil de configuration pour spécifier `ssl-client.crt` et `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
```

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Windows

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

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

2. À l'aide d'un PowerShell interpréteur, utilisez l'outil de configuration pour spécifier `ssl-client.crt` et`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
```

Création d'une application

Créez des applications et travaillez avec des clés à l'aide de AWS CloudHSM.

Pour commencer à créer et à utiliser des clés dans votre nouveau cluster, vous devez d'abord créer un utilisateur du module de sécurité matérielle (HSM) à l'aide de l'Utilitaire de gestion CloudHSM (CMU). Pour plus d'informations, consultez Comprendre les tâches de gestion des utilisateurs HSM, Commencer à utiliser l'interface de ligne de commande (CLI) AWS CloudHSM et Comment gérer les utilisateurs HSM.

> **Note**
>
> Si vous utilisez le SDK client 3, utilisez l'Utilitaire de gestion CloudHSM (CMU) au lieu de la CLI CloudHSM.

Une fois les utilisateurs HSM en place, vous pouvez vous connecter au HSM et créer et utiliser des clés avec l'une des options suivantes :

- Utilisation d'un Utilitaire de gestion des clés, d'un outil de ligne de commande
- Création d'une application C à l'aide de la bibliothèque PKCS #11
- Création d'une application Java à l'aide du fournisseur JCE
- Utilisation du moteur dynamique OpenSSL directement depuis la ligne de commande
- Utilisation du moteur dynamique OpenSSL pour le déchargement du protocole TLS avec les serveurs Web NGINX et Apache

- Utilisez les fournisseurs CNG et KSP pour les utiliser avec AWS CloudHSM Microsoft Windows Server Certificate Authority (CA)
- Utilisez les fournisseurs CNG et KSP pour utiliser l'outil AWS CloudHSM Microsoft Sign
- Utilisation des fournisseurs CNG et KSP pour le déchargement TLS avec le serveur Web Internet Information Server (IIS)

Les meilleures pratiques pour AWS CloudHSM

Appliquez les meilleures pratiques décrites dans cette rubrique pour une utilisation efficace AWS CloudHSM.

Table des matières

Gestion du cluster

Suivez les meilleures pratiques décrites dans cette section lors de la création, de l'accès et de la gestion de votre AWS CloudHSM cluster.

Adaptez votre cluster pour faire face aux pics de trafic

Plusieurs facteurs peuvent influencer le débit maximal que votre cluster peut gérer, notamment la taille de l'instance client, la taille du cluster, la topographie du réseau et les opérations cryptographiques requises pour votre cas d'utilisation.

Pour commencer, reportez-vous à la rubrique AWS CloudHSM Rendement pour obtenir des estimations de performances sur les tailles et configurations de clusters courantes. Nous vous recommandons de tester la charge de votre cluster avec le pic de charge que vous prévoyez afin de déterminer si votre architecture actuelle est résiliente et à la bonne échelle.

Concevez votre cluster pour une haute disponibilité

Ajoutez de la redondance pour tenir compte de la maintenance : vous AWS pouvez remplacer votre HSM pour une maintenance planifiée ou s'il détecte un problème. En règle générale, la taille de votre cluster doit avoir au moins +1 redondance. Par exemple, si vous avez besoin de deux HSM pour que votre service fonctionne aux heures de pointe, la taille idéale de votre cluster sera alors de trois. Si vous suivez les meilleures pratiques en matière de disponibilité, ces remplacements de HSM

ne devraient pas avoir d'impact sur votre service. Cependant, les opérations en cours sur le HSM remplacé peuvent échouer et doivent être réessayées.

Répartissez vos HSM sur de nombreuses zones de disponibilité : réfléchissez à la manière dont votre service pourra fonctionner en cas de panne de zone de disponibilité. AWS vous recommande de répartir vos HSM sur autant de zones de disponibilité que possible. Pour un cluster comportant trois HSM, vous devez répartir les HSM sur trois zones de disponibilité. En fonction de votre système, il se peut que vous ayez besoin d'une redondance supplémentaire.

Disposer d'au moins trois HSM pour garantir la durabilité des clés nouvellement générées

Pour les applications qui nécessitent la durabilité des clés nouvellement générées, nous recommandons d'avoir au moins trois HSM répartis dans différentes zones de disponibilité d'une région.

Accès sécurisé à votre cluster

Utilisez des sous-réseaux privés pour limiter l'accès à votre instance : lancez vos HSM et vos instances clientes dans les sous-réseaux privés de votre VPC. Cela limite l'accès à vos HSM depuis le monde extérieur.

Utilisez des points de terminaison VPC pour accéder aux API : le plan de AWS CloudHSM données a été conçu pour fonctionner sans avoir besoin d'accéder à Internet ou aux API AWS. Si votre instance client a besoin d'accéder à l' AWS CloudHSM API, vous pouvez utiliser des points de terminaison VPC pour accéder à l'API sans avoir besoin d'un accès Internet sur votre instance client. Pour plus d'informations, consultez AWS CloudHSM et points de terminaison VPC.

Reconfigurez le protocole SSL pour sécuriser les communications client-serveur : AWS CloudHSM utilise le protocole TLS pour établir une connexion avec votre HSM. Après avoir initialisé votre cluster, vous pouvez remplacer le certificat TLS par défaut et la clé utilisés pour établir la connexion TLS externe. Pour plus d'informations, consultez Améliorez la sécurité de votre serveur Web grâce au déchargement SSL/TLS dans AWS CloudHSM.

Réduisez les coûts en adaptant vos besoins

Il n'y a aucun coût initial d'utilisation AWS CloudHSM. Vous payez un tarif horaire pour chaque HSM que vous lancez jusqu'à ce que vous mettiez fin au HSM. Si votre service ne nécessite pas une

utilisation continue de AWS CloudHSM, vous pouvez réduire les coûts en réduisant (en supprimant) vos HSM à zéro lorsqu'ils ne sont pas nécessaires. Lorsque des HSM sont à nouveau nécessaires, vous pouvez restaurer vos HSM à partir d'une sauvegarde. Si, par exemple, votre charge de travail vous oblige à signer du code une fois par mois, en particulier le dernier jour du mois, vous pouvez agrandir votre cluster avant, le réduire en supprimant vos HSM une fois le travail terminé, puis restaurer votre cluster pour effectuer à nouveau les opérations de signature à la fin du mois suivant.

AWS CloudHSM effectue automatiquement des sauvegardes périodiques des HSM du cluster. Lorsque vous ajoutez un nouveau HSM à une date ultérieure, la dernière sauvegarde AWS CloudHSM sera restaurée sur le nouveau HSM afin que vous puissiez reprendre l'utilisation à l'endroit où vous l'avez laissée. Pour calculer vos coûts AWS CloudHSM d'architecture, consultez la section AWS CloudHSM Tarification.

Ressources connexes :

- Présentation générale des sauvegardes
- Politique de conservation des sauvegardes
- Copier des sauvegardes entre AWS les régions

Gestion des utilisateurs HSM

Suivez les meilleures pratiques décrites dans cette section pour gérer efficacement les utilisateurs de votre AWS CloudHSM cluster. Les utilisateurs HSM sont distincts des utilisateurs IAM. Les utilisateurs et entités IAM dotés d'une politique basée sur l'identité avec les autorisations appropriées peuvent créer des HSM en interagissant avec les ressources via l'API AWS. Une fois le HSM créé, vous devez utiliser les informations d'identification de l'utilisateur HSM pour authentifier les opérations sur le HSM. Pour un guide détaillé des utilisateurs de HSM, voirGestion des utilisateurs HSM dans AWS CloudHSM.

Protégez les informations d'identification de vos utilisateurs HSM

Il est impératif de protéger de manière sécurisée les informations d'identification de vos utilisateurs HSM, car les utilisateurs HSM sont les entités qui peuvent accéder à votre HSM et effectuer des opérations de cryptographie et de gestion sur celui-ci. AWS CloudHSM n'a pas accès à vos informations d'identification d'utilisateur HSM et ne sera pas en mesure de vous aider si vous perdez l'accès à celles-ci.

Disposer d'au moins deux administrateurs pour empêcher le verrouillage

Pour éviter d'être exclu de votre cluster, nous vous recommandons d'avoir au moins deux administrateurs au cas où un mot de passe d'administrateur serait perdu. Dans ce cas, vous pouvez utiliser l'autre administrateur pour réinitialiser le mot de passe.

> **Note**
>
> Dans le SDK client 5, les administrateurs sont synonymes de responsables du chiffrement (CoS) dans le SDK client 3.

Activer le quorum pour toutes les opérations de gestion des utilisateurs

Le quorum vous permet de définir un nombre minimum d'administrateurs qui doivent approuver une opération de gestion des utilisateurs avant que celle-ci puisse avoir lieu. En raison des privilèges dont disposent les administrateurs, nous vous recommandons d'activer le quorum pour toutes les opérations de gestion des utilisateurs. Cela peut limiter l'impact potentiel si l'un de vos mots de passe d'administrateur est compromis. Pour plus d'informations, consultez la section Gestion du quorum.

Créez plusieurs utilisateurs de cryptomonnaies, chacun avec des autorisations limitées

En séparant les responsabilités des utilisateurs de cryptomonnaies, aucun utilisateur n'a le contrôle total de l'ensemble du système. Pour cette raison, nous vous recommandons de créer plusieurs utilisateurs de cryptomonnaies et de limiter les autorisations de chacun. Cela se fait généralement en confiant aux différents utilisateurs de cryptomonnaies des responsabilités et des actions distinctes qu'ils exécutent (par exemple, un utilisateur de cryptographie est chargé de générer et de partager les clés avec d'autres utilisateurs de cryptomonnaies qui les utilisent ensuite dans votre application).

Ressources connexes :

- partage de clés
- annuler le partage d'une clé

Gestion des clés HSM

Suivez les meilleures pratiques décrites dans cette section lors de la gestion des clés d'entrée AWS CloudHSM.

Choisissez le bon type de clé

Lorsque vous utilisez une clé de session, vos transactions par seconde (TPS) seront limitées à un HSM où la clé existe. Des HSM supplémentaires dans votre cluster n'augmenteront pas le débit des demandes pour cette clé. Si vous utilisez une clé de jeton pour la même application, vos demandes seront équilibrées entre tous les HSM disponibles dans votre cluster. Pour plus d'informations, consultez Principaux paramètres de synchronisation et de durabilité dans AWS CloudHSM.

Gérez les limites de stockage des clés

Les HSM ont des limites quant au nombre maximum de jetons et de clés de session qui peuvent être stockés simultanément sur un HSM. Pour plus d'informations sur les limites de stockage des clés, consultezAWS CloudHSM quotas. Si votre application requiert une quantité supérieure à la limite, vous pouvez utiliser une ou plusieurs des stratégies suivantes pour gérer efficacement les clés :

Utilisez un encapsulage sécurisé pour stocker vos clés dans un magasin de données externe : grâce à l'encapsulage de clés sécurisé, vous pouvez dépasser la limite de stockage des clés en stockant toutes vos clés dans un magasin de données externe. Lorsque vous devez utiliser cette clé, vous pouvez la déballer dans le HSM en tant que clé de session, utiliser la clé pour l'opération requise, puis supprimer la clé de session. Les données clés d'origine restent stockées en toute sécurité dans votre banque de données pour être utilisées chaque fois que vous en avez besoin. L'utilisation de clés fiables pour ce faire maximise votre protection.

Répartissez les clés entre les clusters : une autre stratégie pour surmonter la limite de stockage des clés consiste à stocker vos clés dans plusieurs clusters. Dans cette approche, vous maintenez un mappage des clés stockées dans chaque cluster. Utilisez ce mappage pour acheminer les demandes de vos clients vers le cluster avec la clé requise. Pour plus d'informations sur la connexion à plusieurs clusters à partir de la même application cliente, consultez les rubriques suivantes :

- Connexion à plusieurs clusters avec le fournisseur JCE
- Connexion à plusieurs emplacements avec PKCS #11

Gestion et sécurisation de l'emballage des clés

Les clés peuvent être marquées comme extractibles ou non extractibles via l'attribut. `EXTRACTABLE` Par défaut, les clés HSM sont marquées comme extractibles.

Les clés extractibles sont des clés dont l'exportation depuis le HSM est autorisée par encapsulage de clés. Les clés encapsulées sont chiffrées et doivent être déballées à l'aide de la même clé d'encapsulation avant de pouvoir être utilisées. Les clés non extractibles ne peuvent en aucun cas être exportées depuis le HSM. Il n'existe aucun moyen de rendre extractible une clé non extractible. Pour cette raison, il est important de déterminer si vous avez besoin que vos clés soient extractibles ou non et de définir l'attribut clé correspondant en conséquence.

Si vous avez besoin d'un encapsulage de clés dans votre application, vous devez utiliser un encapsulage de clés sécurisé pour limiter la capacité de vos utilisateurs HSM à encapsuler/déballer uniquement les clés qui ont été explicitement marquées comme fiables par un administrateur. Pour plus d'informations, consultez les rubriques relatives à l'encapsulation de clés fiablesGestion des clés dans AWS CloudHSM.

Ressources connexes

- Fonctions d'encapsulage et de désencapsulage
- Fonctions de chiffrement pour JCE
- Attributs de clé Java pris en charge
- Attributs de clé de la CLI CloudHSM

Intégration d'applications

Suivez les meilleures pratiques décrites dans cette section pour optimiser la manière dont votre application s'intègre à votre AWS CloudHSM cluster.

Démarrez votre SDK client

Avant que le SDK client puisse se connecter à votre cluster, celui-ci doit être amorcé. Lorsque vous démarrez des adresses IP vers votre cluster, nous vous recommandons d'utiliser le `--cluster-id` paramètre dans la mesure du possible. Cette méthode remplit votre configuration avec toutes les adresses IP HSM de votre cluster sans qu'il soit nécessaire de suivre chaque adresse individuelle. Cela renforce la résilience de l'initialisation de votre application en cas de maintenance d'un HSM ou d'une panne de zone de disponibilité. Pour en savoir plus, consultez Amorcez le SDK client.

Authentifiez-vous pour effectuer des opérations

Dans AWS CloudHSM, vous devez vous authentifier auprès de votre cluster avant de pouvoir effectuer la plupart des opérations, telles que les opérations cryptographiques.

Authentification avec la CLI CloudHSM : vous pouvez vous authentifier avec la CLI CloudHSM en utilisant le mode commande unique ou le mode interactif. Utilisez la login commande pour vous authentifier en mode interactif. Pour vous authentifier en mode commande unique, vous devez définir les variables environnementales `CLOUDHSM_ROLE` et`CLOUDHSM_PIN`. Pour plus de détails sur cette procédure, reportez-vous àMode commande unique. AWS CloudHSM recommande de stocker en toute sécurité vos informations d'identification HSM lorsque votre application ne les utilise pas.

Authentifiez-vous avec PKCS #11 : Dans PKCS #11, vous vous connectez à l'aide de l'API C_Login après avoir ouvert une session à l'aide de C_. OpenSession Vous n'avez besoin d'effectuer qu'un seul C_Login par slot (cluster). Une fois connecté, vous pouvez ouvrir des sessions supplémentaires à l'aide de C_ OpenSession sans avoir à effectuer d'autres opérations de connexion. Pour des exemples d'authentification auprès de PKCS #11, consultez. Exemples de code pour la bibliothèque PKCS #11

Authentifiez-vous avec JCE : le fournisseur AWS CloudHSM JCE prend en charge les connexions implicites et explicites. La méthode qui vous convient dépend de votre cas d'utilisation. Dans la mesure du possible, nous vous recommandons d'utiliser la connexion implicite, car le SDK gère automatiquement l'authentification si votre application est déconnectée de votre cluster et doit être réauthentifiée. L'utilisation de la connexion implicite vous permet également de fournir des informations d'identification à votre application lorsque vous utilisez une intégration qui ne vous permet pas de contrôler le code de votre application. Pour en savoir plus sur les méthodes de connexion, consultezFournir des informations d'identification au fournisseur JCE.

Authentifiez-vous avec OpenSSL : avec le moteur dynamique OpenSSL, vous fournissez des informations d'identification par le biais de variables d'environnement. AWS CloudHSM recommande de stocker en toute sécurité vos informations d'identification HSM lorsque votre application ne les utilise pas. Dans la mesure du possible, vous devez configurer votre environnement pour récupérer et définir systématiquement ces variables d'environnement sans saisie manuelle. Pour plus de détails sur l'authentification avec OpenSSL, consultez. Installation du moteur dynamique OpenSSL

Gérez efficacement les clés de votre application

Utilisez les attributs clés pour contrôler ce que les clés peuvent faire : lors de la génération d'une clé, utilisez les attributs clés pour définir un ensemble d'autorisations qui autoriseront ou refuseront des

types d'opérations spécifiques pour cette clé. Nous recommandons que les clés soient générées avec le moins d'attributs nécessaires pour effectuer leur tâche. Par exemple, une clé AES utilisée pour le chiffrement ne doit pas également être autorisée à extraire des clés du HSM. Pour plus d'informations, consultez nos pages d'attributs pour les SDK clients suivants :

- Attributs de clés PKCS #11
- Attributs de clés JCE

Dans la mesure du possible, mettez en cache les objets clés pour minimiser la latence : les opérations de recherche de clés interrogeront tous les HSM de votre cluster. Cette opération est coûteuse et ne s'adapte pas au nombre de HSM de votre cluster.

- Avec PKCS #11, vous pouvez trouver des clés à l'aide de l'`C_FindObjects`API.
- Avec JCE, vous trouvez les clés à l'aide du KeyStore.

Pour des performances optimales, il est AWS recommandé d'utiliser les commandes de recherche de touches (comme findKey etliste des clés) une seule fois au démarrage de l'application et de mettre en cache l'objet clé renvoyé dans la mémoire de l'application. Si vous avez besoin de cet objet clé ultérieurement, vous devez le récupérer de votre cache au lieu de demander cet objet pour chaque opération, ce qui alourdira considérablement les performances.

Utiliser le multi-threading

AWS CloudHSM prend en charge les applications multithread, mais il y a certaines choses à garder à l'esprit en ce qui concerne les applications multithread.

Avec PKCS #11, vous ne devez initialiser la bibliothèque PKCS #11 (appel`C_Initialize`) qu'une seule fois. Chaque fil doit se voir attribuer sa propre session (`C_OpenSession`). Il n'est pas recommandé d'utiliser la même session dans plusieurs threads.

Avec JCE, le AWS CloudHSM fournisseur ne doit être initialisé qu'une seule fois. Ne partagez pas les instances d'objets SPI entre les threads. Par exemple, Cipher, Signature, Digest, Mac KeyFactory ou KeyGenerator les objets ne doivent être utilisés que dans le contexte de leur propre thread.

Gérer les erreurs de régulation

Vous pouvez rencontrer des erreurs de régulation HSM dans les circonstances suivantes :

- Votre cluster n'est pas correctement dimensionné pour gérer les pics de trafic.
- Votre cluster n'est pas dimensionné avec une redondance +1 lors des événements de maintenance.
- Les pannes de zone de disponibilité entraînent une réduction du nombre de HSM disponibles dans votre cluster.

Consultez Limitation du HSM pour plus d'informations sur la meilleure façon de gérer ce scénario.

Pour vous assurer que votre cluster est correctement dimensionné et qu'il ne sera pas limité, il est AWS recommandé de tester la charge dans votre environnement en fonction de votre pic de trafic attendu.

Intégrez les nouvelles tentatives sur les opérations du cluster

AWS peut remplacer votre HSM pour des raisons opérationnelles ou de maintenance. Afin de rendre votre application résiliente face à de telles situations, il est AWS recommandé d'implémenter une logique de nouvelle tentative côté client sur toutes les opérations acheminées vers votre cluster. Les tentatives ultérieures portant sur des opérations ayant échoué en raison de remplacements devraient réussir.

Mettre en œuvre des stratégies de reprise après sinistre

En réponse à un événement, il peut être nécessaire de déplacer votre trafic hors d'un cluster ou d'une région dans son ensemble. Les sections suivantes décrivent plusieurs stratégies pour y parvenir.

Utilisez le peering VPC pour accéder à votre cluster depuis un autre compte ou une autre région : vous pouvez utiliser le peering VPC pour accéder à votre AWS CloudHSM cluster depuis un autre compte ou une autre région. Pour plus d'informations sur la façon de configurer cela, consultez Qu'est-ce que le peering VPC ? dans le guide de peering VPC. Une fois que vous avez établi vos connexions d'appairage et configuré vos groupes de sécurité de manière appropriée, vous pouvez communiquer avec les adresses IP HSM de la même manière que vous le feriez normalement.

Connectez-vous à plusieurs clusters à partir de la même application : le fournisseur JCE, la bibliothèque PKCS #11 et la CLI du SDK client 5 prennent en charge la connexion à plusieurs clusters à partir de la même application. Par exemple, vous pouvez avoir deux clusters actifs, chacun situé dans des régions différentes, et votre application peut se connecter aux deux en même temps et équilibrer la charge entre les deux dans le cadre des opérations normales. Si votre application

n'utilise pas le SDK client 5 (le dernier SDK), vous ne pouvez pas vous connecter à plusieurs clusters à partir de la même application. Vous pouvez également maintenir un autre cluster opérationnel et, en cas de panne régionale, transférer votre trafic vers l'autre cluster afin de minimiser les temps d'arrêt. Consultez les pages respectives pour plus de détails :

- Connexion à plusieurs emplacements avec PKCS #11
- Connexion à plusieurs clusters avec le fournisseur JCE
- Connexion à plusieurs clusters à l'aide de la CLI

Restaurer un cluster à partir d'une sauvegarde : vous pouvez créer un nouveau cluster à partir de la sauvegarde d'un cluster existant. Pour plus d'informations, consultez Gestion des AWS CloudHSM sauvegardes.

Surveillance

Cette section décrit les différents mécanismes que vous pouvez utiliser pour surveiller votre cluster et votre application. Pour plus de détails sur la surveillance, voirSurveillance AWS CloudHSM.

Surveiller les journaux des clients

Chaque SDK client écrit des journaux que vous pouvez surveiller. Pour plus d'informations sur la journalisation des clients, consultezUtilisation des journaux du SDK client.

Sur les plateformes conçues pour être éphémères, telles qu'Amazon ECS, la collecte des journaux des clients à partir d'un fichier peut s'avérer difficile. AWS Lambda Dans ces situations, il est recommandé de configurer la journalisation de votre SDK client pour écrire des journaux sur la console. La plupart des services collecteront automatiquement ces résultats et les publieront sur Amazon CloudWatch Logs pour que vous puissiez les conserver et les consulter.

Si vous utilisez une intégration tierce en plus du SDK AWS CloudHSM client, vous devez vous assurer de configurer ce package logiciel pour qu'il enregistre également sa sortie sur la console. Dans le cas contraire, la sortie du SDK AWS CloudHSM client peut être capturée par ce package et écrite dans son propre fichier journal.

Consultez le Outil de configuration du SDK client 5 pour savoir comment configurer les options de journalisation dans votre application.

Surveiller les journaux d'audit

AWS CloudHSM publie des journaux d'audit sur votre CloudWatch compte Amazon. Les journaux d'audit proviennent du HSM et permettent de suivre certaines opérations à des fins d'audit.

Vous pouvez utiliser les journaux d'audit pour suivre toutes les commandes de gestion invoquées sur votre HSM. Par exemple, vous pouvez déclencher une alarme lorsque vous remarquez qu'une opération de gestion inattendue est effectuée.

Pour plus d'informations, consultez Fonctionnement de la journalisation d'audit HSM.

Moniteur AWS CloudTrail

AWS CloudHSM est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans AWS CloudHSM. AWS CloudTrail capture tous les appels d'API AWS CloudHSM sous forme d'événements. Les appels capturés incluent des appels provenant de la AWS CloudHSM console et des appels de code vers les opérations de l' AWS CloudHSM API.

Vous pouvez l'utiliser AWS CloudTrail pour auditer tout appel d'API envoyé au plan de AWS CloudHSM contrôle afin de vous assurer qu'aucune activité indésirable n'a lieu sur votre compte.

Consultez Travailler avec AWS CloudTrail et AWS CloudHSM pour plus de détails.

Surveillez les CloudWatch statistiques d'Amazon

Vous pouvez utiliser CloudWatch les métriques Amazon pour surveiller votre AWS CloudHSM cluster en temps réel. Les métriques peuvent être regroupées par région, ID de cluster ou ID HSM et ID de cluster.

À l'aide CloudWatch des métriques Amazon, vous pouvez configurer les CloudWatch alarmes Amazon pour vous avertir de tout problème potentiel susceptible d'avoir un impact sur votre service. Nous vous recommandons de configurer les alarmes pour surveiller les éléments suivants :

- Vous vous approchez de votre limite de clés sur un HSM
- Approche de la limite du nombre de sessions HSM sur un HSM
- Approche de la limite du nombre d'utilisateurs HSM sur un HSM
- Différences dans le nombre d'utilisateurs ou de clés HSM pour identifier les problèmes de synchronisation

- HSM défectueux pour faire évoluer votre cluster jusqu'à ce que AWS CloudHSM le problème soit résolu

Pour en savoir plus, consultez Utilisation d'Amazon CloudWatch Logs et AWS CloudHSM d'Audit Logs.

Gestion des AWS CloudHSM clusters

Vous pouvez gérer vos AWS CloudHSM clusters à partir de la AWS CloudHSM console, de l'un des AWS SDK ou des outils de ligne de commande. Pour plus d'informations, consultez les rubriques suivantes.

Pour créer un cluster, consultez Premiers pas.

Architecture du cluster

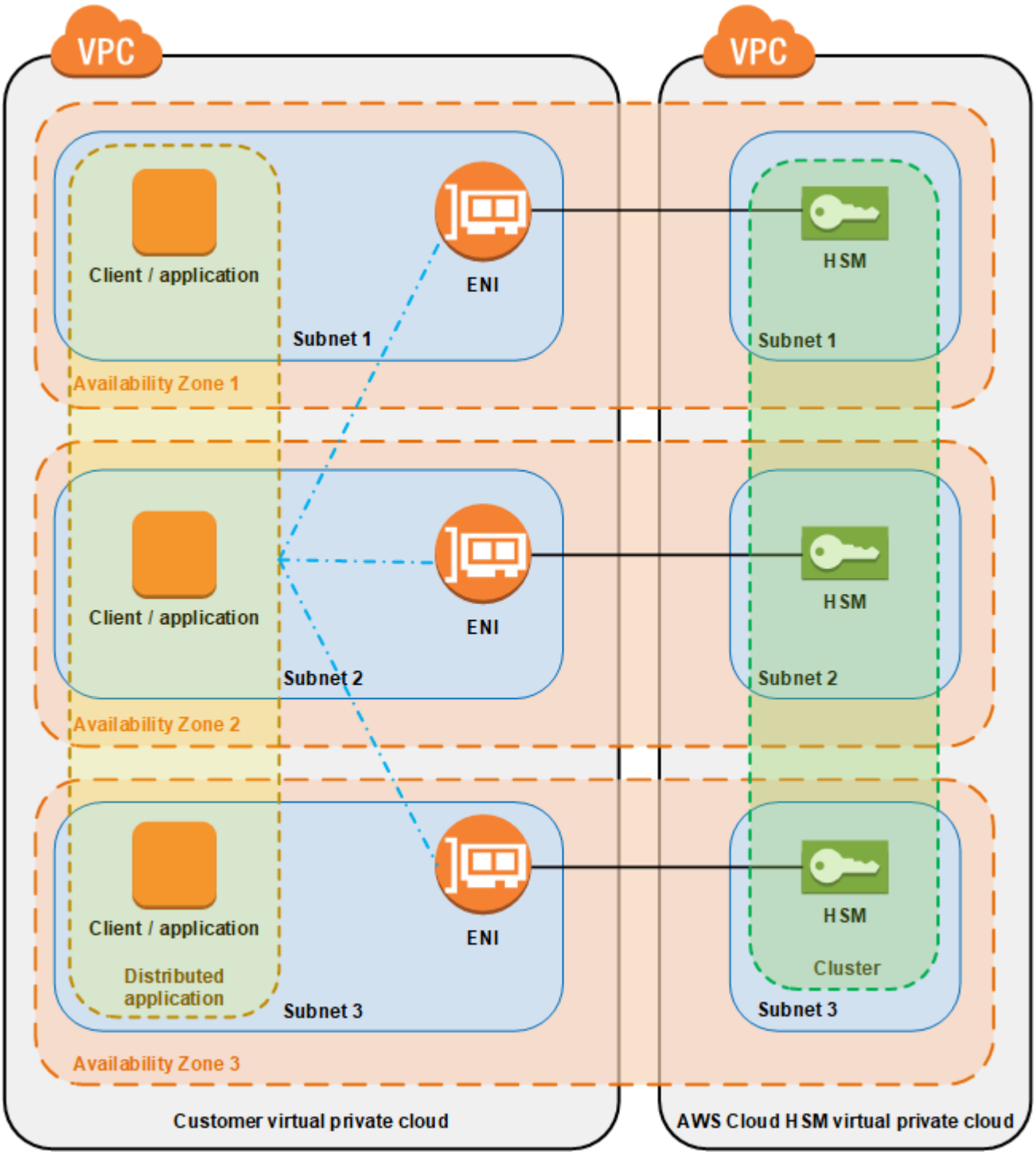
Lorsque vous créez un cluster, vous spécifiez un Amazon Virtual Private Cloud (VPC) dans votre AWS compte et un ou plusieurs sous-réseaux dans ce VPC. Nous vous recommandons de créer un sous-réseau dans chaque zone de disponibilité (AZ) de la AWS région que vous avez choisie. Vous pouvez créer des sous-réseaux privés lorsque vous créez un VPC. Pour en savoir plus, veuillez consulter la section Création d'un cloud privé virtuel (VPC).

Chaque fois que vous créez un HSM, vous devez spécifier le cluster et la zone de disponibilité de celui-ci. En plaçant le HSM dans des zones de disponibilité distinctes, vous obtenez de la redondance et de la haute disponibilité en cas d'indisponibilité d'une zone de disponibilité.

Lorsque vous créez un HSM, AWS CloudHSM place une Elastic Network Interface (ENI) dans le sous-réseau spécifié de votre AWS compte. L'interface réseau Elastic est l'interface permettant d'interagir avec le HSM. Le HSM réside dans un VPC distinct, dans AWS un compte détenu par. AWS CloudHSM Le HSM et son interface réseau correspondante se trouvent dans la même zone de disponibilité.

Pour interagir avec les HSM d'un cluster, vous avez besoin du logiciel AWS CloudHSM client. En règle générale, vous installez le client sur les instances Amazon EC2, aussi appelées instances client, qui résident dans le même VPC que les ENI HSM, comme illustré dans la figure suivante. Techniquement, ce n'est cependant pas obligatoire. Vous pouvez installer le client sur n'importe quel ordinateur compatible, à condition qu'il puisse se connecter aux ENI HSM. Le client communique avec les HSM individuels de votre cluster via leurs ENI.

La figure suivante représente un AWS CloudHSM cluster avec trois HSM, chacun dans une zone de disponibilité différente du VPC.



Synchronisation du cluster

Dans un AWS CloudHSM cluster, AWS CloudHSM synchronise les touches des différents HSM. Vous n'avez pas besoin de faire quoi que ce soit pour synchroniser les clés sur vos HSM. Pour synchroniser les utilisateurs et les politiques de chaque HSM, mettez à jour le fichier de configuration

du AWS CloudHSM client avant de gérer les utilisateurs du HSM. Pour plus d'informations, consultez Conserver la synchronisation des utilisateurs HSM.

Lorsque vous ajoutez un nouveau HSM à un cluster, effectuez AWS CloudHSM une sauvegarde de toutes les clés, utilisateurs et politiques d'un HSM existant. Il restaure ensuite la sauvegarde sur le nouvel HSM. Les deux HSM sont ainsi synchronisés.

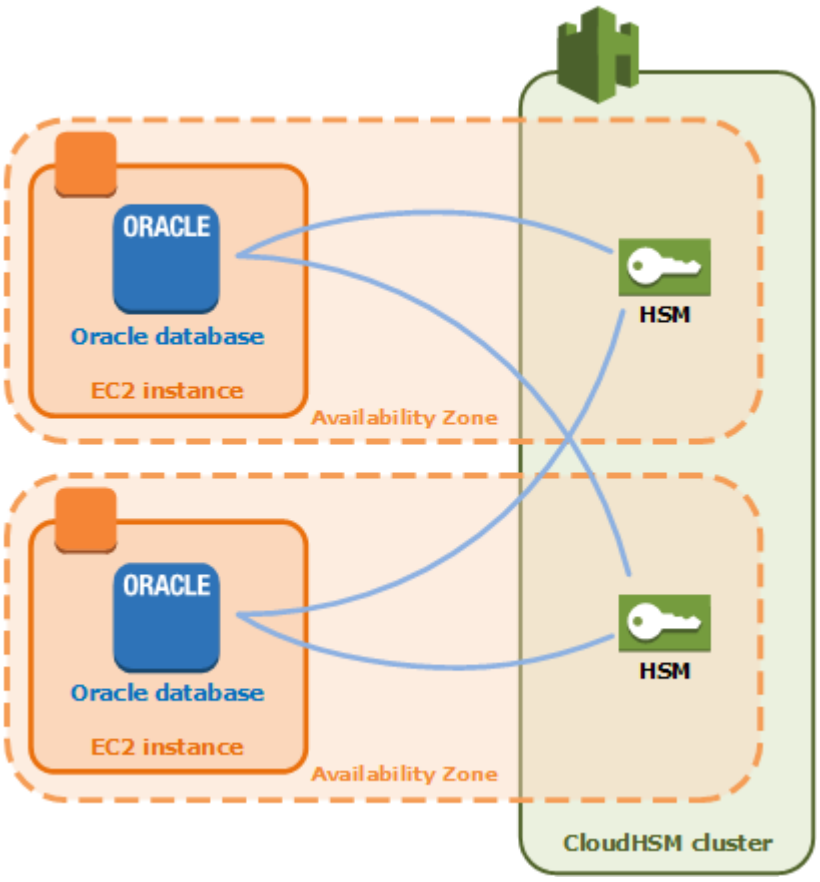
Si les HSM d'un cluster ne sont pas synchronisés, il les AWS CloudHSM resynchronise automatiquement. Pour ce faire, AWS CloudHSM utilise les informations d'identification de l'utilisateur de l'appliance. Cet utilisateur existe sur tous les HSM fournis par AWS CloudHSM et dispose d'autorisations limitées. Il peut obtenir un hachage d'objets sur le HSM, et extraire et insérer des objets masqués (chiffrés). AWS ne peut pas afficher ou modifier vos utilisateurs ou vos clés, et ne peut pas effectuer d'opérations de chiffrement à l'aide de ces clés.

Haute disponibilité et équilibrage de charge du cluster

Lorsque vous créez un AWS CloudHSM cluster avec plusieurs HSM, vous obtenez automatiquement un équilibrage de charge. L'équilibrage de charge signifie que le client AWS CloudHSM répartit les opérations de chiffrement entre tous les HSM du cluster en fonction de la capacité de chacun à prendre en charge un traitement supplémentaire.

Lorsque vous créez les HSM dans différentes zones de AWS disponibilité, vous bénéficiez automatiquement d'une haute disponibilité. La haute disponibilité signifie que vous bénéficiez d'une fiabilité supérieure, car aucun HSM individuel ne constitue de point de défaillance unique. Nous vous recommandons de disposer d'au moins deux HSM par cluster, chaque HSM se trouvant dans différentes zones de disponibilité au sein d'une AWS région.

Par exemple, la figure suivante montre une application de base de données Oracle distribuée dans deux zones de disponibilité différentes. Les instances de base de données stockent leurs clés principales dans un cluster qui inclut un HSM dans chaque zone de disponibilité. AWS CloudHSM synchronise automatiquement les clés avec les deux HSM afin qu'elles soient immédiatement accessibles et redondantes.



Connectez le SDK client au cluster AWS CloudHSM

Pour vous connecter au cluster avec le SDK client 5 ou le SDK client 3, vous devez d'abord effectuer deux opérations :

- Disposer d'un certificat d'émission sur l'instance EC2
- Démarrez le SDK client sur le cluster

Placez le certificat émetteur sur chaque instance EC2

Vous créez le certificat émetteur lorsque vous initialisez le cluster. Copiez le certificat émetteur à l'emplacement par défaut de la plateforme sur chaque instance EC2 connectée au cluster.

Linux

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

Windows

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

Spécifier l'emplacement du certificat émetteur

Avec le SDK client 5, vous utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

PKCS #11 library

Pour placer le certificat émetteur sous Linux pour un SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

Pour placer le certificat émetteur sous Windows pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

OpenSSL Dynamic Engine

Pour placer le certificat émetteur sous Linux pour un SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

JCE provider

Pour placer le certificat émetteur sous Linux pour un SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

Pour placer le certificat émetteur sous Windows pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

CloudHSM CLI

Pour placer le certificat émetteur sous Linux pour un SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

Pour placer le certificat émetteur sous Windows pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

Pour plus d'informations, veuillez consulter L'outil de configuration.

Pour plus d'informations sur l'initialisation du cluster ou sur la création et la signature du certificat, veuillez consulter Initialiser le cluster.

Amorcez le SDK client

Le processus d'amorçage est différent selon la version du SDK client que vous utilisez, mais vous devez disposer de l'adresse IP de l'un des modules de sécurité matériels (HSM) du cluster. Vous pouvez utiliser l'adresse IP de n'importe quel HSM connecté à votre cluster. Une fois le SDK client connecté, il récupère les adresses IP de tous les HSM supplémentaires et effectue des opérations d'équilibrage de charge et de synchronisation des clés côté client.

Pour obtenir une adresse IP pour le cluster

Pour obtenir une adresse IP pour un HSM (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Pour changer de région AWS, utilisez le Sélecteur de région dans l'angle supérieur droit de la page.
3. Pour ouvrir la page détaillée du cluster, choisissez l'ID du cluster dans le tableau des clusters.
4. Pour obtenir l'adresse IP, dans l'onglet HSM, choisissez l'une des adresses IP répertoriées sous Adresse IP ENI.

Pour obtenir une adresse IP pour un HSM (CLI)

- Obtenez l'adresse IP d'un HSM à l'aide de la describe-clusters commande de la CLI. Dans la sortie de la commande, l'adresse IP des HSM sont les valeurs de `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
```

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

Pour plus d'informations sur l'amorçage, voir l'outil de configuration.

Pour démarrer le SDK client 5

PKCS #11 library

Pour démarrer une instance EC2 Linux pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

Pour démarrer une instance EC2 Windows pour le client SDK 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

OpenSSL Dynamic Engine

Pour démarrer une instance EC2 Linux pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

JCE provider

Pour démarrer une instance EC2 Linux pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

Pour démarrer une instance EC2 Windows pour le client SDK 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

CloudHSM CLI

Pour démarrer une instance EC2 Linux pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP du ou des HSM de votre cluster.

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

Pour démarrer une instance EC2 Windows pour le client SDK 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP du ou des HSM de votre cluster.

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

> **Note**
>
> vous pouvez utiliser le paramètre `--cluster-id` à la place de`-a <HSM_IP_ADDRESSES>`. Pour connaître les conditions d'utilisation de `--cluster-id`, consultez Outil de configuration du SDK client 5.

Pour démarrer le SDK client 3

Pour démarrer une instance EC2 Linux pour le SDK client 3

- configureÀ utiliser pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

Pour démarrer une instance EC2 Windows pour le SDK client 3

- configureÀ utiliser pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

Pour plus d'informations sur la configuration, veuillez consulter ???.

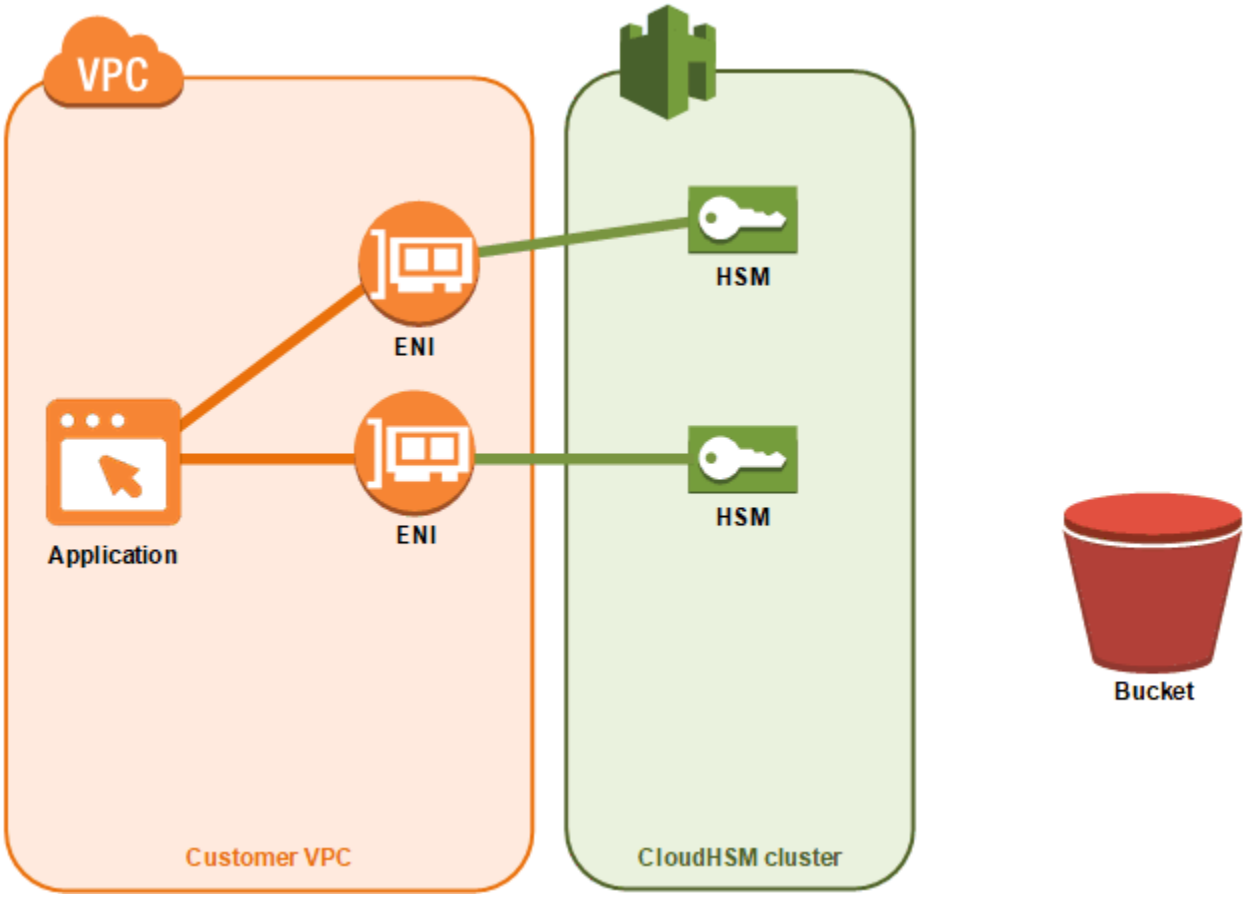
Ajouter ou supprimer des HSM dans un cluster AWS CloudHSM

Pour augmenter ou diminuer la taille de votre AWS CloudHSM cluster, ajoutez ou supprimez des HSM à l'aide de la AWS CloudHSM console, de l'un des AWS SDK ou des outils de ligne de commande. Nous vous recommandons de tester la charge de votre cluster pour déterminer le pic de charge auquel vous devez vous attendre, puis d'y ajouter un ou plusieurs HSM supplémentaires pour garantir une haute disponibilité.

Rubriques

Ajout d'un HSM

La figure suivante illustre les événements qui se produisent lorsque vous ajoutez un HSM à un cluster.



1. Vous ajoutez un nouvel HSM à un cluster. Les procédures suivantes expliquent comment procéder à partir de la AWS CloudHSM console, de la AWS Command Line Interface (CLI) et de l'AWS CloudHSM API.

   Il s'agit de la seule action que vous effectuez. Les événements restants se produisent automatiquement.

2. AWS CloudHSM crée une copie de sauvegarde d'un HSM existant dans le cluster. Pour plus d'informations, consultez Sauvegardes.

3. AWS CloudHSM restaure la sauvegarde sur le nouveau HSM. Cela permet de s'assurer que le HSM est synchronisé avec les autres HSM du cluster.
4. Les HSM existants dans le cluster notifient au AWS CloudHSM client la présence d'un nouveau HSM dans le cluster.
5. Le client établit une connexion au nouvel HSM.

Pour ajouter un HSM (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Choisissez un cluster pour le HSM que vous ajoutez.
3. Sous l'onglet HSM, choisissez Create HSM.
4. Choisissez une zone de disponibilité (AZ) pour le HSM que vous créez. Ensuite, choisissez Créer.

Pour ajouter un HSM (CLI)

- À l'invite de commande, exécutez la commande create-hsm, en spécifiant un ID de cluster et une zone de disponibilité pour le HSM que vous créez. Si vous ne connaissez pas l'ID de cluster de votre cluster favori, exécutez la commande describe-clusters. Spécifiez la zone de disponibilité dans le format suivant : `us-east-2a`, `us-east-2b`, etc.

```
$ 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"
    }
}
```

Pour ajouter un HSM (AWS CloudHSM API)

- Envoyez une demande CreateHsm, en spécifiant l'ID de cluster et une zone de disponibilité pour le HSM que vous créez.

Suppression d'un HSM

Vous pouvez supprimer un HSM à l'aide de la AWS CloudHSM console, de la CLI ou de l' AWS CloudHSM API.

Pour supprimer un HSM (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Choisissez le cluster qui contient le HSM que vous supprimez.
3. Sous l'onglet HSM, choisissez le HSM que vous supprimez. Ensuite, choisissez Delete HSM.
4. Confirmez que vous voulez supprimer le HSM. Ensuite, choisissez Supprimer.

Pour supprimer un HSM (CLI)

- À partir d'une invite de commande, exécutez la commande delete-hsm. Transmettez l'ID du cluster qui contient le HSM que vous supprimez et l'un des identificateurs HSM suivants :
  - ID du HSM (`--hsm-id`)
  - Adresse IP du HSM (`--eni-ip`)
  - ID d'interface réseau Elastic du HSM (`--eni-id`)

  Si vous ne connaissez pas les valeurs de ces identificateurs, exécutez la commande describe-clusters.

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

Pour supprimer un HSM (AWS CloudHSM API)

- Envoyez une demande DeleteHsm, en spécifiant l'ID de cluster et un identificateur pour le HSM que vous supprimez.

Supprimer un AWS CloudHSM cluster

Pour pouvoir supprimer un cluster, vous devez supprimer tous les HSM du cluster. Pour plus d'informations, consultez Suppression d'un HSM.

Après avoir supprimé tous les HSM, vous pouvez supprimer un cluster à l'aide de la AWS CloudHSM console, de la AWS Command Line Interface (CLI) ou de l' AWS CloudHSM API.

Pour supprimer un cluster (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Choisissez le cluster à supprimer. Ensuite, choisissez Delete cluster.
3. Confirmez que vous voulez supprimer le cluster, puis choisissez Delete.

Pour supprimer un cluster (CLI)

- À l'invite de commande, exécutez la commande delete-cluster, en spécifiant l'ID du cluster à supprimer. Si vous ne connaissez pas l'ID du cluster, exécutez la commande describe-clusters.

```
$ 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"
    }
}
```

Pour supprimer un cluster AWS CloudHSM (API)

- Envoyez une demande DeleteCluster, en spécifiant l'ID du cluster à supprimer.

Création de AWS CloudHSM clusters à partir de sauvegardes

Pour restaurer un AWS CloudHSM cluster à partir d'une sauvegarde, suivez les étapes décrites dans cette rubrique. Ce cluster contiendra les mêmes utilisateurs, éléments de clé, certificats, configuration et stratégies que la sauvegarde que vous avez restaurée. Pour plus d'informations sur la gestion des sauvegardes, consultez Gestion des sauvegardes.

Création de clusters à partir de sauvegardes (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Choisissez Créer un cluster.
3. Dans la section Configuration de cluster, effectuez les opérations suivantes :
    a. Pour VPC, choisissez un VPC pour le cluster que vous créez.
    b. Pour AZ(s), choisissez un sous-réseau privé pour chaque zone de disponibilité que vous ajoutez au cluster.
4. Dans la section Cluster source, effectuez les opérations suivantes :
    a. Choisissez Restore cluster from existing backup.
    b. Choisissez la sauvegarde que vous restaurez.
5. Choisissez Suivant : vérification.
6. Vérifiez la configuration de votre cluster, puis choisissez Create cluster.
7. Spécifiez la durée pendant laquelle le service doit conserver les sauvegardes.

Acceptez la période de conservation par défaut de 90 jours ou saisissez une nouvelle valeur comprise entre 7 et 379 jours. Le service supprimera automatiquement les sauvegardes de ce cluster plus anciennes que la valeur que vous spécifiez ici. Cette valeur peut être modifiée par la suite. Pour plus d'informations, consultez Configuration de la rétention des sauvegardes.

8. Choisissez Suivant.
9. (Facultatif) Saisissez une clé de balise et une valeur de balise facultative. Pour ajouter plusieurs balises au cluster, sélectionnez Ajouter une balise.
10. Choisissez Examiner.
11. Vérifiez la configuration de votre cluster, puis choisissez Create cluster (Créer un cluster).

> **Tip**
>
> Pour créer un HSM dans ce cluster contenant les mêmes utilisateurs, éléments clés, certificats, configurations et politiques que ceux de la sauvegarde que vous avez restaurée, ajoutez un HSM au cluster.

Création de clusters à partir de sauvegardes (CLI)

Pour déterminer l'ID de sauvegarde, exécutez la commande describe-backups.

- À partir d'une invite de commande, exécutez la commande create-cluster. Spécifiez le type d'instance HSM, les ID de sous-réseau des sous-réseaux sur lesquels vous envisagez de créer des HSM, et l'ID de sauvegarde de la sauvegarde que vous restaurez.

```
$ 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"
    }
}
```

Création de clusters à partir de sauvegardes (AWS CloudHSM API)

Reportez-vous à la rubrique suivante pour savoir comment créer des clusters à partir de sauvegardes à l'aide de l'API.

- CreateCluster

Gestion des AWS CloudHSM sauvegardes

AWS CloudHSM effectue des sauvegardes périodiques de votre cluster au moins une fois toutes les 24 heures. Chaque sauvegarde contient des copies chiffrées des données suivantes :

- Utilisateurs (CO, CU et AU)
- Matériel de clé et certificats
- Configuration et stratégies du module de sécurité matérielle (HSM)

Vous ne pouvez pas demander au service d'effectuer des sauvegardes, mais vous pouvez effectuer certaines actions qui obligent le service à créer une sauvegarde. Le service effectue une sauvegarde lorsque vous effectuez l'une des actions suivante :

- Activation d'un cluster
- Ajout d'un HSM à un cluster actif
- Suppression d'un HSM d'un cluster actif

AWS CloudHSM supprime les sauvegardes conformément à la politique de conservation des sauvegardes que vous avez définie lors de la création de clusters. Pour plus d'informations sur la gestion de la stratégie de rétention des sauvegardes, consultez Configuration de la rétention des sauvegardes.

Rubriques

Utilisation des sauvegardes

Lorsque vous ajoutez un HSM à un cluster qui contenait auparavant un ou plusieurs HSM actifs, le service restaure la dernière sauvegarde sur le nouvel HSM. Utilisez des sauvegardes pour gérer les HSM que vous utilisez rarement. Lorsque vous n'avez pas besoin d'utiliser le HSM, vous pouvez le supprimer pour déclencher une sauvegarde. Plus tard, lorsque vous aurez besoin du HSM, créez-

en un nouveau dans le même cluster, et cette action restaurera la sauvegarde que vous avez créée précédemment avec l'opération de suppression du HSM.

Suppression des clés expirées ou des utilisateurs inactifs

Vous pouvez souhaiter supprimer certains matériaux de chiffrement non désirés de votre environnement, par exemple des clés expirées ou des utilisateurs inactifs. Ce processus se divise en deux parties : Tout d'abord, supprimez ces documents de votre HSM. Supprimez ensuite toutes les sauvegardes existantes. Ce processus garantit que vous ne restaurez pas les informations supprimées lors de l'initialisation d'un nouveau cluster à partir d'une sauvegarde. Pour plus d'informations, consultez the section called "Suppression et restauration de sauvegardes".

Prise en compte de reprise après sinistre

Vous pouvez créer un cluster à partir d'une sauvegarde. Vous pouvez le faire pour définir un point de récupération pour votre cluster. Désignez une sauvegarde contenant tous les utilisateurs, les éléments clés et les certificats que vous souhaitez ajouter à votre point de restauration, puis utilisez cette sauvegarde pour créer un nouveau cluster. Pour de plus amples informations sur la création d'un cluster à partir d'une sauvegarde, veuillez consulter Création de clusters à partir de sauvegardes.

Vous pouvez également copier une sauvegarde d'un cluster dans une autre région, qui pourra ensuite être utilisé pour créer un cluster comme clone de l'original. Vous pouvez le faire pour un certain nombre de raisons, y compris pour simplifier le processus de reprise après sinistre. Pour plus d'information sur la copie d'une sauvegarde dans plusieurs régions, consultez Copie de sauvegardes dans plusieurs régions.

Suppression et restauration de sauvegardes

Après avoir supprimé une sauvegarde, le service la conserve pendant sept jours, période pendant laquelle vous pouvez la restaurer. Après cette période de sept jours, vous ne pourrez plus restaurer la sauvegarde. Pour de plus amples informations sur la gestion des sauvegardes, consultez Gestion des sauvegardes.

Supprimer et restaurer des sauvegardes (console)

Pour supprimer une sauvegarde (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Pour changer de région AWS, utilisez le Sélecteur de région dans l'angle supérieur droit de la page.
3. Dans le volet de navigation, choisissez Sauvegardes.
4. Choisissez une sauvegarde à supprimer.
5. Pour supprimer la sauvegarde sélectionnée, choisissez Actions, Supprimer.

   La boîte de dialogue Supprimer les sauvegardes s'affiche.
6. Sélectionnez Delete (Supprimer).

   L'état de la sauvegarde passe sur `PENDING_DELETE`. Vous pouvez restaurer une sauvegarde en attente de suppression jusqu'à 7 jours après en avoir demandé la suppression.

Pour restaurer une sauvegarde (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Pour changer de région AWS, utilisez le Sélecteur de région dans l'angle supérieur droit de la page.
3. Dans le volet de navigation, choisissez Sauvegardes.
4. Choisissez une sauvegarde dans l'état `PENDING_DELETE` à restaurer.
5. Pour restaurer la sauvegarde sélectionnée, choisissez Actions, Restaurer.

Supprimer et restaurer des sauvegardes (CLI)

Vérifiez l'état d'une sauvegarde ou recherchez son identifiant à l'aide de la describe-backups commande de la CLI.

Pour supprimer une sauvegarde (CLI)

- À partir d'une invite de commande, exécutez la commande delete-backup, en transmettant l'ID de la sauvegarde à supprimer.

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

Pour restaurer une sauvegarde (CLI)

- Pour restaurer une sauvegarde, exécutez la commande restore-backup, en transmettant l'ID d'une sauvegarde qui est à l'état `PENDING_DELETION`.

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

Pour répertorier les sauvegardes (CLI)

- Si vous souhaitez afficher une liste de toutes les sauvegardes dont l'état est `PENDING_DELETION`, exécutez la commande describe-backups et incluez `states=PENDING_DELETION` comme filtre.

```
$ aws cloudhsmv2 describe-backups --filters states=PENDING_DELETION
{
```

```
    "Backups": [
        {
            "BackupId": "backup-ro5c4er4aac",
            "BackupState": "PENDING_DELETION",
            "CreateTimestamp": 1534461854.64,
            "ClusterId": "cluster-dygnwhmscg5",
            "DeleteTimestap": 1536339805.522,
        }
}
```

Supprimer et restaurer des sauvegardes (AWS CloudHSM API)

Consultez les rubriques suivantes pour apprendre à supprimer et restaurer des sauvegardes à l'aide de l'API.

- DeleteBackup
- RestoreBackup

Configuration de la politique AWS CloudHSM de conservation des sauvegardes

À l'exception des clusters créés avant le 18 novembre 2020, la stratégie de rétention des sauvegardes par défaut pour les clusters est de 90 jours. Vous pouvez définir cette période sur un nombre compris entre 7 et 379 jours. AWS CloudHSM ne supprime pas la dernière sauvegarde d'un cluster. Pour de plus amples informations sur la gestion des sauvegardes, consultez Gestion des sauvegardes.

Présentation de la stratégie de rétention des sauvegardes

AWS CloudHSM purge les sauvegardes en fonction de la politique de conservation des sauvegardes que vous avez définie lors de la création d'un cluster. La stratégie de rétention des sauvegardes s'applique aux clusters. Si vous déplacez une sauvegarde vers une autre région, cette sauvegarde n'est plus associée à un cluster et ne fait l'objet d'aucune stratégie de rétention des sauvegardes. Vous devez supprimer manuellement toutes les sauvegardes non associées à un cluster. AWS CloudHSM ne supprime pas la dernière sauvegarde d'un cluster.

AWS CloudTrail signale les sauvegardes marquées pour suppression. Vous pouvez restaurer les sauvegardes purgées par le service de la même manière que vous restaureriez les sauvegardes

supprimées manuellement. Pour éviter une situation de course, vous devez modifier la stratégie de rétention des sauvegardes du cluster avant de restaurer une sauvegarde supprimée par le service. Si vous souhaitez conserver la même stratégie de rétention et conserver certaines sauvegardes, vous pouvez spécifier que le service exclut les sauvegardes de la stratégie de rétention des sauvegardes du cluster.

Exemption relative aux clusters existants

AWS CloudHSM a lancé la conservation gérée des sauvegardes le 18 novembre 2020. Les clusters créés avant le 18 novembre 2020 sont soumis à une stratégie de rétention des sauvegardes de 90 jours, plus l'âge du cluster. Par exemple, si vous avez créé un cluster le 18 novembre 2019, le service attribuera à votre cluster une stratégie de rétention des sauvegardes d'un an plus 90 jours (455 jours).

> **Note**
>
> Vous pouvez désactiver complètement la rétention gérée des sauvegardes en contactant le support (https://aws.amazon.com/support).

Configuration de la rétention des sauvegardes (console)

Pour configurer la stratégie de rétention des sauvegardes (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Pour changer de région AWS, utilisez le Sélecteur de région dans l'angle supérieur droit de la page.
3. Cliquez sur l'ID de cluster d'un cluster à l'état actif pour gérer la stratégie de rétention des sauvegardes pour ce cluster.
4. Pour modifier la stratégie de rétention des sauvegardes, choisissez Actions, Modifier la période de rétention des sauvegardes.

   La boîte de dialogue Modifier la période de rétention des sauvegardes s'affiche.
5. Dans Période de rétention des sauvegardes (en jours), saisissez une valeur comprise entre 7 et 379 jours.
6. Choisissez Modifier la période de rétention des sauvegardes.

Pour exclure ou inclure une sauvegarde de la stratégie de rétention des sauvegardes (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Pour afficher vos sauvegardes, choisissez Sauvegardes dans le panneau de navigation.
3. Cliquez sur l'ID de sauvegarde d'une sauvegarde dont l'état est Prêt à exclure ou à inclure.
4. Sur la page Détails de la sauvegarde, effectuez l'une des actions suivante.
   - Pour exclure une sauvegarde dont la date est indiquée dans l'Heure d'expiration, choisissez Actions, puis Désactiver l'expiration.
   - Pour inclure une sauvegarde qui n'expire pas, choisissez Actions, Utiliser la stratégie de rétention du cluster.

Configuration de la conservation des sauvegardes (CLI)

Vérifiez l'état d'une sauvegarde ou recherchez son identifiant à l'aide de la describe-backups commande de la CLI.

Pour configurer la politique de conservation des sauvegardes (CLI)

- À partir d'une invite de commande, exécutez la commande modify-cluster. Spécifiez l'ID du cluster et la stratégie de rétention des sauvegardes.

```
$ 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"
   }
}
```

Pour exclure une sauvegarde de la politique de conservation des sauvegardes (CLI)

- À partir d'une invite de commande, exécutez la commande modify-backup-attributes. Spécifiez l'ID de sauvegarde et définissez l'indicateur never expires pour préserver la sauvegarde.

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

Pour inclure une sauvegarde dans la politique de conservation des sauvegardes (CLI)

- À partir d'une invite de commande, exécutez la commande modify-backup-attributes. Spécifiez l'ID de sauvegarde et définissez l' no-never-expires indicateur pour inclure la sauvegarde dans la politique de conservation des sauvegardes, ce qui signifie que le service finira par supprimer la sauvegarde.

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

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

Configuration de la conservation des sauvegardes (AWS CloudHSM API)

Consultez les rubriques suivantes pour apprendre à gérer la rétention des sauvegardes à l'aide de l'API.

- ModifyCluster
- ModifyBackupAttributes

Copier des sauvegardes entre AWS les régions

Vous pouvez copier des sauvegardes d'une région à l'autre pour de nombreuses raisons, notamment la résilience entre les régions, les charges de travail mondiales et la reprise après sinistre. Une fois que vous avez copié les sauvegardes, elles apparaissent dans la région de destination avec le statut `CREATE_IN_PROGRESS`. Une fois l'exécution terminée, le statut de la sauvegarde devient `READY`. Si la copie échoue, le statut de la sauvegarde devient `DELETED`. Vérifiez vos paramètres d'entrée pour rechercher des erreurs et vous assurer que la sauvegarde source spécifiée n'est pas dans un état `DELETED` avant de réexécuter l'opération. Pour plus d'informations sur les sauvegardes ou les modalités de création d'un cluster à partir d'une sauvegarde, consultez Gestion des sauvegardes ou Création de clusters à partir de sauvegardes.

Notez ce qui suit :

- Pour copier une sauvegarde de cluster dans une région de destination, votre compte doit disposer des autorisations de politique IAM appropriées. Pour copier la sauvegarde dans une autre région, votre politique IAM doit permettre l'accès à la région source dans laquelle la sauvegarde est située. Une fois copiée entre les régions, votre politique IAM doit autoriser l'accès à la région de destination afin d'interagir avec la sauvegarde copiée, qui inclut l'utilisation de l'opération CreateCluster. Pour plus d'informations, consultez Création d'administrateurs IAM.

- Le cluster d'origine et le cluster qui peut être créé à partir d'une sauvegarde dans la région de destination ne sont pas liés. Vous devez gérer chacun de ces clusters de manière indépendante. Pour plus d'informations, consultez Gestion des clusters .
- Les sauvegardes ne peuvent pas être copiées entre les régions AWS restreintes et les régions standard. Les sauvegardes peuvent être copiées entre les AWS GovCloud régions (USA Est) et AWS GovCloud (USA Ouest).

Copier des sauvegardes vers différentes régions (console)

Pour copier des sauvegardes vers différentes régions (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Pour changer de région AWS, utilisez le Sélecteur de région dans l'angle supérieur droit de la page.
3. Dans le volet de navigation, choisissez Sauvegardes.
4. Choisissez une sauvegarder à copier dans une région différente.
5. Pour copier la sauvegarde sélectionnée, choisissez Actions, puis Copier la sauvegarde vers une autre région.

   La boîte de dialogue Copier la sauvegarde dans une autre région apparaît.
6. Dans Région de destination, choisissez une région dans Sélectionner une région.
7. (Facultatif) Saisissez une clé de balise et une valeur de balise facultative. Pour ajouter plusieurs balises au cluster, sélectionnez Ajouter une balise.
8. Choisissez Copier la sauvegarde.

Copier des sauvegardes vers différentes régions (CLI)

Pour identifier l'ID de sauvegarde, exécutez la commande describe-backups.

Pour copier des sauvegardes vers différentes régions (CLI)

- À partir d'une invite de commande, exécutez la commande copy-backup-to-region. Spécifiez la région de destination et l'ID de sauvegarde de la sauvegarde source. Si vous spécifiez un ID de sauvegarde, la sauvegarde associée est copiée.

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

Copier des sauvegardes vers différentes régions (AWS CloudHSM API)

Reportez-vous à la rubrique suivante pour savoir comment copier des sauvegardes vers différentes régions à l'aide de l'API.

- CopyBackupToRegion

Ressources de balisage AWS CloudHSM

Une étiquette est une étiquette que vous attribuez à une AWS ressource. Vous pouvez attribuer des balises à vos clusters AWS CloudHSM . Chaque balise est constituée d'une clé de balise et d'une valeur de balise que vous définissez. Par exemple, vous pouvez choisir la clé de balise centre de coûts et la valeur de balise 12345. Les clés de balise doivent être uniques pour chaque cluster.

Vous pouvez utiliser les balises à diverses fins. Généralement, elles sont utilisées pour classer vos coûts AWS par catégorie et en effectuer le suivi. Vous pouvez appliquer des balises associées à des catégories métier (telles que les centres de coûts, les noms d'applications ou les propriétaires) pour organiser les coûts relatifs à divers services. Lorsque vous ajoutez des balises à vos AWS ressources, AWS génère un rapport de répartition des coûts avec l'utilisation et les coûts agrégés par balises. Vous pouvez utiliser ce rapport pour visualiser vos AWS CloudHSM coûts en termes de projets ou d'applications, au lieu de visualiser tous les AWS CloudHSM coûts sur une seule ligne.

Pour en savoir plus sur l'utilisation des balises pour l'allocation des coûts, veuillez consulter Utilisation des balises de répartition des coûts dans le Guide de l'utilisateur AWS Billing .

Vous pouvez utiliser la console AWS CloudHSM, ou l'un des kits SDK ou outils de ligne de commande AWS, pour ajouter, mettre à jour, afficher et supprimer des balises.

Rubriques

- Ajout ou mise à jour de balises
- Établissement d'une liste de balises
- Suppression de balises

Ajout ou mise à jour de balises

Vous pouvez ajouter ou mettre à jour des balises à partir de la AWS CloudHSM console, de la AWS Command Line Interface (CLI) ou de l' AWS CloudHSM API.

Pour ajouter ou mettre à jour des balises (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Choisissez le cluster que vous balisez.

3. Choisissez Tags.
4. Pour ajouter une balise, procédez comme suit :
   a. Choisissez Edit Tag (Modifier une balise) puis Add Tag (Ajouter une balise).
   b. Dans Tag Key (Clé de balise), précisez la clé de la balise.
   c. (Facultatif) Dans Value (Valeur), entrez une valeur pour la balise.
   d. Choisissez Enregistrer.
5. Pour mettre à jour une balise, procédez comme suit :
   a. Choisissez Edit Tag (Modifier une balise).

      > **Note**
      >
      > Si vous mettez à jour la clé de balise d'une balise existante, la console supprime la balise existante et en crée une nouvelle.

   b. Tapez la nouvelle valeur de balise.
   c. Choisissez Enregistrer.

Pour ajouter ou mettre à jour des balises (CLI)

1. À l'invite de commande, exécutez la commande tag-resource, en spécifiant les balises et l'ID du cluster que vous balisez. Si vous ne connaissez pas l'ID du cluster, exécutez la commande describe-clusters.

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

2. Pour mettre à jour les balises, utilisez la même commande, mais spécifiez une clé de balise existante. Lorsque vous spécifiez une nouvelle valeur de balise pour une balise existante, la balise est remplacée par la nouvelle valeur.

Pour ajouter ou mettre à jour des balises (AWS CloudHSM API)

- Envoyez une demande TagResource. Spécifiez les balises et l'ID du cluster que vous balisez.

Établissement d'une liste de balises

Vous pouvez répertorier les balises d'un cluster à partir de la AWS CloudHSM console, de la CLI ou de l' AWS CloudHSM API.

Pour afficher les balises (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Choisissez le cluster dont vous voulez afficher les balises.
3. Choisissez Tags.

Pour répertorier les tags (CLI)

- À l'invite de commande, exécutez la commande list-tags, en spécifiant les balises et l'ID du cluster dont vous affichez les balises. Si vous ne connaissez pas l'ID du cluster, exécutez la commande describe-clusters.

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

Pour répertorier les tags (AWS CloudHSM API)

- Envoyez une demande ListTags, en spécifiant l'ID du cluster dont vous affichez les balises.

Suppression de balises

Vous pouvez supprimer des balises d'un cluster à l'aide de la AWS CloudHSM console, de la CLI ou de l' AWS CloudHSM API.

Pour supprimer les balises (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Choisissez le cluster dont vous supprimez les balises.
3. Choisissez Tags.
4. Choisissez Edit Tag (Modifier la balise), puis Remove tag (Supprimer la balise) pour la balise que vous souhaitez supprimer.
5. Choisissez Enregistrer.

Pour supprimer des balises (CLI)

- À l'invite de commande, exécutez la commande untag-resource, en spécifiant les clés des balises que vous supprimez et l'ID du cluster dont vous supprimez les balises. Lorsque vous utilisez la CLI pour supprimer des balises, spécifiez uniquement les clés des balises, pas les valeurs des balises.

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

Pour supprimer des balises (AWS CloudHSM API)

- Envoyez une UntagResourcedemande dans l' AWS CloudHSM API, en spécifiant l'ID du cluster et les balises que vous souhaitez supprimer.

Gestion des utilisateurs et des clés HSM dans AWS CloudHSM

Avant de pouvoir utiliser votre AWS CloudHSM cluster pour le cryptotraitement, vous devez créer des utilisateurs et des clés sur les HSM de votre cluster. Consultez les rubriques suivantes pour plus d'informations sur la gestion des utilisateurs et des clés HSM dans AWS CloudHSM. Vous pouvez également apprendre à utiliser l'authentification par quorum (également connue sous le nom de contrôle d'accès M sur N).

Rubriques

Gestion des utilisateurs HSM dans AWS CloudHSM

Dans AWS CloudHSM, vous devez utiliser les outils de ligne de commande de la CLI CloudHSM ou de l'utilitaire de gestion CloudHSM (CMU) pour créer et gérer les utilisateurs sur votre HSM. La CLI CloudHSM est conçue pour être utilisée avec les dernières séries de versions de SDK, tandis que le CMU est conçu pour être utilisé avec les séries de versions de SDK précédentes.

Rubriques

Gestion des utilisateurs HSM à l'aide de la CLI CloudHSM

Utilisez les outils de ligne de commande de la CLI CloudHSM pour créer et gérer les utilisateurs de votre HSM avec le dernier SDK.

Rubriques

- Utilisation de la CLI CloudHSM pour gérer la MFA
- Utilisation de la CLI CloudHSM pour gérer l'authentification par quorum (contrôle d'accès M sur N)

Comprendre les utilisateurs HSM

La plupart des opérations que vous effectuez sur le HSM nécessitent les informations d'identification d'un utilisateur HSM. Le HSM authentifie chaque utilisateur HSM et chaque utilisateur HSM possède un type qui détermine les opérations que vous pouvez effectuer sur le HSM en tant qu'utilisateur.

> **Note**
>
> Les utilisateurs HSM sont distincts des utilisateurs IAM. Les utilisateurs IAM qui disposent des informations d'identification correctes peuvent créer des HSM en interagissant avec les ressources via l'API AWS. Une fois le HSM créé, vous devez utiliser les informations d'identification de l'utilisateur HSM pour authentifier les opérations sur le HSM.

Types d'utilisateur

- Administrateur non activé
- Administrateur
- Utilisateur de chiffrement (CU)
- utilisateur de l'appareil (AU)

Administrateur non activé

Dans la CLI CloudHSM, l'administrateur non activé est un utilisateur temporaire qui n'existe que sur le premier HSM d'un cluster AWS CloudHSM qui n'a jamais été activé. Pour activer un cluster, exécutez la commande cluster activate dans la CLI CloudHSM. Après avoir exécuté cette commande, les administrateurs non activés sont invités à modifier le mot de passe. Après avoir modifié le mot de passe, l'administrateur non activé devient administrateur.

Administrateur

Dans la CLI CloudHSM, l'administrateur peut effectuer des opérations de gestion des utilisateurs. Par exemple, il peut créer et supprimer des utilisateurs, et modifier les mots de passe des utilisateurs. Pour plus d'informations sur les administrateurs, veuillez consulter le Tableau Autorisations des utilisateurs HSM.

Utilisateur de chiffrement (CU)

Un utilisateur de chiffrement (CU) peut effectuer les opérations de chiffrement et de gestion des clés suivantes.

- Gestion des clés - Créer, supprimer, partager, importer et exporter des clés de chiffrement.
- Opérations de chiffrement - Utiliser les clés de chiffrement pour le chiffrement, le déchiffrement, la signature, la vérification, et plus encore.

Pour plus d'informations, consultez le Tableau Autorisations des utilisateurs HSM.

utilisateur de l'appareil (AU)

L'utilisateur de l'appliance (AU) peut effectuer des opérations de clonage et de synchronisation sur les HSM de votre cluster. AWS CloudHSM utilise l'AU pour synchroniser les HSM d'un AWS CloudHSM cluster. L'AU existe sur tous les HSM fournis par AWS CloudHSM et dispose d'autorisations limitées. Pour plus d'informations, consultez le Tableau Autorisations des utilisateurs HSM.

AWS ne peut effectuer aucune opération sur vos HSM. AWS ne peut pas afficher ou modifier vos utilisateurs ou vos clés et ne peut effectuer aucune opération cryptographique à l'aide de ces clés.

Tableau Autorisations des utilisateurs HSM

Le tableau suivant répertorie les opérations HSM triées selon le type d'utilisateur ou de session HSM qui peut effectuer l'opération.

	Administrateur	Utilisateur de chiffrement (CU)	utilisateur de l'appareil (AU)	Session non authentifiée
Obtention d'informations de base sur le cluster[1]	 Oui	 Oui	 Oui	 Oui
Changement de son mot de passe	 Oui	 Oui	 Oui	Ne s'applique pas

	Administrateur	Utilisateur de chiffrement (CU)	utilisateur de l'appareil (AU)	Session non authentifiée
Changement du mot de passe d'un utilisateur	 Oui	 Non	 Non	 Non
Ajout et suppression d'utilisateurs	 Oui	 Non	 Non	 Non
Obtention du statut de la synchronisation[2]	 Oui	 Oui	 Oui	 Non
Extraction et insertion d'objets masqués[3]	 Oui	 Oui	 Oui	 Non
Fonctions de gestion des clés[4]	 Non	 Oui	 Non	 Non
Chiffrement et déchiffrement	 Non	 Oui	 Non	 Non
Connexion et vérification	 Non	 Oui	 Non	 Non

	Administrateur	Utilisateur de chiffrement (CU)	utilisateur de l'appareil (AU)	Session non authentifiée
Génération de synthèses et de HMAC	 Non	 Oui	 Non	 Non

- [1] Les informations de base sur le cluster comprennent le nombre de HSM dans le cluster ainsi que l'adresse IP, le modèle, le numéro de série, l'ID d'appareil, l'ID de microprogramme, etc. de chaque HSM.
- [2] L'utilisateur peut obtenir un ensemble de résumés (hachages) qui correspondent aux clés du HSM. Une application peut comparer ces ensembles pour comprendre le statut de la synchronisation des HSM dans un cluster.
- [3] Les objets masqués sont des clés qui sont chiffrées avant de quitter le HSM. Elles ne peuvent pas être déchiffrées en dehors du HSM. Elles ne sont déchiffrées que lorsqu'elles sont insérées dans un HSM qui se trouve dans le même cluster que le HSM duquel elles ont été extraites. Une application peut extraire et insérer des objets masqués afin de synchroniser les HSM dans un cluster.
- [4] Les fonctions de gestion des clés incluent la création, la suppression, l'encapsulage, le désencapsulage et la modification des attributs de clés.

Utilisation de la CLI CloudHSM pour gérer les utilisateurs

Cette rubrique fournit des step-by-step instructions sur la gestion des utilisateurs du module de sécurité matérielle (HSM) à l'aide de la CLI CloudHSM. Pour plus d'informations sur les utilisateurs de la CLI CloudHSM ou du HSM, consultez CLI CloudHSM et Utilisation de la CLI CloudHSM.

Sections

- Comprendre la gestion des utilisateurs HSM avec la CLI CloudHSM
- Télécharger la CLI CloudHSM
- Comment gérer les utilisateurs HSM à l'aide de la CLI CloudHSM

Comprendre la gestion des utilisateurs HSM avec la CLI CloudHSM

Pour gérer les utilisateurs HSM, vous devez vous connecter au HSM avec le nom d'utilisateur et le mot de passe d'un administrateur. Seuls les administrateurs peuvent gérer les utilisateurs. Le HSM contient un administrateur par défaut appelé admin. Vous définissez le mot de passe pour admin lorsque vous avez activé le cluster.

Pour utiliser la CLI CloudHSM, vous devez utiliser l'outil de configuration pour mettre à jour la configuration locale. Pour obtenir des instructions sur l'exécution de l'outil de configuration avec la CLI CloudHSM, consultez Mise en route avec l'interface de ligne de commande (CLI) CloudHSM. Le paramètre `-a` vous oblige à ajouter l'adresse IP d'un HSM dans votre cluster. Si vous avez plusieurs HSM, vous pouvez utiliser n'importe quelle adresse IP. Cela garantit que la CLI CloudHSM peut propager toutes les modifications que vous apportez sur l'ensemble du cluster. N'oubliez pas que la CLI CloudHSM utilise son fichier local pour suivre les informations du cluster. Si le cluster a changé depuis la dernière fois que vous avez utilisé la CLI CloudHSM depuis un hôte particulier, vous devez ajouter ces modifications au fichier de configuration local stocké sur cet hôte. Ne supprimez jamais un HSM lorsque vous utilisez la CLI CloudHSM.

Pour obtenir une adresse IP pour un HSM (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Pour changer de région AWS, utilisez le Sélecteur de région dans l'angle supérieur droit de la page.
3. Pour ouvrir la page détaillée du cluster, choisissez l'ID du cluster dans le tableau des clusters.
4. Pour obtenir l'adresse IP, dans l'onglet HSM, choisissez l'une des adresses IP répertoriées sous Adresse IP ENI.

Pour obtenir une adresse IP pour un HSM (CLI)

- Obtenez l'adresse IP d'un HSM à l'aide de la describe-clusters commande de la CLI. Dans la sortie de la commande, l'adresse IP des HSM sont les valeurs de `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


{
    "Clusters": [
```

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

Télécharger la CLI CloudHSM

La dernière version de la CLI CloudHSM est disponible pour les tâches de gestion des utilisateurs HSM pour le SDK client 5. Pour télécharger et installer la CLI CloudHSM, suivez les instructions de la section Installation et configuration de la CLI CloudHSM.

Comment gérer les utilisateurs HSM à l'aide de la CLI CloudHSM

Cette section inclut les commandes de base permettant de gérer les utilisateurs HSM à l'aide de la CLI CloudHSM.

> **Note**
>
> Remarque : les commandes utilisateur de la CLI CloudHSM sont répertoriées dans la référence des commandes utilisateur de la CLI CloudHSM

Rubriques

- Pour créer un administrateur
- Pour créer un utilisateur de chiffrement
- Pour répertorier tous les utilisateurs HSM du cluster
- Pour modifier les mots de passe utilisateur HSM
- Pour supprimer des utilisateurs HSM

Pour créer un administrateur

Suivez ces étapes pour créer un administrateur.

1. Utilisez la commande suivante pour démarrer le mode interactif de la CLI CloudHSM.

   Linux

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

   Windows

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

2. Utilisez la commande login et connectez-vous au cluster en tant qu'administrateur.

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

3. Le système vous invite à saisir votre mot de passe. Entrez le mot de passe, et le résultat indique que la commande a été exécutée avec succès.

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

4. Entrez la commande suivante pour créer un administrateur.

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

5. Saisissez le mot de passe du nouvel utilisateur.
6. Entrez à nouveau le mot de passe pour confirmer que le mot de passe que vous avez saisi est correct.

Pour créer un utilisateur de chiffrement

Exécutez la procédure ci-dessous pour créer un utilisateur.

1. Utilisez la commande suivante pour démarrer le mode interactif de la CLI CloudHSM.

Linux

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

Windows

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

2. Utilisez la commande login et connectez-vous au cluster en tant qu'administrateur.

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

3. Le système vous invite à saisir votre mot de passe. Entrez le mot de passe, et le résultat indique que la commande a été exécutée avec succès.

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

4. Saisissez la commande suivante pour créer un utilisateur de chiffrement :

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

5. Saisissez le mot de passe du nouvel utilisateur de chiffrement.
6. Entrez à nouveau le mot de passe pour confirmer que le mot de passe que vous avez saisi est correct.

Pour répertorier tous les utilisateurs HSM du cluster

Utilisez la commande user list pour répertorier tous les utilisateurs du cluster. Vous n'avez pas besoin de vous connecter pour exécuter user list. Tous les types d'utilisateurs peuvent répertorier des utilisateurs.

Procédez comme suit pour répertorier tous les utilisateurs du cluster

1. Utilisez la commande suivante pour démarrer le mode interactif de la CLI CloudHSM.

   Linux

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

   Windows

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

2. Saisissez la commande suivante pour répertorier tous les utilisateurs du cluster :

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

Pour plus d'informations sur user list, veuillez consulter la rubrique Répertorier les utilisateurs.

Pour modifier les mots de passe utilisateur HSM

Utilisez la commande user change-password pour modifier un mot de passe.

Les types d'utilisateurs et les mots de passe sont sensibles à la casse, les noms d'utilisateurs, non.

Les utilisateurs de chiffrement (CU) et les utilisateurs d'appliance (AU) peuvent modifier uniquement leur propre mot de passe. Pour modifier le mot de passe d'un autre utilisateur, vous devez vous connecter en tant qu'administrateur. Vous ne pouvez pas modifier le mot de passe d'un utilisateur qui est actuellement connecté..

Pour modifier votre propre mot de passe

1. Utilisez la commande suivante pour démarrer le mode interactif de la CLI CloudHSM.

   Linux

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

   Windows

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

2. Utilisez la commande login et connectez-vous en tant qu'utilisateur avec le mot de passe que vous souhaitez modifier.

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

3. Entrer le mot de passe de l'utilisateur.

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

4. Entrez la commande user change-password.

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

5. Saisissez le nouveau mot de passe.
6. Saisissez à nouveau le nouveau mot de passe.

Pour modifier le mot de passe d'un autre utilisateur

1. Utilisez la commande suivante pour démarrer le mode interactif de la CLI CloudHSM.

   Linux

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

   Windows

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

2. Utilisez la commande login et connectez-vous en tant qu'administrateur.

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

3. Entrez le mot de passe de l'administrateur.

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

4. Saisissez la commande user change-password ainsi que le nom d'utilisateur de l'utilisateur dont vous souhaitez changer le mot de passe.

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

5. Saisissez le nouveau mot de passe.
6. Saisissez à nouveau le nouveau mot de passe.

Pour plus d'informations sur user change-password, veuillez consulter changer le mot de passe d'un utilisateur.

Pour supprimer des utilisateurs HSM

Utilisez user delete pour supprimer un utilisateur. Vous devez vous connecter en tant qu'administrateur pour supprimer un autre utilisateur.

> **Tip**
>
> Vous ne pouvez pas supprimer les utilisateurs de chiffrement (CU) qui possèdent des clés.

Pour supprimer un utilisateur

1. Utilisez la commande suivante pour démarrer le mode interactif de la CLI CloudHSM.

   Linux

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

Windows

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

2. Utilisez la commande login et connectez-vous au cluster en tant qu'administrateur.

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

3. Le système vous invite à saisir votre mot de passe. Entrez le mot de passe, et le résultat indique que la commande a été exécutée avec succès.

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

4. Utilisez la commande user delete pour supprimer l'utilisateur.

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

Pour plus d'informations sur user delete, veuillez consulter Supprimer un utilisateur.

Utilisation de la CLI CloudHSM pour gérer la MFA

Pour plus de sécurité, nous vous recommandons de configurer l'authentification multifactorielle (MFA) pour mieux protéger le cluster. Pour plus d'informations, consultez les rubriques ci-dessous.

Rubriques

- Comprendre la MFA pour les utilisateurs de HSM
- Utilisation de la MFA pour les utilisateurs de HSM

Comprendre la MFA pour les utilisateurs de HSM

Lorsque vous vous connectez à un cluster avec un compte utilisateur HSM compatible MFA, vous fournissez votre mot de passe à la CLI CloudHSM (le premier facteur, ce que vous savez) et la CLI CloudHSM vous fournit un jeton et vous invite à le faire signer.

Pour fournir le deuxième facteur, c'est-à-dire ce que vous avez, vous signez le jeton avec une clé privée provenant d'une paire de clés que vous avez déjà créée et associée à l'utilisateur HSM. Pour accéder au cluster, vous devez fournir le jeton signé à la CLI CloudHSM.

Pour plus d'informations sur la configuration de la MFA pour un utilisateur, veuillez consulter Configuration de la MFA pour la CLI CloudHSM

Authentification par quorum et MFA

Le cluster utilise la même clé pour l'authentification par quorum et pour l'authentification MFA. Cela signifie qu'un utilisateur dont la MFA est activée est effectivement enregistré pour le contrôle d'accès MofN ou Quroum. Pour utiliser correctement l'authentification MFA et le quorum pour le même utilisateur HSM, tenez compte des points suivants :

- Si vous utilisez l'authentification par quorum pour un utilisateur aujourd'hui, vous devez utiliser la même paire de clés que celle que vous avez créée pour l'utilisateur du quorum afin d'activer le MFA pour cet utilisateur.
- Si vous ajoutez l'exigence MFA pour un utilisateur non MFA qui n'est pas un utilisateur d'authentification par quorum, vous enregistrez cet utilisateur en tant qu'utilisateur enregistré Quroum (MofN) avec authentification MFA.
- Si vous supprimez l'exigence MFA ou modifiez le mot de passe d'un utilisateur MFA qui est également un utilisateur enregistré avec authentification par quorum, vous supprimerez également l'enregistrement de l'utilisateur en tant qu'utilisateur du quorum (MofN).
- Si vous supprimez l'exigence MFA ou modifiez le mot de passe d'un utilisateur MFA qui est également un utilisateur utilisant l'authentification par quorum, mais que vous souhaitez toujours que cet utilisateur participe à l'authentification par quorum, vous devez l'enregistrer à nouveau en tant qu'utilisateur du quorum (MofN).

Pour de plus amples informations sur l'authentification par quorum, veuillez consulter Gestion du quorum (M sur N).

Utilisation de la MFA pour les utilisateurs de HSM

Cette rubrique fournit des informations et des instructions relatives à l'utilisation de la CLI CloudHSM pour gérer l'authentification multifactorielle (MFA). Pour plus d'informations sur la CLI CloudHSM, veuillez consulter Interface de ligne de commande (CLI) CloudHSM.

Rubriques

- Exigences relatives à la paire de clés MFA
- Configuration de la MFA pour la CLI CloudHSM
- Créez des utilisateurs avec la MFA activée
- Connectez-vous aux utilisateurs dont la MFA est activée
- Rotation des touches pour les utilisateurs dont l'authentification MFA est activée
- Désenregistrer une clé publique MFA pour les utilisateurs administrateurs lorsque la clé publique MFA est enregistrée
- Référence du fichier de jetons

Pour plus d'informations sur le travail avec des utilisateurs HSM, veuillez consulter Interface de ligne de commande (CLI) CloudHSM

Exigences relatives à la paire de clés MFA

Pour activer la MFA pour un utilisateur HSM, vous pouvez créer une nouvelle paire de clés ou utiliser une clé existante répondant aux exigences suivantes :

- Type de clé : asymétrique
- Utilisation de clé : signature et vérification
- Spécification de clé : RSA_2048
- L'algorithme de signature inclut : sha256WithRSAEncryption

> **Note**
>
> Si vous utilisez l'authentification par quorum ou si vous envisagez d'utiliser l'authentification par quorum, veuillez consulter Authentification par quorum et MFA

Vous pouvez utiliser la CLI CloudHSM et la paire de clés pour créer un nouvel utilisateur administrateur avec la MFA activée.

Configuration de la MFA pour la CLI CloudHSM

Procédez comme suit pour configurer l'authentification MFA pour la CLI CloudHSM.

1. Pour configurer la MFA à l'aide de la stratégie de signature de jeton, vous devez d'abord générer une clé privée RSA de 2 048 bits et la clé publique associée.

```
$ 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. À l'aide de la CLI CloudHSM, connectez-vous à votre compte utilisateur.

```
$ 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. Ensuite, exécutez la commande pour modifier votre stratégie MFA. Vous devez fournir le paramètre `--token`. Ce paramètre indique un fichier dans lequel seront écrits des jetons non signés.

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

4. Vous avez maintenant un fichier contenant des jetons non signés qui doivent être signés : `unsigned-tokens.json`. Le nombre de jetons dans ce fichier dépend du nombre de HSM de votre cluster. Chaque jeton représente un HSM. Ce fichier est au format JSON et contient des jetons qui doivent être signés pour prouver que vous disposez d'une clé privée.

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

5. L'étape suivante consiste à signer ces jetons avec la clé privée créée à l'étape 1. Replacez les signatures dans le fichier. Tout d'abord, vous devez extraire et décoder les jetons codés en base64.

```
$ 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. Vous disposez désormais de jetons binaires que vous pouvez signer à l'aide de la clé privée RSA créée à l'étape 1.

```
$ 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. Maintenant, vous avez les signatures binaires des jetons. Vous devez les encoder en base64 et les replacer dans votre fichier de jetons.

```
$ 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. Enfin, vous pouvez copier-coller les valeurs base64 dans votre fichier de jetons :

```
{
  "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. Maintenant que votre fichier de jetons contient toutes les signatures requises, vous pouvez continuer. Entrez le nom du fichier contenant les jetons signés et appuyez sur la touche Entrée. Enfin, entrez le chemin de votre clé publique.

```
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"
  }
}
```

Vous avez maintenant configuré votre utilisateur avec la MFA.

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

```
    ],
    "cluster-coverage": "full"
},
```

Créez des utilisateurs avec la MFA activée

Procédez comme suit pour créer des utilisateurs avec la MFA activée.

1. Utilisez la CLI CloudHSM pour vous connecter au HSM en tant qu'administrateur.
2. Utilisez la commande user create pour créer l'utilisateur de votre choix. Suivez ensuite les étapes décrites dans Configuration de la MFA pour la CLI CloudHSM pour configurer la MFA pour l'utilisateur.

Connectez-vous aux utilisateurs dont la MFA est activée

Procédez comme suit pour connecter les utilisateurs dont la MFA est activée.

1. Utilisez la commande login mfa-token-sign de la CLI CloudHSM pour démarrer le processus de connexion avec MFA pour un utilisateur dont la MFA est activée.

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

2. Entrez votre mot de passe. Vous serez ensuite invité à saisir le chemin d'accès au fichier de jetons qui contient des paires de jetons non signés/signés, les jetons signés étant ceux générés à l'aide de votre clé privée.

```
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. Lorsque vous êtes invité à saisir le chemin du fichier de jeton signé, vous pouvez inspecter le fichier de jeton non signé dans un terminal séparé. Identifiez le fichier contenant des jetons non signés qui doivent être signés : `unsigned-tokens.json`. Le nombre de jetons dans ce fichier dépend du nombre de HSM de votre cluster. Chaque jeton représente un HSM. Ce fichier est au format JSON et contient des jetons qui doivent être signés pour prouver que vous disposez d'une clé privée.

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

4. Signez les jetons non signés avec la clé privée créée à l'étape 2. Vous devez d'abord extraire et décoder les jetons codés en base64.

```
$ 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. Vous avez maintenant des jetons binaires. Signez-les à l'aide de la clé privée RSA que vous avez créée précédemment à l'étape 1 de la configuration de la MFA.

```
$ 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. Vous avez maintenant les signatures binaires des jetons. Codez-les en base64 et replacez-les dans votre fichier de jetons.

```
$ 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. Enfin, copiez et collez les valeurs base64 dans votre fichier de jetons :

```
{
  "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. Maintenant que votre fichier de jetons contient toutes les signatures requises, vous pouvez continuer. Entrez le nom du fichier contenant les jetons signés et appuyez sur la touche Entrée. Vous devriez maintenant vous connecter avec succès.

```
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>"
  }
}
```

Rotation des touches pour les utilisateurs dont l'authentification MFA est activée

Procédez comme suit pour faire pivoter les clés pour les utilisateurs dont l'authentification MFA est activée.

<result>

Vous avez signé le fichier de jeton au format JSON généré avec votre clé privée et enregistré une nouvelle clé publique MFA.
</result>

1. Utilisez la CLI CloudHSM pour vous connecter au HSM en tant qu'administrateur ou en tant qu'utilisateur spécifique pour lequel la MFA est activée (voir Connexion des utilisateurs avec MFA activée pour plus de détails).
2. Ensuite, exécutez la commande pour modifier votre stratégie MFA. Vous devez fournir le paramètre `--token`. Ce paramètre indique un fichier dans lequel seront écrits des jetons non signés.

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

3. Identifiez le fichier contenant des jetons non signés qui doivent être signés : `unsigned-tokens.json`. Le nombre de jetons dans ce fichier dépend du nombre de HSM de votre cluster. Chaque jeton représente un HSM. Ce fichier est au format JSON et contient des jetons qui doivent être signés pour prouver que vous disposez d'une clé privée. Il s'agira de la nouvelle clé privée issue de la nouvelle paire de clés publique/privée RSA que vous souhaitez utiliser pour faire pivoter la clé publique actuellement enregistrée.

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

4. Signez ces jetons avec la clé privée que vous avez créée précédemment lors de l'installation. Nous devons d'abord extraire et décoder les jetons codés en base64.

```
$ 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. Vous avez maintenant des jetons binaires. Signez-les à l'aide de la clé privée RSA que vous avez créée précédemment lors de l'installation.

```
$ 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. Vous avez maintenant les signatures binaires des jetons. Codez-les en base64 et replacez-les dans votre fichier de jetons.

```
$ 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. Enfin, copiez et collez les valeurs base64 dans votre fichier de jetons :

```
{
  "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. Maintenant que votre fichier de jetons contient toutes les signatures requises, vous pouvez continuer. Entrez le nom du fichier contenant les jetons signés et appuyez sur la touche Entrée. Enfin, saisissez le chemin de votre nouvelle clé publique. Vous verrez maintenant ce qui suit dans le cadre de la sortie de user list.

```
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"
  }
}
```

Nous avons maintenant configuré notre utilisateur avec la MFA.

```
{
```

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

Désenregistrer une clé publique MFA pour les utilisateurs administrateurs lorsque la clé publique MFA est enregistrée

Procédez comme suit pour désenregistrer une clé publique MFA pour les utilisateurs administrateurs lorsque la clé publique MFA est enregistrée.

1. Utilisez la CLI CloudHSM pour vous connecter au HSM en tant qu'administrateur avec la MFA activée.
2. Utilisez la commande user change-mfa token-sign pour supprimer la MFA pour un utilisateur.

```
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"
  }
}
```

Référence du fichier de jetons

Le fichier de jetons généré lors de l'enregistrement d'une clé publique MFA ou lors de la tentative de connexion à l'aide de l'authentification MFA comprend les éléments suivants :

- Jetons : un tableau de paires de jetons non signés/signés codés en base64 sous la forme de littéraux d'objets JSON.
- Non signé : jeton codé en base64 et haché SHA256.
- Signé : jeton signé codé en base64 (signature) du jeton non signé, à l'aide de la clé privée RSA 2 048 bits.


```
{
  "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
    }
  ]
}
```

Utilisation de la CLI CloudHSM pour gérer l'authentification par quorum (contrôle d'accès M sur N)

Les HSM de votre AWS CloudHSM cluster prennent en charge l'authentification par quorum, également connue sous le nom de contrôle d'accès M of N. Avec l'authentification par quorum, aucun utilisateur individuel sur le HSM ne peut effectuer d'opérations contrôlées par quorum sur le HSM. À la place, un nombre minimal d'utilisateurs HSM (au moins 2) doivent coopérer pour effectuer ces opérations. Avec l'authentification par quorum, vous pouvez ajouter une couche de protection supplémentaire en demandant l'approbation de plusieurs utilisateurs HSM.

L'authentification par quorum peut contrôler les opérations suivantes :

- La gestion des utilisateurs HSM par l'administrateur - La création et la suppression d'utilisateurs HSM, et la modification du mot de passe d'un autre utilisateur HSM. Pour plus d'informations, consultez Utilisation de l'authentification par quorum pour les administrateurs.

Les rubriques suivantes fournissent plus d'informations sur l'authentification par quorum dans AWS CloudHSM.

Rubriques

Présentation de l'authentification par quorum avec stratégie de signature par jeton

Les étapes suivantes résument les processus d'authentification par quorum. Pour connaître les étapes et les outils spécifiques, consultez Utilisation de l'authentification par quorum pour les administrateurs.

1. Chaque utilisateur HSM crée une clé asymétrique pour la signature. Les utilisateurs s'en chargent en dehors du HSM, en veillant à protéger la clé de manière appropriée.

2. Chaque utilisateur HSM se connecte au HSM et enregistre la partie publique de sa clé de signature (la clé publique) avec le HSM.
3. Lorsqu'un utilisateur HSM veut effectuer une opération contrôlée par quorum, il se connecte au HSM et obtient un jeton de quorum.
4. L'utilisateur HSM donne le jeton de quorum à un ou plusieurs autres utilisateurs HSM et demande leur approbation.
5. Les autres utilisateurs HSM approuvent en utilisant leurs clés pour signer de façon chiffrée le jeton de quorum. Cela se produit en dehors du HSM.
6. Lorsque l'utilisateur du HSM dispose du nombre d'approbations requis, il se connecte au HSM et exécute l'opération de contrôle du quorum avec l'argument --approval, en fournissant le fichier de jeton de quorum signé, qui contient toutes les approbations nécessaires (signatures).
7. Le HSM utilise les clés publiques enregistrées de chaque signataire pour vérifier les signatures. Si les signatures sont valides, le HSM approuve le jeton et l'opération contrôlée par le quorum est exécutée.

Détails supplémentaires concernant l'authentification par quorum

Notez les informations supplémentaires suivantes sur l'utilisation de l'authentification par quorum dans AWS CloudHSM.

- Un utilisateur HSM peut signer son propre jeton de quorum, c'est-à-dire que l'utilisateur demandeur peut fournir l'une des approbations requises pour l'authentification par quorum.
- Vous choisissez le nombre minimal d'approbateurs de quorum pour les opérations contrôlées par quorum. Le plus petit nombre que vous pouvez choisir est deux (2), et le plus grand nombre que vous pouvez choisir est huit (8).
- Le HSM peut stocker jusqu'à 1 024 jetons de quorum. Si le HSM possède déjà 1 024 jetons lorsque vous essayez d'en créer un nouveau, le HSM efface l'un des jetons ayant expiré. Par défaut, les jetons expirent dix minutes après leur création.
- Si l'authentification MFA est activée, le cluster utilise la même clé pour l'authentification par quorum et pour l'authentification multifactorielle (MFA). Pour plus d'informations sur l'utilisation de l'authentification par quorum et de l'authentification 2FA, consultez la section Utilisation de la CLI CloudHSM pour gérer l'authentification MFA.
- Chaque HSM ne peut contenir qu'un seul jeton par service à la fois.

Noms et types de services prenant en charge l'authentification par quorum

Services d'administrateur : l'authentification par quorum est utilisée pour les services dotés de privilèges d'administrateur tels que la création d'utilisateurs, la suppression d'utilisateurs, la modification des mots de passe des utilisateurs, la définition des valeurs de quorum et la désactivation des fonctionnalités de quorum et de MFA.

Chaque type de service est ensuite décomposé en un nom de service éligible, qui contient un ensemble spécifique d'opérations de service prises en charge par le quorum qui peuvent être effectuées.

Nom du service	Type de service	Opérations de service
utilisateur	Administrateur	• créer un utilisateur • supprimer un utilisateur • modifier un mot de passe utilisateur • modifier la MFA d'un utilisateur
quorum	Administrateur	• signe du jeton quorum set-quorum-value

Utilisation de l'authentification par quorum pour les administrateurs : première configuration

Les rubriques suivantes décrivent les étapes que vous devez suivre pour configurer votre module de sécurité matériel (HSM) afin que les administrateurs puissent utiliser l'authentification par quorum. Vous devez suivre ces étapes une seule fois lorsque vous configurez l'authentification par quorum pour les administrateurs. Une fois que vous avez terminé ces étapes, consultez Utilisation de l'authentification par quorum pour les administrateurs.

Rubriques

- Prérequis
- Création et enregistrement d'une clé pour la signature
- Définition de la valeur minimale de quorum sur le HSM

Prérequis

Pour comprendre cet exemple, vous devez bien connaître la CLI CloudHSM. Dans cet exemple, le AWS CloudHSM cluster possède deux HSM, chacun ayant les mêmes administrateurs, comme indiqué dans le résultat suivant de la user list commande. Pour plus d'informations sur la création d'utilisateurs, consultez Utilisation de la CLI CloudHSM.

```
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"
      }
    ]
  }
}
```

Création et enregistrement d'une clé pour la signature

Pour utiliser l'authentification par quorum, chaque administrateur doit effectuer toutes les étapes suivantes :

Rubriques

- Créer une paire de clés RSA
- Créez et signez un jeton d'enregistrement
- Enregistrez la clé publique avec le HSM

Créer une paire de clés RSA

Il existe de nombreuses manières différentes de créer et de protéger une paire de clés. Les exemples suivants montrent comment procéder avec OpenSSL.

Example — Créer une clé privée à l'aide d'OpenSSL

L'exemple suivant montre comment utiliser OpenSSL pour créer une clé RSA de 2048 bits, protégée par une phrase secrète. Pour utiliser cet exemple, remplacez *<admin.key>* par le nom du fichier dans lequel vous souhaitez stocker la clé.

```
$ 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:
```

Générez ensuite la clé publique à l'aide de la clé privée que vous venez de créer.

Example — Crée une clé publique avec OpenSSL

L'exemple suivant montre comment utiliser OpenSSL pour créer une clé publique à partir de la clé privée que vous venez de créer.

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

Créez et signez un jeton d'enregistrement

Vous créez un jeton et vous le signez avec la clé privée que vous venez de générer à l'étape précédente.

Example — Créez un jeton d'enregistrement

1. Utilisez la commande suivante pour démarrer la CLI CloudHSM :

   Linux

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

   Windows

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

2. Créez un jeton d'enregistrement en exécutant la commande quorum token-sign generate :

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

3. La commande quorum token-sign generate génère un jeton d'enregistrement sur le chemin de fichier spécifié. Inspectez le fichier du jeton :

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

Le chemin d'accès au fichier se compose des éléments suivants :

- approval_data : jeton de données aléatoire codé en base64 dont les données brutes ne dépassent pas le maximum de 245 octets.
- non signé : un jeton codé en base64 et haché SHA256 de approval_data.
- signé : jeton signé codé en base64 (signature) du jeton non signé, utilisant la clé privée RSA 2048 bits précédemment générée avec OpenSSL.

Vous signez le jeton non signé avec la clé privée pour démontrer que vous avez accès à la clé privée. Vous aurez besoin du fichier de jeton d'enregistrement entièrement rempli d'une signature et de la clé publique pour enregistrer l'administrateur en tant qu'utilisateur du quorum auprès du AWS CloudHSM cluster.

Example — Signez le jeton d'enregistrement non signé

1. Décodez le jeton non signé codé en base64 et placez-le dans un fichier binaire :

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

2. Utilisez OpenSSL et la clé privée pour signer le jeton d'enregistrement non signé désormais binaire et créer un fichier de signature binaire :

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

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

3. Encodez la signature binaire en base64 :

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

4. Copiez et collez la signature codée en base64 dans le fichier de jeton :

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

Enregistrez la clé publique avec le HSM

Après avoir créé une clé, l'administrateur doit enregistrer la clé publique auprès du AWS CloudHSM cluster.

Pour enregistrer une clé publique avec le HSM

1. Utilisez la commande suivante pour démarrer la CLI CloudHSM.

   Linux

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

   Windows

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

2. À l'aide de la CLI CloudHSM, connectez-vous en tant qu'administrateur.

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

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

3. Utilisez la commande user change-quorum token-sign register pour enregistrer la clé publique. Pour plus d'informations, consultez l'exemple suivant ou utilisez la commande help user change-quorum token-sign register.

Example — Enregistrez une clé publique auprès AWS CloudHSM du cluster

L'exemple suivant montre comment utiliser la commande user change-quorum token-sign register de la CLI CloudHSM pour enregistrer la clé publique d'un administrateur avec le HSM. Pour utiliser cette commande, l'administrateur doit être connecté au HSM. Remplacez les valeurs suivantes par les vôtres :

```
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 : chemin d'accès au fichier PEM à clé publique
> Obligatoire : oui
> /path/tokenfile : le chemin du fichier avec le jeton signé par la clé privée de l'utilisateur
> Obligatoire : oui

Une fois que tous les administrateurs ont enregistré leurs clés publiques, le résultat de la commande user list l'indique dans le champ de quorum, indiquant la stratégie de quorum activée utilisée, comme indiqué ci-dessous :

```
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"
      }
    ]
  }
}
```

Définition de la valeur minimale de quorum sur le HSM

Pour utiliser l'authentification par quorum, un administrateur doit se connecter au HSM, puis définir la valeur minimale de quorum. Il s'agit du nombre minimal d'approbations d'administrateur qui sont nécessaires pour effectuer des opérations de gestion des utilisateurs HSM. Tout administrateur sur le HSM peut définir la valeur minimale de quorum, y compris les administrateurs qui n'ont pas enregistré de clé pour la signature. Vous pouvez modifier la valeur minimale de quorum à tout moment ; pour plus d'informations, veuillez consulter Modifier la valeur minimale..

Pour définir la valeur minimale de quorum sur le HSM

1. Utilisez la commande suivante pour démarrer la CLI CloudHSM.

Linux

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

Windows

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

2. À l'aide de la CLI CloudHSM, connectez-vous en tant qu'administrateur.

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

3. Utilisez la commande signe du jeton quorum set-quorum-value pour définir la valeur minimale de quorum. Pour plus d'informations, consultez l'exemple suivant ou utilisez la commande help quorum token-sign set-quorum-value.

Example - Définition de la valeur minimale de quorum sur le HSM

Cet exemple utilise une valeur minimale de quorum de deux (2). Vous pouvez choisir n'importe quelle valeur comprise entre deux (2) et huit (8), jusqu'au nombre total d'administrateurs sur le HSM. Dans cet exemple, le HSM compte quatre (4) administrateurs, la valeur maximale possible est donc de quatre (4).

Pour utiliser l'exemple de commande suivant, remplacez le nombre final (*<2>*) par la valeur minimale de quorum que vous préférez.

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

Dans cet exemple, le service identifie le service HSM dont vous définissez la valeur minimale de quorum. La commande signe du jeton quorum list-quorum-values répertorie les types, les noms et les descriptions des services HSM inclus dans le service.

Services d'administrateur : l'authentification par quorum est utilisée pour les services dotés de privilèges d'administrateur tels que la création d'utilisateurs, la suppression d'utilisateurs, la modification des mots de passe des utilisateurs, la définition des valeurs de quorum et la désactivation des fonctionnalités de quorum et de MFA.

Chaque type de service est ensuite décomposé en un nom de service éligible, qui contient un ensemble spécifique d'opérations de service prises en charge par le quorum qui peuvent être effectuées.

Nom du service	Type de service	Opérations de service
utilisateur	Administrateur	• créer un utilisateur • supprimer un utilisateur • modifier un mot de passe utilisateur • modifier la MFA d'un utilisateur
quorum	Administrateur	• signe du jeton quorum set-quorum-value

Utilisez la commande quorum token-sign list-quorum-values pour obtenir la valeur minimale du quorum pour un service.

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

La sortie de la commande quorum token-sign list-quorum-values précédente indique que la valeur minimale de quorum pour le service des utilisateurs HSM, responsable des opérations de gestion des utilisateurs, est désormais égale à deux (2). Une fois que vous avez terminé ces étapes, consultez Utilisation du quorum (M sur N).

Utilisation de l'authentification par quorum pour les administrateurs

Un administrateur du HSM peut configurer l'authentification par quorum pour les opérations suivantes dans le AWS CloudHSM cluster :

- user create
- Supprimer un utilisateur
- user change-password
- user change-mfa

Une fois le AWS CloudHSM cluster configuré pour l'authentification par quorum, les administrateurs ne peuvent pas effectuer eux-mêmes les opérations de gestion des utilisateurs HSM. L'exemple suivant montre la sortie lorsqu'un administrateur tente de créer un nouvel utilisateur sur le HSM. La commande échoue avec une erreur indiquant que l'authentification par quorum est requise.

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

Pour effectuer une opération de gestion d'utilisateur HSM, un administrateur doit exécuter les tâches suivantes :

Rubriques

- Obtention d'un jeton de quorum
- Obtention des signatures des administrateurs en charge de l'approbation
- Approuver le jeton sur le AWS CloudHSM cluster et exécuter une opération de gestion des utilisateurs

Obtention d'un jeton de quorum

Tout d'abord, l'administrateur doit utiliser la CLI CloudHSM pour demander un jeton de quorum.

Pour obtenir un jeton de quorum

1. Utilisez la commande suivante pour démarrer la CLI CloudHSM.

   Linux

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

   Windows

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

2. Utilisez la commande login et connectez-vous au cluster en tant qu'administrateur.

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

3. Utilisez la commande quorum token-sign generate pour générer un jeton de quorum. Pour plus d'informations, consultez l'exemple suivant ou utilisez la commande help quorum token-sign generate.

Example — Génère un jeton de quorum

Cet exemple obtient un jeton de quorum pour l'administrateur avec le nom d'utilisateur `admin` et enregistre le jeton dans un fichier nommé `admin.token`. Pour utiliser l'exemple de commande, remplacez ces valeurs par les vôtres :

- *`<admin>`* — Le nom de l'administrateur qui reçoit le jeton. Il doit s'agir du même administrateur que celui qui est connecté au HSM et qui exécute cette commande.
- *`<admin.token>`* – Le nom du fichier à utiliser pour stocker le jeton de quorum.

Dans la commande suivante, `user` identifie le nom de service pour lequel vous pouvez utiliser le jeton que vous générez. Dans ce cas, le jeton concerne les opérations de gestion des utilisateurs HSM (`user` service). .

```
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"
  }
}
```

La commande quorum token-sign generate génère un jeton de quorum de service utilisateur sur le chemin de fichier spécifié. Le fichier du jeton peut être inspecté :

```
$cat </path/admin.token>
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW/
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAABAAMAAAAUA
+b23gAAAAAAAAAA",
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": []
}
```

Le chemin d'accès au fichier se compose des éléments suivants :

- approval_data : jeton de données brutes codé en base64 généré par le HSM.
- jeton : jeton codé en base64 et haché SHA-256 du approval_data
- signatures : tableau de jetons signés codés en base64 (signatures) du jeton non signé, où chaque signature d'un approbateur prend la forme d'un objet JSON littéral :

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

Chaque signature est créée à partir du résultat d'un approbateur utilisant sa clé privée RSA 2048 bits correspondante dont la clé publique a été enregistrée auprès du HSM.

L'existence du jeton de quorum du service utilisateur généré peut être confirmée sur le cluster CloudHSM en exécutant la commande quorum token-sign list :

```
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"
      }
    ]
  }
}
```

L'heure `token-timeout-seconds` indique le délai d'expiration en secondes pour qu'un jeton généré soit approuvé avant son expiration.

Obtention des signatures des administrateurs en charge de l'approbation

Un administrateur qui possède un jeton de quorum doit obtenir le jeton approuvé par d'autres administrateurs. Pour donner leur approbation, les autres administrateurs utilisent leur clé de signature pour signer le jeton de façon cryptographique. Ils le font en dehors du module HSM.

Il existe de nombreuses façons différentes de signer le jeton. L'exemple suivant montre comment procéder avec OpenSSL. Pour utiliser un autre outil de signature, assurez-vous que l'outil utilise la clé privée de l'administrateur (clé de signature) pour signer un hachage SHA-256 du jeton.

Example - Obtention des signatures des administrateurs en charge de l'approbation

Dans cet exemple, l'administrateur qui possède le jeton (`admin`) a besoin d'au moins deux (2) approbations. L'exemple de commandes suivant montrent comment deux (2) administrateurs peuvent utiliser OpenSSL pour signer le jeton de manière cryptographique.

1. Décodez le jeton non signé codé en base64 et placez-le dans un fichier binaire :

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

2. Utilisez OpenSSL et la clé privée correspondante de l'approbateur (`admin3`) pour signer le jeton non signé du quorum désormais binaire pour le service utilisateur et créer un fichier de signature binaire :

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

3. Encodez la signature binaire en base64 :

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

4. Enfin, copiez et collez la signature codée en base64 dans le fichier de jeton, en suivant le format littéral d'objet JSON spécifié précédemment pour la signature de l'approbateur :

```
{
  "version": "2.0",
  "approval_data": "AAEAAwAAABgAAAAAAAAAAJ9eFkfcP3mNzJAlfK
+OWbNhZG1pbgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABj5vbeAAAAAAAAAAAAAQADAAAAFQAAAAAAAAAW
v5Euk83amq1fij0zyvD2FkbWluAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAGPm9t4AAAAAAAAAAAABAAMAA
+b23gAAAAAAAAAA",
  "token": "0l2LZkmAHZyAc1hPhyckOoVW33aGrgG77qmDHWQ3CJ8=",
  "signatures": [
    {
      "username": "admin2",
      "signature": "O6qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+OsDtvYujhEU4aOFVLcrUFmyB/CX9OQmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1nOXwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWWdtDgfT7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
```

```
ssktwyruGFLpXs1n0tJOEglGhx2qbYTs+omKWZdORl5WIWEXW3IXw/
Dg5vVObrNpvG0eZKO8nSMc27+cyPySc+ZbNw=="
    },
    {
      "username": "admin3",
      "signature": "O6qx7/mUaVkYYVr1PW7l8JJko+Kh3e8zBIqdk3tAiNy+1rW
+OsDtvYujhEU4aOFVLcrUFmyB/CX9OQmgJLgx/pyK+ZPEH+GoJGqk9YZ7X1nOXwZRP9g7hKV
+7XCtg9TuDFtHYWDpBfz2jWiu2fXfX4/
jTs4f2xIfFPIDKcSP8fhxjQ63xEcCf1jzGha6rDQMu4xUWWdtDgfT7um7EJ9dXNoHqLB7cTzphaubNaEFbFPXQ1siGm
ssktwyruGFLpXs1n0tJOEglGhx2qbYTs+omKWZdORl5WIWEXW3IXw/
Dg5vVObrNpvG0eZKO8nSMc27+cyPySc+ZbNw=="
    }
  ]
}
```


Approuver le jeton sur le AWS CloudHSM cluster et exécuter une opération de gestion des utilisateurs

Une fois qu'un administrateur a obtenu les approbations/signatures nécessaires, comme indiqué dans la section précédente, il peut fournir ce jeton au cluster AWS CloudHSM avec l'une des opérations de gestion des utilisateurs suivantes :

- créer
- supprimer
- change-password
- user change-mfa

Pour plus d'informations sur l'utilisation de ces commandes, consultez Utilisation de la CLI CloudHSM.

Au cours de la transaction, le jeton sera approuvé au sein du AWS CloudHSM cluster et exécutera l'opération de gestion des utilisateurs demandée. Le succès de l'opération de gestion des utilisateurs dépend à la fois d'un jeton de quorum approuvé valide et d'une opération de gestion des utilisateurs valide.

L'administrateur peut utiliser le jeton pour une seule opération. Lorsque cette opération réussit, le jeton n'est plus valide. Pour effectuer une autre opération de gestion des utilisateurs HSM, l'administrateur doit répéter le processus décrit ci-dessus. C'est-à-dire que l'administrateur doit

générer un nouveau jeton de quorum, obtenir de nouvelles signatures des approbateurs et approuver et consommer le nouveau jeton sur le HSM avec l'opération de gestion des utilisateurs demandée.

> **Note**
>
> Le jeton de quorum n'est valide que tant que votre session de connexion en cours est ouverte. Si vous vous déconnectez de la CLI CloudHSM ou si le réseau se déconnecte, le jeton n'est plus valide. De même, un jeton autorisé ne peut être utilisé que dans la CLI CloudHSM. Il ne peut pas être utilisé pour s'authentifier dans une autre application.

Example Création d'un nouvel utilisateur en tant qu'administrateur

Dans l'exemple suivant, un administrateur connecté crée un utilisateur sur le HSM.

```
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"
  }
}
```

L'administrateur saisit ensuite la commande user list pour confirmer la création du nouvel utilisateur :

```
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"
      }
    ]
  }
}
```

Si l'administrateur essaie d'effectuer une autre opération de gestion de l'utilisateur HSM, l'opération échoue avec une erreur d'authentification par quorum :

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

Comme indiqué ci-dessous, la commande quorum token-sign list indique que l'administrateur n'a aucun jeton approuvé. Pour effectuer une autre opération de gestion des utilisateurs HSM, l'administrateur doit générer un nouveau jeton de quorum, obtenir de nouvelles signatures de la part des approbateurs et exécuter l'opération de gestion des utilisateurs souhaitée avec l'argument --approval pour fournir le jeton de quorum à approuver et à utiliser pendant l'exécution de l'opération de gestion des utilisateurs.

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

```
}
```

Modifier la valeur minimale du quorum pour les administrateurs

Après avoir défini la valeur minimale du quorum de sorte que les administrateurs puissent utiliser l'authentification par quorum, vous pouvez souhaiter modifier la valeur minimale du quorum. Le HSM vous permet de modifier la valeur minimale du quorum uniquement lorsque le nombre d'approbateurs est égal ou supérieur à la valeur minimale du quorum. Par exemple, si la valeur minimale du quorum est deux (2), au moins deux (2) administrateurs doivent donner leur approbation pour une modification de la valeur minimale du quorum.

> **Note**
>
> La valeur de quorum du service utilisateur doit toujours être inférieure à la valeur de quorum du service de quorum. Pour plus d'informations sur les noms de service, tels que le service quorum et le service utilisateur, veuillez consulter Noms et types de services prenant en charge l'authentification par quorum.

Pour obtenir l'approbation du quorum en vue de modifier la valeur minimale du quorum, vous avez besoin d'un jeton de quorum pour le quorum service utilisant la commande quorum token-sign set-quorum-value. Pour générer un jeton de quorum pour le quorum service utilisant la commande quorum token-sign set-quorum-value, le service de quorum doit être supérieur à un (1). Cela signifie que, avant de pouvoir modifier la valeur minimale du quorum pour le service d'utilisateur, vous devrez peut-être modifier la valeur minimale du quorum pour service de quorum.

Pour modifier la valeur minimale du quorum pour les administrateurs

1. Utilisez la commande suivante pour démarrer le mode interactif de la CLI CloudHSM.

   Linux

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

   Windows

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

2. Utilisez la commande login et connectez-vous au cluster en tant qu'administrateur.

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

3. Utilisez la commande quorum token-sign list-quorum-values pour obtenir la valeur minimale du quorum pour tous les noms de service. Pour de plus amples informations, consultez l'exemple ci-dessous.
4. Si la valeur minimale du quorum pour le service de quorum est inférieure à celle pour le service d'utilisateur, utilisez la commande quorum token-sign set-quorum-value pour modifier la valeur spécifiée pour le service de quorum. Changez la valeur pour le service de quorum sur un (1) qui est une valeur égale ou supérieure à celle utilisée pour le service d'utilisateur. Pour plus d'informations, consultez l'exemple suivant.
5. Générez un jeton de quorum, en veillant à spécifier le service de quorum en tant que service pour lequel vous pouvez utiliser le jeton.
6. Obtenir des approbations (signatures) de la part d'autres administrateurs.
7. Approuvez le jeton sur le AWS CloudHSM cluster et exécutez une opération de gestion des utilisateurs. .
8. Utilisez la commande quorum token-sign set-quorum-value pour obtenir la valeur minimale du quorum pour le service d'utilisateur.

Example - Obtenir des valeurs minimale de quorum et modifier la valeur pour le service de quorum

L'exemple de commande suivant montre que la valeur minimale du quorum pour le service d'utilisateur est actuellement deux (2).

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

Pour modifier la valeur minimale du quorum pour le service de quorum, utilisez la commande quorum token-sign set-quorum-value pour définir une valeur égale ou supérieure à celle utilisée pour le service d'utilisateur. L'exemple suivant définit la valeur minimale du quorum pour le service de quorum à deux (2), valeur identique à celle définie pour le service d'utilisateur.

```
aws-cloudhsm > quorum token-sign set-quorum-value --service quorum --value 2{
```

```
  "error_code": 0,
  "data": "Set quorum value successful"
}
```

Les commandes suivantes montrent que la valeur minimale du quorum est désormais deux (2) pour le service d'utilisateur et le service de quorum.

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

Gestion des utilisateurs HSM avec l'Utilitaire de gestion CloudHSM (CMU)

Dans AWS CloudHSM, vous devez utiliser les outils de ligne de commande de la CLI CloudHSM ou de l'utilitaire de gestion CloudHSM (CMU) pour créer et gérer les utilisateurs sur votre HSM. La CLI CloudHSM est conçue pour être utilisée avec le dernier SDK, tandis que le CMU est conçu pour être utilisé avec les SDK précédents.

Rubriques

- Comprendre les utilisateurs HSM
- Tableau Autorisations des utilisateurs HSM
- Utilisation de l'Utilitaire de gestion CloudHSM (CMU) pour gérer les utilisateurs
- Utilisation de l'Utilitaire de gestion CloudHSM (CMU) pour gérer l'authentification à deux facteurs (2FA) pour les responsables du chiffrement
- Utilisation de l'Utilitaire de gestion CloudHSM (CMU) pour gérer l'authentification par quorum (contrôle d'accès M sur N)

Comprendre les utilisateurs HSM

La plupart des opérations que vous effectuez sur le HSM nécessitent les informations d'identification d'un utilisateur HSM. Le HSM authentifie chaque utilisateur HSM et chaque utilisateur HSM possède un type qui détermine les opérations que vous pouvez effectuer sur le HSM en tant qu'utilisateur.

> **Note**
>
> Les utilisateurs HSM sont distincts des utilisateurs IAM. Les utilisateurs IAM qui disposent des informations d'identification correctes peuvent créer des HSM en interagissant avec les ressources via l'API AWS. Une fois le HSM créé, vous devez utiliser les informations d'identification de l'utilisateur HSM pour authentifier les opérations sur le HSM.

Types d'utilisateur

- Responsable du pré-chiffrement (PRECO)
- Responsable de chiffrement (CO)
- Utilisateur de chiffrement (CU)
- utilisateur de l'appareil (AU)

Responsable du pré-chiffrement (PRECO)

Dans l'Utilitaire de gestion du cloud (CMU) et dans l'Utilitaire de gestion des clés (KMU), le PRECO est un utilisateur temporaire qui n'existe que sur le premier HSM d'un cluster AWS CloudHSM . Le premier HSM d'un nouveau cluster contient un utilisateur PRECO indiquant que ce cluster n'a jamais été activé. Pour activer un cluster, vous devez exécuter le cloudhsm-cli et exécuter la commande cluster activate. Connectez-vous au HSM et modifiez le mot de passe du PRECO. Lorsque vous modifiez le mot de passe, cet utilisateur devient un responsable de chiffrement (CO).

Responsable de chiffrement (CO)

Dans l'Utilitaire de gestion du cloud (CMU) et dans l'Utilitaire de gestion des clés (KMU), un responsable de chiffrement (CO) peut effectuer des opérations de gestion des utilisateurs. Par exemple, il peut créer et supprimer des utilisateurs, et modifier les mots de passe des utilisateurs. Pour de plus amples informations sur les utilisateurs CO, veuillez consulter Tableau Autorisations des utilisateurs HSM. Lorsque vous activez un nouveau cluster, l'utilisateur passe de Responsable du pré-chiffrement (PRECO) à responsable de chiffrement (CO).

Utilisateur de chiffrement (CU)

Un utilisateur de chiffrement (CU) peut effectuer les opérations de chiffrement et de gestion des clés suivantes.

- Gestion des clés - Créer, supprimer, partager, importer et exporter des clés de chiffrement.

- Opérations de chiffrement - Utiliser les clés de chiffrement pour le chiffrement, le déchiffrement, la signature, la vérification, et plus encore.

Pour plus d'informations, consultez le Tableau Autorisations des utilisateurs HSM.

utilisateur de l'appareil (AU)

L'utilisateur de l'appliance (AU) peut effectuer des opérations de clonage et de synchronisation sur les HSM de votre cluster. AWS CloudHSM utilise l'AU pour synchroniser les HSM d'un AWS CloudHSM cluster. L'AU existe sur tous les HSM fournis par AWS CloudHSM et dispose d'autorisations limitées. Pour plus d'informations, consultez le Tableau Autorisations des utilisateurs HSM.

AWS ne peut effectuer aucune opération sur vos HSM. AWS ne peut pas afficher ou modifier vos utilisateurs ou vos clés et ne peut effectuer aucune opération cryptographique à l'aide de ces clés.

Tableau Autorisations des utilisateurs HSM

Le tableau suivant répertorie les opérations HSM triées selon le type d'utilisateur ou de session HSM qui peut effectuer l'opération.

	Responsable de chiffrement (CO)	Utilisateur de chiffrement (CU)	utilisateur de l'appareil (AU)	Session non authentifiée
Obtention d'informations de base sur le cluster[1]	 Oui	 Oui	 Oui	 Oui
Changement de son mot de passe	 Oui	 Oui	 Oui	Ne s'applique pas
Changement du mot de passe d'un utilisateur	 Oui	 Non	 Non	 Non

	Responsable de chiffrement (CO)	Utilisateur de chiffrement (CU)	utilisateur de l'appareil (AU)	Session non authentifiée
Ajout et suppression d'utilisateurs	 Oui	 Non	 Non	 Non
Obtention du statut de la synchronisation[2]	 Oui	 Oui	 Oui	 Non
Extraction et insertion d'objets masqués[3]	 Oui	 Oui	 Oui	 Non
Fonctions de gestion des clés[4]	 Non	 Oui	 Non	 Non
Chiffrement et déchiffrement	 Non	 Oui	 Non	 Non
Connexion et vérification	 Non	 Oui	 Non	 Non
Génération de synthèses et de HMAC	 Non	 Oui	 Non	 Non

- [1] Les informations de base sur le cluster comprennent le nombre de HSM dans le cluster ainsi que l'adresse IP, le modèle, le numéro de série, l'ID d'appareil, l'ID de microprogramme, etc. de chaque HSM.
- [2] L'utilisateur peut obtenir un ensemble de résumés (hachages) qui correspondent aux clés du HSM. Une application peut comparer ces ensembles pour comprendre le statut de la synchronisation des HSM dans un cluster.
- [3] Les objets masqués sont des clés qui sont chiffrées avant de quitter le HSM. Elles ne peuvent pas être déchiffrées en dehors du HSM. Elles ne sont déchiffrées que lorsqu'elles sont insérées dans un HSM qui se trouve dans le même cluster que le HSM duquel elles ont été extraites. Une application peut extraire et insérer des objets masqués afin de synchroniser les HSM dans un cluster.
- [4] Les fonctions de gestion des clés incluent la création, la suppression, l'encapsulage, le désencapsulage et la modification des attributs de clés.

Utilisation de l'Utilitaire de gestion CloudHSM (CMU) pour gérer les utilisateurs

Cette rubrique fournit des step-by-step instructions sur la gestion des utilisateurs du module de sécurité matérielle (HSM) à l'aide de l'utilitaire de gestion CloudHSM (CMU), un outil de ligne de commande fourni avec le SDK client. Pour plus d'informations sur le CMU ou les utilisateurs HSM, veuillez consulter Utilitaire de gestion CloudHSM et Comprendre les utilisateurs HSM.

Sections

- Comprendre la gestion des utilisateurs HSM avec le CMU
- Télécharger l'Utilitaire de gestion CloudHSM
- Comment gérer les utilisateurs du HSM avec le CMU

Comprendre la gestion des utilisateurs HSM avec le CMU

Pour gérer les utilisateurs HSM, vous devez vous connecter au HSM avec le nom d'utilisateur et le mot de passe d'un responsable de chiffrement (CO). Seul les CO peuvent gérer d'autres utilisateurs. Le HSM contient un responsable du chiffrement par défaut (CO) appelé admin Vous définissez le mot de passe pour admin lorsque vous avez activé le cluster.

Pour utiliser le CMU, vous devez utiliser l'outil de configuration pour mettre à jour la configuration locale. Le CMU crée sa propre connexion au cluster et cette connexion n'est pas consciente du cluster. Pour suivre les informations du cluster, le CMU gère un fichier de configuration local. Cela

signifie que chaque fois que vous utilisez le CMU, vous devez d'abord mettre à jour le fichier de configuration en exécutant l'outil de ligne de commande configure avec le paramètre `--cmu`. Si vous utilisez le SDK client 3.2.1 ou une version antérieure, vous devez utiliser un paramètre différent de `--cmu`. Pour plus d'informations, consultez the section called "Utilisation du CMU avec le SDK client 3.2.1 et versions antérieures".

Le paramètre `--cmu` vous oblige à ajouter l'adresse IP d'un HSM dans votre cluster. Si vous avez plusieurs HSM, vous pouvez utiliser n'importe quelle adresse IP. Cela garantit que le CMU peut propager les modifications que vous apportez sur l'ensemble du cluster. N'oubliez pas que le CMU utilise son fichier local pour suivre les informations du cluster. Si le cluster a changé depuis la dernière fois que vous avez utilisé le CMU depuis un hôte particulier, vous devez ajouter ces modifications au fichier de configuration local stocké sur cet hôte. N'ajoutez ni ne supprimez jamais de HSM lorsque vous utilisez la CMU.

Pour obtenir une adresse IP pour un HSM (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Pour changer de région AWS, utilisez le Sélecteur de région dans l'angle supérieur droit de la page.
3. Pour ouvrir la page détaillée du cluster, choisissez l'ID du cluster dans le tableau des clusters.
4. Pour obtenir l'adresse IP, dans l'onglet HSM, choisissez l'une des adresses IP répertoriées sous Adresse IP ENI.

Pour obtenir une adresse IP pour un HSM (CLI)

- Obtenez l'adresse IP d'un HSM à l'aide de la describe-clusters commande de la CLI. Dans la sortie de la commande, l'adresse IP des HSM sont les valeurs de `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


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

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

Utilisation du CMU avec le SDK client 3.2.1 et versions antérieures

Avec le SDK client 3.3.0, la prise en charge du `--cmu` paramètre AWS CloudHSM a été ajoutée, ce qui simplifie le processus de mise à jour du fichier de configuration de la CMU. Si vous utilisez une version de CMU issue du SDK client 3.2.1 ou d'une version antérieure, vous devez continuer à utiliser les paramètres `-a` et `-m` pour mettre à jour le fichier de configuration. Pour plus d'informations sur ces paramètres, veuillez consulter l'Outil de configuration.

Télécharger l'Utilitaire de gestion CloudHSM

La dernière version de CMU est disponible pour les tâches de gestion des utilisateurs HSM, que vous utilisiez le SDK client 5 ou le SDK client 3.

Pour télécharger et installer le CMU

- Téléchargez et installez le 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. Téléchargez l'Utilitaire de gestion CloudHSM.
2. Exécutez le programme d'installation de la CMU (AWSCloudHSMManagementUtil-latest.msi) avec les privilèges d'administrateur Windows.

Windows Server 2012 R2

1. Téléchargez l'Utilitaire de gestion CloudHSM.
2. Exécutez le programme d'installation de la CMU (AWSCloudHSMManagementUtil-latest.msi) avec les privilèges d'administrateur Windows.

Windows Server 2016

1. Téléchargez l'Utilitaire de gestion CloudHSM.
2. Exécutez le programme d'installation de la CMU (AWSCloudHSMManagementUtil-latest.msi) avec les privilèges d'administrateur Windows.

Comment gérer les utilisateurs du HSM avec le CMU

Cette section inclut les commandes de base pour gérer les utilisateurs HSM avec le CMU.

Pour créer des utilisateurs HSM

Utilisez createUser pour créer de nouveaux utilisateurs sur le HSM. Vous devez vous connecter en tant que CO pour créer un utilisateur.

Pour créer un nouvel utilisateur CO

1. Utilisez l'outil de configuration pour mettre à jour la configuration du CMU.

Linux

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

Windows

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

2. Démarrer le 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. Connectez-vous au HSM en tant qu'utilisateur CO.

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

   Assurez-vous que le nombre de connexions répertoriées par le CMU correspond au nombre de HSM du cluster. Si ce n'est pas le cas, déconnectez-vous et recommencez.

4. Utilisez createUser pour créer un utilisateur CO nommé **example_officer** dont le mot de passe est **password1**.

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

   L'utilitaire CMU vous invite à réaliser l'opération de création d'utilisateurs.

   ```
   ************************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. Tapez **y**.

Pour créer un nouvel utilisateur CU

1. Utilisez l'outil de configuration pour mettre à jour la configuration du CMU.

   Linux

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

   Windows

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

2. Démarrer le 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. Connectez-vous au HSM en tant qu'utilisateur CO.

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

   Assurez-vous que le nombre de connexions répertoriées par le CMU correspond au nombre de HSM du cluster. Si ce n'est pas le cas, déconnectez-vous et recommencez.

4. Utilisez createUser pour créer un utilisateur CU nommé **example_user** dont le mot de passe est **password1**.

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

L'utilitaire CMU vous invite à réaliser l'opération de création d'utilisateurs.

```
**********************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. Tapez **y**.

Pour plus d'informations sur createUser, veuillez consulter Créer un utilisateur.

Pour répertorier tous les utilisateurs HSM du cluster

Utilisez la commande listUsers pour répertorier tous les utilisateurs du cluster. Vous n'avez pas besoin de vous connecter pour exécuter listUsers et tous les types d'utilisateurs peuvent répertorier des utilisateurs.

Pour répertorier tous les utilisateurs du cluster

1. Utilisez l'outil de configuration pour mettre à jour la configuration du CMU.

   Linux

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

   Windows

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

2. Démarrer le 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. Utilisez listUsers pour répertorier tous les utilisateurs du cluster.

```
aws-cloudhsm>listUsers
```

L'utilitaire CMU répertorie tous les utilisateurs du cluster.

```
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
```

Pour de plus amples informations sur listUsers, veuillez consulter répertorier les utilisateurs.

Pour modifier les mots de passe utilisateur HSM

Pour changer un mot de passe, utilisez changePswd.

Les types d'utilisateurs et les mots de passe sont sensibles à la casse, les noms d'utilisateurs, non.

Les utilisateurs de chiffrement (CU) et les utilisateurs d'appliance (AU) peuvent modifier uniquement leur propre mot de passe. Pour modifier le mot de passe d'un autre utilisateur, vous devez vous connecter en tant que CO. Vous ne pouvez pas modifier le mot de passe d'un utilisateur qui est actuellement connecté.

Pour modifier votre propre mot de passe

1. Utilisez l'outil de configuration pour mettre à jour la configuration du CMU.

   Linux

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

   Windows

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

2. Démarrer le 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. Connectez-vous au HSM.

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

Assurez-vous que le nombre de connexions répertoriées par le CMU correspond au nombre de HSM du cluster. Si ce n'est pas le cas, déconnectez-vous et recommencez.

4. Pour modifier votre propre mot de passe utilisez changePswd.

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

Le CMU fournit une invite sur l'opération de modification du mot de passe.

```
**********************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. Tapez **y**.

Le CMU fournit une invite sur l'opération de modification du mot de passe.

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

Pour modifier le mot de passe d'un autre utilisateur

1. Utilisez l'outil de configuration pour mettre à jour la configuration du CMU.

Linux

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

Windows

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

2. Démarrer le 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. Connectez-vous au HSM en tant qu'utilisateur CO.

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

   Assurez-vous que le nombre de connexions répertoriées par le CMU correspond au nombre de HSM du cluster. Si ce n'est pas le cas, déconnectez-vous et recommencez.

4. Utilisez changePswd pour modifier le mot de passe d'un autre utilisateur.

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

   Le CMU fournit une invite sur l'opération de modification du mot de passe.

   ```
   ************************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. Tapez **y**.

   Le CMU fournit une invite sur l'opération de modification du mot de passe.

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

Pour plus d'informations sur changePswd, veuillez consulter Modifier le mot de passe.

Pour supprimer des utilisateurs HSM

Utilisez deleteUser pour supprimer un utilisateur. Vous devez vous connecter en tant que CO pour supprimer un autre utilisateur.

> **Tip**
>
> Vous ne pouvez pas supprimer les utilisateurs de chiffrement (CU) qui possèdent des clés.

Pour supprimer un utilisateur

1. Utilisez l'outil de configuration pour mettre à jour la configuration du CMU.

   Linux

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

   Windows

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

2. Démarrer le 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. Connectez-vous au HSM en tant qu'utilisateur CO.

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

Assurez-vous que le nombre de connexions répertoriées par le CMU correspond au nombre de HSM du cluster. Si ce n'est pas le cas, déconnectez-vous et recommencez.

4. Utilisez deleteUser pour supprimer un utilisateur.

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

Le CMU supprime l'utilisateur.

```
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)
```

Pour plus d'informations sur deleteUser, veuillez consulter Supprimer un utilisateur.

Utilisation de l'Utilitaire de gestion CloudHSM (CMU) pour gérer l'authentification à deux facteurs (2FA) pour les responsables du chiffrement

Pour plus de sécurité, vous pouvez configurer l'authentification à deux facteurs (2FA) pour mieux protéger le cluster. Vous ne pouvez activer l'authentification à deux facteurs que pour les responsables de chiffrement (CO).

> **Note**
>
> Vous ne pouvez pas activer l'authentification à deux facteurs pour les utilisateurs de chiffrement (CU) ou les applications. L'authentification à deux facteurs (2FA) est réservée aux utilisateurs de CO.

Rubriques

- Comprendre l'authentification à deux facteurs pour les utilisateurs HSM
- Utilisation de l'authentification à deux facteurs pour les utilisateurs HSM

Comprendre l'authentification à deux facteurs pour les utilisateurs HSM

Lorsque vous vous connectez à un cluster avec un compte HSM (Hardware Service Module) compatible 2FA, vous fournissez votre mot de passe à cloudhsm-mgmt_util (CMU) (le premier facteur, ce que vous connaissez), et l'utilitaire CMU vous fournit un jeton et vous invite à le faire signer. Pour fournir le deuxième facteur, c'est-à-dire ce que vous avez, vous signez le jeton avec une clé privée provenant d'une paire de clés que vous avez déjà créée et associée à l'utilisateur HSM. Pour accéder au cluster, vous devez fournir le jeton signé à l'utilitaire CMU.

Authentification par quorum et 2FA

Le cluster utilise la même clé pour l'authentification par quorum et pour l'authentification à deux facteurs. Cela signifie qu'un utilisateur dont l'authentification à deux facteurs est activée est effectivement enregistré pour le contrôle d'accès M of N (MofN). Pour utiliser correctement l'authentification 2FA et l'authentification par quorum pour le même utilisateur HSM, tenez compte des points suivants :

- Si vous utilisez l'authentification par quorum pour un utilisateur aujourd'hui, vous devez utiliser la même paire de clés que celle que vous avez créée pour l'utilisateur du quorum afin d'activer l'authentification à deux facteurs pour cet utilisateur.
- Si vous ajoutez l'exigence 2FA pour un utilisateur non-2FA qui n'est pas un utilisateur d'authentification par quorum, vous enregistrez cet utilisateur en tant qu'utilisateur MofN avec authentification 2FA.
- Si vous supprimez l'exigence 2FA ou modifiez le mot de passe d'un utilisateur 2FA qui est également un utilisateur d'authentification par quorum, vous supprimerez également l'enregistrement de l'utilisateur du quorum en tant qu'utilisateur MofN.
- Si vous supprimez l'exigence 2FA ou modifiez le mot de passe d'un utilisateur 2FA qui est également un utilisateur utilisant l'authentification par quorum, mais que vous souhaitez tout de même que cet utilisateur participe à l'authentification par quorum, vous devez l'enregistrer à nouveau en tant qu'utilisateur MofN.

Pour de plus amples informations sur l'authentification par quorum, veuillez consulter Utilisation du CMU pour gérer l'authentification du quorum.

Utilisation de l'authentification à deux facteurs pour les utilisateurs HSM

Cette section décrit comment utiliser 2FA pour les utilisateurs HSM, notamment en créant des utilisateurs 2FA HSM, en faisant pivoter les clés et en se connectant au HSM en tant qu'utilisateurs

compatibles 2FA. Pour plus d'informations sur l'utilisation d'utilisateurs HSM, consultez ???, ???, ???, ??? et ???.

Création d'utilisateurs 2FA

Pour activer 2FA pour un utilisateur HSM, utilisez une clé répondant aux exigences suivantes.

Exigences relatives aux paires de clés 2FA

Vous pouvez créer une nouvelle paire de clés ou utiliser une clé existante répondant aux exigences suivantes.

- Type de clé : asymétrique
- Utilisation de clé : signature et vérification
- Spécification de clé : RSA_2048
- L'algorithme de signature inclut :
  - `sha256WithRSAEncryption`

> **Note**
>
> Si vous utilisez l'authentification par quorum ou si vous envisagez d'utiliser l'authentification par quorum, consultez the section called “Authentification par quorum et 2FA”.

Utilisez l'utilitaire CMU et la paire de clés pour créer un nouvel utilisateur CO pour lequel 2FA est activé.

Pour créer des utilisateurs CO pour lesquels 2FA est activé

1. Dans un terminal, effectuez l'une des étapes suivantes :

   a. Accédez à votre HSM et connectez-vous à l'Utilitaire de gestion CloudHSM :

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

   b. Connectez-vous en tant que CO et utilisez la commande suivante pour créer un nouvel utilisateur MFA avec 2FA :

   ```
   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. Laissez le terminal ci-dessus dans cet état. N'appuyez pas sur Entrée et n'entrez aucun nom de fichier.

2. Dans un autre terminal, effectuez les opérations suivantes :

a. Accédez à votre HSM et connectez-vous à l'Utilitaire de gestion CloudHSM :

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

b. Générez une paire de clés publique-privée à l'aide des commandes suivantes :

```
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. Exécutez la commande suivante pour installer une fonctionnalité de requête JSON permettant d'extraire le résumé du fichier authdata :

```
sudo yum install jq
```

d. Pour extraire la valeur du résumé, recherchez d'abord les données suivantes dans le fichier authdata :

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

> **Note**
>
> Le résumé obtenu est codé en base64, mais pour signer le résumé, vous devez d'abord décoder puis signer le fichier. La commande suivante décodera le résumé et stockera le contenu décodé dans « digest1.bin »
>
> ```
> cat authdata | jq '.Data[0].Digest' | cut -c2- | rev | cut -c2- | rev |
>  base64 -d > digest1.bin
> ```

e. Convertissez le contenu de la clé publique en ajoutant « \n » et en supprimant les espaces comme indiqué ici :

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

> **Important**
>
> La commande ci-dessus montre comment « \n » est ajouté immédiatement après BEGIN PUBLIC KEY-----, les espaces entre « \n » et le premier caractère de la clé publique sont supprimés, « \n » est ajouté avant -----END PUBLIC KEY et les espaces sont supprimés entre « \n » et la fin de la clé publique.

Il s'agit du format PEM pour la clé publique qui est accepté dans le fichier authdata.

f. Collez le contenu du format pem de la clé publique dans la section clé publique du fichier authdata.

```
vi authdata
```

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

g. Signez le fichier du jeton à l'aide de la commande suivante :

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

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

> **Note**
>
> Comme indiqué dans la commande ci-dessus, utilisez openssl pkeyutl plutôt que openssl dgst pour signer.

h. Ajoutez le résumé signé dans le fichier Authdata dans le champ « Signature ».

```
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. Retournez au premier terminal et appuyez sur **`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)
```

Gestion de l'authentification à deux facteurs pour les utilisateurs HSM

Utilisez le changement de mot de passe pour modifier le mot de passe d'un utilisateur 2FA, pour activer ou désactiver l'authentification 2FA, ou pour faire pivoter la clé 2FA. Chaque fois que vous activez l'authentification à deux facteurs, vous devez fournir une clé publique pour les connexions à deux facteurs.

Le changement de mot de passe exécute l'un des scénarios suivants :

- Modification du mot de passe d'un utilisateur 2FA
- Modification du mot de passe d'un utilisateur non-2FA
- Ajout de 2FA à un utilisateur non-2FA
- Suppression de l'authentification à deux facteurs d'un utilisateur 2FA
- Rotation de la clé pour un utilisateur 2FA

Vous pouvez également combiner des tâches. Par exemple, vous pouvez supprimer l'authentification 2FA d'un utilisateur et modifier le mot de passe en même temps, ou vous pouvez faire pivoter la clé 2FA et modifier le mot de passe utilisateur.

Pour modifier les mots de passe ou faire pivoter les clés pour les utilisateurs CO pour lesquels 2FA est activé

1. Utilisez l'utilitaire CMU pour vous connecter au HSM en tant que CO avec 2FA activé.
2. Utilisez changePswd pour modifier le mot de passe ou faire pivoter la clé pour les utilisateurs CO pour lesquels l'authentification à deux facteurs est activée. Utilisez le paramètre `-2fa` et incluez un emplacement dans le système de fichiers pour que le système puisse écrire le fichier `authdata`. Ce fichier inclut un résumé pour chaque HSM du cluster.

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

   L'utilitaire CMU vous invite à utiliser la clé privée pour signer les résumés du fichier `authdata` et à renvoyer les signatures avec la clé publique.
3. Utilisez la clé privée pour signer les résumés du fichier `authdata`, ajoutez les signatures et la clé publique au fichier `authdata` au format JSON, puis indiquez à l'utilitaire CMU l'emplacement du fichier `authdata`. Pour plus d'informations, consultez the section called "Référence de configuration".

   > **Note**
   >
   > Le cluster utilise la même clé pour l'authentification par quorum et l'authentification à deux facteurs. Si vous utilisez l'authentification par quorum ou si vous envisagez d'utiliser l'authentification par quorum, consultez the section called "Authentification par quorum et 2FA".

Pour désactiver l'authentification à deux facteurs pour les utilisateurs CO pour lesquels l'authentification à deux facteurs est activée

1. Utilisez l'utilitaire CMU pour vous connecter au HSM en tant que CO avec 2FA activé.
2. Utilisez changePswd pour supprimer l'authentification à deux facteurs pour les utilisateurs CO pour lesquels l'authentification à deux facteurs est activée.

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

   L'utilitaire CMU vous invite à confirmer l'opération de modification du mot de passe.

> **Note**
>
> Si vous supprimez l'exigence 2FA ou modifiez le mot de passe d'un utilisateur 2FA qui est également un utilisateur d'authentification par quorum, vous supprimerez également l'enregistrement de l'utilisateur du quorum en tant qu'utilisateur MofN. Pour plus d'informations sur les utilisateurs du quorum et 2FA, consultez the section called "Authentification par quorum et 2FA".

3. Tapez **y**.

   L'utilitaire CMU confirme l'opération de modification du mot de passe.

Référence de configuration

Voici un exemple des propriétés 2FA du fichier `authdata` pour la demande générée par l'utilitaire CMU et pour vos réponses.

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

Données

Nœud de niveau supérieur. Contient un nœud subordonné pour chaque HSM du cluster. Apparaît dans les demandes et les réponses pour toutes les commandes 2FA.

Digest

C'est ce que vous devez signer pour fournir l'authentification à deux facteurs. CMU générée dans les demandes pour toutes les commandes 2FA.

HsmId

L'identifiant de votre HSM. Apparaît dans les demandes et les réponses pour toutes les commandes 2FA.

PublicKey

La partie clé publique de la paire de clés que vous avez générée a été insérée sous forme de chaîne au format PEM. Vous le saisissez dans les réponses pour createUser et changePswd.

Signature

Le résumé signé codé en Base64. Vous le saisissez dans les réponses pour toutes les commandes 2FA.

Version

La version du fichier de données d'authentification au format JSON. Apparaît dans les demandes et les réponses pour toutes les commandes 2FA.

Utilisation de l'Utilitaire de gestion CloudHSM (CMU) pour gérer l'authentification par quorum (contrôle d'accès M sur N)

Les HSM de votre AWS CloudHSM cluster prennent en charge l'authentification par quorum, également connue sous le nom de contrôle d'accès M of N. Avec l'authentification par quorum, aucun utilisateur individuel sur le HSM ne peut effectuer d'opérations contrôlées par quorum sur le HSM. À la place, un nombre minimal d'utilisateurs HSM (au moins 2) doivent coopérer pour effectuer ces opérations. Avec l'authentification par quorum, vous pouvez ajouter une couche de protection supplémentaire en demandant l'approbation de plusieurs utilisateurs HSM.

L'authentification par quorum peut contrôler les opérations suivantes :

- La gestion des utilisateurs HSM par les responsables de chiffrement (CO) - La création et la suppression d'utilisateurs HSM, et la modification du mot de passe d'un autre utilisateur HSM. Pour plus d'informations, consultez Utilisation de l'authentification par quorum pour les responsables de chiffrement.

Les rubriques suivantes fournissent plus d'informations sur l'authentification par quorum dans AWS CloudHSM.

Rubriques

Présentation de l'authentification par quorum

Les étapes suivantes résument les processus d'authentification par quorum. Pour connaître les étapes et les outils spécifiques, consultez Utilisation de l'authentification par quorum pour les responsables de chiffrement.

1. Chaque utilisateur HSM crée une clé asymétrique pour la signature. Il s'en charge en dehors du HSM, en veillant à protéger la clé de manière appropriée.
2. Chaque utilisateur HSM se connecte au HSM et enregistre la partie publique de sa clé de signature (la clé publique) avec le HSM.
3. Lorsqu'un utilisateur HSM veut effectuer une opération contrôlée par quorum, il se connecte au HSM et obtient un jeton de quorum.
4. L'utilisateur HSM donne le jeton de quorum à un ou plusieurs autres utilisateurs HSM et demande leur approbation.
5. Les autres utilisateurs HSM approuvent en utilisant leurs clés pour signer de façon chiffrée le jeton de quorum. Cela se produit en dehors du HSM.
6. Lorsque l'utilisateur HSM dispose du nombre requis d'approbations, il se connecte au HSM et donne le jeton de quorum et les approbations (signatures) au HSM.
7. Le HSM utilise les clés publiques enregistrées de chaque signataire pour vérifier les signatures. Si les signatures sont valides, le HSM approuve le jeton.
8. L'utilisateur HSM peut désormais effectuer une opération contrôlée par quorum.

Détails supplémentaires concernant l'authentification par quorum

Notez les informations supplémentaires suivantes sur l'utilisation de l'authentification par quorum dans AWS CloudHSM.

- Un utilisateur HSM peut signer son propre jeton de quorum, c'est-à-dire que l'utilisateur demandeur peut fournir l'une des approbations requises pour l'authentification par quorum.
- Vous choisissez le nombre minimal d'approbateurs de quorum pour les opérations contrôlées par quorum. Le plus petit nombre que vous pouvez choisir est deux (2), et le plus grand nombre que vous pouvez choisir est huit (8).
- Le HSM peut stocker jusqu'à 1 024 jetons de quorum. Si le HSM possède déjà 1 024 jetons lorsque vous essayez d'en créer un nouveau, le HSM efface l'un des jetons ayant expiré. Par défaut, les jetons expirent dix minutes après leur création.
- Le cluster utilise la même clé pour l'authentification par quorum et pour l'authentification à deux facteurs (2FA). Pour plus d'informations sur l'utilisation de l'authentification par quorum et de l'authentification 2FA, consultez Authentification par quorum et 2FA.

Utilisation de l'authentification par quorum pour les responsables de chiffrement : première configuration

Les rubriques suivantes décrivent les étapes que vous devez suivre pour configurer votre module de sécurité matériel (HSM) afin que les responsables de chiffrement (CO) puissent utiliser l'authentification par quorum. Vous devez suivre ces étapes une seule fois lorsque vous configurez l'authentification par quorum pour les responsables de chiffrement. Une fois que vous avez terminé ces étapes, consultez Utilisation de l'authentification par quorum pour les responsables de chiffrement.

Rubriques

- Prérequis
- Création et enregistrement d'une clé pour la signature
- Définition de la valeur minimale de quorum sur le HSM

Prérequis

Pour comprendre cet exemple, vous devez connaître l'outil de ligne de commande cloudhsm_mgmt_util (CMU). Dans cet exemple, le AWS CloudHSM cluster possède deux HSM,

chacun avec le même CoS, comme indiqué dans le résultat suivant de la listUsers commande. Pour plus d'informations sur la création d'utilisateurs, consultez Gestion des utilisateurs HSM.

```
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
```

Création et enregistrement d'une clé pour la signature

Pour utiliser l'authentification par quorum, chaque CO doit effectuer toutes les étapes suivantes :

Rubriques

- Créer une paire de clés RSA
- Créez et signez un jeton d'enregistrement
- Enregistrez la clé publique avec le HSM

Créer une paire de clés RSA

Il existe de nombreuses manières différentes de créer et de protéger une paire de clés. Les exemples suivants montrent comment procéder avec OpenSSL.

Example — Créer une clé privée à l'aide d'OpenSSL

L'exemple suivant montre comment utiliser OpenSSL pour créer une clé RSA de 2048 bits, protégée par une phrase secrète. Pour utiliser cet exemple, remplacez *officer1.key* par le nom du fichier dans lequel vous souhaitez stocker la clé.

```
$ 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:
```

Générez ensuite la clé publique à l'aide de la clé privée que vous venez de créer.

Example — Crée une clé publique avec OpenSSL

L'exemple suivant montre comment utiliser OpenSSL pour créer une clé publique à partir de la clé privée que vous venez de créer.

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

Créez et signez un jeton d'enregistrement

Vous créez un jeton et vous le signez avec la clé privée que vous venez de générer à l'étape précédente.

Example — Crée un jeton

Le jeton d'enregistrement est simplement un fichier contenant des données aléatoires ne dépassant pas la taille maximale de 245 octets. Vous signez le jeton avec la clé privée pour démontrer que vous avez accès à la clé privée. La commande suivante utilise echo pour rediriger une chaîne vers un fichier.

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

Signez le jeton et enregistrez-le dans un fichier de signature. Vous aurez besoin du jeton signé, du jeton non signé et de la clé publique pour enregistrer le CO en tant qu'utilisateur du MofN auprès du HSM.

Example — Signez le jeton

Utilisez OpenSSL et la clé privée pour signer le jeton d'enregistrement et créer le fichier de signature.

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

Enregistrez la clé publique avec le HSM

Après avoir créé une clé, le responsable de chiffrement doit enregistrer la partie publique de la clé (clé publique) avec le HSM.

Pour enregistrer une clé publique avec le HSM

1. Utilisez la commande suivante pour démarrer l'outil de ligne de commande cloudhsm_mgmt_util.

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

2. Utilisez la commande loginHSM pour vous connecter aux HSM en tant que CO. Pour plus d'informations, consultez ???.

3. Utilisez la commande registerQuorumPubKey pour enregistrer la clé publique.
   Pour plus d'informations, consultez l'exemple suivant ou utilisez la commande help registerQuorumPubKey.

Example - Enregistrer une clé publique avec le HSM

L'exemple suivant montre comment utiliser la commande registerQuorumPubKey de l'outil de ligne de commande cloudhsm_mgmt_util pour enregistrer la clé publique d'un responsable de chiffrement avec le HSM. Pour utiliser cette commande, le responsable de chiffrement doit être connecté au HSM. Remplacez les valeurs suivantes par les vôtres :

```
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>

Le chemin d'accès à un fichier contenant un jeton d'enregistrement non signé. Peut avoir n'importe quelle donnée aléatoire d'une taille de fichier maximale de 245 octets.

Obligatoire : oui

<officer1.token.sig>

Le chemin d'accès à un fichier contenant le hachage signé par le mécanisme SHA256_PKCS du jeton d'enregistrement.

Obligatoire : oui

<officer1.pub>

Le chemin d'accès au fichier contenant la clé publique d'une paire de clés asymétriques RSA-2048. Utilisez la clé privée pour signer le jeton d'enregistrement.

Obligatoire : oui

Une fois que tous les responsables de chiffrement ont enregistré leurs clés publiques, la sortie de la commande listUsers montre cela dans la colonne `MofnPubKey`, comme indiqué dans l'exemple suivant.

```
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
```

Définition de la valeur minimale de quorum sur le HSM

Pour utiliser l'authentification par quorum pour les responsables de chiffrement, un responsable de chiffrement doit se connecter au HSM, puis définir la valeur minimale de quorum, également appelée valeur m. Il s'agit du nombre minimal d'approbations de responsable de chiffrement qui sont nécessaires pour effectuer des opérations de gestion des utilisateurs HSM. Tout responsable de chiffrement sur le HSM peut définir la valeur minimale de quorum, y compris les responsables de chiffrement qui n'ont pas enregistré de clé pour la signature. Vous pouvez modifier la valeur minimale de quorum à tout moment ; pour plus d'informations, consultez Modifier la valeur minimale..

Pour définir la valeur minimale de quorum sur le HSM

1. Utilisez la commande suivante pour démarrer l'outil de ligne de commande cloudhsm_mgmt_util.

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

2. Utilisez la commande loginHSM pour vous connecter aux HSM en tant que CO. Pour plus d'informations, consultez ???.
3. Utilisez la commande setMValue pour définir la valeur minimale de quorum. Pour plus d'informations, consultez l'exemple suivant ou utilisez la commande help setMValue.

Example - Définition de la valeur minimale de quorum sur le HSM

Cet exemple utilise une valeur minimale de quorum de deux. Vous pouvez choisir n'importe quelle valeur comprise entre deux (2) et huit (8), jusqu'au nombre total de CO sur le HSM. Dans cet exemple, le HSM a six CO, de sorte que la valeur maximale possible est six.

Pour utiliser l'exemple de commande suivant, remplacez le nombre final (*2*) par la valeur minimale de quorum que vous préférez.

```
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
```

Dans l'exemple précédent, le premier nombre (3) identifie le service HSM dont vous définissez la valeur minimale de quorum.

Le tableau suivant répertorie les identifiants de service HSM ainsi que leurs noms, descriptions et commandes incluses dans le service.

Identifiant du service	Service Name	Description du service	Commandes HSM
3	USER_MGMT	Gestion des utilisateurs HSM	• createUser • deleteUser • changePswd (s'applique uniquement lors de la modification du mot de passe d'un autre utilisateur HSM)
4	MISC_CO	Service CO divers	• setMValue

Pour obtenir la valeur minimale de quorum pour un service, utilisez la commande getMValue, comme dans l'exemple suivant.

```
aws-cloudhsm>getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

La sortie de la commande getMValue précédente indique que la valeur minimale de quorum pour les opérations de gestion des utilisateurs HSM (service 3) est désormais égale à deux.

Une fois que vous avez terminé ces étapes, consultez Utilisation de l'authentification par quorum pour les responsables de chiffrement.

Utilisation de l'authentification par quorum pour les responsables de chiffrement

Un responsable de chiffrement (CO) sur le HSM peut configurer l'authentification par quorum pour les opérations suivantes sur le HSM :

- Création d'utilisateurs HSM
- Suppression d'utilisateurs HSM
- Modification du mot de passe d'un autre utilisateur HSM

Une fois que le HSM est configuré pour l'authentification par quorum, les responsables de chiffrement ne peuvent pas effectuer d'opérations de gestion des utilisateurs par eux-mêmes. L'exemple suivant montre la sortie lorsqu'un responsable de chiffrement tente de créer un nouvel utilisateur sur le HSM. La commande échoue avec une erreur `RET_MXN_AUTH_FAILED`, ce qui indique que l'authentification par quorum a échoué.

```
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
```

Pour effectuer une opération de gestion d'utilisateur HSM, un responsable de chiffrement doit exécuter les tâches suivantes :

1. Obtenir un jeton de quorum.
2. Obtenir des approbations (signatures) de la part d'autres responsables de chiffrement.
3. Approuver le jeton sur le HSM.
4. Effectuer l'opération de gestion des utilisateurs HSM.

Si vous n'avez pas encore configuré le module HSM pour l'authentification par quorum pour les responsables de chiffrement, faites-le maintenant. Pour plus d'informations, consultez Première configuration.

Obtention d'un jeton de quorum

Tout d'abord, le CO doit utiliser l'outil de ligne de commande cloudhsm_mgmt_util pour demander un jeton de quorum.

Pour obtenir un jeton de quorum

1. Utilisez la commande suivante pour démarrer l'outil de ligne de commande cloudhsm_mgmt_util.

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

2. Utilisez la commande loginHSM pour vous connecter aux HSM en tant que CO. Pour plus d'informations, consultez ???.
3. Utilisez la commande getToken pour obtenir un jeton de quorum. Pour plus d'informations, consultez l'exemple suivant ou utilisez la commande help getToken.

Example - Obtenir un jeton de quorum.

Cet exemple obtient un jeton de quorum pour le responsable de chiffrement avec le nom d'utilisateur officer1 et enregistre le jeton dans un fichier nommé `officer1.token`. Pour utiliser l'exemple de commande, remplacez ces valeurs par les vôtres :

- *`officer1`* – Le nom du CO qui obtient le jeton. Il doit s'agir du même responsable de chiffrement que celui qui est connecté au HSM et qui exécute cette commande.
- *`officer1.token`* – Le nom du fichier à utiliser pour stocker le jeton de quorum.

Dans la commande suivante, 3 identifie le service pour lequel vous pouvez utiliser le jeton que vous obtenez. Dans ce cas, le jeton concerne les opérations de gestion des utilisateurs HSM (service 3). Pour plus d'informations, consultez Définition de la valeur minimale de quorum sur le HSM.

```
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
```

Obtention des signatures des CO en charge de l'approbation

Un responsable de chiffrement qui possède un jeton de quorum doit obtenir le jeton approuvé par d'autres responsables de chiffrement. Pour donner leur approbation, les autres responsables de chiffrement utilisent leur clé de signature pour signer le jeton de façon cryptographique. Ils le font en dehors du module HSM.

Il existe de nombreuses façons différentes de signer le jeton. L'exemple suivant montre comment procéder avec OpenSSL. Pour utiliser un autre outil de signature, assurez-vous que l'outil utilise la clé privée du responsable de chiffrement (clé de signature) pour signer un hachage SHA-256 du jeton.

Example - Obtention des signatures des responsables de chiffrement en charge de l'approbation

Dans cet exemple, le CO qui possède le jeton (officer1) a besoin d'au moins deux approbations. L'exemple suivant montrent comment deux CO peuvent utiliser OpenSSL pour signer le jeton de manière cryptographique.

Dans la première commande, officer1 signe son propre jeton. Pour utiliser les commandes de l'exemple suivant, remplacez ces valeurs par les vôtres :

- *officer1.key* et *officer2.key* – Le nom du fichier qui contient la clé de signature du CO.
- *officer1.token.sig1* et *officer1.token.sig2* – Le nom du fichier à utiliser pour stocker la signature. Assurez-vous de sauvegarder chaque signature dans un fichier différent.
- *officer1.token* – Le nom du fichier qui contient le jeton que le responsable de chiffrement signe.

```
$ openssl dgst -sha256 -sign officer1.key -out officer1.token.sig1 officer1.token
```

```
Enter pass phrase for officer1.key:
```

Dans la commande suivante, officer2 signe le même jeton.

```
$ openssl dgst -sha256 -sign officer2.key -out officer1.token.sig2 officer1.token
Enter pass phrase for officer2.key:
```

Approbation du jeton signé sur le HSM

Une fois qu'un responsable de chiffrement obtient le nombre minimal d'approbations (signatures) d'autres CO, alors il doit approuver le jeton signé sur le HSM.

Pour approuver le jeton signé sur le HSM

1. Créez un fichier d'approbation de jeton. Pour plus d'informations, consultez l'exemple suivant.
2. Utilisez la commande suivante pour démarrer l'outil de ligne de commande cloudhsm_mgmt_util.

   ```
   $ /opt/cloudhsm/bin/cloudhsm_mgmt_util /opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg
   ```
3. Utilisez la commande loginHSM pour vous connecter aux HSM en tant que CO. Pour plus d'informations, consultez ???.
4. Utilisez la commande approveToken pour approuver le jeton signé, en transmettant le fichier d'approbation du jeton. Pour plus d'informations, consultez l'exemple suivant.

Example - Création d'un fichier d'approbation de jeton et approbation du jeton signé sur le HSM

Le fichier d'approbation de jeton est un fichier texte dans un format particulier que le HSM nécessite. Le fichier contient des informations sur le jeton, ses approbateurs et leurs signatures. L'exemple suivant montre un exemple de fichier d'approbation de jeton.

```
# 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;
```

Après avoir créé le fichier, le jeton d'approbation CO utilise l'outil de ligne de commande cloudhsm_mgmt_util pour vous connecter au HSM. Le CO utilise ensuite la commande approveToken pour approuver le jeton, comme indiqué dans l'exemple suivant. Remplacez *approval.txt* par le nom du fichier d'approbation du jeton.

```
aws-cloudhsm>approveToken approval.txt
approveToken success on server 0(10.0.2.14)
approveToken success on server 1(10.0.1.4)
```

Lorsque cette commande réussit, le module HSM a approuvé le jeton de quorum. Pour vérifier le statut d'un jeton, utilisez la commande listTokens, comme indiqué dans l'exemple suivant. La sortie de la commande affiche que le jeton a le nombre requis d'approbations.

La durée de validité du jeton indique la durée pendant laquelle le jeton est assuré de demeurer sur le module HSM. Même après que la durée de validité du jeton s'est écoulée (zéro seconde), vous pouvez continuer d'utiliser le jeton.

```
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
```

Utilisation du jeton pour les opérations de gestion des utilisateurs

Une fois qu'un CO possède un jeton avec le nombre requis d'approbations, comme illustré dans la section précédente, le CO peut effectuer l'une des opérations de gestion d'utilisateur HSM suivantes :

- Créer un utilisateur HSM avec la commande createUser
- Supprimer un utilisateur HSM avec la commande deleteUser
- Modifier le mot de passe d'un autre utilisateur HSM avec la commande changePswd

Pour plus d'informations sur l'utilisation de ces commandes, consultez Gestion des utilisateurs HSM.

Le CO peut utiliser le jeton pour une seule opération. Lorsque cette opération réussit, le jeton n'est plus valide. Pour effectuer une autre opération de gestion de l'utilisateur HSM, le CO doit obtenir un

nouveau jeton de quorum, obtenir de nouvelles signatures des approbateurs et approuver le nouveau jeton sur le HSM.

> **Note**
>
> Le jeton MofN n'est valide que tant que votre session de connexion en cours est ouverte. Si vous vous déconnectez de cloudhsm_mgmt_util ou si la connexion réseau est déconnectée, le jeton n'est plus valide. De même, un jeton autorisé ne peut être utilisé que dans cloudhsm_mgmt_util, il ne peut pas être utilisé pour s'authentifier dans une autre application.

Dans l'exemple de commande suivant, le CO crée un utilisateur sur le HSM.

```
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
```

Une fois que la commande précédente a réussi, une commande listUsers affiche le nouvel utilisateur.

```
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
```

Si le CO essaie d'effectuer une autre opération de gestion de l'utilisateur HSM, l'opération échoue avec une erreur d'authentification par quorum, comme illustré dans l'exemple suivant.

```
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
```

La commande listTokens montre que le CO n'a pas de jetons approuvés, comme illustré dans l'exemple suivant. Pour effectuer une autre opération de gestion de l'utilisateur HSM, le CO doit obtenir un nouveau jeton de quorum, obtenir de nouvelles signatures des approbateurs et approuver le nouveau jeton sur le HSM.

```
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
```

Modifier la valeur minimale du quorum pour les responsables de chiffrement

Après avoir défini la valeur minimale du quorum de sorte que les responsables de chiffrement puissent utiliser l'authentification par quorum, vous pouvez souhaiter modifier la valeur minimale du quorum. Le HSM vous permet de modifier la valeur minimale du quorum uniquement lorsque le nombre d'approbateurs est égal ou supérieur à la valeur minimale du quorum. Par exemple, si la valeur minimale du quorum est deux, au moins deux responsables de chiffrement doivent donner leur approbation pour une modification de la valeur minimale du quorum.

Pour obtenir l'approbation du quorum en vue de modifier la valeur minimale du quorum, vous avez besoin d'un jeton de quorum pour la commande setMValue (service 4). Pour obtenir un jeton de quorum pour la commande setMValue (service 4), la valeur minimale du quorum pour service 4 doit être supérieure à un. Cela signifie que, avant de pouvoir modifier la valeur minimale du quorum pour les responsables de chiffrement (service 3), vous devrez peut-être modifier la valeur minimale du quorum pour service 4.

Le tableau suivant répertorie les identifiants de service HSM ainsi que leurs noms, descriptions et commandes incluses dans le service.

Identifiant du service	Service Name	Description du service	Commandes HSM
3	USER_MGMT	Gestion des utilisateurs HSM	• createUser • deleteUser • changePswd (s'applique uniquement lors de la modification du mot de passe d'un autre utilisateur HSM)
4	MISC_CO	Service CO divers	• setMValue

Pour modifier la valeur minimale du quorum pour les responsables de chiffrement

1. Utilisez la commande suivante pour démarrer l'outil de ligne de commande cloudhsm_mgmt_util.

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

2. Utilisez la commande loginHSM pour vous connecter aux HSM en tant que CO. Pour plus d'informations, consultez ???.
3. Utilisez la commande getMValue pour obtenir la valeur minimale du quorum pour service 3. Pour plus d'informations, consultez l'exemple suivant.
4. Utilisez la commande getMValue pour obtenir la valeur minimale du quorum pour service 4. Pour plus d'informations, consultez l'exemple suivant.
5. Si la valeur minimale du quorum pour service 4 est inférieure à celle pour service 3, utilisez la commande setMValue pour modifier la valeur spécifiée pour service 4. Indiquez, pour service 4, une valeur égale ou supérieure à celle utilisée pour service 3. Pour plus d'informations, consultez l'exemple suivant.
6. Obtenez un jeton de quorum, en veillant à spécifier le service 4 en tant que service pour lequel vous pouvez utiliser le jeton.
7. Obtenir des approbations (signatures) de la part d'autres responsables de chiffrement.
8. Approuver le jeton sur le HSM.

9. Utilisez la commande setMValue pour modifier la valeur minimale du quorum pour service 3 (opérations de gestion des utilisateurs réalisées par les responsables de chiffrement).

Example - Obtenir des valeurs minimale de quorum et modifier la valeur pour service 4

L'exemple de commande suivant montre que la valeur minimale du quorum pour service 3 est actuellement deux.

```
aws-cloudhsm>getMValue 3
MValue of service 3[USER_MGMT] on server 0 : [2]
MValue of service 3[USER_MGMT] on server 1 : [2]
```

L'exemple de commande suivant montre que la valeur minimale du quorum pour service 4 est actuellement un.

```
aws-cloudhsm>getMValue 4
MValue of service 4[MISC_CO] on server 0 : [1]
MValue of service 4[MISC_CO] on server 1 : [1]
```

Pour modifier la valeur minimale du quorum pour service 4, utilisez la commande setMValue pour définir une valeur égale ou supérieure à celle utilisée pour service 3. L'exemple suivant définit la valeur minimale du quorum pour service 4 à deux (2), valeur identique à celle définie pour service 3.

```
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
```

Les commandes suivantes montrent que la valeur minimale du quorum est désormais deux pour service 3 et service 4.

```
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]
```

Gestion des clés dans AWS CloudHSM

Dans AWS CloudHSM, utilisez l'une des méthodes suivantes pour gérer les clés des HSM de votre cluster :

- Bibliothèque PKCS #11
- Fournisseur JCE
- Fournisseurs KSP et CNG
- CLI CloudHSM

Pour gérer les clés, connectez-vous au HSM avec le nom d'utilisateur et le mot de passe d'un utilisateur de chiffrement (CU). Seul un CU peut créer une clé. Le CU qui crée une clé possède et gère cette clé.

Rubriques

Principaux paramètres de synchronisation et de durabilité dans AWS CloudHSM

Cette rubrique décrit les paramètres de synchronisation des clés AWS CloudHSM, les problèmes courants auxquels les clients sont confrontés lorsqu'ils utilisent des clés dans un cluster et les stratégies visant à rendre les clés plus durables.

Rubriques

Concepts

Clés de jetons

Clés persistantes que vous créez lors des opérations de génération, d'importation ou de déballage des clés. AWS CloudHSM synchronise les clés de jeton au sein d'un cluster.

Clés de session

Clés éphémères qui n'existent que sur un seul module de sécurité matériel (HSM) du cluster. AWS CloudHSM ne synchronise pas les clés de session au sein d'un cluster.

Synchronisation des clés côté client

Processus côté client qui clone les clés de jeton que vous créez lors des opérations de génération, d'importation ou de désencapsulage des clés. Vous pouvez rendre les clés à jeton plus durables en exécutant un cluster comportant au moins deux HSM.

Synchronisation des clés côté serveur

Clone régulièrement les clés de chaque HSM du cluster. Ne nécessite aucune gestion.

Paramètres de durabilité des clés du client

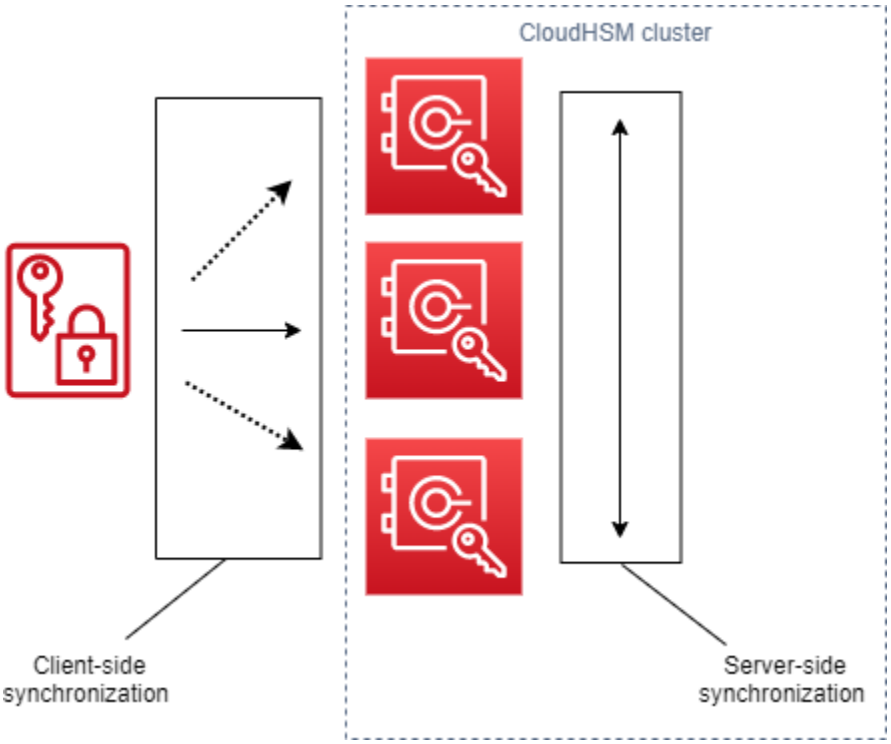
Paramètres que vous configurez sur le client qui ont un impact sur la durabilité des clés. Ces paramètres fonctionnent différemment dans le SDK client 5 et dans le SDK client 3.

- Dans le SDK client 5, utilisez ce paramètre pour exécuter un seul cluster HSM.
- Dans le SDK client 3, utilisez ce paramètre pour spécifier le nombre de HSM nécessaires à la réussite des opérations de création de clés.

Présentation de la synchronisation des clés

AWS CloudHSM utilise la synchronisation des clés pour cloner les clés de jeton sur tous les HSM d'un cluster. Vous créez des clés de jeton en tant que clés persistantes lors des opérations de

génération, d'importation ou de désencapsulage de clés. Pour distribuer ces clés au sein du cluster, CloudHSM propose une synchronisation des clés côté client et côté serveur.



L'objectif de la synchronisation des clés, à la fois côté serveur et côté client, est de distribuer les nouvelles clés dans le cluster le plus rapidement possible après les avoir créées. Cela est important car les appels suivants que vous effectuez pour utiliser de nouvelles clés peuvent être routés vers n'importe quel HSM disponible dans le cluster. Si l'appel que vous effectuez est acheminé vers un HSM sans la clé, l'appel échoue. Vous pouvez atténuer ce type d'échec en spécifiant que vos applications réessayent les appels suivants effectués après les opérations de création de clés. Le temps nécessaire à la synchronisation peut varier en fonction de la charge de travail de votre cluster et des autres actifs incorporels. Utilisez CloudWatch des métriques pour déterminer le calendrier que votre application doit utiliser dans ce type de situation. Pour plus d'informations, consultez la section CloudWatch Métriques.

Le défi de la synchronisation des clés dans un environnement cloud réside dans la durabilité des clés. Vous créez des clés sur un seul HSM et vous commencez souvent à les utiliser immédiatement. Si le HSM sur lequel vous créez les clés tombe en panne avant que les clés n'aient été clonées sur un autre HSM du cluster, vous perdez les clés et l'accès à tout ce qui est chiffré par les clés. Pour atténuer ce risque, nous proposons une synchronisation côté client. La synchronisation côté client est un processus côté client qui clone les clés que vous créez lors des opérations de génération, d'importation ou de désencapsulage des clés. Le clonage des clés au fur et à mesure de leur création les rend plus durables. Bien entendu, vous ne pouvez pas cloner des clés dans un cluster avec un

seul HSM. Pour améliorer la durabilité des clés, nous vous recommandons également de configurer votre cluster pour qu'il utilise au moins deux HSM. Grâce à la synchronisation côté client et à un cluster avec deux HSM, vous pouvez relever le défi de la durabilité des clés dans un environnement cloud.

Utilisation des paramètres de durabilité des clés du client

La synchronisation des clés est principalement un processus automatique, mais vous pouvez gérer les paramètres de durabilité des clés côté client. Les paramètres de durabilité des clés côté client fonctionnent différemment dans le SDK client 5 et le SDK client 3.

- Dans le SDK client 5, nous introduisons le concept de quorums de disponibilité des clés, qui vous oblige à exécuter des clusters avec un minimum de deux HSM. Vous pouvez utiliser les paramètres de durabilité des clés côté client pour vous soustraire à l'exigence des deux HSM. Pour plus d'informations sur les quorums, veuillez consulter the section called “Concepts du SDK client 5”.
- Dans le SDK client 3, vous utilisez les paramètres de durabilité des clés côté client pour spécifier le nombre de HSM sur lesquels la création de clés doit réussir pour que l'opération globale soit considérée comme réussie.

Paramètres de durabilité des clés du SDK client 5

Dans le SDK client 5, la synchronisation des clés est un processus entièrement automatique. Avec un quorum de disponibilité des clés, les clés nouvellement créées doivent exister sur deux HSM du cluster pour que votre application puisse utiliser la clé. Pour utiliser un quorum de disponibilité des clés, votre cluster doit avoir au moins deux HSM.

Si la configuration de votre cluster ne répond pas aux principales exigences de durabilité, toute tentative de création ou d'utilisation d'une clé de jeton échouera avec le message d'erreur suivant dans les journaux :

```
Key <key handle> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

Vous pouvez utiliser les paramètres de configuration du client pour désactiver le quorum de disponibilité des clés. Vous pouvez choisir de ne pas exécuter un cluster avec un seul HSM, par exemple.

Concepts du SDK client 5

Quorum de disponibilité des clés

AWS CloudHSM indique le nombre de HSM d'un cluster sur lesquels les clés doivent exister avant que votre application puisse les utiliser. Nécessite des clusters avec au moins deux HSM.

Gestion des paramètres de durabilité des clés du client

Pour gérer les paramètres de durabilité des clés client, vous devez utiliser l'outil de configuration du SDK client 5.

PKCS #11 library

Pour désactiver la durabilité de la clé du client pour le SDK client 5 sous Linux

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Pour désactiver la durabilité de la clé client pour le SDK client 5 sous Windows

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-
availability-check
```

OpenSSL Dynamic Engine

Pour désactiver la durabilité de la clé du client pour le SDK client 5 sous Linux

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

JCE provider

Pour désactiver la durabilité de la clé du client pour le SDK client 5 sous Linux

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Pour désactiver la durabilité de la clé client pour le SDK client 5 sous Windows

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-
availability-check
```

CloudHSM CLI

Pour désactiver la durabilité de la clé du client pour le SDK client 5 sous Linux

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Pour désactiver la durabilité de la clé client pour le SDK client 5 sous Windows

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-
availability-check
```

Paramètres de durabilité des clés du SDK client 3

Dans le SDK client 3, la synchronisation des clés est principalement un processus automatique, mais vous pouvez utiliser les paramètres de durabilité des clés client pour améliorer la durabilité des clés. Vous spécifiez le nombre de HSM sur lesquels la création de clé doit réussir pour que l'opération globale soit considérée comme réussie. La synchronisation côté client s'efforce toujours de cloner les clés de chaque HSM du cluster, quel que soit le paramètre que vous choisissez. Votre paramètre impose la création de clés selon le nombre de HSM que vous spécifiez. Si vous spécifiez une valeur et que le système ne peut pas répliquer la clé sur ce nombre de HSM, le système nettoie automatiquement tout contenu de clé indésirable et vous pouvez réessayer.

> **Important**
>
> Si vous ne définissez pas les paramètres de durabilité des clés client (ou si vous utilisez la valeur par défaut de 1), vos clés risquent de se perdre. Si votre HSM actuel tombe en panne avant que le service côté serveur n'ait cloné cette clé sur un autre HSM, vous perdez le contenu de la clé.

Pour optimiser la durabilité des clés, pensez à spécifier au moins deux HSM pour la synchronisation côté client. N'oubliez pas que quel que soit le nombre de HSM que vous spécifiez, la charge de travail de votre cluster reste la même. La synchronisation côté client s'efforce toujours de cloner les clés de chaque HSM du cluster.

Recommandations

- Minimum : deux HSM par cluster
- Maximum : un de moins que le nombre total de HSM dans votre cluster

Si la synchronisation côté client échoue, le service client nettoie toutes les clés indésirables qui ont pu être créées et qui le sont désormais. Ce nettoyage est une solution au mieux qui ne fonctionne pas toujours. Si le nettoyage échoue, il se peut que vous deviez supprimer les éléments de clés indésirables. Pour de plus amples informations, veuillez consulter Échecs de synchronisation des clés.

Configuration du fichier de configuration pour la durabilité de la clé client

Pour définir les paramètres de durabilité de la clé client, vous devez modifier `cloudhsm_client.cfg`.

Pour modifier le fichier de configuration du client

1. Ouvrir `cloudhsm_client.cfg`.

   Linux :

   ```
   /opt/cloudhsm/etc/cloudhsm_client.cfg
   ```

   Windows:

   ```
   C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
   ```

2. Dans le `client` nœud du fichier, ajoutez `create_object_minimum_nodes` et spécifiez une valeur pour le nombre minimum de HSM sur lesquels les clés AWS CloudHSM doivent être créées avec succès pour que les opérations de création de clés réussissent.

   ```
   "create_object_minimum_nodes" : 2
   ```

   > **Note**
   >
   > L'outil de ligne de commande key_mgmt_util (KMU) dispose d'un paramètre supplémentaire pour la durabilité des clés client. Pour plus d'informations, consultez the section called "KMU et synchronisation côté client".

Référence de configuration

Voici les propriétés de synchronisation côté client, présentées dans un extrait du `cloudhsm_client.cfg` :

```
{
    "client": {
        "create_object_minimum_nodes" : 2,
        ...
    },
```

```
    ...
}
```

create_object_minimum_nodes

Spécifie le nombre minimum de HSM requis pour que les opérations de génération, d'importation ou de désencapsulage de clés soient considérées comme réussies. Si elle est défini, la valeur par défaut est 1. Cela signifie que pour chaque opération de création de clé, le service côté client tente de créer des clés sur chaque HSM du cluster, mais pour réussir, il suffit de créer une seule clé sur un HSM du cluster.

KMU et synchronisation côté client

Si vous créez des clés à l'aide de l'outil de ligne de commande key_mgmt_util (KMU), vous utilisez un paramètre de ligne de commande facultatif (`-min_srv`) pour limiter le nombre de HSM sur lesquels cloner des clés. Si vous spécifiez le paramètre de ligne de commande et une valeur dans le fichier de configuration, respectez AWS CloudHSM la plus grande des deux valeurs.

Pour plus d'informations, consultez les rubriques suivantes :

- GendSA KeyPair
- GèneCC KeyPair
- Genre RSA KeyPair
- genSymKey
- importPrivateKey
- importPubKey
- imSymKey
- insertMaskedObject
- unWrapKey

Synchronisation des clés entre les clusters clonés

La synchronisation côté client et côté serveur sert uniquement à synchroniser les clés au sein d'un même cluster. Si vous copiez la sauvegarde d'un cluster vers une autre région, vous pouvez utiliser la commande syncKey du cloudhsm-mgmt_util (CMU) pour synchroniser les clés entre les clusters.

Vous pouvez utiliser des clusters clonés pour assurer la redondance entre régions ou pour simplifier votre processus de reprise après sinistre. Pour plus d'informations, veuillez consulter syncKey.

Enveloppe de clés AES AWS CloudHSM

Cette rubrique décrit les options d'encapsulation des clés AES AWS CloudHSM. La fonction d'encapsulage de clés AES utilise une clé AES (clé d'encapsulage) pour encapsuler un autre type de clé (clé cible). L'encapsulage des clés permet de protéger les clés stockées ou de transmettre des clés sur des réseaux non sécurisés.

Rubriques

Algorithmes pris en charge

AWS CloudHSM propose trois options d'encapsulation des clés AES, chacune basée sur la manière dont la touche cible est rembourrée avant d'être enroulée. Le remplissage est effectué automatiquement, conformément à l'algorithme que vous utilisez, lorsque vous appelez l'encapsulage de la clé. Le tableau suivant répertorie les algorithmes pris en charge et les détails associés pour vous aider à choisir un mécanisme d'encapsulage approprié pour votre application.

Algorithme d'encapsulage des clés AES	Spécification de	Types de clé cibles pris en charge	Schéma de remplissage	AWS CloudHSM Disponibilité des clients
Encapsulage des clés AES avec remplissage à l'aide de zéros	RFC 5649 et SP 800 - 38F	Tous	Ajoute des zéros après les bits de clé, si nécessaire, pour bloquer l'alignement	SDK 3.1 et versions ultérieures
Encapsulage des clés AES sans remplissage	RFC 3394 et SP 800 - 38F	Clés avec blocage de l'alignement comme AES et 3DES	Aucun	SDK 3.1 et versions ultérieures

Algorithme d'encapsulage des clés AES	Spécification de	Types de clé cibles pris en charge	Schéma de remplissage	AWS CloudHSM Disponibilité des clients
Encapsulage des clés AES avec remplissage PKCS #5	Aucun	Tous	Au moins 8 octets sont ajoutés selon le schéma de remplissage PKCS #5 pour bloquer l'alignement	Tous

Pour découvrir comment utiliser, dans votre application, les algorithmes d'encapsulage des clés AES indiqués dans le tableau précédent, consultez Fonctionnement de l'encapsulage des clés AES dans AWS CloudHSM.

Présentation des vecteurs d'initialisation dans l'encapsulage des clés AES

Avant l'encapsulage, CloudHSM ajoute un vecteur d'initialisation (IV) à la clé cible pour l'intégrité des données. Chaque algorithme d'encapsulage de la clé est soumis à des restrictions spécifiques sur le type de vecteur d'initialisation autorisé. Pour installer l'IV AWS CloudHSM, deux options s'offrent à vous :

- Implicite : définir l'IV sur NULL pour que CloudHSM utilise la valeur par défaut de cet algorithme pour les opérations d'encapsulage et de désencapsulage (recommandé)
- Explicite : définir l'IV en passant la valeur de l'IV par défaut à la fonction d'encapsulage de la clé

> **Important**
>
> Vous devez comprendre quel vecteur d'initialisation vous utilisez dans votre application. Pour désencapsuler la clé, vous devez fournir le même IV que celui que vous avez utilisé pour l'encapsuler. Si vous utilisez un IV implicite pour l'encapsulage, un IV implicite est nécessaire pour le désencapsulage. Avec un IV implicite, CloudHSM utilise la valeur par défaut pour le désencapsulage.

Le tableau suivant décrit les valeurs autorisées pour les IV, spécifiées par l'algorithme d'encapsulage.

Algorithme d'encapsulage des clés AES	IV implicite	IV explicite
Encapsulage des clés AES avec remplissage à l'aide de zéros	Obligatoire Valeur par défaut : (IV calculé en interne sur la base de la spécification)	Non autorisée
Encapsulage des clés AES sans remplissage	Autorisée (recommandé) Valeur par défaut : `0xA6A6A6A6A6A6A6A6`	Autorisé Seule cette valeur est acceptée : `0xA6A6A6A6A6A6A6A6`
Encapsulage des clés AES avec remplissage PKCS #5	Autorisée (recommandé) Valeur par défaut : `0xA6A6A6A6A6A6A6A6`	Autorisé Seule cette valeur est acceptée : `0xA6A6A6A6A6A6A6A6`

Utilisation de l'encapsulation des touches AES AWS CloudHSM

Pour encapsuler ou désencapsuler des clés, procédez comme suit :

- Dans la bibliothèque PCKS #11, sélectionnez le mécanisme approprié pour les fonctions `C_WrapKey` et `C_UnWrapKey`, comme indiqué dans le tableau suivant.
- Dans le fournisseur JCE, sélectionnez l'algorithme, le mode et la combinaison de remplissage appropriés, en implémentant les méthodes de chiffrement `Cipher.WRAP_MODE` et `Cipher.UNWRAP_MODE`, comme indiqué dans le tableau suivant.
- Dans la CLI CloudHSM, choisissez l'algorithme approprié dans la liste étui pour clés des algorithmes déballage des clés pris en charge, comme indiqué dans le tableau suivant.
- Dans key_mgmt_util (KMU), utilisez les commandes wrapKey et unWrapKey avec les valeurs m appropriées, comme indiqué dans le tableau suivant.

Algorithme d'encapsulage des clés AES	Mécanisme PKCS #11	Méthode Java	Sous-commande CLI	Argument de l'utilitaire de gestion des clés (KMU)
Encapsulage des clés AES avec remplissage à l'aide de zéros	• `CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD` (mécanisme défini par le fournisseur)	`AESWrap/ECB/ZeroPadding`	aes-zero-pad	m = 6
Encapsulage des clés AES sans remplissage	• `CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD` (mécanisme défini par le fournisseur)	`AESWrap/ECB/NoPadding`	aes-no-pad	m = 5
Encapsulage des clés AES avec remplissage PKCS #5	• `CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD` (mécanisme défini par le fournisseur)	`AESWrap/ECB/PKCS5Padding`	aes-pkcs5-pad	m = 4

Utilisation de clés fiables dans AWS CloudHSM

AWS CloudHSM prend en charge l'encapsulation fiable des clés pour protéger les clés de données contre les menaces internes. Cette rubrique décrit comment créer des clés fiables pour sécuriser les données.

Rubriques

Présentation des clés fiables

Une clé fiable est une clé qui est utilisée pour encapsuler d'autres clés et que les administrateurs et les responsables de chiffrement (CO) identifient spécifiquement comme étant fiable à l'aide de l'attribut `CKA_TRUSTED`. En outre, les administrateurs et les responsables de chiffrement (CO) utilisent `CKA_UNWRAP_TEMPLATE` et des attributs associés pour spécifier les actions que les clés de données peuvent effectuer une fois qu'elles ont été désencapsulées par une clé fiable. Les clés de données désencapsulées par la clé fiable doivent également contenir ces attributs pour que l'opération de désencapsulage réussisse, ce qui permet de garantir que les clés de données désencapsulées ne sont autorisées que pour l'usage que vous souhaitez.

Utilisez l'attribut `CKA_WRAP_WITH_TRUSTED` pour identifier toutes les clés de données que vous souhaitez encapsuler avec des clés fiables. Cela vous permet de restreindre les clés de données afin que les applications ne puissent utiliser que des clés fiables pour les désencapsuler. Une fois que vous avez défini cet attribut sur les clés de données, il devient en lecture seule et vous ne pouvez pas le modifier. Lorsque ces attributs sont en place, les applications ne peuvent déballer vos clés de données qu'avec les clés auxquelles vous faites confiance, et les désencapsulages se traduisent toujours par des clés de données dotées d'attributs qui limitent la manière dont ces clés peuvent être utilisées.

Attributs de clés fiables

Les attributs suivants vous permettent de marquer une clé comme étant fiable, de spécifier qu'une clé de données ne peut être encapsulée et désencapsulée qu'avec une clé fiable, et de contrôler ce que peut faire une clé de données une fois qu'elle a été désencapsulée :

- `CKA_TRUSTED` : Appliquez cet attribut (en plus de `CKA_UNWRAP_TEMPLATE`) à la clé qui encapsulera les clés de données pour indiquer qu'un administrateur ou un responsable de chiffrement (CO) a fait preuve de la diligence nécessaire et qu'il fait confiance à cette clé. Seul un administrateur ou un CO peut définir `CKA_TRUSTED`. L'utilisateur de chiffrement (CU) possède la clé, mais seul un CO peut définir son attribut `CKA_TRUSTED`.

- `CKA_WRAP_WITH_TRUSTED` : Appliquez cet attribut à une clé de données exportable pour indiquer que vous ne pouvez encapsuler cette clé qu'avec des clés marquées comme `CKA_TRUSTED`. Une fois `CKA_WRAP_WITH_TRUSTED` défini sur true, l'attribut passe en lecture seule et vous ne pouvez ni le modifier ni le supprimer.
- `CKA_UNWRAP_TEMPLATE` : Appliquez cet attribut à la clé d'encapsulage (en plus de `CKA_TRUSTED`) pour spécifier les noms et valeurs d'attribut que le service doit automatiquement appliquer aux clés de données qu'il désencapsule. Lorsqu'une application soumet une clé pour désencapsulage, elle peut fournir également son propre modèle de désencapsulage. Si vous spécifiez un modèle de désencapsulage et que l'application fournit son propre modèle de désencapsulage, le HSM utilise les deux modèles pour appliquer des noms et des valeurs d'attribut à la clé. Toutefois, si une valeur dans le `CKA_UNWRAP_TEMPLATE` pour la clé d'encapsulage entre en conflit avec un attribut fourni par l'application lors de la demande de désencapsulage, cette dernière échoue.

Pour plus d'informations sur les attributs, veuillez consulter les rubriques suivantes :

- Attributs de clés PKCS #11
- Attributs de clés JCE
- Attributs de clés CLI CloudHSM

Comment utiliser des clés fiables pour encapsuler des clés de données

Pour utiliser une clé fiable pour encapsuler une clé de données, vous devez suivre trois étapes de base :

1. Pour la clé de données que vous prévoyez d'encapsuler avec une clé fiable, définissez son attribut `CKA_WRAP_WITH_TRUSTED` sur true.
2. Pour la clé fiable avec laquelle vous prévoyez d'encapsuler la clé de données, définissez son attribut `CKA_TRUSTED` sur true.
3. Utilisez la clé fiable pour encapsuler la clé de données.

Étape 1 : définir le **`CKA_WRAP_WITH_TRUSTED`** de la clé de données sur true

Pour la clé de données que vous souhaitez encapsuler, choisissez l'une des options suivantes pour définir l'attribut `CKA_WRAP_WITH_TRUSTED` de la clé sur true. Cela restreint la clé de données afin que les applications ne puissent utiliser que des clés fiables pour l'encapsuler.

Option 1 : si vous générez une nouvelle clé, définissez **`CKA_WRAP_WITH_TRUSTED`** sur true

Générez une clé à l'aide de PKCS #11, JCE ou CloudHSM CLI. Consultez les exemples suivants pour plus de détails.

PKCS #11

Pour générer une clé avec PKCS #11, vous devez définir l'attribut `CKA_WRAP_WITH_TRUSTED` de la clé sur true. Comme indiqué dans l'exemple suivant, procédez en incluant cet attribut dans le `CK_ATTRIBUTE template` de la clé puis en lui attribuant la valeur true :

```
CK_BYTE_PTR label = "test_key";
CK_ATTRIBUTE template[] = {
        {CKA_WRAP_WITH_TRUSTED, &true_val,         sizeof(CK_BBOOL)},
        {CKA_LABEL,             label,             strlen(label)},
        ...
};
```

Pour plus d'informations, veuillez consulter nos exemples publics illustrant la génération de clés avec PKCS #11.

JCE

Pour générer une clé avec JCE, vous devez définir l'attribut `WRAP_WITH_TRUSTED` de la clé sur true. Comme indiqué dans l'exemple suivant, procédez en incluant cet attribut dans le `KeyAttributesMap` de la clé puis en lui attribuant la valeur true :

```
final String label = "test_key";
final KeyAttributesMap keySpec = new KeyAttributesMap();
keySpec.put(KeyAttribute.WRAP_WITH_TRUSTED, true);
keySpec.put(KeyAttribute.LABEL, label);
...
```

Pour plus d'informations, veuillez consulter nos exemples publics illustrant la génération de clés avec JCE.

CloudHSM CLI

Pour générer une clé avec la CLI CloudHSM, vous devez définir l'attribut `wrap-with-trusted` de la clé sur true. Pour ce faire, incluez `wrap-with-trusted=true` dans l'argument approprié pour la commande de génération de clés :

- Pour les clés symétriques, ajoutez `wrap-with-trusted` à l'argument `attributes`.

- Pour les clés publiques, ajoutez `wrap-with-trusted` à l'argument `public-attributes`.
- Pour les clés privées, ajoutez `wrap-with-trusted` à l'argument `private-attributes`.

Pour de plus amples informations sur la génération de paires de clés, veuillez consulter clé generate-asymmetric-pair .

Pour plus d'informations sur la génération de clés symétriques, veuillez consulter Clé generate-symmetric .

Option 2 : Si vous utilisez une clé existante, utilisez la CLI CloudHSM pour la définir son **`CKA_WRAP_WITH_TRUSTED`** sur true

Pour attribuer la valeur true à l'attribut `CKA_WRAP_WITH_TRUSTED` d'une clé existante, procédez comme suit :

1. Utilisez la commande login pour vous connecter en tant qu'utilisateur de chiffrement (CU).
2. Utilisez la commande clé set-attribute pour définir l'attribut `wrap-with-trusted` de la clé sur 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"
  }
}
```

Étape 2 : définir le **`CKA_TRUSTED`** de la clé fiable sur true

Pour qu'une clé soit une clé fiable, son attribut `CKA_TRUSTED` doit être défini sur true. Pour ce faire, vous pouvez utiliser la CLI CloudHSM ou l'Utilitaire de gestion CloudHSM (CMU).

- Si vous utilisez la CLI CloudHSM pour définir l'attribut `CKA_TRUSTED` d'une clé, veuillez consulter Comment marquer une clé comme fiable avec la CLI CloudHSM.
- Si vous utilisez la CMU pour définir l'attribut `CKA_TRUSTED` d'une clé, veuillez consulter Comment marquer une clé comme fiable auprès de l'utilitaire CMU.

Étape 3. Utilisez la clé fiable pour encapsuler la clé de données

Pour associer la clé de données référencée à l'étape 1 à la clé fiable que vous avez définie à l'étape 2, veuillez consulter les liens suivants pour obtenir des exemples de code. Chacun montre comment encapsuler les clés.

- AWS CloudHSM Exemples de PKCS #11
- AWS CloudHSM Exemples JCE

Comment désencapsuler une clé de données avec une clé fiable

Pour désencapsuler une clé de données, vous avez besoin qu'une clé fiable ait `CKA_UNWRAP` défini sur true. Pour être une telle clé, elle doit également répondre aux critères suivants :

- L'attribut `CKA_TRUSTED` de la clé doit être défini sur true.
- La clé doit utiliser des attributs `CKA_UNWRAP_TEMPLATE` et associés pour spécifier les actions que les clés de données peuvent effectuer une fois qu'elles sont désencapsulées. Si, par exemple, vous souhaitez qu'une clé désencapsulée ne soit pas exportable, vous définissez `CKA_EXPORTABLE = FALSE` dans le cadre de `CKA_UNWRAP_TEMPLATE`.

> **Note**
>
> `CKA_UNWRAP_TEMPLATE` n'est disponible qu'avec PKCS #11.

Lorsqu'une application soumet une clé pour désencapsulage, elle peut fournir également son propre modèle de désencapsulage. Si vous spécifiez un modèle de désencapsulage et que l'application fournit son propre modèle de désencapsulage, le HSM utilise les deux modèles pour appliquer des noms et des valeurs d'attribut à la clé. Toutefois, si, au cours de la demande de désencapsulage, une valeur du `CKA_UNWRAP_TEMPLATE` de la clé fiable entre en conflit avec un attribut fourni par l'application, la demande de désencapsulage échoue.

Pour voir un exemple de désencapsulage d'une clé de données avec une clé fiable, reportez-vous à cet exemple PKCS #11.

Gestion des clés à l'aide de la CLI CloudHSM

Si vous utilisez la dernière série de versions de SDK, utilisez la CLI CloudHSM pour gérer les clés de votre cluster. AWS CloudHSM Pour en savoir plus, consultez les rubriques ci-dessous.

- L'utilisation de clés fiables décrit comment utiliser la CLI CloudHSM pour créer des clés fiables afin de sécuriser les données.
- La génération de clés inclut des instructions sur la création de clés, notamment des clés symétriques, des clés RSA et des clés EC.
- La suppression de clés décrit la manière dont les propriétaires de clés suppriment les clés.
- Le partage et l'annulation du partage des clés expliquent comment les propriétaires des clés partagent et annule le partage des clés.
- Le filtrage des clés fournit des instructions sur la façon d'utiliser les filtres pour rechercher des clés.

Utilisation de la CLI CloudHSM pour générer des clés

Avant de générer une clé, vous devez démarrer la CLI CloudHSM et vous connecter en tant qu'utilisateur de chiffrement (CU). Pour générer des clés sur le HSM, utilisez la commande qui correspond au type de clé que vous souhaitez générer.

Rubriques

- Générer des clés symétriques
- Générer des clés asymétriques
- Rubriques en relation

Générer des clés symétriques

Utilisez les commandes répertoriées dans Clé generate-symmetric pour générer des clés symétriques. Pour consulter toutes les options disponibles, utilisez la commande help key generate-symmetric.

Générer une clé AES

Utilisez la commande key generate-symmetric aes pour générer des clés AES. Pour consulter toutes les options disponibles, utilisez la commande help key generate-symmetric aes.

Example

L'exemple suivant génère une clé AES de 32 octets.

```
aws-cloudhsm > key generate-symmetric aes \
    --label aes-example \
    --key-length-bytes 32
```

Arguments

<LABEL>

Spécifie l'étiquette définie par l'utilisateur pour la clé AES.

Obligatoire : oui

<KEY-LENGTH-BYTES>

Spécifie la taille de la clé en octets.

Valeurs valides :

- 16, 24 et 32

Obligatoire : oui

<KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs de clés à définir pour la clé AES générée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple, `token=true`)

Pour obtenir la liste des attributs AWS CloudHSM clés pris en charge, consultezAttributs de clé de la CLI CloudHSM.

Obligatoire : non

<SESSION>

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée. Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Pour remplacer une clé de session par une clé persistante (jeton), utilisez key set-attribute.

Par défaut, lorsque des clés sont générées, il s'agit de clés persistantes/de jeton. L'utilisation de <SESSION> modifie cela, pour garantir qu'une clé générée avec cet argument est une session/éphémère

Obligatoire : non

Générer une clé secrète générique

Utilisez la commande key generate-symmetric generic-secret pour générer des clés secrètes génériques. Pour consulter toutes les options disponibles, utilisez la commande help key generate-symmetric generic-secret.

Example

L'exemple suivant génère une clé secrète générique de 32 octets.

```
aws-cloudhsm > key generate-symmetric generic-secret \
    --label generic-secret-example \
    --key-length-bytes 32
```

Arguments

<LABEL>

Spécifie une étiquette définie par l'utilisateur pour la clé secrète générique.

Obligatoire : oui

<KEY-LENGTH-BYTES>

Spécifie la taille de la clé en octets.

Valeurs valides :

- 1 à 800

Obligatoire : oui

<KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs de clés à définir pour la clé secrète générique générée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple, `token=true`)

Pour obtenir la liste des attributs AWS CloudHSM clés pris en charge, consultezAttributs de clé de la CLI CloudHSM.

Obligatoire : non

<SESSION>

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée. Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Pour remplacer une clé de session par une clé persistante (jeton), utilisez key set-attribute.

Par défaut, lorsque des clés sont générées, il s'agit de clés persistantes/de jeton. L'utilisation de <SESSION> modifie cela, pour garantir qu'une clé générée avec cet argument est une session/ éphémère

Obligatoire : non

Générer des clés asymétriques

Utilisez les commandes répertoriées dans clé generate-asymmetric-pair pour générer des paires de clés asymétriques.

Générer une clé RSA

Utilisez la commande key generate-asymmetric-pair rsa pour générer une paire de clés RSA. Pour consulter toutes les options disponibles, utilisez la commande help key generate-asymmetric-pair rsa.

Example

L'exemple suivant génère une paire de clés RSA 2048 bits.

```
aws-cloudhsm > key generate-asymmetric-pair rsa \
    --public-exponent 65537 \
    --modulus-size-bits 2048 \
    --public-label rsa-public-example \
    --private-label rsa-private-example
```

Arguments

<PUBLIC_LABEL>

Spécifie une étiquette définie par l'utilisateur pour la clé publique.

Obligatoire : oui

<PRIVATE_LABEL>

Spécifie l'étiquette définie par l'utilisateur pour la clé privée.

Obligatoire : oui

<MODULUS_SIZE_BITS>

Indique la longueur du module en bits. La valeur minimale est de 2048.

Obligatoire : oui

<PUBLIC_EXPONENT>

Spécifie l' exposant public. La valeur doit être un entier impair supérieur ou égal à 65 537.

Obligatoire : oui

<PUBLIC_KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs de clés à définir pour la clé publique RSA générée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple, `token=true`).

Pour obtenir la liste des attributs AWS CloudHSM clés pris en charge, consultezAttributs de clé de la CLI CloudHSM.

Obligatoire : non

<SESSION>

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée. Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Pour remplacer une clé de session par une clé persistante (jeton), utilisez key set-attribute.

Par défaut, lorsque des clés sont générées, il s'agit de clés persistantes/de jeton. L'utilisation de <SESSION> modifie cela, pour garantir qu'une clé générée avec cet argument est une session/éphémère

Obligatoire : non

Générez des paires de clés EC (Elliptic Curve Cryptography)

Utilisez la commande key generate-asymmetric-pair ec pour générer une paire de clés EC. Pour voir toutes les options disponibles, notamment une liste des courbes elliptiques prises en charge, utilisez la commande help key generate-asymmetric-pair ec.

Example

L'exemple suivant génère une paire de clés EC à l'aide de la courbe elliptique Secp384r1.

```
aws-cloudhsm > key generate-asymmetric-pair ec \
    --curve secp384r1 \
    --public-label ec-public-example \
    --private-label ec-private-example
```

Arguments

<PUBLIC_LABEL>

Spécifie une étiquette définie par l'utilisateur pour la clé publique. La taille maximale autorisée `label` est de 127 caractères pour le SDK client 5.11 et versions ultérieures. Le SDK client 5.10 et les versions antérieures sont limités à 126 caractères.

Obligatoire : oui

<PRIVATE_LABEL>

Spécifie l'étiquette définie par l'utilisateur pour la clé privée. La taille maximale autorisée `label` est de 127 caractères pour le SDK client 5.11 et versions ultérieures. Le SDK client 5.10 et les versions antérieures sont limités à 126 caractères.

Obligatoire : oui

<CURVE>

Spécifie l'identifiant de la courbe elliptique.

Valeurs valides :

- prime256v1
- secp256r1
- secp224r1
- secp384r1
- secp256k1
- secp521r1

Obligatoire : oui

<PUBLIC_KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs de clés à définir pour la clé publique EC générée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple, `token=true`).

Pour obtenir la liste des attributs AWS CloudHSM clés pris en charge, consultezAttributs de clé de la CLI CloudHSM.

Obligatoire : non

<PRIVATE_KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs de clés à définir pour la clé privée EC générée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple, `token=true`).

Pour obtenir la liste des attributs AWS CloudHSM clés pris en charge, consultezAttributs de clé de la CLI CloudHSM.

Obligatoire : non

<SESSION>

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée. Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Pour remplacer une clé de session par une clé persistante (jeton), utilisez key set-attribute.

Par défaut, les clés générées sont des clés persistantes (jetons). La transmission vers <SESSION> change cela, garantissant qu'une clé générée avec cet argument est une clé de session (éphémère).

Obligatoire : non

Rubriques en relation

- Attributs de clé de la CLI CloudHSM
- clé generate-asymmetric-pair
- Clé generate-symmetric

Utilisation de la CLI CloudHSM pour supprimer des clés

Utilisez l'exemple présenté dans cette rubrique pour supprimer une clé à l'aide de la CLI CloudHSM. Seul le propriétaire de la clé peut la supprimer.

Rubriques

- Exemple : Supprimer une clé
- Rubriques en relation

Exemple : Supprimer une clé

1. Exécutez la commande key list pour identifier la clé que vous voulez supprimer :

```
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. Après avoir identifié la clé, exécutez la commande key delete avec l'attribut unique `label` de la clé pour supprimer la clé :

```
aws-cloudhsm > key delete --filter attr.label="my_key_to_delete"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

3. Exécutez la commande key list avec l'attribut unique `label` de la clé et confirmez que la clé a été supprimée. Comme le montre l'exemple suivant, aucune clé portant l'étiquette `my_key_to_delete` ne se trouve dans le cluster HSM :

```
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
  }
}
```

Rubriques en relation

- Attributs de clé de la CLI CloudHSM
- supprimer une clé

Utilisation de la CLI CloudHSM pour partager et annuler le partage de clés

Utilisez les commandes de cette rubrique pour partager ou annuler le partage de clés dans la CLI CloudHSM. Dans AWS CloudHSM, l'utilisateur cryptographique (CU) qui crée la clé en est propriétaire. Le propriétaire peut utiliser les commandes key share et key unshare pour partager ou annuler le partage de la clé avec d'autres CU. Les utilisateurs avec lesquels la clé est partagée peuvent utiliser la clé dans des opérations de chiffrement, mais ils ne peuvent pas l'exporter, la supprimer ni la partager avec d'autres utilisateurs.

Avant de pouvoir partager une clé, vous devez vous connecter au HSM en tant qu'utilisateur de chiffrement (CU) qui possède la clé.

Rubriques

- Exemple : Partage et annulation du partage d'une clé
- Rubriques en relation

Exemple : Partage et annulation du partage d'une clé

Example

L'exemple suivant montre comment partager et annuler le partage d'une clé avec un utilisateur de chiffrement (CU) `alice`. Outre les commandes key unshare et key share, les commandes de partage et d'annulation du partage nécessitent également une clé spécifique utilisant les filtres de clé de la CLI CloudHSM et le nom d'utilisateur spécifique de l'utilisateur avec lequel la clé sera partagée ou non partagée.

1. Commencez par exécuter la commande key list avec un filtre pour renvoyer une clé spécifique et voir avec qui la clé est déjà partagée.

```
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. Consultez le résultat `shared-users` pour identifier avec qui la clé est actuellement partagée.
3. Pour partager cette clé avec un utilisateur de chiffrement (CU)`alice`, entrez la commande suivante :

```
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"
  }
}
```

Notez qu'en plus de la commande key share, cette commande utilise l'étiquette unique de la clé et le nom de l'utilisateur avec lequel la clé sera partagée.

4. Exécutez la commande key list pour confirmer que la clé a été partagée avec `alice` :

```
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. Pour annuler le partage de la même clé avec `alice`, exécutez la commande unshare suivante :

```
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"
  }
}
```

   Notez qu'en plus de la commande key unshare, cette commande utilise l'étiquette unique de la clé et le nom de l'utilisateur avec lequel la clé sera partagée.

6. Exécutez à nouveau la commande key list et confirmez que la clé n'a pas été partagée avec l'utilisateur de chiffrement `alice` :

```
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
  }
}
```


Rubriques en relation

- Attributs de clé de la CLI CloudHSM
- partage de clés
- annuler le partage d'une clé
- Utilisation de la CLI CloudHSM pour filtrer les clés

Utilisation de la CLI CloudHSM pour filtrer les clés

Utilisez les commandes de clé suivantes pour utiliser les mécanismes de filtration de clés standardisés pour la CLI CloudHSM.

- key list
- key delete
- key share
- key unshare
- key set-attribute

Pour sélectionner et/ou filtrer les clés à l'aide de la CLI CloudHSM, les commandes de clé utilisent un mécanisme de filtration standardisé basé sur Attributs de clé de la CLI CloudHSM. Une touche ou un jeu de touches peut être spécifié dans les raccourcis clavier à l'aide d'un ou de plusieurs AWS CloudHSM attributs permettant d'identifier une ou plusieurs touches. Le mécanisme de filtrage des clés fonctionne uniquement sur les clés que l'utilisateur actuellement connecté possède et partage, ainsi que sur toutes les clés publiques du AWS CloudHSM cluster.

Rubriques

Prérequis

Pour filtrer les clés, vous devez être connecté en tant qu'utilisateur de chiffrement (CU).

Filtrer pour trouver une seule clé

Notez que dans les exemples suivants, chaque attribut utilisé comme filtre doit être écrit sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE`. Par exemple, si vous souhaitez filtrer en fonction de l'attribut d'étiquette, vous allez écrire `attr.label=my_label`.

Example Utilisez un seul attribut pour trouver une seule clé

Cet exemple montre comment filtrer en fonction d'une seule clé unique en utilisant un seul attribut d'identification.

```
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 Utilisez plusieurs attributs pour trouver une seule clé

L'exemple suivant montre comment trouver une clé unique à l'aide de plusieurs attributs de clé.

```
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 Filtrer pour trouver un jeu de clés

L'exemple suivant montre comment filtrer pour trouver un ensemble de clés RSA privées.

```
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
  }
}
```

Erreurs de filtration

Certaines opérations de clés ne peuvent être effectuées que sur une seule clé à la fois. Pour ces opérations, la CLI CloudHSM génère une erreur si les critères de filtration ne sont pas suffisamment affinés et si plusieurs clés correspondent aux critères. Un exemple de ce type est illustré ci-dessous avec la suppression de clé.

Example Erreur de filtration en cas de correspondance avec un trop grand nombre de clés

```
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."
}
```

Rubriques en relation

- Attributs de clé de la CLI CloudHSM

Comment marquer une clé comme fiable avec la CLI CloudHSM

Le contenu de cette section fournit des instructions sur l'utilisation de la CLI CloudHSM pour marquer une clé comme étant fiable.

1. À l'aide de la commande CLI CloudHSM login, connectez-vous en tant qu'utilisateur de chiffrement(CU).
2. Utilisez la commande key list pour identifier la référence de clé de la clé que vous souhaitez marquer comme fiable. L'exemple de code répertorie la clé avec l'étiquette `key_to_be_trusted`.

```
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. À l'aide de la commande logout, déconnectez-vous en tant qu'utilisateur de chiffrement (CU).
4. À l'aide de la commande login, connectez-vous en tant qu'administrateur.
5. À l'aide de la commande key set-attribute avec la référence de clé que vous avez identifiée à l'étape 2, définissez la valeur fiable de la clé sur true :

```
aws-cloudhsm > key set-attribute --filter key-reference=<Key Reference> --name
 trusted --value true
{
  "error_code": 0,
  "data": {
    "message": "Attribute set successfully"
  }
```

```
}
```

Gestion des clés avec la KMU et la CMU

Si vous utilisez la dernière série de versions de SDK, utilisez la CLI CloudHSM pour gérer les clés de votre cluster. AWS CloudHSM

Si vous utilisez la série de versions précédentes du SDK, vous pouvez gérer les clés des HSM de votre AWS CloudHSM cluster à l'aide de l'outil de ligne de commande key_mgmt_util. Avant de pouvoir gérer les clés, vous devez démarrer le AWS CloudHSM client, démarrer key_mgmt_util et vous connecter aux HSM. Pour plus d'informations, veuillez consulter Démarrer avec key_mgmt_util.

- L'utilisation de clés fiables décrit comment utiliser les attributs de bibliothèque PKCS #11 et la CMU pour créer des clés fiables afin de sécuriser les données.
- La génération de clés contient des instructions sur la génération de clés, notamment des clés symétriques, des clés RSA et des clés EC.
- L'importation de clés fournit des informations sur la manière dont les propriétaires de clés importent les clés.
- L'exportation de clés fournit des informations sur la manière dont les propriétaires de clés exportent les clés.
- La suppression de clés fournit des informations sur la manière dont les propriétaires de clés suppriment les clés.
- Le partage et l'annulation du partage des clés expliquent comment les propriétaires des clés partagent et annule le partage des clés.

Génération de clés

Pour générer des clés sur le HSM, utilisez la commande qui correspond au type de clé que vous souhaitez générer.

Rubriques

- Générer des clés symétriques
- Générer des paire de clés RSA
- Générer des paires de clés ECC (Elliptic Curve Cryptography)

Générer des clés symétriques

Utilisez la genSymKeycommande pour générer des clés AES et d'autres types de clés symétriques. Pour consulter toutes les options disponibles, utilisez la commande genSymKey -h.

L'exemple suivant crée une clé AES 256 bits.

```
Command: genSymKey -t 31 -s 32 -l aes256
CfmGenerateSymmetricKey returned: 0x00 : HSM Return: SUCCESS

Symmetric Key Created.  Key Handle: 524295

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Générer des paire de clés RSA

Pour générer une paire de clés RSA, utilisez la commande KeyPairGenRSA. Pour consulter toutes les options disponibles, utilisez la commande genRSAKeyPair -h.

L'exemple suivant génère une paire de clés RSA 2048 bits.

```
Command: genRSAKeyPair -m 2048 -e 65537 -l rsa2048
CfmGenerateKeyPair returned: 0x00 : HSM Return: SUCCESS

CfmGenerateKeyPair:    public key handle: 524294    private key handle: 524296

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Générer des paires de clés ECC (Elliptic Curve Cryptography)

Pour générer une paire de clés ECC, utilisez la commande KeyPairGeneCC. Pour voir toutes les options disponibles, notamment une liste des courbes elliptiques prises en charge, utilisez la commande genECCKeyPair -h.

L'exemple suivant génère une paire de clés ECC à l'aide de la courbe elliptique P-384 définie dans la publication NIST FIPS 186-4.

```
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
```

Importation de clés

Pour importer des clés secrètes—c'est-à-dire des clés symétriques et clés privées asymétriques—dans le HSM, vous devez d'abord créer une clé d'encapsulage sur le HSM. Vous pouvez importer des clés publiques directement sans clé d'encapsulage.

Rubriques

- Importer des clés secrètes
- Importer des clés publiques

Importer des clés secrètes

Effectuez les étapes suivantes pour importer une clé secrète. Avant d'importer une clé secrète, enregistrez-la dans un fichier. Enregistrez les clés symétriques sous la forme d'octets bruts et clés privées asymétriques au format PEM.

Cet exemple montre comment importer une clé secrète en texte brut à partir d'un fichier dans le HSM. Pour importer une clé chiffrée d'un fichier dans le HSM, utilisez la unWrapKeycommande.

Pour importer une clé secrète

1. Utilisez la genSymKeycommande pour créer une clé d'encapsulation. La commande suivante crée une clé d'encapsulage AES 128 bits valable uniquement pour la session en cours. Vous pouvez utiliser une clé de session ou une clé persistante en tant que clé d'encapsulage.

   ```
   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. Utilisez l'une des commandes suivantes, selon le type de clé secrète que vous importez.

   - Pour importer une clé symétrique, utilisez la imSymKeycommande. La commande suivante importe une clé AES depuis un fichier nommé `aes256.key` à l'aide de la clé d'encapsulage créée à l'étape précédente. Pour consulter toutes les options disponibles, utilisez la commande imSymKey -h.

     ```
     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
     ```

   - Pour importer une clé privée asymétrique, utilisez la importPrivateKeycommande. La commande suivante importe une clé privée depuis un fichier nommé `rsa2048.key` à l'aide de la clé d'encapsulage créée à l'étape précédente. Pour consulter toutes les options disponibles, utilisez la commande importPrivateKey -h.

     ```
     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
```

Importer des clés publiques

Utilisez la importPubKeycommande pour importer une clé publique. Pour consulter toutes les options disponibles, utilisez la commande importPubKey -h.

L'exemple suivant importe une clé publique RSA depuis un fichier nommé `rsa2048.pub`.

```
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
```

Exportation de clés

Pour exporter des clés secrètes—c'est-à-dire des clés symétriques et clés privées asymétriques—à partir du HSM, vous devez d'abord créer une clé d'encapsulage. Vous pouvez exporter des clés publiques directement sans clé d'encapsulage.

Seul le propriétaire de la clé peut l'exporter. Les utilisateurs avec lesquels la clé est partagée peuvent utiliser la clé dans des opérations de chiffrement, mais ils ne peuvent pas l'exporter. Lorsque vous exécutez cet exemple, veillez à exporter une clé que vous avez créée.

> **Important**
>
> La exSymKeycommande écrit une copie en texte clair (non chiffrée) de la clé secrète dans un fichier. Le processus d'exportation nécessite une clé d'encapsulage, mais la clé figurant dans le fichier n'est pas une clé encapsulée. Pour exporter une copie encapsulée (chiffrée) d'une clé, utilisez la commande wrapKey.

Rubriques

- Exporter des clés secrètes

- Exporter des clés publiques

Exporter des clés secrètes

Effectuez les étapes suivantes pour exporter une clé secrète.

Pour exporter une clé secrète

1. Utilisez la genSymKeycommande pour créer une clé d'encapsulation. La commande suivante crée une clé d'encapsulage AES 128 bits valable uniquement pour la session en cours.

```
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. Utilisez l'une des commandes suivantes, selon le type de clé secrète que vous exportez.

   - Pour exporter une clé symétrique, utilisez la exSymKeycommande. La commande suivante exporte une clé AES vers un fichier nommé `aes256.key.exp`. Pour consulter toutes les options disponibles, utilisez la commande exSymKey -h.

```
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**
>
> La sortie de la commande indique qu'une « clé symétrique encapsulée » est écrite dans le fichier de sortie. Toutefois, le fichier de sortie contient une clé en texte brut (non encapsulée). Pour exporter une clé encapsulée (chiffrée) dans un fichier, utilisez la commande wrapKey.

- Pour exporter une clé privée, utilisez la commande exportPrivateKey. La commande suivante exporte une clé privée vers un fichier nommé `rsa2048.key.exp`. Pour consulter toutes les options disponibles, utilisez la commande exportPrivateKey -h.

```
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
```

Exporter des clés publiques

Utilisez la commande exportPubKey pour exporter une clé publique. Pour consulter toutes les options disponibles, utilisez la commande exportPubKey -h.

L'exemple suivant exporte une clé publique RSA vers un fichier nommé `rsa2048.pub.exp`.

```
Command: exportPubKey -k 524294 -out rsa2048.pub.exp
PEM formatted public key is written to rsa2048.pub.key

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

Suppression des clés

Utilisez la commande deleteKey pour supprimer une clé, comme dans l'exemple suivant. Seul le propriétaire de la clé peut la supprimer.

```
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
```

Partage et annulation du partage des clés

Dans AWS CloudHSM, le CU qui crée la clé en est propriétaire. Le propriétaire gère la clé, peut l'exporter et la supprimer, et peut utiliser la clé dans des opérations de chiffrement. Le propriétaire

peut également partager la clé avec d'autres utilisateurs de chiffrement. Les utilisateurs avec lesquels la clé est partagée peuvent utiliser la clé dans des opérations de chiffrement, mais ils ne peuvent pas l'exporter, la supprimer ni la partager avec d'autres utilisateurs.

Vous pouvez partager des clés avec d'autres utilisateurs du CU lorsque vous créez la clé, par exemple en utilisant le `-u` paramètre des commandes genSymKeyou genRSA KeyPair. Pour partager des clés existantes avec un autre utilisateur HSM, utilisez l'outil de ligne de commande cloudhsm_mgmt_util. Cette tâche diffère de la plupart des tâches documentées dans cette section, qui utilisent l'outil de ligne de commande key_mgmt_util.

Avant de partager une clé, vous devez démarrer cloudhsm-mgmt_util, activer le end-to-end chiffrement et vous connecter aux HSM. Pour partager une clé, connectez-vous au HSM en tant qu'utilisateur de chiffrement (CU) qui possède la clé. Seuls les propriétaires de clé peuvent partager une clé.

Utilisez la commande shareKey pour partager une clé ou en annuler le partage, en spécifiant le descripteur de la clé et l'ID du ou des utilisateurs. Pour partager une clé avec plusieurs utilisateurs ou en annuler le partage, saisissez les ID de ces utilisateurs en les séparant par une virgule. Pour partager une clé, utilisez 1 comme dernier paramètre de commande, comme dans l'exemple suivant. Pour annuler le partage, utilisez `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)
```

L'exemple suivant montre la syntaxe de la commande shareKey.

```
aws-cloudhsm>shareKey <key handle> <user ID> <Boolean: 1 for share, 0 for unshare>
```

Comment marquer une clé comme fiable auprès de l'utilitaire CMU

Le contenu de cette section fournit des instructions sur l'utilisation de la CMU pour marquer une clé comme fiable.

1. À l'aide de la commande loginHSM, connectez-vous en tant que responsable de chiffrement (CO).
2. Utilisez la commande setAttribute avec `OBJ_ATTR_TRUSTED` (valeur `134`) défini sur true (`1`).

```
setAttribute <Key Handle> 134 1
```

Gestion des clusters clonés

Utilisez l'Utilitaire de gestion CloudHSM (CMU) pour synchroniser un cluster dans une région distante, si le cluster de cette région a été créé à l'origine à partir de la sauvegarde d'un cluster dans une autre région. Supposons que vous ayez copié un cluster vers une autre région (destination) et que vous souhaitiez ensuite synchroniser les modifications depuis le cluster d'origine (source). Dans de tels scénarios, vous utilisez l'utilitaire CMU pour synchroniser les clusters. Pour ce faire, créez un nouveau fichier de configuration CMU, spécifiez les modules de sécurité matériels (HSM) des deux clusters dans le nouveau fichier, puis utilisez l'utilitaire CMU pour vous connecter au cluster avec ce fichier.

Pour utiliser l'utilitaire CMU sur des clusters clonés

1. Créez une copie de votre fichier de configuration actuel et remplacez le nom de la copie par un autre nom.

   Par exemple, utilisez les emplacements de fichiers suivants pour localiser et créer une copie de votre fichier de configuration actuel, puis remplacez le nom de la copie `cloudhsm_mgmt_config.cfg` par `syncConfig.cfg`.

   - Linux : `/opt/cloudhsm/etc/cloudhsm_mgmt_config.cfg`
   - Windows: `C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_mgmt_config.cfg`
2. Dans la copie renommée, ajoutez l'adresse IP ENI (Elastic Network Interface) du HSM de destination (le HSM de la région étrangère qui doit être synchronisé). Nous vous recommandons d'ajouter le HSM de destination sous le HSM source.

```
{
    ...
    "servers": [
        {
            ...
            "hostname": "<ENI Source IP>",
            ...
        },
        {
            ...
            "hostname": "<ENI Destination IP>",
            ...
        }
    ]
}
```

Pour en savoir plus sur la manière d'obtenir l'adresse IP, consultez the section called "Obtenir une adresse IP pour un HSM".

3. Initialisez l'utilitaire CMU avec le nouveau fichier de configuration :

   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. Vérifiez les messages de statut retournés pour vous assurer que l'utilitaire CMU est connecté à tous les HSM souhaités et déterminez quelle adresse IP ENI renvoyée correspond à chaque cluster. Utilisez SyncUser et SyncKey pour synchroniser manuellement les utilisateurs et les clés. Pour plus d'informations, consultez SyncUser et SyncKey.

Obtenir une adresse IP pour un HSM

Utilisez cette section pour obtenir une adresse IP pour un HSM.

Pour obtenir une adresse IP pour un HSM (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Pour changer de région AWS, utilisez le Sélecteur de région dans l'angle supérieur droit de la page.
3. Pour ouvrir la page détaillée du cluster, choisissez l'ID du cluster dans le tableau des clusters.
4. Pour obtenir l'adresse IP, dans l'onglet HSM, choisissez l'une des adresses IP répertoriées sous Adresse IP ENI.

Pour obtenir une adresse IP pour un HSM (CLI)

- Obtenez l'adresse IP d'un HSM à l'aide de la describe-clusters commande de la CLI. Dans la sortie de la commande, l'adresse IP des HSM sont les valeurs de `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


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

Rubriques en relation

- syncUser
- syncKey
- Copie d'une sauvegarde dans plusieurs régions

AWS CloudHSM outils de ligne de commande

Cette rubrique décrit les outils de ligne de commande disponibles pour la gestion et l'utilisation d' AWS CloudHSM.

Rubriques

Comprendre les outils de ligne de commande

Outre l'interface de ligne de commande (CLI) AWS que vous utilisez pour gérer vos ressources AWS, elle AWS CloudHSM propose des outils de ligne de commande pour créer et gérer des utilisateurs HSM et des clés sur vos HSM. AWS CloudHSM Vous utilisez la CLI habituelle pour gérer votre cluster et les outils de ligne de commande CloudHSM pour gérer votre HSM.

Voici les différents outils de ligne de commande :

Pour gérer les HSM et les clusters

Commandes CloudHSMv2 dans la CLI et applets de commande PowerShell HSM2 dans le module AWSPowerShell

- Ces outils obtiennent, créent, suppriment et balisent des AWS CloudHSM clusters et des HSM :
- Pour utiliser les commandes contenues dans les commandes CloudHSMv2 de la CLI, vous devez installer et configurer la CLI.
- Les PowerShell applets de commande HSM2 du AWSPowerShell module sont disponibles dans un module Windows et un PowerShell module Core multiplateforme. PowerShell

Pour gérer des utilisateurs HSM

CLI CloudHSM

- Utilisez la CLI CloudHSM pour créer, supprimer et répertorier des utilisateurs, ainsi que pour modifier les mots de passe des utilisateurs et mettre à jour l'authentification multifactorielle (MFA). Elle n'est pas incluse dans le logiciel client AWS CloudHSM. Pour obtenir des conseils sur l'installation de cet outil, consultez la section Installation et configuration de la CLI CloudHSM.

Outils d'assistance

Deux outils vous aident à utiliser les AWS CloudHSM outils et les bibliothèques de logiciels :

- Configurez les mises à jour de vos fichiers de configuration client CloudHSM à l'aide de l'outil de configuration. Cela permet AWS CloudHSM de synchroniser les HSM dans un cluster.

  AWS CloudHSM propose deux versions principales, et le SDK client 5 est la plus récente. Il offre de nombreux avantages par rapport au SDK client 3 (la série précédente).
- pkpspeed mesure la performance de votre matériel HSM indépendamment des bibliothèques logicielles.

Outils pour les SDK précédents

Utilisez l'outil de gestion des clés (KMU) pour créer, supprimer, importer et exporter des clés symétriques et des paires de clés asymétriques :

- key_mgmt_util. Cet outil est inclus dans le logiciel client AWS CloudHSM .

Utilisez l'outil de gestion CloudHSM (CMU) pour créer et supprimer des utilisateurs HSM, notamment en implémentant l'authentification par quorum pour les tâches de gestion des utilisateurs

- cloudhsm_mgmt_util. Cet outil est inclus dans le logiciel client AWS CloudHSM .

Outil de configuration

AWS CloudHSM synchronise automatiquement les données entre tous les modules de sécurité matériels (HSM) d'un cluster. L'outil configure met à jour les données HSM dans les fichiers de

configuration utilisés par les mécanismes de synchronisation. Utilisez configure pour actualiser les données HSM avant d'utiliser les outils de ligne de commande, en particulier lorsque les HSM du cluster ont changé.

AWS CloudHSM inclut deux versions principales du SDK client :

- SDK client 5 : il s'agit de notre dernier SDK client par défaut. Pour plus d'informations sur ses avantages, consultez Avantages du SDK client 5.
- SDK client 3 : il s'agit de notre ancien SDK client. Il inclut un ensemble complet de composants pour la compatibilité des applications basées sur les plateformes et les langages ainsi que des outils de gestion.

Pour obtenir des instructions sur la migration du SDK client 3 vers le SDK client 5, consultez. Migration du SDK client 3 vers le SDK client 5

Rubriques

- Outil de configuration du SDK client 5
- Outil de configuration du SDK client 3

Outil de configuration du SDK client 5

Utilisez l'outil de configuration du SDK client 5 pour mettre à jour les fichiers de configuration côté client.

Chaque composant du SDK client 5 inclut un outil de configuration avec un indicateur du composant dans le nom de fichier de l'outil de configuration. Par exemple, la bibliothèque PKCS #11 pour le SDK client 5 inclut un outil de configuration nommé `configure-pkcs11` sous Linux ou `configure-pkcs11.exe` Windows.

Syntaxe

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]
```

Configurations avancées

Pour obtenir la liste des configurations avancées spécifiques à l'outil de configuration du SDK client 5, reportez-vous à la section Configurations avancées de l'outil de configuration du SDK client 5.

> **Important**
>
> Après avoir apporté des modifications à votre configuration, vous devez redémarrer votre application pour que les modifications prennent effet.

Exemples

Ces exemples illustrent comment utiliser l'outil de configuration pour le SDK client 5.

Démarrage du SDK client 5

Example

Cet exemple utilise le paramètre `-a` pour mettre à jour les données HSM pour le SDK client 5. Pour utiliser le paramètre `-a`, vous devez disposer de l'adresse IP de l'un des HSM de votre cluster.

PKCS #11 library

Pour démarrer une instance EC2 Linux pour un SDK client 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

Pour démarrer une instance EC2 Windows pour le client SDK 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

OpenSSL Dynamic Engine

Pour démarrer une instance EC2 Linux pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

JCE provider

Pour démarrer une instance EC2 Linux pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

Pour démarrer une instance EC2 Windows pour le client SDK 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster.

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

CloudHSM CLI

Pour démarrer une instance EC2 Linux pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP du ou des HSM de votre cluster.

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

Pour démarrer une instance EC2 Windows pour le client SDK 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP du ou des HSM de votre cluster.

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

> **Note**
>
> vous pouvez utiliser le paramètre `--cluster-id` à la place de`-a <HSM_IP_ADDRESSES>`. Pour connaître les conditions d'utilisation de `--cluster-id`, consultez Outil de configuration du SDK client 5.

Pour plus d'informations sur le paramètre `-a`, consultez the section called "Paramètres".

Spécifiez le cluster, la région et le point de terminaison pour le SDK client 5

Example

Cet exemple utilise le paramètre `cluster-id` pour démarrer le SDK client 5 en effectuant un appel `DescribeClusters`.

PKCS #11 library

Pour démarrer une instance EC2 Linux pour le SDK client 5 avec **`cluster-id`**

- Utilisez l'ID du cluster `cluster-1234567` pour spécifier l'adresse IP d'un HSM dans votre cluster.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id cluster-1234567
```

Pour démarrer une instance EC2 Windows pour le SDK client 5 avec **`cluster-id`**

- Utilisez l'ID du cluster `cluster-1234567` pour spécifier l'adresse IP d'un HSM dans votre cluster.

```
"C:\Program Files\Amazon\CloudHSM\configure-pkcs11.exe" --cluster-
id cluster-1234567
```

OpenSSL Dynamic Engine

Pour démarrer une instance EC2 Linux pour le SDK client 5 avec **`cluster-id`**

- Utilisez l'ID du cluster `cluster-1234567` pour spécifier l'adresse IP d'un HSM dans votre cluster.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id cluster-1234567
```

JCE provider

Pour démarrer une instance EC2 Linux pour le SDK client 5 avec **`cluster-id`**

- Utilisez l'ID du cluster `cluster-1234567` pour spécifier l'adresse IP d'un HSM dans votre cluster.

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id cluster-1234567
```

Pour démarrer une instance EC2 Windows pour le SDK client 5 avec **`cluster-id`**

- Utilisez l'ID du cluster `cluster-1234567` pour spécifier l'adresse IP d'un HSM dans votre cluster.

```
"C:\Program Files\Amazon\CloudHSM\configure-jce.exe" --cluster-
id cluster-1234567
```

CloudHSM CLI

Pour démarrer une instance EC2 Linux pour le SDK client 5 avec **`cluster-id`**

- Utilisez l'ID du cluster `cluster-1234567` pour spécifier l'adresse IP d'un HSM dans votre cluster.

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id cluster-1234567
```

Pour démarrer une instance EC2 Windows pour le SDK client 5 avec **`cluster-id`**

- Utilisez l'ID du cluster `cluster-1234567` pour spécifier l'adresse IP d'un HSM dans votre cluster.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --cluster-
id cluster-1234567
```

Vous pouvez utiliser les paramètres `--region` et `--endpoint` en combinaison avec le paramètre `cluster-id` pour spécifier la manière dont le système effectue l'appel `DescribeClusters`. Par exemple, si la région du cluster est différente de celle configurée par défaut par votre CLI AWS, vous devez utiliser le paramètre `--region` pour utiliser cette région. En outre, il est possible de spécifier le point de terminaison d' AWS CloudHSM API à utiliser pour l'appel, ce qui peut être nécessaire pour diverses configurations réseau, telles que l'utilisation de points de terminaison d'interface VPC qui n'utilisent pas le nom d'hôte DNS par défaut pour. AWS CloudHSM

PKCS #11 library

Pour démarrer une instance EC2 Linux avec un point de terminaison et une région personnalisés

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster avec une région et un point de terminaison personnalisés.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --cluster-id cluster-1234567 --
region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

Pour démarrer une instance EC2 Windows avec un point de terminaison et une région

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster avec une région et un point de terminaison personnalisés.

```
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

Pour démarrer une instance EC2 Linux avec un point de terminaison et une région personnalisés

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster avec une région et un point de terminaison personnalisés.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --cluster-id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

JCE provider

Pour démarrer une instance EC2 Linux avec un point de terminaison et une région personnalisés

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster avec une région et un point de terminaison personnalisés.

```
$ sudo /opt/cloudhsm/bin/configure-jce --cluster-id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

Pour démarrer une instance EC2 Windows avec un point de terminaison et une région

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster avec une région et un point de terminaison personnalisés.

```
"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

Pour démarrer une instance EC2 Linux avec un point de terminaison et une région personnalisés

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster avec une région et un point de terminaison personnalisés.

```
$ sudo /opt/cloudhsm/bin/configure-cli --cluster-id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

Pour démarrer une instance EC2 Windows avec un point de terminaison et une région

- Utilisez l'outil de configuration pour spécifier l'adresse IP d'un HSM dans votre cluster avec une région et un point de terminaison personnalisés.

```
"C:\Program Files\Amazon\CloudHSM\configure-cli.exe" --cluster-id cluster-1234567 --region us-east-1 --endpoint https://cloudhsmv2.us-east-1.amazonaws.com
```

Pour plus d'informations sur les paramètres `--cluster-id`, `--region` et `--endpoint`, consultez the section called “Paramètres”.

Mettre à jour la clé et le certificat client pour l'authentification mutuelle client-serveur TLS

Example

Cet exemple montre comment utiliser les `--server-client-key-file` paramètres `server-client-cert-file` et pour reconfigurer le protocole SSL en spécifiant une clé personnalisée et un certificat SSL pour AWS CloudHSM

PKCS #11 library

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Linux

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. Utilisez l'outil de configuration pour spécifier `ssl-client.crt` et `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
```

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Windows

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. À l'aide d'un PowerShell interpréteur, utilisez l'outil de configuration pour spécifier `ssl-client.crt` et`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

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Linux

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. Utilisez l'outil de configuration pour spécifier `ssl-client.crt` et `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

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Linux

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. Utilisez l'outil de configuration pour spécifier `ssl-client.crt` et `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
```

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Windows

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. À l'aide d'un PowerShell interpréteur, utilisez l'outil de configuration pour spécifier `ssl-client.crt` et`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

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Linux

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. Utilisez l'outil de configuration pour spécifier `ssl-client.crt` et `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
```

Pour utiliser un certificat et une clé personnalisés pour l'authentification mutuelle client-serveur TLS avec le SDK client 5 sous Windows

1. Copiez votre clé et votre certificat dans le répertoire approprié.

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

2. À l'aide d'un PowerShell interpréteur, utilisez l'outil de configuration pour spécifier `ssl-client.crt` et`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
```

Pour plus d'informations sur les paramètres `server-client-cert-file` et `--server-client-key-file`, consultez the section called “Paramètres”.

Désactiver les paramètres de durabilité de la clé du client

Example

Cet exemple utilise le paramètre `--disable-key-availability-check` pour désactiver les paramètres de durabilité de la clé du client. Pour exécuter un cluster avec un seul HSM, vous devez désactiver les paramètres de durabilité de la clé du client.

PKCS #11 library

Pour désactiver la durabilité de la clé du client pour le SDK client 5 sous Linux

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --disable-key-availability-check
```

Pour désactiver la durabilité de la clé client pour le SDK client 5 sous Windows

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-pkcs11.exe" --disable-key-
availability-check
```

OpenSSL Dynamic Engine

Pour désactiver la durabilité de la clé du client pour le SDK client 5 sous Linux

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
$ sudo /opt/cloudhsm/bin/configure-dyn --disable-key-availability-check
```

JCE provider

Pour désactiver la durabilité de la clé du client pour le SDK client 5 sous Linux

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
$ sudo /opt/cloudhsm/bin/configure-jce --disable-key-availability-check
```

Pour désactiver la durabilité de la clé client pour le SDK client 5 sous Windows

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-jce.exe" --disable-key-
availability-check
```

CloudHSM CLI

Pour désactiver la durabilité de la clé du client pour le SDK client 5 sous Linux

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
$ sudo /opt/cloudhsm/bin/configure-cli --disable-key-availability-check
```

Pour désactiver la durabilité de la clé client pour le SDK client 5 sous Windows

- Utilisez l'outil de configuration pour désactiver les paramètres de durabilité de la clé du client.

```
"C:\Program Files\Amazon\CloudHSM\bin\configure-cli.exe" --disable-key-
availability-check
```

Pour plus d'informations sur le paramètre `--disable-key-availability-check`, consultez the section called "Paramètres".

Gestion des options de journalisation

Example

Le SDK client 5 utilise les paramètres `log-file`, `log-level`, `log-rotation` et `log-type` pour gérer la journalisation.

> **Note**
>
> Pour configurer votre SDK pour des environnements sans serveur tels qu'AWS Fargate ou AWS Lambda, nous vous recommandons de configurer votre type de journal sur. AWS CloudHSM `term` Les journaux du client seront générés `stderr` et capturés dans le groupe de CloudWatch journaux des journaux configuré pour cet environnement.

PKCS #11 library

Emplacement de journalisation par défaut

- Si vous ne spécifiez pas d'emplacement pour le fichier, le système écrit les journaux à l'emplacement par défaut suivant :

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-pkcs11.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-pkcs11.log
  ```

Pour configurer le niveau de journalisation et laisser les autres options de journalisation définies par défaut

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-level info
  ```
  ```

Pour configurer les options de journalisation des fichiers

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type file --log-file <file name with path> --log-rotation daily --log-level info
  ```

Pour configurer les options de journalisation du terminal

- ```
  $ sudo /opt/cloudhsm/bin/configure-pkcs11 --log-type term --log-level info
  ```

OpenSSL Dynamic Engine

Emplacement de journalisation par défaut

- Si vous ne spécifiez pas d'emplacement pour le fichier, le système écrit les journaux à l'emplacement par défaut suivant :

  Linux

  ```
  stderr
  ```

Pour configurer le niveau de journalisation et laisser les autres options de journalisation définies par défaut

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-level info
  ```

Pour configurer les options de journalisation des fichiers

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type <file name> --log-file file --log-rotation daily --log-level info
  ```

Pour configurer les options de journalisation du terminal

- ```
  $ sudo /opt/cloudhsm/bin/configure-dyn --log-type term --log-level info
  ```
  ```

JCE provider

Emplacement de journalisation par défaut

- Si vous ne spécifiez pas d'emplacement pour le fichier, le système écrit les journaux à l'emplacement par défaut suivant :

  Linux

  ```
  /opt/cloudhsm/run/cloudhsm-jce.log
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\cloudhsm-jce.log
  ```

Pour configurer le niveau de journalisation et laisser les autres options de journalisation définies par défaut

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-level info
  ```

Pour configurer les options de journalisation des fichiers

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type file --log-file <file name> --
  log-rotation daily --log-level info
  ```

Pour configurer les options de journalisation du terminal

- ```
  $ sudo /opt/cloudhsm/bin/configure-jce --log-type term --log-level info
  ```

CloudHSM CLI

Emplacement de journalisation par défaut

- Si vous ne spécifiez pas d'emplacement pour le fichier, le système écrit les journaux à l'emplacement par défaut suivant :
  ```

Linux

```
/opt/cloudhsm/run/cloudhsm-cli.log
```

Windows

```
C:\Program Files\Amazon\CloudHSM\cloudhsm-cli.log
```

Pour configurer le niveau de journalisation et laisser les autres options de journalisation définies par défaut

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-level info
  ```

Pour configurer les options de journalisation des fichiers

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type file --log-file <file name> --log-rotation daily --log-level info
  ```

Pour configurer les options de journalisation du terminal

- ```
  $ sudo /opt/cloudhsm/bin/configure-cli --log-type term --log-level info
  ```

Pour plus d'informations sur les paramètres `log-file`, `log-level`, `log-rotation` et `log-type` consultez the section called “Paramètres”.

Placez le certificat émetteur pour le SDK client 5

Example

Cet exemple utilise le paramètre `--hsm-ca-cert` pour mettre à jour l'emplacement du certificat émetteur pour le SDK client 5.
  ```

PKCS #11 library

Pour placer le certificat émetteur sous Linux pour un SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

Pour placer le certificat émetteur sous Windows pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

OpenSSL Dynamic Engine

Pour placer le certificat émetteur sous Linux pour un SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

JCE provider

Pour placer le certificat émetteur sous Linux pour un SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

Pour placer le certificat émetteur sous Windows pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

CloudHSM CLI

Pour placer le certificat émetteur sous Linux pour un SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

Pour placer le certificat émetteur sous Windows pour le SDK client 5

- Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur.

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

Pour plus d'informations sur le paramètre `--hsm-ca-cert`, consultez the section called "Paramètres".

Paramètres

-a ***<adresse IP d'ENI>***

Ajoute l'adresse IP spécifiée aux fichiers de configuration du SDK client 5. Entrez n'importe quelle adresse IP ENI d'un HSM du cluster. Pour plus d'informations sur l'utilisation de cette option, consultez la section Démarrer un SDK client 5.

Obligatoire : oui

-- hsm-ca-cert <customerCA certificate file path>

Chemin d'accès au répertoire stockant le certificat de l'autorité de certification (CA) utilisé pour connecter les instances clientes EC2 au cluster. Il s'agit du fichier que vous avez créé lorsque vous avez initialisé le cluster. Par défaut, le système recherche ce fichier dans l'emplacement suivant :

Linux

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

Windows

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

Pour plus d'informations sur l'initialisation du cluster ou le placement du certificat, consultez ??? et ???.

Obligatoire : non

--cluster-id ***<cluster ID>***

Effectue un appel `DescribeClusters` pour rechercher toutes les adresses IP de l'interface réseau Elastic (ENI) du HSM du cluster associées à l'ID du cluster. Le système ajoute les adresses IP ENI aux fichiers AWS CloudHSM de configuration.

> **Note**
>
> Si vous utilisez le `--cluster-id` paramètre à partir d'une instance EC2 au sein d'un VPC qui n'a pas accès à l'Internet public, vous devez créer un point de terminaison VPC

d'interface auquel vous connecter. AWS CloudHSM Pour plus d'informations sur les points de terminaison d'un VPC, consultez ???.

Obligatoire : non

--endpoint ***<endpoint>***

Spécifiez le point de terminaison de l' AWS CloudHSM API utilisé pour effectuer l'`DescribeClusters`appel. Vous devez définir cette option en combinaison avec `--cluster-id`.

Obligatoire : non

--region ***<region>***

Spécifiez la région de votre cluster. Vous devez définir cette option en combinaison avec `--cluster-id`.

Si vous ne fournissez pas le paramètre `--region`, le système choisit la région en essayant de lire les variables d'environnement `AWS_DEFAULT_REGION` ou `AWS_REGION`. Si ces variables ne sont pas définies, le système vérifie la région associée à votre profil dans votre fichier AWS Config (généralement `~/.aws/config`), sauf si vous avez spécifié un autre fichier dans la variable d'environnement `AWS_CONFIG_FILE`. Si aucune des options ci-dessus n'est définie, le système utilise par défaut la région `us-east-1`.

Obligatoire : non

-- server-client-cert-file <client certificate file path>

Chemin d'accès au certificat client utilisé pour l'authentification mutuelle client-serveur TLS.

N'utilisez cette option que si vous ne souhaitez pas utiliser la clé par défaut et le certificat SSL/TLS inclus dans le SDK client 5. Vous devez définir cette option en combinaison avec `--server-client-key-file`.

Obligatoire : non

-- server-client-key-file <client key file path>

Chemin d'accès à la clé client utilisée pour l'authentification mutuelle client-serveur TLS.

N'utilisez cette option que si vous ne souhaitez pas utiliser la clé par défaut et le certificat SSL/TLS inclus dans le SDK client 5. Vous devez définir cette option en combinaison avec `--server-client-cert-file`.

Obligatoire : non

--log-level ***<error | warn | info | debug | trace>***

Spécifie le niveau de journalisation minimal que le système doit écrire dans le fichier journal. Chaque niveau inclut les niveaux précédents, où « error » est le niveau minimum et « trace » le niveau maximum. Cela signifie que si vous spécifiez des erreurs, le système écrit uniquement les erreurs dans le journal. Si vous spécifiez le suivi, le système écrit les erreurs, les avertissements, les messages d'information (info) et de débogage dans le journal. Pour plus d'informations, consultez Journalisation du SDK client 5.

Obligatoire : non

--log-rotation***<daily | weekly>***

Spécifie la fréquence à laquelle le système fait pivoter les journaux. Pour plus d'informations, consultez Journalisation du SDK client 5.

Obligatoire : non

--log-file ***<file name with path>***

Spécifie l'endroit où le système écrira le fichier journal. Pour plus d'informations, consultez Journalisation du SDK client 5.

Obligatoire : non

--log-type ***<term | file>***

Spécifie si le système doit écrire le journal dans un fichier ou un terminal. Pour plus d'informations, consultez Journalisation du SDK client 5.

Obligatoire : non

-h | --help

Affiche l'aide.

Obligatoire : non

-v | --version

Affiche la version.

Obligatoire : non

--disable-key-availability-check

Drapeau pour désactiver le quorum de disponibilité des clés. Utilisez cet indicateur pour indiquer que le quorum de disponibilité des clés AWS CloudHSM doit être désactivé et que vous pouvez utiliser des clés qui n'existent que sur un seul HSM du cluster. Pour plus d'informations sur l'utilisation de cet indicateur pour définir le quorum de disponibilité des clés, consultez ???.

Obligatoire : non

--enable-key-availability-check

Drapeau pour activer le quorum de disponibilité des clés. Utilisez cet indicateur pour indiquer que vous AWS CloudHSM devez utiliser le quorum de disponibilité des clés et ne pas vous autoriser à utiliser des clés tant que ces clés n'existent pas sur deux HSM du cluster. Pour plus d'informations sur l'utilisation de cet indicateur pour définir le quorum de disponibilité des clés, consultez ???.

Activée par défaut.

Obligatoire : non

-- disable-validate-key-at -init

Améliore les performances en spécifiant que vous pouvez ignorer un appel d'initialisation afin de vérifier les autorisations sur une clé pour les appels suivants. À utiliser avec précaution.

Contexte : Certains mécanismes de la bibliothèque PKCS #11 prennent en charge les opérations en plusieurs parties dans le cadre desquelles un appel d'initialisation vérifie si vous pouvez utiliser la clé pour les appels suivants. Cela nécessite un appel de vérification au HSM, ce qui ajoute de la latence au fonctionnement global. Cette option vous permet de désactiver l'appel suivant et d'améliorer potentiellement les performances.

Obligatoire : non

-- enable-validate-key-at -init

Spécifie que vous devez utiliser un appel d'initialisation pour vérifier les autorisations sur une clé pour les appels suivants. Il s'agit de l'option par défaut. Utilisez `enable-validate-key-at-init` pour reprendre ces appels d'initialisation après les avoir suspendus via `disable-validate-key-at-init`.

Obligatoire : non

Rubriques en relation

- DescribeClustersOpération d'API
- CLI AWS describe-clusters
- Get-HSM2Cluster PowerShell applet de commande
- Démarrage du SDK client 5
- AWS CloudHSM Points de terminaison d'un VPC
- Gestion des paramètres de durabilité des clés du SDK client 5
- Journalisation du SDK client 5

Configurations avancées pour l'outil de configuration du SDK client 5

L'outil de configuration du SDK client 5 inclut des configurations avancées qui ne font pas partie des fonctionnalités générales utilisées par la plupart des clients. Les configurations avancées offrent des fonctionnalités supplémentaires.

- Configurations avancées pour PKCS #11
  - Connexion à plusieurs emplacements avec PKCS #11
  - Commandes de nouvelle tentative pour PKCS #11
- Configurations avancées pour JCE
  - Connexion à plusieurs clusters avec le fournisseur JCE
  - Commandes de nouvelle tentative pour JCE
  - Extraction de clés à l'aide de JCE
- Configurations avancées pour openSSL
  - Commandes de nouvelle tentative pour OpenSSL
- Configurations avancées pour l'interface de ligne de commande (CLI)
  - Connexion à plusieurs clusters à l'aide de la CLI

Outil de configuration du SDK client 3

Utilisez l'outil de configuration du SDK client 3 pour démarrer le démon client et configurer CloudHSM Management Utility.

Syntaxe

```
configure -h | --help
          -a <ENI IP address>
          -m [-i <daemon_id>]
          --ssl --pkey <private key file> --cert <certificate file>
          --cmu <ENI IP address>
```

Exemples

Ces exemples illustrent comment utiliser l'outil configure.

Example : Mettre à jour les données HSM pour le AWS CloudHSM client et key_mgmt_util

Cet exemple utilise le `-a` paramètre de configure pour mettre à jour les données HSM pour le AWS CloudHSM client et key_mgmt_util. Pour utiliser le paramètre `-a`, vous devez disposer de l'adresse IP de l'un des HSM de votre cluster. Utilisez la console ou la CLI AWS pour obtenir l'adresse IP.

Pour obtenir une adresse IP pour un HSM (console)

1. Ouvrez la AWS CloudHSM console à l'adresse https://console.aws.amazon.com/cloudhsm/home.
2. Pour changer de région AWS, utilisez le Sélecteur de région dans l'angle supérieur droit de la page.
3. Pour ouvrir la page détaillée du cluster, choisissez l'ID du cluster dans le tableau des clusters.
4. Pour obtenir l'adresse IP, dans l'onglet HSM, choisissez l'une des adresses IP répertoriées sous Adresse IP ENI.

Pour obtenir une adresse IP pour un HSM (CLI)

- Obtenez l'adresse IP d'un HSM à l'aide de la describe-clusters commande de la CLI. Dans la sortie de la commande, l'adresse IP des HSM sont les valeurs de `EniIp`.

```
$  aws cloudhsmv2 describe-clusters


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

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

Pour mettre à jour les données HSM

1. Avant de mettre à jour le `-a` paramètre, arrêtez le AWS CloudHSM client. Ceci empêche les conflits qui peuvent se produire lorsque l'outil configure modifie le fichier de configuration du client. Si le client est déjà arrêté, cette commande n'a pas d'effet et vous pouvez l'utiliser dans un script.

   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

- Pour le Client Windows version 1.1.2 et ultérieure :

  ```
  C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
  ```

- Pour les clients Windows version 1.1.1 et antérieure :

  Utilisez Ctrl + C dans la fenêtre de commande dans laquelle vous avez démarré le AWS CloudHSM client.

2. Cette étape utilise le paramètre `-a` de l'outil configure pour ajouter l'adresse IP d'ENI `10.0.0.9` aux fichiers de configuration.

   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. Ensuite, redémarrez le AWS CloudHSM client. Lorsque le client démarre, il utilise l'adresse IP d'ENI dans son fichier de configuration pour interroger le cluster. Ensuite, il écrit les adresses IP d'ENI de tous les HSM du cluster dans le fichier `cluster.info`.

   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

- Pour le Client Windows version 1.1.2 et ultérieure :

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Pour les clients Windows version 1.1.1 et antérieure :

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Lorsque la commande est terminée, les données HSM utilisées par le AWS CloudHSM client et key_mgmt_util sont complètes et exactes.

Example : Mettre à jour des données HSM pour l'utilitaire CMU à partir du SDK client 3.2.1 et versions antérieures

Cet exemple utilise la commande `-m` configure pour copier les données HSM mises à jour depuis le fichier `cluster.info` vers le fichier `cloudhsm_mgmt_util.cfg` utilisé par cloudhsm_mgmt_util. Utilisez-la avec l'utilitaire CMU fourni avec le SDK client 3.2.1 et versions antérieures.

- Avant d'exécuter le`-m`, arrêtez le AWS CloudHSM client, exécutez la `-a` commande, puis redémarrez le AWS CloudHSM client, comme indiqué dans l'exemple précédent. Cela garantit que les données copiées dans le fichier `cloudhsm_mgmt_util.cfg` depuis le fichier `cluster.info` sont complètes et précises.

  Linux

  ```
  $ sudo /opt/cloudhsm/bin/configure -m
  ```

  Windows

  ```
  C:\Program Files\Amazon\CloudHSM\bin\ configure.exe -m
  ```

Example : Mettre à jour des données HSM pour l'utilitaire CMU à partir du SDK client 3.3.0 et versions ultérieures

Cet exemple utilise le paramètre `--cmu` de la commande configure pour mettre à jour les données HSM pour l'utilitaire CMU. Utilisez-le avec l'utilitaire CMU fourni avec le SDK client 3.3.0 et versions ultérieures. Pour plus d'informations sur l'utilisation de l'utilitaire CMU, consultez les sections Utilisation de l'Utilitaire de gestion CloudHSM (CMU) pour gérer les utilisateurs et Utilisation de la CMU avec le SDK client 3.2.1 et versions antérieures.

- Utilisez le paramètre `--cmu` pour transmettre l'adresse IP d'un HSM de votre cluster.

  Linux

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

  Windows

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

Paramètres

-h | --help

Affiche la syntaxe de la commande.

Obligatoire : oui

-a *<adresse IP d'ENI>*

Ajoute l'adresse IP d'interface réseau Elastic (ENI) du HSM spécifié dans les fichiers de configuration AWS CloudHSM . Entrez l'adresse IP d'ENI d'un des HSM du cluster. Peu importe lequel vous sélectionnez.

Pour obtenir les adresses IP ENI des HSM de votre cluster, utilisez l'DescribeClustersopération, la commande CLI describe-clusters ou Get-HSM2Cluster PowerShell l'applet de commande.

> **Note**
>
> Avant d'exécuter la `-a` configure commande, arrêtez le AWS CloudHSM client. Ensuite, lorsque la `-a` commande est terminée, redémarrez le AWS CloudHSM client. Pour plus de détails, consultez les exemples.

Ce paramètre modifie les fichiers de configuration suivants :

- `/opt/cloudhsm/etc/cloudhsm_client.cfg`: utilisé par le AWS CloudHSM client et key_mgmt_util.
- `/opt/cloudhsm/etc/cloudhsm_mgmt_util.cfg` : utilisé par cloudhsm-mgmt_util.

Lorsque le AWS CloudHSM client démarre, il utilise l'adresse IP ENI dans son fichier de configuration pour interroger le cluster et mettre à jour le `cluster.info` fichier (`/opt/cloudhsm/daemon/1/cluster.info`) avec les adresses IP ENI correctes pour tous les HSM du cluster.

Obligatoire : oui

-m

Met à jour les adresses IP d'ENI des HSM dans le fichier de configuration utilisé par l'utilitaire CMU.

> **Note**
>
> Le paramètre `-m` est destiné à être utilisé avec l'utilitaire CMU à partir du SDK client 3.2.1 et des versions antérieures. Pour l'utilitaire CMU issu du SDK client 3.3.0 et versions ultérieures, consultez le paramètre `--cmu`, qui simplifie le processus de mise à jour des données HSM pour l'utilitaire CMU.

Lorsque vous mettez à jour le `-a` paramètre du AWS CloudHSM client, configure puis que vous le démarrez, le démon client interroge le cluster et met à jour les `cluster.info` fichiers avec les adresses IP HSM correctes pour tous les HSM du cluster. L'exécution de la commande `-m` configure finalise la mise à jour en copiant les adresses IP HSM depuis le fichier `cluster.info` vers le fichier de configuration `cloudhsm_mgmt_util.cfg` utilisé par cloudhsm_mgmt_util.

Assurez-vous d'exécuter la `-a` configure commande et de redémarrer le AWS CloudHSM client avant d'exécuter la `-m` commande. Cela garantit que les données copiées dans `cloudhsm_mgmt_util.cfg` à partir de `cluster.info` sont complètes et précises.

Obligatoire : oui

-i

Spécifie un autre démon client. La valeur par défaut représente le client AWS CloudHSM .

Par défaut: 1

Obligatoire : non

--ssl

Remplacez la clé et les certificats SSL du cluster avec la clé privée et le certificat spécifiés. Lorsque vous utilisez ce paramètre, les paramètres `--pkey` et `--cert` sont obligatoires.

Obligatoire : non

--pkey

Spécifie la nouvelle clé privée. Entrez le chemin et le nom du fichier qui contient la clé privée.

Obligatoire oui si --ssl est spécifié. Sinon, ne l'utilisez pas.

--cert

Spécifie le nouveau certificat. Entrez le chemin et le nom du fichier qui contient le certificat. Le certificat doit se lier au certificat `customerCA.crt`, le certificat auto-signé utilisé pour initialiser le cluster. Pour plus d'informations, consultez Initialiser le cluster.

Obligatoire oui si --ssl est spécifié. Sinon, ne l'utilisez pas.

--cmu ***<adresse IP d'ENI>***

Combine les paramètres `-a` et `-m` en un seul paramètre. Ajoute l'adresse IP HSM Elastic Network Interface (ENI) spécifiée aux fichiers de AWS CloudHSM configuration, puis met à jour le fichier de configuration CMU. Entrez une adresse IP provenant de n'importe quel HSM du cluster. Pour le SDK client 3.2.1 et versions antérieures, consultez Utilisation de l'utilitaire CMU avec le SDK client 3.2.1 et versions antérieures.

Obligatoire : oui

Rubriques en relation

- Configuration de key_mgmt_util

Interface de ligne de commande (CLI) CloudHSM

La CLI CloudHSM aide les administrateurs à gérer les utilisateurs et les utilisateurs de chiffrement à gérer les clés de leur cluster. Il inclut des outils qui peuvent être utilisés pour créer, supprimer et répertorier des utilisateurs, modifier les mots de passe des utilisateurs, mettre à jour l'authentification multifactorielle (MFA) des utilisateurs. Il inclut également des commandes qui génèrent, suppriment, importent et exportent des clés, obtiennent et définissent des attributs, trouvent des clés et effectuent des opérations cryptographiques.

Pour obtenir la liste définie des utilisateurs de la CLI CloudHSM, consultez Gestion des utilisateurs HSM à l'aide de la CLI CloudHSM . Pour obtenir une liste définie des attributs clés de la CLI CloudHSM, consultez. Attributs de clé de la CLI CloudHSM Pour plus d'informations sur l'utilisation de la CLI CloudHSM pour gérer les clés, consultez. Gestion des clés à l'aide de la CLI CloudHSM

Pour une mise en route rapide, consultez Mise en route avec l'interface de ligne de commande (CLI) CloudHSM. Pour plus d'informations sur les commandes de la CLI CloudHSM et des exemples d'utilisation des commandes, consultez Référence pour les commandes de la CLI CloudHSM.

Rubriques

Plateformes prises en charge par l'Interface de ligne de commande (CLI) CloudHSM

Prise en charge de Linux

Plateformes prises en charge	Architecture x86_64	Architecture ARM
Amazon Linux 2	Oui	Oui
Amazon Linux 2023	Oui	Oui
CentOS 7 (7,8+)	Oui	Non
Red Hat Enterprise Linux 7 (7.8+)	Oui	Non
Red Hat Enterprise Linux 8 (8.3 ou version ultérieure)	Oui	Non
Red Hat Enterprise Linux 9 (version 9.2+)	Oui	Oui
Ubuntu 20.04 LTS	Oui	Non
Ubuntu 22.04 LTS	Oui	Oui

Remarque : le SDK 5.4.2 est la dernière version à prendre en charge la plate-forme CentOS 8. Pour plus d'informations, veuillez consulter le site Web CentOS.

Prise en charge de Windows

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Mise en route avec l'interface de ligne de commande (CLI) CloudHSM

L'interface de ligne de commande (CLI) CloudHSM vous permet de gérer les utilisateurs de votre cluster. AWS CloudHSM Utilisez cette rubrique pour démarrer avec les tâches de gestion des utilisateurs HSM, telles que la création d'utilisateurs, la liste des utilisateurs et la connexion de la CLI CloudHSM au cluster.

Installation de la CLI CloudHSM

Utilisez les commandes suivantes pour télécharger et installer CloudHSM CLI.

Amazon Linux 2

Amazon Linux 2 sur architecture x86_64 :

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

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

Amazon Linux 2 sur architecture ARM64 :

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

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

Amazon Linux 2023

Amazon Linux 2023 sur une architecture x86_64 :

```
$ 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
```

Amazon Linux 2023 sur architecture ARM64 :

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

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

CentOS 7 (7.8+)

CentOS 7 sur architecture x86_64 :

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

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

RHEL 7 (7.8+)

RHEL 7 sur architecture x86_64 :

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

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

RHEL 8 (8.3+)

RHEL 8 sur architecture x86_64 :

```
$ 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+)

RHEL 9 sur une architecture x86_64 :

```
$ 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
```

RHEL 9 sur l'architecture ARM64 :

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

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

Ubuntu 20.04 LTS

Ubuntu 20.04 LTS sur architecture x86_64 :

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

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

Ubuntu 22.04 LTS

Ubuntu 22.04 LTS sur architecture x86_64 :

```
$ 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
```

Ubuntu 22.04 LTS sur architecture ARM64 :

```
$ 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

Pour Windows Server 2016 sur une architecture x86_64, ouvrez PowerShell en tant qu'administrateur et exécutez la commande suivante :

```
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

Pour Windows Server 2019 sur une architecture x86_64, ouvrez PowerShell en tant qu'administrateur et exécutez la commande suivante :

```
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
```

Utilisez les commandes suivantes pour configurer CloudHSM CLI.

Pour démarrer une instance EC2 Linux pour un SDK client 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP du ou des HSM de votre cluster.

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

Pour démarrer une instance EC2 Windows pour le client SDK 5

- Utilisez l'outil de configuration pour spécifier l'adresse IP du ou des HSM de votre cluster.

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

Utilisation de la CLI CloudHSM

1. Utilisez la commande suivante pour démarrer la CLI CloudHSM.

   Linux

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

   Windows

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

2. Utilisez la commande login pour vous connecter au cluster. Tous les utilisateurs peuvent utiliser cette commande.

   La commande de l'exemple suivant se connecte à admin, qui est le compte admin par défaut. Vous définissez le mot de passe de l'utilisateur lors de l'activation du cluster.

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

   Le système vous invite à saisir votre mot de passe. Vous entrez le mot de passe, et le résultat indique que la commande a été exécutée avec succès.

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

3. Exécutez la commande user list pour répertorier tous les utilisateurs du cluster.

```
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. Utilisez user create pour créer un utilisateur CU nommé **example_user**.

   Vous pouvez créer des CU car, lors d'une étape précédente, vous vous êtes connecté en tant qu'utilisateur administrateur. Seuls les utilisateurs administrateur peuvent effectuer des tâches de gestion des utilisateurs, telles que la création et la suppression d'utilisateurs et la modification des mots de passe des autres utilisateurs.

```
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. Utilisez user list pour répertorier tous les utilisateurs du cluster.

```
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. Utilisez la logout commande pour vous déconnecter du AWS CloudHSM cluster.

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

7. Utilisez la CLI pour exécuter la commande quit.

```
aws-cloudhsm > quit
```

Modes de commande interactifs et uniques

Dans la CLI CloudHSM, vous pouvez exécuter des commandes de deux manières différentes : en mode commande unique et en mode interactif. Le mode interactif est conçu pour les utilisateurs et le mode commande unique est conçu pour les scripts.

> **Note**
>
> Toutes les commandes fonctionnent en mode interactif et en mode commande unique.

Mode interactif

Utilisez les commandes suivantes pour démarrer le mode interactif de la CLI CloudHSM

Linux

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

Windows

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

Lorsque vous utilisez la CLI en mode interactif, vous pouvez vous connecter à un compte utilisateur à l'aide de la commande login.

Pour afficher toutes les commandes de la CLI CloudHSM, exécutez la commande suivante :

```
aws-cloudhsm > help
```

Pour obtenir la syntaxe d'une commande de la CLI CloudHSM, exécutez la commande suivante :

```
aws-cloudhsm >  help <command-name>
```

Par exemple, pour obtenir la liste des utilisateurs sur les HSM, tapez user list.

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

Pour terminer votre session dans la CLI CloudHSM, exécutez la commande suivante :

```
aws-cloudhsm > quit
```

Mode commande unique

Si vous exécutez la CLI CloudHSM en mode commande unique, vous devez définir deux variables d'environnement pour fournir des informations d'identification : CLOUDHSM_PIN et CLOUDHSM_ROLE :

```
$ export CLOUDHSM_ROLE=admin
```

```
$ export CLOUDHSM_PIN=admin_username:admin_password
```

Ensuite, vous pouvez exécuter des commandes à l'aide des informations d'identification stockées dans votre environnement.

```
$ cloudhsm-cli user change-password --username alice --role crypto-user
Enter password:
Confirm password:
{
    "error_code": 0,
    "data": {
      "username": "alice",
      "role": "crypto-user"
    }
}
```

Attributs de clé de la CLI CloudHSM

Cette rubrique décrit comment utiliser la CLI CloudHSM pour définir des attributs clés. Un attribut de clé de la CLI CloudHSM peut définir le type d'une clé, le mode de fonctionnement d'une clé ou le libellé d'une clé. Certains attributs définissent des caractéristiques uniques (le type d'une clé, par exemple). Les autres attributs peuvent être définis sur vrai ou faux. Leur modification active ou désactive une partie des fonctionnalités de la clé.

Pour des exemples montrant comment utiliser les attributs clés, consultez les commandes répertoriées sous la commande parent clé.

Attributs pris en charge

Il est conseillé de spécifier uniquement des valeurs pour les attributs que vous souhaitez restreindre. Si vous n'indiquez pas de valeur, la CLI CloudHSM utilise la valeur par défaut spécifiée dans le tableau ci-dessous.

Le tableau suivant répertorie les principaux attributs, les valeurs possibles, les valeurs par défaut et les remarques associées. Une cellule vide dans la colonne Valeur indique qu'il n'y a aucune valeur par défaut attribuée à l'attribut.

Attribut de la CLI CloudHSM	Valeur	Modifiable avec une clé set-attribute	Réglable lors de la création de la clé
`always-sensitive`	La valeur est `True` si `sensitive` a toujours été défini sur `True` et n'a jamais changé.	Non	Non
`check-value`	Valeur de contrôle de clé de la clé. Pour plus d'informations, consultez Informations supplémentaires.	Non	Non
`class`	Les valeurs possibles sont : `secret-key` , `public-key` et `private-key` .	Non	Oui
`curve`	Courbe elliptique utilisée pour générer la paire de clés EC. Valeurs valides : `secp224r1` ,	Non	Réglable avec RSA, non paramétrable avec EC

Attribut de la CLI CloudHSM	Valeur	Modifiable avec une clé set-attribute	Réglable lors de la création de la clé
	`secp256r1` , `prime256v1` , `secp384r1` , `secp256k1` et `secp521r1`		
`decrypt`	Par défaut : `False`	Oui	Oui
`derive`	Par défaut : `False`	Oui	Oui
`destroyable`	Par défaut : `True`	Oui	Oui
`ec-point`	Pour les clés EC, codage DER de la valeur ECPoint ANSI X9.62 « Q » au format hexadécimal. Pour les autres types de clé, cet attribut n'existe pas.	Non	Non
`encrypt`	Par défaut : `False`	Oui	Oui
`extractable`	Par défaut : `True`	Non	Oui
`id`	Par défaut : Empty	Non	Oui
`key-length-bytes`	Nécessaire pour générer une clé AES. Valeurs valides : 16,24, et 32 octets.	Non	Non
`key-type`	Les valeurs possibles sont : `aes`, `rsa` et `ec`	Non	Oui

Attribut de la CLI CloudHSM	Valeur	Modifiable avec une clé set-attribute	Réglable lors de la création de la clé
`label`	Par défaut : Empty	Oui	Oui
`local`	Par défaut : `True` pour les clés générées dans le HSM, `False` pour les clés importées dans le HSM.	Non	Non
`modifiable`	Par défaut : `True`	Non	Non
`modulus`	Module qui a été utilisé pour générer une paire de clés RSA. Pour les autres types de clé, cet attribut n'existe pas.	Non	Non
`modulus-size-bits`	Requis pour générer une paire de clés RSA. La valeur minimale est `2048`.	Non	Réglable avec RSA, non paramétrable avec EC
`never-extractable`	La valeur est `True` si extractable n'a jamais été défini sur `False`. La valeur est `False` si extractable a déjà été défini sur `True`.	Non	Non
`private`	Par défaut : `True`	Non	Oui

Attribut de la CLI CloudHSM	Valeur	Modifiable avec une clé set-attribute	Réglable lors de la création de la clé
`public-exponent`	Requis pour générer une paire de clés RSA. Valeurs valides : la valeur doit être un nombre impair supérieur ou égal à `65537`.	Non	Réglable avec RSA, non paramétrable avec EC
`sensitive`	Par défaut : • La valeur est `True` pour les clés AES et les clés privées EC et RSA. • La valeur est `False` pour les clés publiques EC et RSA.	Non	Paramétrable avec des clés privées, non paramétrable avec des clés publiques.
`sign`	Par défaut : • La valeur est `True` pour les clés AES. • La valeur est `False` pour les clés RSA et EC.	Oui	Oui
`token`	Par défaut : `False`	Non	Oui
`trusted`	Par défaut : `False`	Oui	Non
`unwrap`	Par défaut : `False`	Oui	Oui

Attribut de la CLI CloudHSM	Valeur	Modifiable avec une clé set-attribute	Réglable lors de la création de la clé
`unwrap-template`	Les valeurs doivent utiliser le modèle d'attribut appliqué à toute clé désencapsulée à l'aide de cette clé d'encapsulage.	Oui	Non
`verify`	Par défaut : • La valeur est `True` pour les clés AES. • La valeur est `False` pour les clés RSA et EC.	Oui	Oui
`wrap`	Par défaut : `False`	Oui	Oui
`wrap-template`	Les valeurs doivent utiliser le modèle d'attribut pour correspondre à la clé encapsulée à l'aide de cette clé d'encapsulage.	Oui	Non
`wrap-with-trusted`	Par défaut : `False`	Oui	Oui

Informations supplémentaires

Valeur de contrôle

La valeur de contrôle est un hachage ou une somme de contrôle de 3 octets d'une clé générée lorsque le HSM importe ou génère une clé. Vous pouvez également calculer une valeur de contrôle en dehors du HSM, par exemple après avoir exporté une clé. Vous pouvez ensuite

comparer les valeurs de contrôle pour confirmer l'identité et l'intégrité de la clé. Pour obtenir la valeur de contrôle d'une clé, utilisez la liste des clés avec l'indicateur détaillé.

AWS CloudHSM utilise les méthodes standard suivantes pour générer une valeur de contrôle :

- Clés symétriques : les 3 premiers octets du résultat du chiffrement d'un bloc zéro avec la clé.
- Paires de clés asymétriques : les 3 premiers octets du hachage SHA-1 de la clé publique.
- Clés HMAC : la KCV pour les clés HMAC n'est pas prise en charge pour le moment.

Rubriques en relation

- clé
- Référence pour les commandes de la CLI CloudHSM

Migrer de la CMU et de la KMU du SDK client 3 vers la CLI CloudHSM du SDK client 5

Utilisez cette rubrique pour migrer les flux de travail qui utilisent les outils de ligne de commande du SDK client 3, l'utilitaire de gestion CloudHSM (CMU) et l'utilitaire de gestion des clés (KMU), afin d'utiliser à la place l'outil de ligne de commande du SDK client 5, CloudHSM CLI.

Dans AWS CloudHSM, les applications client exécutent des opérations cryptographiques à l'aide du kit de développement logiciel (SDK) AWS CloudHSM client. Le SDK client 5 est le SDK principal auquel de nouvelles fonctionnalités et un support de plateforme continuent d'être ajoutés. Cette rubrique fournit des informations spécifiques à la migration du SDK client 3 vers le SDK client 5 pour les outils de ligne de commande.

Le SDK client 3 inclut deux outils de ligne de commande distincts : le CMU pour la gestion des utilisateurs et le KMU pour la gestion des clés et l'exécution des opérations avec les clés. Le SDK client 5 consolide les fonctions de la CMU et de la KMU (outils proposés avec le SDK client 3) en un seul outil, le. Interface de ligne de commande (CLI) CloudHSM Les opérations de gestion des utilisateurs se trouvent sous les sous-commandes utilisateur etquorum. Les opérations de gestion des clés se trouvent sous la sous-commande clé, et les opérations cryptographiques se trouvent sous la sous-commande cryptographique. Voir Référence pour les commandes de la CLI CloudHSM pour une liste complète des commandes.

> **Note**
>
> Si, dans le SDK client 3, vous vous êtes appuyé sur syncKey les syncUser fonctionnalités de synchronisation entre clusters, continuez à utiliser la CMU. La CLI CloudHSM du SDK client 5 ne prend actuellement pas en charge cette fonctionnalité.

Pour obtenir des instructions sur la migration vers le SDK client 5, consultez. Migration du SDK client 3 vers le SDK client 5 Pour connaître les avantages de la migration, voirAvantages du SDK client 5.

Configurations avancées pour CLI

L'interface de ligne de AWS CloudHSM commande (CLI) inclut la configuration avancée suivante, qui ne fait pas partie des configurations générales utilisées par la plupart des clients. Ces configurations fournissent des fonctionnalités supplémentaires.

- Connexion à plusieurs clusters

Connexion à plusieurs clusters à l'aide de la CLI

Avec le SDK client 5, vous pouvez configurer la AWS CloudHSM CLI pour autoriser les connexions à plusieurs clusters CloudHSM à partir d'une seule instance de CLI.

Suivez les instructions de cette rubrique pour utiliser l'interface de ligne de AWS CloudHSM commande (CLI) et utiliser la fonctionnalité multi-clusters pour vous connecter à plusieurs clusters.

Rubriques

- Conditions préalables relatives à plusieurs clusters
- Configuration de la CLI pour une fonctionnalité multicluster
- configure-cli ajouter un cluster
- configure-cli Remove-Cluster
- Utilisation de plusieurs clusters

Conditions préalables relatives à plusieurs clusters

- Au moins deux AWS CloudHSM clusters auxquels vous souhaitez vous connecter, ainsi que leurs certificats de cluster.

- Une instance EC2 avec des groupes de sécurité correctement configurés pour se connecter à tous les clusters ci-dessus. Pour plus d'informations sur la configuration d'un cluster et de l'instance client, reportez-vous à la section Mise en route avec AWS CloudHSM.
- Pour configurer la fonctionnalité multi-clusters, vous devez déjà avoir téléchargé et installé la AWS CloudHSM CLI. Si vous ne l'avez pas déjà fait, consultez les instructions figurant dans ???.
- Vous ne pourrez pas accéder à un cluster configuré avec `./configure-cli[.exe] -a` puisqu'il ne sera pas associé à un`cluster-id`. Vous pouvez le reconfigurer en suivant `config-cli add-cluster` les instructions de ce guide.

Configuration de la CLI pour une fonctionnalité multicluster

Pour configurer votre AWS CloudHSM CLI pour la fonctionnalité multi-clusters, procédez comme suit :

1. Identifiez les clusters auxquels vous souhaitez vous connecter.
2. Ajoutez ces clusters à la configuration de votre AWS CloudHSM CLI à l'aide de la sous-commande configure-cli, `add-cluster` comme décrit ci-dessous.
3. Redémarrez tous les processus AWS CloudHSM CLI pour que la nouvelle configuration prenne effet.

configure-cli ajouter un cluster

Lorsque vous vous connectez à plusieurs clusters, utilisez la `configure-cli add-cluster` commande pour ajouter un cluster à votre configuration.

Syntaxe

```
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]
```

Exemples

Ajoutez un cluster à l'aide du paramètre **`cluster-id`**

Example

Utilisez le paramètre `configure-cli add-cluster` ainsi que le paramètre `cluster-id` pour ajouter un cluster (avec l'ID de `cluster-1234567`) à votre configuration.

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**
>
> Si l'utilisation de `configure-cli add-cluster` avec le paramètre `cluster-id` n'entraîne pas l'ajout du cluster, reportez-vous à l'exemple suivant pour une version plus longue de cette commande qui nécessite également des paramètres `--region` et `--endpoint` pour identifier le cluster ajouté. Si, par exemple, la région du cluster est différente de celle configurée par défaut dans votre interface de ligne de commande AWS, vous devez utiliser le paramètre `--region` pour utiliser la bonne région. En outre, il est possible de spécifier le point de terminaison d' AWS CloudHSM API à utiliser pour l'appel, ce qui peut être nécessaire pour diverses configurations réseau, telles que l'utilisation de points de terminaison d'interface VPC qui n'utilisent pas le nom d'hôte DNS par défaut pour. AWS CloudHSM

Ajouter un cluster à l'aide des paramètres **`cluster-id`**, **`endpoint`** et **`region`**

Example

Utilisez les paramètres `configure-cli add-cluster` ainsi que `cluster-id`, `endpoint` et `region` pour ajouter un cluster (avec l'ID de `cluster-1234567`) à votre configuration.

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
```

Pour plus d'informations sur les paramètres `--cluster-id`, `--region` et `--endpoint`, consultez the section called “Paramètres”.

Paramètres

--cluster-id ***<Cluster ID>***

Effectue un appel `DescribeClusters` pour rechercher toutes les adresses IP de l'interface réseau Elastic (ENI) du HSM du cluster associées à l'ID du cluster. Le système ajoute les adresses IP ENI aux fichiers AWS CloudHSM de configuration.

> **Note**
>
> Si vous utilisez le `--cluster-id` paramètre depuis une instance EC2 au sein d'un VPC qui n'a pas accès à l'Internet public, vous devez créer un point de terminaison VPC d'interface auquel vous connecter. AWS CloudHSM Pour plus d'informations sur les points de terminaison d'un VPC, veuillez consulter ???.

Obligatoire : oui

--endpoint ***<Endpoint>***

Spécifiez le point de terminaison de l' AWS CloudHSM API utilisé pour effectuer l'`DescribeClusters`appel. Vous devez définir cette option en combinaison avec `--cluster-id`.

Obligatoire : non

-- hsm-ca-cert <HsmCA Certificate Filepath>

Spécifie le chemin du fichier vers le certificat HSM CA.

Obligatoire : non

--region ***<Region>***

Spécifiez la région de votre cluster. Vous devez définir cette option en combinaison avec `--cluster-id`.

Si vous ne fournissez pas le paramètre `--region`, le système choisit la région en essayant de lire les variables d'environnement `AWS_DEFAULT_REGION` ou `AWS_REGION`. Si ces variables ne sont pas définies, le système vérifie la région associée à votre profil dans votre fichier AWS Config (généralement `~/.aws/config`), sauf si vous avez spécifié un autre fichier dans la variable d'environnement `AWS_CONFIG_FILE`. Si aucune des options ci-dessus n'est définie, le système utilise par défaut la région `us-east-1`.

Obligatoire : non

-- server-client-cert-file <Client Certificate Filepath>

Chemin d'accès au certificat client utilisé pour l'authentification mutuelle client-serveur TLS.

N'utilisez cette option que si vous ne souhaitez pas utiliser la clé par défaut et le certificat SSL/TLS inclus dans le SDK client 5. Vous devez définir cette option en combinaison avec `--server-client-key-file`.

Obligatoire : non

-- server-client-key-file <Client Key Filepath>

Chemin d'accès à la clé client utilisée pour l'authentification mutuelle client-serveur TLS.

N'utilisez cette option que si vous ne souhaitez pas utiliser la clé par défaut et le certificat SSL/TLS inclus dans le SDK client 5. Vous devez définir cette option en combinaison avec `--server-client-cert-file`.

Obligatoire : non

configure-cli Remove-Cluster

Lorsque vous vous connectez à plusieurs clusters à l'aide de la CLI, utilisez la `configure-cli remove-cluster` commande pour supprimer un cluster de votre configuration.

Syntaxe

```
configure-cli remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

Exemples

Supprimer un cluster à l'aide du paramètre **`cluster-id`**

Example

Utilisez le paramètre `configure-cli remove-cluster` ainsi que le paramètre `cluster-id` pour supprimer un cluster (avec l'ID de `cluster-1234567`) de votre configuration.

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
```

Pour plus d'informations sur le paramètre `--cluster-id`, consultez the section called “Paramètres”.

Paramètre

--cluster-id ***<Cluster ID>***

ID du cluster à supprimer de la configuration.

Obligatoire : oui

Utilisation de plusieurs clusters

Après avoir configuré plusieurs clusters avec la AWS CloudHSM CLI, utilisez la `cloudhsm-cli` commande pour interagir avec eux.

Exemples

Définition d'une valeur par défaut **`cluster-id`** lors de l'utilisation du mode interactif

Example

Utilisez le paramètre ??? ainsi que le `cluster-id` paramètre pour définir un cluster par défaut (avec l'ID de`cluster-1234567`) à partir de votre configuration.

Linux

```
$ cloudhsm-cli interactive --cluster-id cluster-1234567
```

Windows

```
C:\Program Files\Amazon\CloudHSM\> .\cloudhsm-cli.exe interactive --cluster-
id cluster-1234567
```

Configuration du **`cluster-id`** lors de l'exécution d'une seule commande

Example

Utilisez le `cluster-id` paramètre pour définir le cluster (avec l'ID de`cluster-1234567`) à ??? partir duquel vous souhaitez accéder.

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
```

Référence pour les commandes de la CLI CloudHSM

La CLI CloudHSM aide les administrateurs à gérer les utilisateurs de leur cluster. AWS CloudHSM La CLI CloudHSM peut être exécutée dans deux modes : le mode interactif et le mode de commande unique. Pour une mise en route rapide, consultez Mise en route avec l'interface de ligne de commande (CLI) CloudHSM.

Pour exécuter la plupart des commandes de la CLI CloudHSM, vous devez démarrer la CLI CloudHSM et vous connecter au HSM. Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de la CLI CloudHSM. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Les rubriques suivantes décrivent les commandes de la CLI CloudHSM :

Command	Description	Type d'utilisateur
activation du cluster	Active un cluster CloudHSM et confirme qu'il s'agit d'un nouveau cluster. Cela doit être fait avant toute autre opération .	Administrateur non activé
cluster hsm-info	Répertoriez les HSM de votre cluster.	Tout [1], y compris les utilisateurs non authentifiés. La connexion n'est pas requise.
signe cryptographique ecdsa	Génère une signature à l'aide d'une clé privée EC et du mécanisme de signature ECDSA.	Utilisateurs de chiffrement (CU)
signe cryptographique rsa-pkcs	Génère une signature à l'aide d'une clé privée RSA et du mécanisme de signature RSA-PKCS.	CU
signe cryptographique rsa-pkcs-pss	Génère une signature à l'aide d'une clé privée RSA et du mécanisme de signature RSA-PKCS-PSS.	CU
Crypto Verify ECDSA	Confirme qu'un fichier a été signé dans le HSM à l'aide d'une clé publique donnée. Vérifie que la signature	CU

Command	Description	Type d'utilisateur
	a été générée à l'aide du mécanisme de signature ECDSA. Compare un fichier signé à un fichier source et détermine si les deux sont liés cryptographiquement en fonction d'une clé publique ecdsa et d'un mécanisme de signature donnés.	
vérification cryptographique rsa-pkcs	Confirme qu'un fichier a été signé dans le HSM à l'aide d'une clé publique donnée. Vérifie que la signature a été générée à l'aide du mécanisme de signature RSA-PKCS. Compare un fichier signé à un fichier source et détermine si les deux sont liés cryptographiquement en fonction d'une clé publique RSA et d'un mécanisme de signature donnés.	CU

Command	Description	Type d'utilisateur
vérification cryptographique rsa-pkcs-pss	Confirme qu'un fichier a été signé dans le HSM à l'aide d'une clé publique donnée. Vérifie que la signature a été générée à l'aide du mécanisme de signature RSA-PKCS-PSS. Compare un fichier signé à un fichier source et détermine si les deux sont liés cryptogra phiquement en fonction d'une clé publique RSA et d'un mécanisme de signature donnés.	CU
Key delete	Supprime une clé de votre AWS CloudHSM cluster.	CU
key generate-file	Génère un fichier clé dans votre AWS CloudHSM cluster.	CU
clé generate-asymmetric-pair RSA	Génère une paire de clés RSA asymétrique dans votre AWS CloudHSM cluster.	CU
clé generate-asymmetric-pair ec	Génère une paire de clés à courbe elliptique asymétrique (EC) dans votre cluster. AWS CloudHSM	CU
clé generate-symmetric aes	Génère une clé AES symétrique dans votre AWS CloudHSM cluster.	CU

Command	Description	Type d'utilisateur
key generate-symmetric generic-secret	Génère une clé secrète générique symétrique dans votre AWS CloudHSM cluster.	CU
clé d'importation pem	Importe une clé au format PEM dans un HSM. Vous pouvez l'utiliser pour importer des clés publiques générées hors du HSM.	CU
Key list	Trouve toutes les clés de l'utilisateur actuel présent dans votre AWS CloudHSM cluster.	CU
répliquer la clé	Répliquez une clé d'un cluster source vers un cluster de destination cloné.	CU
key set-attribute	Définit les attributs des clés de votre AWS CloudHSM cluster.	Les CU peuvent exécuter cette commande et les administrateurs peuvent définir l'attribut de confiance.
key share	Partage une clé avec les autres UC de votre AWS CloudHSM cluster.	CU
Key unshare	Annule le partage d'une clé avec les autres UC de votre AWS CloudHSM cluster.	CU
déballez les clés aes-gcm	Déballe une clé de charge utile dans le cluster à l'aide de la clé d'encapsulation AES et du mécanisme de déballage AES-GCM.	CU

Command	Description	Type d'utilisateur
déballage des clés aes-no-pad	Déballe une clé de charge utile dans le cluster à l'aide de la clé d'encapsulation AES et du mécanisme de déballage AES-NO-PAD.	CU
déballez la clé aes-pkcs5-pad	Déballe une clé de charge utile à l'aide de la clé d'encapsulage AES et du mécanisme de déballage AES-PKCS5-PAD.	CU
déballage des clés aes-zero-pad	Déballe une clé de charge utile dans le cluster à l'aide de la clé d'encapsulation AES et du mécanisme de déballage AES-ZERO-PAD.	CU
déballage des clés cloudhsm-aes-gcm	Déballe une clé de charge utile dans le cluster à l'aide de la clé d'encapsulation AES et du mécanisme de déballage CLOUDHSM-AES-GCM.	CU
clé unwrap rsa-aes	Déballe une clé de charge utile à l'aide d'une clé privée RSA et du mécanisme de déballage RSA-AES.	CU
clé unwrap rsa-oaep	Déballe une clé de charge utile à l'aide de la clé privée RSA et du mécanisme de déballage RSA-OAEP.	CU

Command	Description	Type d'utilisateur
clé unwrap rsa-pkcs	Déballe une clé de charge utile à l'aide de la clé privée RSA et du mécanisme de déballage RSA-PKCS.	CU
porte-clés aes-gcm	Enveloppe une clé de charge utile à l'aide d'une clé AES sur le HSM et du mécanisme d'encapsulage AES-GCM.	CU
étui pour clés aes-no-pad	Enveloppe une clé de charge utile à l'aide d'une clé AES sur le HSM et du mécanisme d'encapsulage AES-NO-PAD.	CU
porte-clés AES-PKCS5-PAD	Encapsule une clé de charge utile à l'aide d'une clé AES sur le HSM et du mécanisme d'encapsulage AES-PKCS5-PAD.	CU
étui pour clés aes-zero-pad	Enveloppe une clé de charge utile à l'aide d'une clé AES sur le HSM et du mécanisme d'encapsulage AES-ZERO-PAD.	CU
étui pour clés cloudhsm-aes-gcm	Enveloppe une clé de charge utile à l'aide d'une clé AES sur le HSM et du mécanisme d'encapsulage CLOUDHSM-AES-GCM.	UCs

Command	Description	Type d'utilisateur
porte-clés rsa-aes	Encapsule une clé de charge utile à l'aide d'une clé publique RSA sur le HSM et du mécanisme d'encapsulage RSA-AES.	CU
porte-clés RSA-OAEP	Encapsule une clé de charge utile à l'aide d'une clé publique RSA sur le HSM et du mécanisme d'encapsulage RSA-OAEP.	CU

Command	Description	Type d'utilisateur
La key wrap rsa-pkcs commande encapsule une clé de charge utile à l'aide d'une clé publique RSA sur le HSM et du mécanisme d'encapsulation. `RSA-PKCS` L'`extractable` attribut de la clé de charge utile doit être défini sur. `true` Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur cryptographique (CU) qui a créé la clé, peut encapsuler la clé. Les utilisateurs qui partagent la clé peuvent l'utiliser dans des opérations cryptographiques. Pour utiliser la key wrap rsa-pkcs commande, vous devez d'abord disposer d'une clé RSA dans votre AWS CloudHSM cluster. Vous pouvez générer une paire de clés RSA à l'aide de la clé generate-asymmetric-pair commande et de l'`wrap`attribut définis sur. `true` Type utilisateur Les types d'utilisateur suivants peuvent exécuter cette commande. • Utilisateurs de chiffrement (CU) Prérequis	Encapsule une clé de charge utile à l'aide d'une clé publique RSA sur le HSM et du mécanisme d'encapsulage RSA-PKCS.	CU

Command	Description	Type d'utilisateur
login	Connectez-vous à votre AWS CloudHSM cluster.	Administrateur, utilisateur du chiffrement (CU) et utilisateur de l'appareil (AU)
logout	Déconnectez-vous de votre AWS CloudHSM cluster.	Administrateur, CU et utilisateur de l'appareil (AU)
quorum token-sign delete	Supprime un ou plusieurs jetons pour un service autorisé par quorum.	Administrateur
quorum token-sign generate	Génère un jeton pour un service autorisé par quorum.	Administrateur
quorum token-sign list	Répertorie tous les jetons de quorum token-sign présents dans votre cluster CloudHSM.	Tout [1], y compris les utilisateurs non authentifiés. La connexion n'est pas requise.
signe du jeton quorum list-quorum-values	Répertorie les valeurs de quorum définies dans votre cluster CloudHSM.	Tout [1], y compris les utilisateurs non authentifiés. La connexion n'est pas requise.
quorum token-sign list-timeouts	Obtient le délai d'expiration du jeton en secondes pour tous les types de jetons.	Administrateur et utilisateur de chiffrement
signe du jeton quorum set-quorum-value	Définit une nouvelle valeur de quorum pour un service autorisé par quorum.	Administrateur
quorum token-sign set-timeout	Définit le délai d'expiration du jeton en secondes pour chaque type de jeton.	Administrateur

Command	Description	Type d'utilisateur
user change-mfa	Modifie la stratégie d'authentification multifactorielle (MFA) d'un utilisateur.	Administrateur, CU
user change-password	Modifie les mots de passe des utilisateurs sur les HSM. Tout utilisateur peut modifier son propre mot de passe. Les administrateur peuvent modifier le mot de passe de n'importe quel utilisateur.	Administrateur, CU
user create	Crée un utilisateur dans votre AWS CloudHSM cluster.	Administrateur
user delete	Supprime un utilisateur de votre AWS CloudHSM cluster.	Administrateur
user list	Répertorie les utilisateurs de votre AWS CloudHSM cluster.	Tout [1], y compris les utilisateurs non authentifiés. La connexion n'est pas requise.
user change-quorum token-sign register	Enregistre la stratégie de quorum token-sign par quorum pour un utilisateur.	Administrateur

Annotations

- [1] Tous les utilisateurs incluent tous les rôles répertoriés et les utilisateurs non connectés.

cluster

cluster est une catégorie parent pour un groupe de commandes qui, lorsqu'elles sont combinées à la catégorie parent, créent une commande spécifique aux utilisateurs. Actuellement, la catégorie utilisateur comprend les commandes suivantes :

- activation du cluster
- cluster hsm-info

activation du cluster

Utilisez la commande cluster activate de l'interface de ligne de commande CloudHSM pour activer un nouveau cluster. Cette commande doit être exécutée avant que le cluster puisse être utilisé pour effectuer des opérations cryptographiques.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- administrateur non activé

Syntaxe

Cette commande n'a pas de paramètres.

```
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')
```

Exemple

Cette commande active votre cluster en définissant le mot de passe initial de votre utilisateur administrateur.

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

Rubriques en relation

- Créer un utilisateur
- Supprimer un utilisateur
- Modifier un mot de passe utilisateur

cluster hsm-info

Utilisez la commande cluster hsm-info dans l'interface de ligne de commande de CloudHSM pour répertorier les HSM de votre cluster. Vous n'avez pas besoin d'être connecté à la CLI CloudHSM pour exécuter cette commande.

> **Note**
>
> Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration utilisés par le AWS CloudHSM client et les outils de ligne de commande. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Tous les utilisateurs. Vous n'avez pas besoin d'être connecté pour exécuter cette commande.

Syntaxe

```
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
```

Exemple

Cette commande répertorie les HSM présents dans votre AWS CloudHSM cluster.

```
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]"
      }
    ]
  }
}
```

L'objet possède les attributs suivants :

- Fournisseur : nom du fournisseur du HSM.
- Modèle : le numéro de modèle du HSM.
- Numéro de série : le numéro de série du HSM. Cela peut changer en raison des remplacements.
- H ardware-version-major : La version matérielle majeure.
- H ardware-version-minor : La version matérielle mineure.
- F irmware-version-major : La version principale du microprogramme.
- F irmware-version-minor : Version mineure du microprogramme.
- F irmware-build-number : Numéro de version du microprogramme.
- Firmware-id: ID du microprogramme, qui inclut les versions majeures et mineures ainsi que le build.

Rubriques en relation

- activation du cluster

crypto

cryptoest une catégorie parent pour un groupe de commandes qui, lorsqu'elles sont combinées à la catégorie parent, créent une commande spécifique aux opérations cryptographiques. Actuellement, cette catégorie comprend les commandes suivantes :

- signe cryptographique
  - signe cryptographique ecdsa
  - signe cryptographique rsa-pkcs
  - signe cryptographique rsa-pkcs-pss
- vérification cryptographique
  - Crypto Verify ECDSA
  - vérification cryptographique rsa-pkcs
  - vérification cryptographique rsa-pkcs-pss

signe cryptographique

crypto signest une catégorie parent pour un groupe de commandes qui, lorsqu'il est combiné à la catégorie parent, utilise une clé privée choisie dans votre AWS CloudHSM cluster pour générer une signature. crypto signcomporte les sous-commandes suivantes :

- signe cryptographique ecdsa
- signe cryptographique rsa-pkcs
- signe cryptographique rsa-pkcs-pss

Pour l'utilisercrypto sign, vous devez disposer d'une clé privée dans votre HSM. Vous pouvez générer une clé privée à l'aide des commandes suivantes :

- clé generate-asymmetric-pair ec
- clé generate-asymmetric-pair RSA

signe cryptographique ecdsa

La crypto sign ecdsa commande génère une signature à l'aide d'une clé privée EC et du mécanisme de signature ECDSA.

Pour utiliser la crypto sign ecdsa commande, vous devez d'abord disposer d'une clé privée EC dans votre AWS CloudHSM cluster. Vous pouvez générer une clé privée EC à l'aide de la clé generate-asymmetric-pair ec commande dont l'`sign`attribut est défini sur`true`.

> **Note**
>
> Les signatures peuvent être vérifiées à l' AWS CloudHSM aide de vérification cryptographique sous-commandes.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Ces exemples montrent comment générer une signature crypto sign ecdsa à l'aide du mécanisme de signature et de la fonction de SHA256 hachage ECDSA. Cette commande utilise une clé privée dans le HSM.

Example Exemple : génération d'une signature pour les données codées en base 64

```
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 Exemple : génération d'une signature pour un fichier de données

```
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=="
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<DATA>

Données codées en Base64 à signer.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<DATA_PATH>

Spécifie l'emplacement des données à signer.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<HASH_FUNCTION>

Spécifie la fonction de hachage.

Valeurs valides :

- sha1
- sha224
- sha256
- sha384
- sha512

Obligatoire : oui

<KEY_FILTER>

Référence clé (par exemple key-reference=0xabc) ou liste séparée par des espaces d'attributs clés sous la forme attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE pour sélectionner une clé correspondante.

Pour obtenir la liste des attributs clés de la CLI CloudHSM pris en charge, consultez la section Attributs clés de la CLI CloudHSM.

Obligatoire : oui

Rubriques en relation

- signe cryptographique

- vérification cryptographique

signe cryptographique rsa-pkcs

La crypto sign rsa-pkcs commande génère une signature à l'aide d'une clé privée RSA et du mécanisme de signature RSA-PKCS.

Pour utiliser la crypto sign rsa-pkcs commande, vous devez d'abord disposer d'une clé privée RSA dans votre AWS CloudHSM cluster. Vous pouvez générer une clé privée RSA à l'aide de la clé generate-asymmetric-pair RSA commande dont l'`sign`attribut est défini sur. `true`

> **Note**
>
> Les signatures peuvent être vérifiées à l' AWS CloudHSM aide de vérification cryptographique sous-commandes.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Ces exemples montrent comment générer une signature crypto sign rsa-pkcs à l'aide du mécanisme de signature et SHA256 de la fonction de hachage RSA-PKCS. Cette commande utilise une clé privée dans le HSM.

Example Exemple : génération d'une signature pour les données codées en base 64

```
aws-cloudhsm > crypto sign rsa-pkcs --key-filter attr.label=rsa-private --hash-function
 sha256 --data YWJjMTIz
{
  "error_code": 0,
  "data": {
    "key-reference": "0x0000000007008db",
    "signature": "XJ7mRyHnDRYrDWTQuuNb
+5mhoXx7VTsPMjgOQW4iMN7E42eNHj2Q0oovMmBdHUEH0F4HYG8FBJOBhvGuM8J/
z6y41GbowVpUT6WzjnIQs79K9i7i6oR1TYjLnIS3r/zkimuXcS8/ZxyDzru+GO9BUT9FFU/
of9cvu4Oyn6a5+IXuCbKNQs19uASuFARUTZ0a0Ny1CB1MulxUpqGTmI91J6evlP7k/2khwDmJ5E8FEar5/
Cvbn9t21p3Uj561ngTXrYbIZ2KHpef9jQh/cEIvFLG61sexJjQi8EdTxeDA
+I3ITO0qrvvESvA9+Sj7kdG2ceIicFS8/8LwyxiIC31UHQ=="
  }
}
```

Example Exemple : génération d'une signature pour un fichier de données

```
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=="
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<DATA>

Données codées en Base64 à signer.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<DATA_PATH>

Spécifie l'emplacement des données à signer.

Obligatoire : Oui (sauf indication contraire dans les données)

<HASH_FUNCTION>

Spécifie la fonction de hachage.

Valeurs valides :

- sha1
- sha224
- sha256
- sha384
- sha512

Obligatoire : oui

<KEY_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé correspondante.

Pour obtenir la liste des attributs clés de la CLI CloudHSM pris en charge, consultez la section Attributs clés de la CLI CloudHSM.

Obligatoire : oui

Rubriques en relation

- signe cryptographique
- vérification cryptographique

signe cryptographique rsa-pkcs-pss

La crypto sign rsa-pkcs-pss commande génère une signature à l'aide d'une clé privée RSA et du mécanisme de `RSA-PKCS-PSS` signature.

Pour utiliser la crypto sign rsa-pkcs-pss commande, vous devez d'abord disposer d'une clé privée RSA dans votre AWS CloudHSM cluster. Vous pouvez générer une clé privée RSA à l'aide de la clé generate-asymmetric-pair RSA commande dont l'`sign`attribut est défini sur. `true`

> **Note**
>
> Les signatures peuvent être vérifiées à l' AWS CloudHSM aide de vérification cryptographique sous-commandes.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Ces exemples montrent comment générer une signature crypto sign rsa-pkcs-pss à l'aide du mécanisme de signature et de la `RSA-PKCS-PSS` fonction de `SHA256` hachage. Cette commande utilise une clé privée dans le HSM.

Example Exemple : génération d'une signature pour les données codées en base 64

```
aws-cloudhsm > crypto sign rsa-pkcs-pss --key-filter attr.label=rsa-private --hash-
function sha256 --data YWJjMTIz --salt-length 10 --mgf mgf1-sha256
{
  "error_code": 0,
  "data": {
    "key-reference": "0x00000000007008db",
    "signature": "H/z1rYVMzNAa31K4amE5MTiwGxDdCTgQXCJXRBKVOVm7ZuyI0fGE4sT/BUN
+977mQEV2TqtWpTsiF2IpwGM1VfSBRt7h/g4o6YERm1tTQLl7q+AJ7uGGK37zCsWQrAo7Vy8NzPShxekePo/
ZegrB1aHWN1fE8H3IPUKqLuMDI9o1Jq6kM986ExS7YmeOIclcZkyykTWqHLQVL2C3+A2bHJZBqRcM5XoIpk8HkPypjpN
```

```
+m4FNUds30GAemoOMl6asSrEJSthaZWV53OBsDOqzA8Rt8JdhXS+GZp3vNLdL1OTBELDPweXVgAu4dBX0FOvpw/
gg6sNvuaDK4YOBv2fqKg=="
  }
}
```

Example Exemple : génération d'une signature pour un fichier de données

```
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=="
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<DATA>

Données codées en Base64 à signer.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<DATA_PATH>

Spécifie l'emplacement des données à signer.

Obligatoire : Oui (sauf indication contraire dans les données)

<HASH_FUNCTION>

Spécifie la fonction de hachage.

Valeurs valides :

- sha1
- sha224
- sha256
- sha384
- sha512

Obligatoire : oui

<KEY_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé correspondante.

Pour obtenir la liste des attributs clés de la CLI CloudHSM pris en charge, consultez la section Attributs clés de la CLI CloudHSM.

Obligatoire : oui

<MGF>

Spécifie la fonction de génération de masques.

> **Note**
>
> La fonction de hachage de la fonction de génération de masque doit correspondre à la fonction de hachage du mécanisme de signature.

Valeurs valides :

- mgf1-sha1
- mgf1-sha225
- mgf1-sha255
- mgf1-sha385
- mgf1-sha512

Obligatoire : oui

<SALT_LENGTH>

Spécifie la longueur du sel.

Obligatoire : oui

Rubriques en relation

- signe cryptographique
- vérification cryptographique

Rubriques en relation

- vérification cryptographique

vérification cryptographique

crypto verifyest une catégorie parent pour un groupe de commandes qui, lorsqu'il est combiné à la catégorie parent, confirme si un fichier a été signé par une clé donnée. crypto verifycomporte les sous-commandes suivantes :

- Crypto Verify ECDSA
- Crypto Verify RSA-PKCS
- vérification cryptographique rsa-pkcs-pss

La crypto verify commande compare un fichier signé à un fichier source et analyse s'ils sont liés cryptographiquement en fonction d'une clé publique et d'un mécanisme de signature donnés.

> **Note**
>
> Les fichiers peuvent être connectés à l' AWS CloudHSM aide de signe cryptographique cette opération.

Crypto Verify ECDSA

La crypto verify ecdsa commande est utilisée pour effectuer les opérations suivantes :

- Confirmez qu'un fichier a été signé dans le HSM à l'aide d'une clé publique donnée.
- Vérifiez que la signature a été générée à l'aide du mécanisme de signature ECDSA.
- Comparez un fichier signé à un fichier source et déterminez si les deux sont liés cryptographiquement sur la base d'une clé publique ecdsa et d'un mécanisme de signature donnés.

Pour utiliser la crypto verify ecdsa commande, vous devez d'abord disposer d'une clé publique EC dans votre AWS CloudHSM cluster. Vous pouvez importer une clé publique EC à l'aide de la clé d'importation pem commande dont l'`verify`attribut est défini sur`true`.

> **Note**
>
> Vous pouvez générer une signature dans la CLI signe cryptographique CloudHSM à l'aide de sous-commandes.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Ces exemples montrent comment vérifier une signature générée crypto verify ecdsa à l'aide du mécanisme de signature et de la fonction de SHA256 hachage ECDSA. Cette commande utilise une clé publique dans le HSM.

Example Exemple : vérifier une signature codée en Base64 avec des données codées en Base64

```
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 Exemple : vérifier un fichier de signature avec un fichier de données

```
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 Exemple : prouver une fausse relation de signature

Cette commande vérifie si les données situées dans ont `/home/data` été signées par une clé publique avec l'étiquette `ecdsa-public` en utilisant le mécanisme de signature ECDSA pour produire la signature située dans. `/home/signature` Comme les arguments fournis ne constituent pas une véritable relation de signature, la commande renvoie un message d'erreur.

```
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"
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<DATA>

Données codées en Base64 à signer.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<DATA_PATH>

Spécifie l'emplacement des données à signer.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<HASH_FUNCTION>

Spécifie la fonction de hachage.

Valeurs valides :

- sha1
- sha224
- sha256
- sha384
- sha512

Obligatoire : oui

<KEY_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé correspondante.

Pour obtenir la liste des attributs clés de la CLI CloudHSM pris en charge, consultez la section Attributs clés de la CLI CloudHSM.

Obligatoire : oui

<SIGNATURE>

Signature codée en Base64.

Obligatoire : Oui (sauf indication contraire via le chemin de signature)

<SIGNATURE_PATH>

Spécifie l'emplacement de la signature.

Obligatoire : Oui (sauf indication contraire via le chemin de signature)

Rubriques en relation

- signe cryptographique
- vérification cryptographique

vérification cryptographique rsa-pkcs

La crypto verify rsa-pkcs commande est utilisée pour effectuer les opérations suivantes :

- Confirmez qu'un fichier a été signé dans le HSM à l'aide d'une clé publique donnée.
- Vérifiez que la signature a été générée à l'aide du mécanisme de `RSA-PKCS` signature.
- Comparez un fichier signé à un fichier source et déterminez si les deux sont liés cryptographiquement en fonction d'une clé publique RSA et d'un mécanisme de signature donnés.

Pour utiliser la crypto verify rsa-pkcs commande, vous devez d'abord disposer d'une clé publique RSA dans votre AWS CloudHSM cluster.

> **Note**
>
> Vous pouvez générer une signature à l'aide de la signe cryptographique CLI CloudHSM avec les sous-commandes.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Ces exemples montrent comment vérifier une signature générée crypto verify rsa-pkcs à l'aide du mécanisme de signature et SHA256 de la fonction de hachage RSA-PKCS. Cette commande utilise une clé publique dans le HSM.

Example Exemple : vérifier une signature codée en Base64 avec des données codées en Base64

```
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 Exemple : vérifier un fichier de signature avec un fichier de données

```
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 Exemple : prouver une fausse relation de signature

Cette commande vérifie si les données non valides ont été signées par une clé publique avec l'étiquette `rsa-public` en utilisant le mécanisme de signature RSAPKCS pour produire la signature située dans. `/home/signature` Comme les arguments fournis ne constituent pas une véritable relation de signature, la commande renvoie un message d'erreur.

```
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"
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<DATA>

Données codées en Base64 à signer.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<DATA_PATH>

Spécifie l'emplacement des données à signer.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<HASH_FUNCTION>

Spécifie la fonction de hachage.

Valeurs valides :

- sha1
- sha224
- sha256
- sha384
- sha512

Obligatoire : oui

<KEY_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé correspondante.

Pour obtenir la liste des attributs clés de la CLI CloudHSM pris en charge, consultez la section Attributs clés de la CLI CloudHSM.

Obligatoire : oui

<SIGNATURE>

Signature codée en Base64.

Obligatoire : Oui (sauf indication contraire via le chemin de signature)

<SIGNATURE_PATH>

Spécifie l'emplacement de la signature.

Obligatoire : Oui (sauf indication contraire via le chemin de signature)

Rubriques en relation

- signe cryptographique
- vérification cryptographique

vérification cryptographique rsa-pkcs-pss

La crypto sign rsa-pkcs-pss commande est utilisée pour effectuer les opérations suivantes.

- Confirmez qu'un fichier a été signé dans le HSM à l'aide d'une clé publique donnée.
- Vérifiez que la signature a été générée à l'aide du mécanisme de signature RSA-PKCS-PSS.
- Comparez un fichier signé à un fichier source et déterminez si les deux sont liés cryptographiquement en fonction d'une clé publique RSA et d'un mécanisme de signature donnés.

Pour utiliser la crypto verify rsa-pkcs-pss commande, vous devez d'abord disposer d'une clé publique RSA dans votre AWS CloudHSM cluster. Vous pouvez importer une clé publique RSA à l'aide de la commande key import pem (ADD UNWRAP LINK HERE) avec l'`verify`attribut défini sur. `true`

> **Note**
>
> Vous pouvez générer une signature à l'aide de la signe cryptographique CLI CloudHSM avec les sous-commandes.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Ces exemples montrent comment vérifier une signature générée crypto verify rsa-pkcs-pss à l'aide du mécanisme de signature et de la fonction de hachage RSA-PKCS-PSS. SHA256 Cette commande utilise une clé publique dans le HSM.

Example Exemple : vérifier une signature codée en Base64 avec des données codées en Base64

```
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 Exemple : vérifier un fichier de signature avec un fichier de données

```
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 Exemple : prouver une fausse relation de signature

Cette commande vérifie si les données non valides ont été signées par une clé publique avec l'étiquette `rsa-public` en utilisant le mécanisme de signature RSAPKCSPSS pour produire la signature située dans. `/home/signature` Comme les arguments fournis ne constituent pas une véritable relation de signature, la commande renvoie un message d'erreur.

```
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"
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<DATA>

Données codées en Base64 à signer.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<DATA_PATH>

Spécifie l'emplacement des données à signer.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<HASH_FUNCTION>

Spécifie la fonction de hachage.

Valeurs valides :

- sha1
- sha224
- sha256
- sha384
- sha512

Obligatoire : oui

<KEY_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé correspondante.

Pour obtenir la liste des attributs clés de la CLI CloudHSM pris en charge, consultez la section Attributs clés de la CLI CloudHSM.

Obligatoire : oui

<MFG>

Spécifie la fonction de génération de masques.

> **Note**
>
> La fonction de hachage de la fonction de génération de masque doit correspondre à la fonction de hachage du mécanisme de signature.

Valeurs valides :

- mgf1-sha1
- mgf1-sha225
- mgf1-sha255
- mgf1-sha385
- mgf1-sha512

Obligatoire : oui

<SIGNATURE>

Signature codée en Base64.

Obligatoire : Oui (sauf indication contraire via le chemin de signature)

<SIGNATURE_PATH>

Spécifie l'emplacement de la signature.

Obligatoire : Oui (sauf indication contraire via le chemin de signature)

Rubriques en relation

- signe cryptographique
- vérification cryptographique

clé

key est une catégorie parent pour un groupe de commandes qui, lorsqu'elles sont combinées à la catégorie parent, créent une commande spécifique aux clés. Actuellement, cette catégorie comprend les commandes suivantes :

- supprimer une clé

- key generate-file
- clé generate-asymmetric-pair
  - clé generate-asymmetric-pair RSA
  - clé generate-asymmetric-pair ec
- key generate-symmetric
  - key generate-symmetric aes
  - clé generate-symmetric generic-secret
- clé d'importation pem
- liste des clés
- répliquer la clé
- clé set-attribute
- partage de clés
- annuler le partage d'une clé
- déballage des clés
- étui pour clés

supprimer une clé

Utilisez la key delete commande de la CLI CloudHSM pour supprimer une clé AWS CloudHSM d'un cluster. Vous pouvez supprimer une seule clé à la fois. La suppression d'une clé dans une paire de clés n'a aucun effet sur l'autre clé de la paire.

Seul le CU qui a créé la clé et qui en est donc propriétaire peut supprimer la clé. Les utilisateurs qui partagent la clé, sans la posséder, peuvent l'utiliser dans des opérations de chiffrement, mais pas la supprimer.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

```
aws-cloudhsm > key delete --filter attr.label="ec-test-public-key"
{
  "error_code": 0,
  "data": {
    "message": "Key deleted successfully"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé correspondante à supprimer.

Pour obtenir la liste des attributs de clé de la CLI CloudHSM pris en charge, voir Attributs de clé de la CLI CloudHSM

Obligatoire : oui

Rubriques en relation

- liste des clés
- key generate-file
- annuler le partage d'une clé
- Attributs de clé de la CLI CloudHSM
- Utilisation de la CLI CloudHSM pour filtrer les clés

key generate-file

La key generate-file commande exporte une clé asymétrique depuis le HSM. Si la cible est une clé privée, la référence à la clé privée sera exportée au faux format PEM. Si la cible est une clé publique, les octets de la clé publique seront exportés au format PEM.

Le faux fichier PEM, qui ne contient pas le contenu de la clé privée mais fait référence à la clé privée dans le HSM, peut être utilisé pour établir un déchargement SSL/TLS de votre serveur Web vers. AWS CloudHSM Pour plus d'informations, consultez la section Déchargement SSL/TLS.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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')
```

Exemple

Cet exemple montre comment key generate-file générer un fichier clé dans votre AWS CloudHSM cluster.

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"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé correspondante à supprimer.

Pour obtenir la liste des attributs clés de la CLI CloudHSM pris en charge, voir Attributs de clé de la CLI CloudHSM

Obligatoire : non

<ENCODING>

Spécifie le format de codage pour le fichier clé

Obligatoire : oui

<PATH>

Spécifie le chemin du fichier dans lequel le fichier clé sera écrit

Obligatoire : oui

Rubriques en relation

- Attributs de clé de la CLI CloudHSM
- Utilisation de la CLI CloudHSM pour filtrer les clés
- clé generate-asymmetric-pair
- Clé generate-symmetric

clé generate-asymmetric-pair

key generate-asymmetric-pair est une catégorie parent pour un groupe de commandes qui, lorsqu'elles sont combinées à la catégorie parent, créent une commande qui génère des paires de touches asymétriques. Actuellement, cette catégorie comprend les commandes suivantes :

- clé generate-asymmetric-pair ec
- clé generate-asymmetric-pair RSA

clé generate-asymmetric-pair ec

Utilisez la key asymmetric-pair ec commande de la CLI CloudHSM pour générer une paire de clés à courbe elliptique asymétrique (EC) dans votre cluster. AWS CloudHSM

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Ces exemples montrent comment utiliser la commande key generate-asymmetric-pair ec pour créer une paire de clés EC.

Example Exemple : créer une paire de clés EC

```
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 Exemple : créer une paire de clés EC avec des attributs facultatifs

```
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"
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<CURVE>

Spécifie l'identifiant de la courbe elliptique.

- prime256v1
- secp256r1
- secp224r1

- secp384r1
- secp256k1
- secp521r1

Obligatoire : oui

<PUBLIC_KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs clés à définir pour la clé publique EC générée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple, `token=true`)

Pour obtenir la liste des attributs de clés pris en charge, consultez Attributs de clé de la CLI CloudHSM.

Obligatoire : non

<PUBLIC_LABEL>

Spécifie une étiquette définie par l'utilisateur pour public-key. La taille maximale autorisée `label` est de 127 caractères pour le SDK client 5.11 et versions ultérieures. Le SDK client 5.10 et les versions antérieures sont limités à 126 caractères.

Obligatoire : oui

<PRIVATE_KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs clés à définir pour la clé privée EC générée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple, `token=true`)

Pour obtenir la liste des attributs de clés pris en charge, consultez Attributs de clé de la CLI CloudHSM.

Obligatoire : non

<PRIVATE_LABEL>

Spécifie l'étiquette définie par l'utilisateur pour private-key. La taille maximale autorisée `label` est de 127 caractères pour le SDK client 5.11 et versions ultérieures. Le SDK client 5.10 et les versions antérieures sont limités à 126 caractères.

Obligatoire : oui

<SESSION>

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Par défaut, les clés générées sont des clés persistantes (jetons). La transmission vers **<**SESSION**>** change cela, garantissant qu'une clé générée avec cet argument est une clé de session (éphémère).

Obligatoire : non

Rubriques en relation

- Attributs de clé de la CLI CloudHSM
- Utilisation de la CLI CloudHSM pour filtrer les clés

clé generate-asymmetric-pair RSA

Utilisez la key generate-asymmetric-pair rsa commande pour générer une paire de clés RSA asymétrique dans votre AWS CloudHSM cluster.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Ces exemples montrent comment utiliser `key generate-asymmetric-pair rsa` pour créer une paire de clés RSA.

Example Exemple : créer une paire de clés RSA

```
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 Exemple : créer une paire de clés RSA avec des attributs facultatifs

```
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
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<MODULUS_SIZE_BITS>

Indique la longueur du module en bits. La valeur minimale est de 2048.

Obligatoire : oui

<PRIVATE_KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs clés à définir pour la clé privée RSA générée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple, `token=true`)

Pour obtenir la liste des attributs de clés pris en charge, consultez Attributs de clé de la CLI CloudHSM.

Obligatoire : non

<PRIVATE_LABEL>

Spécifie l'étiquette définie par l'utilisateur pour private-key. La taille maximale autorisée `label` est de 127 caractères pour le SDK client 5.11 et versions ultérieures. Le SDK client 5.10 et versions antérieures est limité à 126 caractères.

Obligatoire : oui

<PUBLIC_EXPONENT>

Spécifie l' exposant public. La valeur doit être un entier impair supérieur ou égal à 65 537.

Obligatoire : oui

<PUBLIC_KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs clés à définir pour la clé publique RSA générée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple, `token=true`)

Pour obtenir la liste des attributs de clés pris en charge, consultez .Attributs de clé de la CLI CloudHSM

Obligatoire : non

<PUBLIC_LABEL>

Spécifie une étiquette définie par l'utilisateur pour public-key. La taille maximale autorisée `label` est de 127 caractères pour le SDK client 5.11 et versions ultérieures. Le SDK client 5.10 et versions antérieures est limité à 126 caractères.

Obligatoire : oui

<SESSION>

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Par défaut, les clés générées sont des clés persistantes (jetons). La transmission vers <SESSION> change cela, garantissant qu'une clé générée avec cet argument est une clé de session (éphémère).

Obligatoire : non

Rubriques en relation

- Attributs de clé de la CLI CloudHSM
- Utilisation de la CLI CloudHSM pour filtrer les clés

Clé generate-symmetric

key generate-symmetric est une catégorie parent pour un groupe de commandes qui, lorsqu'elles sont combinées à la catégorie parent, créent une commande qui génère des clés symétriques. Actuellement, cette catégorie comprend les commandes suivantes :

- clé generate-symmetric aes
- clé generate-symmetric generic-secret

clé generate-symmetric aes

La key generate-symmetric aes commande génère une clé AES symétrique dans votre AWS CloudHSM cluster.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Ces exemples montrent comment utiliser la commande key generate-symmetric aes pour créer une clé AES.

Example Exemple : création d'une clé AES

```
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 Exemple : création d'une clé AES avec des attributs facultatifs

```
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
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs de clés à définir pour la clé AES générée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple, `token=true`).

Pour obtenir la liste des attributs de clés pris en charge, consultez .Attributs de clé de la CLI CloudHSM

Obligatoire : non

<KEY-LENGTH-BYTES>

Spécifie la taille de la clé en octets.

Valeurs valides :

- 16, 24 et 32

Obligatoire : oui

<LABEL>

Spécifie une étiquette définie par l'utilisateur pour la clé AES. La taille maximale autorisée `label` est de 127 caractères pour le SDK client 5.11 et versions ultérieures. Le SDK client 5.10 et versions antérieures est limité à 126 caractères.

Obligatoire : oui

<SESSION>

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Par défaut, les clés générées sont des clés persistantes (jetons). La transmission vers <SESSION> change cela, garantissant qu'une clé générée avec cet argument est une clé de session (éphémère).

Obligatoire : non

Rubriques en relation

- Attributs de clé de la CLI CloudHSM
- Utilisation de la CLI CloudHSM pour filtrer les clés

clé generate-symmetric generic-secret

La commande key generate-asymmetric-pair génère une clé secrète générique symétrique dans votre cluster AWS CloudHSM.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Ces exemples montrent comment utiliser la commande key generate-symmetric generic-secret pour créer une clé secrète générique.

Example Exemple : création d'une clé secrète générique

```
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 Exemple : création d'une clé secrète générique avec des attributs facultatifs

```
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
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs de clés à définir pour la clé AES générée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple, `token=true`).

Pour obtenir la liste des attributs de clés pris en charge, consultez .Attributs de clé de la CLI CloudHSM

Obligatoire : non

<KEY-LENGTH-BYTES>

Spécifie la taille de la clé en octets.

Valeurs valides :

- 1 à 800

Obligatoire : oui

<LABEL>

Spécifie une étiquette définie par l'utilisateur pour la clé secrète générique. La taille maximale autorisée `label` est de 127 caractères pour le SDK client 5.11 et versions ultérieures. Le SDK client 5.10 et les versions antérieures sont limités à 126 caractères.

Obligatoire : oui

<SESSION>

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Par défaut, les clés générées sont des clés persistantes (jetons). La transmission vers <SESSION> change cela, garantissant qu'une clé générée avec cet argument est une clé de session (éphémère).

Obligatoire : non

Rubriques en relation

- Attributs de clé de la CLI CloudHSM
- Utilisation de la CLI CloudHSM pour filtrer les clés

clé d'importation pem

La key import pem commande in AWS CloudHSM importe une clé au format PEM dans un HSM. Vous pouvez l'utiliser pour importer des clés publiques générées hors du HSM.

> **Note**
>
> Utilisez la key generate-file commande pour créer un fichier PEM standard à partir d'une clé publique ou pour créer un fichier PEM de référence à partir d'une clé privée.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Cet exemple montre comment utiliser la key import pem commande pour importer une clé publique RSA à partir d'un fichier au format PEM.

Example Exemple : importation d'une clé publique RSA

```
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 Exemple : importation d'une clé publique RSA avec des attributs facultatifs

```
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"
  }
}
```


Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<PATH>

Spécifie le chemin du fichier où se trouve le fichier clé.

Obligatoire : oui

<LABEL>

Spécifie une étiquette définie par l'utilisateur pour la clé importée. La taille maximale autorisée pour `label` est de 126 caractères.

Obligatoire : oui

<KEY_TYPE_CLASS>

Type de clé et classe de clé encapsulée.

Valeurs possibles :

- ec-public
- rsa-public

Obligatoire : oui

<IMPORT_KEY_ATTRIBUTES>

Spécifie une liste séparée par des espaces d'attributs clés à définir pour la clé importée sous la forme `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` (par exemple,`token=true`). Pour obtenir la liste des attributs de clés pris en charge, consultez Attributs de clé de la CLI CloudHSM.

Obligatoire : non

Rubriques en relation

- signe cryptographique
- vérification cryptographique

liste des clés

La key list commande trouve toutes les clés de l'utilisateur actuel présent dans votre AWS CloudHSM cluster. La sortie inclut les clés que l'utilisateur possède et partage, ainsi que toutes les clés publiques des HSM dans le cluster CloudHSM.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Syntaxe

```
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
```

Exemples

Les exemples suivants montrent les différentes manières d'exécuter la commande key list.

Example Exemple : rechercher toutes les clés - par défaut

Cette commande répertorie les clés de l'utilisateur connecté présent dans le AWS CloudHSM cluster.

> **Note**
>
> Par défaut, seules 10 touches de l'utilisateur actuellement connecté sont affichées, et seules les touches `key-reference` et `label` sont affichées en sortie. Utilisez les options de pagination appropriées pour afficher plus ou moins de clés en sortie.

```
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 Exemple : rechercher toutes les clés - détaillé

La sortie inclut les clés que l'utilisateur possède et partage, ainsi que toutes les clés publiques des HSM.

> **Note**
>
> Remarque : Par défaut, seules 10 clés de l'utilisateur actuellement connecté sont affichées. Utilisez les options de pagination appropriées pour afficher plus ou moins de clés en sortie.

```
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 Exemple : retour paginé

L'exemple suivant affiche un sous-ensemble paginé des clés qui ne montre que deux clés. L'exemple fournit ensuite un appel suivant pour afficher les deux touches suivantes.

```
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"
  }
}
```

Pour afficher les 2 touches suivantes, un appel suivant peut être effectué :

```
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"
  }
}
```

Pour d'autres exemples illustrant le fonctionnement du mécanisme de filtration des clés dans la CLI CloudHSM, consultez Utilisation de la CLI CloudHSM pour filtrer les clés.

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner les clés correspondantes à répertorier.

Pour obtenir la liste des attributs clés de la CLI CloudHSM pris en charge, voir Attributs de clé de la CLI CloudHSM

Obligatoire : non

<MAX_ITEMS>

Le nombre total d'éléments à renvoyer dans la sortie de la commande. Si le nombre total d'éléments disponibles est supérieur à la valeur spécifiée, un jeton NextToken est fourni dans la sortie de la commande. Pour reprendre la pagination, fournissez la valeur de next-token dans l'argument starting-token d'une commande suivante.

Obligatoire : non

<STARTING_TOKEN>

Jeton permettant de spécifier où commencer la pagination. Il s'agit du jeton next-token d'une réponse tronquée précédemment.

Obligatoire : non

<VERBOSE>

Le cas échéant, imprime tous les attributs et informations clés pour chaque clé correspondante. Par défaut, chaque clé correspondante affiche uniquement sa référence de clé et son attribut d'étiquette.

Obligatoire : non

Rubriques en relation

- supprimer une clé
- key generate-file
- annuler le partage d'une clé
- Attributs de clé de la CLI CloudHSM

- Utilisation de la CLI CloudHSM pour filtrer les clés

répliquer la clé

La key replicate commande réplique une clé d'un AWS CloudHSM cluster source vers un cluster de destination AWS CloudHSM .

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

> **Note**
>
> Les utilisateurs de cryptomonnaies doivent posséder la clé pour utiliser cette commande.

Prérequis

- Les clusters source et de destination doivent être des clones. Cela signifie que l'un a été créé à partir d'une sauvegarde de l'autre, ou qu'ils ont tous deux été créés à partir d'une sauvegarde commune. Pour plus d'informations, consultez Création de clusters à partir de sauvegardes.
- Le propriétaire de la clé doit exister sur le cluster de destination. En outre, si la clé est partagée avec des utilisateurs, ceux-ci doivent également exister sur le cluster de destination.
- Pour exécuter cette commande, vous devez être connecté en tant que CU sur les clusters source et de destination.
  - En mode commande unique, la commande utilisera les variables d'environnement CLOUDHSM_PIN et CLOUDHSM_ROLE pour s'authentifier sur le cluster source. Pour plus d'informations, consultez Mode commande unique. Pour fournir des informations d'identification pour le cluster de destination, vous devez définir deux variables environnementales supplémentaires : DESTINATION_CLOUDHSM_PIN et DESTINATION_CLOUDHSM_ROLE :

```
$ export DESTINATION_CLOUDHSM_ROLE=crypto-user
```

```
$ export DESTINATION_CLOUDHSM_PIN=username:password
```

- En mode interactif, les utilisateurs devront se connecter explicitement aux clusters source et de destination.

Syntaxe

```
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
```

Exemples

Example Exemple : clé de réplication

Cette commande réplique une clé d'un cluster source vers un cluster de destination cloné.

```
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"
  }
}
```

Arguments

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé correspondante sur le cluster source.

Pour obtenir la liste des attributs clés de la CLI CloudHSM pris en charge, voir Attributs de clé de la CLI CloudHSM

Obligatoire : oui

<SOURCE_CLUSTER_ID>

ID du cluster source.

Obligatoire : oui

<DESTINATION_CLUSTER_ID>

ID du cluster de destination.

Obligatoire : oui

Rubriques en relation

- Connexion à plusieurs clusters à l'aide de la CLI

clé set-attribute

Utilisez la key set-attribute commande pour définir les attributs des clés dans votre AWS CloudHSM cluster. Seul le CU qui a créé la clé et qui en est donc propriétaire peut modifier les attributs de la clé.

Pour obtenir la liste des attributs clés qui peuvent être utilisés dans la CLI CloudHSM, consultez. Attributs de clé de la CLI CloudHSM

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Seuls les utilisateurs de chiffrement (CU) peuvent exécuter cette commande.
- Les administrateurs peuvent définir l'attribut de confiance.

Prérequis

Pour exécuter cette commande, vous devez être connecté en tant que CU. Pour définir l'attribut sécurisé, vous devez être connecté en tant qu'utilisateur administrateur.

Syntaxe

```
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
```

Exemple : définition d'un attribut clé

L'exemple suivant montre comment utiliser la commande key set-attribute pour définir l'étiquette.

Example

1. Utilisez la clé avec l'étiquette `my_key`, comme indiqué ici :

   ```
   aws-cloudhsm > key set-attribute --filter attr.label=my_key --name encrypt --value
    false
   {
     "error_code": 0,
     "data": {
       "message": "Attribute set successfully"
     }
   }
   ```

2. Utilisez la commande key list pour confirmer que l'attribut `encrypt` a changé :

```
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
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<KEY_ATTRIBUTE>

Spécifie le nom de l'attribut de la clé.

Obligatoire : oui

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé correspondante à supprimer.

Pour obtenir la liste des attributs clés de la CLI CloudHSM pris en charge, voir Attributs de clé de la CLI CloudHSM

Obligatoire : non

<KEY_ATTRIBUTE_VALUE>

Spécifie la valeur de l'attribut de la clé.

Obligatoire : oui

<KEY_REFERENCE>

Représentation hexadécimale ou décimale de la clé (comme un handle de clé).

Obligatoire : non

Rubriques en relation

- Utilisation de la CLI CloudHSM pour filtrer les clés
- Attributs de clé de la CLI CloudHSM

partage de clés

La key share commande partage une clé avec les autres unités centrales de votre AWS CloudHSM cluster.

Seul le CU qui a créé la clé et qui en est donc propriétaire peut partager la clé. Les utilisateurs avec lesquels la clé est partagée peuvent utiliser la clé dans des opérations de chiffrement, mais ils ne peuvent pas l'exporter, la supprimer ni la partager ou annuler son partage avec d'autres utilisateurs. De plus, ces utilisateurs ne peuvent pas modifier les attributs de clé.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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')
```

Exemple : partager une clé avec un autre CU

L'exemple suivant montre comment utiliser la commande key share pour partager une clé avec le CU `alice`.

Example

1. Exécutez la commande key share pour partager la clé avec `alice`.

```
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. Exécutez la commande key list.

```
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. Dans la liste ci-dessus, vérifiez que `alice` se trouve dans la liste des `shared-users`

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé correspondante à supprimer.

Pour obtenir la liste des attributs de clés pris en charge, consultez .Attributs de clé de la CLI CloudHSM

Obligatoire : oui

<USERNAME>

Spécifie un nom convivial pour l'utilisateur. La longueur maximale est de 31 caractères. Le seul caractère spécial autorisé est un trait de soulignement (_). Le nom d'utilisateur n'est pas sensible à la casse dans cette commande, il est toujours affiché en minuscules.

Obligatoire : oui

<ROLE>

Spécifie le rôle attribué à cet utilisateur. Ce paramètre est obligatoire. Pour obtenir le rôle de l'utilisateur, utilisez la commande user list. Pour plus d'informations sur les types d'utilisateur sur un HSM, consultez Comprendre les utilisateurs HSM.

Obligatoire : oui

Rubriques en relation

- Utilisation de la CLI CloudHSM pour filtrer les clés
- Attributs de clé de la CLI CloudHSM

annuler le partage d'une clé

La key unshare commande annule le partage d'une clé avec les autres UC de votre AWS CloudHSM cluster.

Seul le CU qui a créé la clé et qui en est donc propriétaire peut annuler le partage de la clé. Les utilisateurs avec lesquels la clé est partagée peuvent utiliser la clé dans des opérations de chiffrement, mais ils ne peuvent pas l'exporter, la supprimer ni la partager ou annuler son partage avec d'autres utilisateurs. De plus, ces utilisateurs ne peuvent pas modifier les attributs de clé.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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')
```

Exemple : annuler le partage d'une clé avec un autre CU

L'exemple suivant montre comment utiliser la commande key unshare pour annuler le partage d'une clé avec le CU`alice`.

Example

1. Exécutez la commande key list et filtrez en fonction de la touche spécifique avec laquelle vous souhaitez annuler le partage avec `alice`.

```
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. Confirmez que `alice` se trouve dans la sortie `shared-users` et exécutez la commande key unshare suivante pour annuler le partage de la clé avec `alice`.

```
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. Exécutez à nouveau la commande `key list` pour confirmer que la clé n'a pas été partagée avec `alice`.

```
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
    }
}
```


Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé correspondante à supprimer.

Pour obtenir la liste des attributs de clés pris en charge, consultez .Attributs de clé de la CLI CloudHSM

Obligatoire : oui

<USERNAME>

Spécifie un nom convivial pour l'utilisateur. La longueur maximale est de 31 caractères. Le seul caractère spécial autorisé est un trait de soulignement (_). Le nom d'utilisateur n'est pas sensible à la casse dans cette commande, il est toujours affiché en minuscules.

Obligatoire : oui

<ROLE>

Spécifie le rôle attribué à cet utilisateur. Ce paramètre est obligatoire. Pour obtenir le rôle de l'utilisateur, utilisez la commande user list. Pour plus d'informations sur les types d'utilisateur sur un HSM, consultez Comprendre les utilisateurs HSM.

Obligatoire : oui

Rubriques en relation

- Utilisation de la CLI CloudHSM pour filtrer les clés
- Attributs de clé de la CLI CloudHSM

déballage des clés

La commande key unwrap parent de la CLI CloudHSM importe une clé privée cryptée (encapsulée) symétrique ou asymétrique d'un fichier vers le HSM. Cette commande est conçue pour importer des clés chiffrées qui ont été encapsulées par la étui pour clés commande, mais elle peut également être utilisée pour désencapsuler des clés encapsulées avec d'autres outils. Toutefois, dans ces cas, nous vous recommandons d'utiliser les bibliothèques de logiciels PKCS # 11 ou JCE pour désencapsuler la clé.

- déballez les clés aes-gcm
- déballage des clés aes-no-pad
- déballez la clé aes-pkcs5-pad
- déballage des clés aes-zero-pad
- déballage des clés cloudhsm-aes-gcm
- clé unwrap rsa-aes
- clé unwrap rsa-oaep
- clé unwrap rsa-pkcs

déballez les clés aes-gcm

La key unwrap aes-gcm commande déballe une clé de charge utile dans le cluster à l'aide de la clé d'encapsulation AES et du mécanisme de déballage. `AES-GCM`

Les clés non emballées peuvent être utilisées de la même manière que les clés générées par AWS CloudHSM. Pour indiquer qu'ils n'ont pas été générés localement, leur `local` attribut est défini sur`false`.

Pour utiliser la key unwrap aes-gcm commande, vous devez disposer de la clé d'encapsulation AES dans votre AWS CloudHSM cluster et son `unwrap` attribut doit être défini sur`true`.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Ces exemples montrent comment utiliser la key unwrap aes-gcm commande à l'aide d'une clé AES avec la valeur unwrap d'attribut définie surtrue.

Example Exemple : désencapsuler une clé de charge utile à partir de données clés encapsulées codées en Base64

```
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 Exemple : désencapsuler une clé de charge utile fournie via un chemin de données

```
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
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé à utiliser pour le déballage.

Obligatoire : oui

<DATA_PATH>

Chemin d'accès au fichier binaire contenant les données clés encapsulées.

Obligatoire : Oui (sauf si fourni via des données codées en Base64)

<DATA>

Données clés encapsulées codées en Base64.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<ATTRIBUTES>

Liste d'attributs clés séparés par des espaces sous la forme de `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour la clé encapsulée.

Obligatoire : non

<AAD>

En tant que valeur GCM Additional Authentiated Data (AAD), en hexadécimal.

Obligatoire : non

<TAG_LENGTH_BITS>

Longueur de la balise Aes GCM en bits.

Obligatoire : oui

<KEY_TYPE_CLASS>

Type de clé et classe de clé encapsulée [valeurs possibles :`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Obligatoire : oui

<LABEL>

Étiquette pour la clé déballée.

Obligatoire : oui

<SESSION>

Crée une clé de session qui n'existe que dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Obligatoire : non

<IV>

Valeur initiale utilisée pour envelopper la clé, en hexadécimal.

Obligatoire : non

Rubriques en relation

- étui pour clés
- déballage des clés

déballage des clés aes-no-pad

La key unwrap aes-no-pad commande déballe une clé de charge utile dans le cluster à l'aide de la clé d'encapsulation AES et du mécanisme de déballage. `AES-NO-PAD`

Les clés non emballées peuvent être utilisées de la même manière que les clés générées par AWS CloudHSM. Pour indiquer qu'ils n'ont pas été générés localement, leur `local` attribut est défini sur`false`.

Pour utiliser la key unwrap aes-no-pad commande, vous devez disposer de la clé d'encapsulation AES dans votre AWS CloudHSM cluster et son `unwrap` attribut doit être défini sur`true`.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Ces exemples montrent comment utiliser la key unwrap aes-no-pad commande à l'aide d'une clé AES avec la valeur unwrap d'attribut définie sur`true`.

Example Exemple : désencapsuler une clé de charge utile à partir de données clés encapsulées codées en Base64

```
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 Exemple : désencapsuler une clé de charge utile fournie via un chemin de données

```
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
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé à déballer.

Obligatoire : oui

<DATA_PATH>

Chemin d'accès au fichier binaire contenant les données clés encapsulées.

Obligatoire : Oui (sauf si fourni via des données codées en Base64)

<DATA>

Données clés encapsulées codées en Base64.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<ATTRIBUTES>

Liste d'attributs clés séparés par des espaces sous la forme de `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour la clé encapsulée.

Obligatoire : non

<KEY_TYPE_CLASS>

Type de clé et classe de clé encapsulée [valeurs possibles :`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Obligatoire : oui

<LABEL>

Étiquette pour la clé déballée.

Obligatoire : oui

<SESSION>

Crée une clé de session qui n'existe que dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Obligatoire : non

Rubriques en relation

- étui pour clés
- déballage des clés

déballez la clé aes-pkcs5-pad

La key unwrap aes-pkcs5-pad commande déballe une clé de charge utile à l'aide de la clé d'encapsulation AES et du mécanisme de déballage. `AES-PKCS5-PAD`

Les clés non emballées peuvent être utilisées de la même manière que les clés générées par AWS CloudHSM. Pour indiquer qu'ils n'ont pas été générés localement, leur `local` attribut est défini sur`false`.

Pour utiliser la key unwrap aes-pkcs5-pad commande, vous devez disposer de la clé d'encapsulation AES dans votre AWS CloudHSM cluster et son `unwrap` attribut doit être défini sur`true`.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Ces exemples montrent comment utiliser la key unwrap aes-pkcs5-pad commande à l'aide d'une clé AES avec la valeur unwrap d'attribut définie sur`true`.

Example Exemple : désencapsuler une clé de charge utile à partir de données clés encapsulées codées en Base64

```
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 Exemple : désencapsuler une clé de charge utile fournie via un chemin de données

```
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
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé à utiliser pour le déballage.

Obligatoire : oui

<DATA_PATH>

Chemin d'accès au fichier binaire contenant les données clés encapsulées.

Obligatoire : Oui (sauf si fourni via des données codées en Base64)

<DATA>

Données clés encapsulées codées en Base64.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<ATTRIBUTES>

Liste d'attributs clés séparés par des espaces sous la forme de `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour la clé encapsulée.

Obligatoire : non

<KEY_TYPE_CLASS>

Type de clé et classe de clé encapsulée [valeurs possibles :`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Obligatoire : oui

<LABEL>

Étiquette pour la clé déballée.

Obligatoire : oui

<SESSION>

Crée une clé de session qui n'existe que dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Obligatoire : non

Rubriques en relation

- étui pour clés
- déballage des clés

déballage des clés aes-zero-pad

La key unwrap aes-zero-pad commande déballe une clé de charge utile dans le cluster à l'aide de la clé d'encapsulation AES et du mécanisme de déballage. `AES-ZERO-PAD`

Les clés non emballées peuvent être utilisées de la même manière que les clés générées par AWS CloudHSM. Pour indiquer qu'ils n'ont pas été générés localement, leur `local` attribut est défini sur`false`.

Pour utiliser la key unwrap aes-no-pad commande, vous devez disposer de la clé d'encapsulation AES dans votre AWS CloudHSM cluster et son `unwrap` attribut doit être défini sur`true`.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Ces exemples montrent comment utiliser la key unwrap aes-zero-pad commande à l'aide d'une clé AES avec la valeur unwrap d'attribut définie surtrue.

Example Exemple : désencapsuler une clé de charge utile à partir de données clés encapsulées codées en Base64

```
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 Exemple : désencapsuler une clé de charge utile fournie via un chemin de données

```
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
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé à déballer.

Obligatoire : oui

<DATA_PATH>

Chemin d'accès au fichier binaire contenant les données clés encapsulées.

Obligatoire : Oui (sauf si fourni via des données codées en Base64)

<DATA>

Données clés encapsulées codées en Base64.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<ATTRIBUTES>

Liste d'attributs clés séparés par des espaces sous la forme de `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour la clé encapsulée.

Obligatoire : non

<KEY_TYPE_CLASS>

Type de clé et classe de clé encapsulée [valeurs possibles :`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Obligatoire : oui

<LABEL>

Étiquette pour la clé déballée.

Obligatoire : oui

<SESSION>

Crée une clé de session qui n'existe que dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Obligatoire : non

Rubriques en relation

- étui pour clés
- déballage des clés

déballage des clés cloudhsm-aes-gcm

La key unwrap cloudhsm-aes-gcm commande déballe une clé de charge utile dans le cluster à l'aide de la clé d'encapsulation AES et du mécanisme de déballage. `CLOUDHSM-AES-GCM`

Les clés non emballées peuvent être utilisées de la même manière que les clés générées par AWS CloudHSM. Pour indiquer qu'ils n'ont pas été générés localement, leur `local` attribut est défini sur`false`.

Pour utiliser la key unwrap cloudhsm-aes-gcm commande, vous devez disposer de la clé d'encapsulation AES dans votre AWS CloudHSM cluster et son `unwrap` attribut doit être défini sur`true`.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Ces exemples montrent comment utiliser la key unwrap cloudhsm-aes-gcm commande à l'aide d'une clé AES avec la valeur `unwrap` d'attribut définie sur`true`.

Example Exemple : désencapsuler une clé de charge utile à partir de données clés encapsulées codées en Base64

```
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 Exemple : désencapsuler une clé de charge utile fournie via un chemin de données

```
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
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé à utiliser pour déballer.

Obligatoire : oui

<DATA_PATH>

Chemin d'accès au fichier binaire contenant les données clés encapsulées.

Obligatoire : Oui (sauf si fourni via des données codées en Base64)

<DATA>

Données clés encapsulées codées en Base64.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<ATTRIBUTES>

Liste d'attributs clés séparés par des espaces sous la forme de `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour la clé encapsulée.

Obligatoire : non

<AAD>

En tant que valeur de données authentifiées supplémentaires (AAD) GCM, en hexadécimal.

Obligatoire : non

<TAG_LENGTH_BITS>

Longueur de la balise Aes GCM en bits.

Obligatoire : oui

<KEY_TYPE_CLASS>

Type de clé et classe de clé encapsulée [valeurs possibles :`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Obligatoire : oui

<LABEL>

Étiquette pour la clé déballée.

Obligatoire : oui

<SESSION>

Crée une clé de session qui n'existe que dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Obligatoire : non

Rubriques en relation

- étui pour clés
- déballage des clés

clé unwrap rsa-aes

La key unwrap rsa-aes commande déballe une clé de charge utile à l'aide d'une clé privée RSA et du mécanisme de déballage. `RSA-AES`

Les clés non emballées peuvent être utilisées de la même manière que les clés générées par AWS CloudHSM. Pour indiquer qu'ils n'ont pas été générés localement, leur `local` attribut est défini sur`false`.

Pour utiliser lekey unwrap rsa-aes, vous devez disposer de la clé privée RSA de la clé d'encapsulation publique RSA dans votre AWS CloudHSM cluster, et son `unwrap` attribut doit être défini sur. `true`

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Ces exemples montrent comment utiliser la key unwrap rsa-aes commande à l'aide de la clé privée RSA avec la valeur unwrap d'attribut définie sur. true

Example Exemple : désencapsuler une clé de charge utile à partir de données clés encapsulées codées en Base64

```
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 Exemple : désencapsuler une clé de charge utile fournie via un chemin de données

```
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
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé à utiliser pour le déballage.

Obligatoire : oui

<DATA_PATH>

Chemin d'accès au fichier binaire contenant les données clés encapsulées.

Obligatoire : Oui (sauf si fourni via des données codées en Base64)

<DATA>

Données clés encapsulées codées en Base64.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<ATTRIBUTES>

Liste d'attributs clés séparés par des espaces sous la forme de `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour la clé encapsulée.

Obligatoire : non

<KEY_TYPE_CLASS>

Type de clé et classe de clé encapsulée [valeurs possibles :`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Obligatoire : oui

<HASH_FUNCTION>

Spécifie la fonction de hachage.

Valeurs valides :

- sha1
- sha224
- sha256
- sha384

- sha512

Obligatoire : oui

<MGF>

Spécifie la fonction de génération de masques.

> **Note**
>
> La fonction de hachage de la fonction de génération de masque doit correspondre à la fonction de hachage du mécanisme de signature.

Valeurs valides :

- mgf1-sha1
- mgf1-sha225
- mgf1-sha255
- mgf1-sha385
- mgf1-sha512

Obligatoire : oui

<LABEL>

Étiquette pour la clé déballée.

Obligatoire : oui

<SESSION>

Crée une clé de session qui n'existe que dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Obligatoire : non

Rubriques en relation

- étui pour clés
- déballage des clés

clé unwrap rsa-oaep

La key unwrap rsa-oaep commande déballe une clé de charge utile à l'aide de la clé privée RSA et du mécanisme de déballage. `RSA-OAEP`

Les clés non emballées peuvent être utilisées de la même manière que les clés générées par AWS CloudHSM. Pour indiquer qu'ils n'ont pas été générés localement, leur `local` attribut est défini sur`false`.

Pour utiliser la key unwrap rsa-oaep commande, vous devez disposer de la clé privée RSA de la clé d'encapsulation publique RSA dans votre AWS CloudHSM cluster, et son `unwrap` attribut doit être défini sur. `true`

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Ces exemples montrent comment utiliser la key unwrap rsa-oaep commande à l'aide de la clé privée RSA avec la valeur unwrap d'attribut définie sur. `true`

Example Exemple : désencapsuler une clé de charge utile à partir de données clés encapsulées codées en Base64

```
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 Exemple : désencapsuler une clé de charge utile fournie via un chemin de données

```
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
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé à utiliser pour le déballage.

Obligatoire : oui

<DATA_PATH>

Chemin d'accès au fichier binaire contenant les données clés encapsulées.

Obligatoire : Oui (sauf si fourni via des données codées en Base64)

<DATA>

Données clés encapsulées codées en Base64.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<ATTRIBUTES>

Liste d'attributs clés séparés par des espaces sous la forme de `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour la clé encapsulée.

Obligatoire : non

<KEY_TYPE_CLASS>

Type de clé et classe de clé encapsulée [valeurs possibles :`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Obligatoire : oui

<HASH_FUNCTION>

Spécifie la fonction de hachage.

Valeurs valides :

- sha1
- sha224
- sha256
- sha384
- sha512

Obligatoire : oui

<MGF>

Spécifie la fonction de génération de masques.

> **Note**
>
> La fonction de hachage de la fonction de génération de masque doit correspondre à la fonction de hachage du mécanisme de signature.

Valeurs valides :

- mgf1-sha1
- mgf1-sha225
- mgf1-sha255
- mgf1-sha385
- mgf1-sha512

Obligatoire : oui

<LABEL>

Étiquette pour la clé déballée.

Obligatoire : oui

<SESSION>

Crée une clé de session qui n'existe que dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Obligatoire : non

Rubriques en relation

- étui pour clés
- déballage des clés

clé unwrap rsa-pkcs

La key unwrap rsa-pkcs commande déballe une clé de charge utile à l'aide de la clé privée RSA et du mécanisme de déballage. `RSA-PKCS`

Les clés non emballées peuvent être utilisées de la même manière que les clés générées par AWS CloudHSM. Pour indiquer qu'ils n'ont pas été générés localement, leur `local` attribut est défini sur`false`.

Pour utiliser le unwrap rsa-pkcs raccourci clavier, vous devez disposer de la clé privée RSA de la clé d'encapsulation publique RSA dans votre AWS CloudHSM cluster, et son `unwrap` attribut doit être défini sur. `true`

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemples

Ces exemples montrent comment utiliser la key unwrap rsa-oaep commande à l'aide d'une clé AES avec la valeur unwrap d'attribut définie surtrue.

Example Exemple : déballer une clé de charge utile à partir de données clés encapsulées codées en Base64

```
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 Exemple : déballer une clé de charge utile fournie via un chemin de données

```
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
      }
    }
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé à utiliser pour déballer.

Obligatoire : oui

<DATA_PATH>

Chemin d'accès au fichier binaire contenant les données clés encapsulées.

Obligatoire : Oui (sauf si fourni via des données codées en Base64)

<DATA>

Données clés encapsulées codées en Base64.

Obligatoire : Oui (sauf indication contraire via le chemin de données)

<ATTRIBUTES>

Liste d'attributs clés séparés par des espaces sous la forme de `KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour la clé encapsulée.

Obligatoire : non

<KEY_TYPE_CLASS>

Type de clé et classe de clé encapsulée [valeurs possibles :`aes`,`des3`,`ec-private`,`generic-secret`,`rsa-private`].

Obligatoire : oui

<LABEL>

Étiquette pour la clé déballée.

Obligatoire : oui

<SESSION>

Crée une clé de session qui n'existe que dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Obligatoire : non

Rubriques en relation

- étui pour clés
- déballage des clés

étui pour clés

La key wrap commande de la CLI CloudHSM exporte une copie chiffrée d'une clé privée symétrique ou asymétrique du HSM vers un fichier. Lorsque vous exécutezkey wrap, vous spécifiez deux éléments : la clé à exporter et le fichier de sortie. La clé à exporter est une clé du HSM qui cryptera (encapsulera) la clé que vous souhaitez exporter.

La key wrap commande ne supprime pas la clé du HSM et ne vous empêche pas de l'utiliser dans des opérations cryptographiques. Vous pouvez exporter la même clé plusieurs fois. Pour réimporter la clé chiffrée dans le HSM, utilisezdéballage des clés. Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur cryptographique (CU) qui a créé la clé, peut encapsuler la clé. Les utilisateurs avec lesquels la clé est partagée ne peuvent l'utiliser que dans le cadre d'opérations cryptographiques.

La key wrap commande comprend les sous-commandes suivantes :

- porte-clés aes-gcm
- étui pour clés aes-no-pad
- porte-clés AES-PKCS5-PAD
- étui pour clés aes-zero-pad
- étui pour clés cloudhsm-aes-gcm
- porte-clés rsa-aes
- porte-clés RSA-OAEP
- porte-clés rsa-pkcs

porte-clés aes-gcm

La key wrap aes-gcm commande encapsule une clé de charge utile à l'aide d'une touche AES sur le HSM et le `AES-GCM` mécanisme d'encapsulation. L'`extractable`attribut de la clé de charge utile doit être défini sur. `true`

Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur cryptographique (CU) qui a créé la clé, peut encapsuler la clé. Les utilisateurs qui partagent la clé peuvent l'utiliser dans des opérations cryptographiques.

Pour utiliser la key wrap aes-gcm commande, vous devez d'abord disposer d'une clé AES dans votre AWS CloudHSM cluster. Vous pouvez générer une clé AES à encapsuler avec la clé generate-symmetric aes commande et l'`wrap`attribut définis sur`true`.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Cet exemple montre comment utiliser la key wrap aes-gcm commande à l'aide d'une touche AES.

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"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<PAYLOAD_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé de charge utile.

Obligatoire : oui

<PATH>

Chemin d'accès au fichier binaire dans lequel les données clés encapsulées seront enregistrées.

Obligatoire : non

<WRAPPING_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé d'encapsulation.

Obligatoire : oui

<AAD>

Valeur des données authentifiées supplémentaires (AAD) AES GCM, en hexadécimal.

Obligatoire : non

<TAG_LENGTH_BITS>

Longueur de la balise AES GCM en bits.

Obligatoire : oui

Rubriques en relation

- étui pour clés
- déballage des clés

étui pour clés aes-no-pad

La key wrap aes-no-pad commande encapsule une clé de charge utile à l'aide d'une touche AES sur le HSM et le `AES-NO-PAD` mécanisme d'encapsulation. L'`extractable`attribut de la clé de charge utile doit être défini sur. `true`

Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur cryptographique (CU) qui a créé la clé, peut encapsuler la clé. Les utilisateurs qui partagent la clé peuvent l'utiliser dans des opérations cryptographiques.

Pour utiliser la key wrap aes-no-pad commande, vous devez d'abord disposer d'une clé AES dans votre AWS CloudHSM cluster. Vous pouvez générer une clé AES pour l'encapsulage à l'aide de la clé generate-symmetric aes commande et de l'`wrap`attribut définis sur`true`.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Cet exemple montre comment utiliser la key wrap aes-no-pad commande à l'aide d'une clé AES avec la valeur `wrap` d'attribut définie sur`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"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<PAYLOAD_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé de charge utile.

Obligatoire : oui

<PATH>

Chemin d'accès au fichier binaire dans lequel les données clés encapsulées seront enregistrées.

Obligatoire : non

<WRAPPING_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé d'encapsulation.

Obligatoire : oui

Rubriques en relation

- étui pour clés
- déballage des clés

porte-clés AES-PKCS5-PAD

La key wrap aes-pkcs5-pad commande encapsule une clé de charge utile à l'aide d'une touche AES sur le HSM et le `AES-PKCS5-PAD` mécanisme d'encapsulation. L'`extractable`attribut de la clé de charge utile doit être défini sur. `true`

Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur cryptographique (CU) qui a créé la clé, peut encapsuler la clé. Les utilisateurs qui partagent la clé peuvent l'utiliser dans des opérations cryptographiques.

Pour utiliser la key wrap aes-pkcs5-pad commande, vous devez d'abord disposer d'une clé AES dans votre AWS CloudHSM cluster. Vous pouvez générer une clé AES pour l'encapsulage à l'aide de la clé generate-symmetric aes commande et de l'`wrap`attribut définis sur`true`.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Cet exemple montre comment utiliser la key wrap aes-pkcs5-pad commande à l'aide d'une clé AES avec la valeur `wrap` d'attribut définie sur`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="
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<PAYLOAD_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé de charge utile.

Obligatoire : oui

<PATH>

Chemin d'accès au fichier binaire dans lequel les données clés encapsulées seront enregistrées.

Obligatoire : non

<WRAPPING_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé d'encapsulation.

Obligatoire : oui

Rubriques en relation

- étui pour clés
- déballage des clés

étui pour clés aes-zero-pad

La key wrap aes-zero-pad commande encapsule une clé de charge utile à l'aide d'une touche AES sur le HSM et le `AES-ZERO-PAD` mécanisme d'encapsulation. L'`extractable`attribut de la clé de charge utile doit être défini sur. `true`

Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur cryptographique (CU) qui a créé la clé, peut encapsuler la clé. Les utilisateurs qui partagent la clé peuvent l'utiliser dans des opérations cryptographiques.

Pour utiliser la key wrap aes-zero-pad commande, vous devez d'abord disposer d'une clé AES dans votre AWS CloudHSM cluster. Vous pouvez générer une clé AES pour l'encapsulage à l'aide de la clé generate-symmetric aes commande dont l'`wrap`attribut est défini sur`true`.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Cet exemple montre comment utiliser la key wrap aes-zero-pad commande à l'aide d'une clé AES avec la valeur `wrap` d'attribut définie sur`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"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<PAYLOAD_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé de charge utile.

Obligatoire : oui

<PATH>

Chemin d'accès au fichier binaire dans lequel les données clés encapsulées seront enregistrées.

Obligatoire : non

<WRAPPING_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé d'encapsulation.

Obligatoire : oui

Rubriques en relation

- étui pour clés
- déballage des clés

étui pour clés cloudhsm-aes-gcm

La key wrap cloudhsm-aes-gcm commande encapsule une clé de charge utile à l'aide d'une touche AES sur le HSM et le `CLOUDHSM-AES-GCM` mécanisme d'encapsulation. L'`extractable`attribut de la clé de charge utile doit être défini sur. `true`

Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur cryptographique (CU) qui a créé la clé, peut encapsuler la clé. Les utilisateurs qui partagent la clé peuvent l'utiliser dans des opérations cryptographiques.

Pour utiliser la key wrap cloudhsm-aes-gcm commande, vous devez d'abord disposer d'une clé AES dans votre AWS CloudHSM cluster. Vous pouvez générer une clé AES à encapsuler avec la clé generate-symmetric aes commande et l'`wrap`attribut définis sur`true`.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Cet exemple montre comment utiliser la key wrap cloudhsm-aes-gcm commande à l'aide d'une touche AES.

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"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<PAYLOAD_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé de charge utile.

Obligatoire : oui

<PATH>

Chemin d'accès au fichier binaire dans lequel les données clés encapsulées seront enregistrées.

Obligatoire : non

<WRAPPING_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé d'encapsulation.

Obligatoire : oui

<AAD>

Valeur des données authentifiées supplémentaires (AAD) AES GCM, en hexadécimal.

Obligatoire : non

<TAG_LENGTH_BITS>

Longueur de la balise AES GCM en bits.

Obligatoire : oui

Rubriques en relation

- étui pour clés
- déballage des clés

porte-clés rsa-aes

La key wrap rsa-aes commande encapsule une clé de charge utile à l'aide d'une clé publique RSA sur le HSM et du mécanisme d'encapsulation RSA-AES. L'`extractable`attribut de la clé de charge utile doit être défini sur. `true`

Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur cryptographique (CU) qui a créé la clé, peut encapsuler la clé. Les utilisateurs qui partagent la clé peuvent l'utiliser dans des opérations cryptographiques.

Pour utiliser la key wrap rsa-aes commande, vous devez d'abord disposer d'une clé RSA dans votre AWS CloudHSM cluster. Vous pouvez générer une paire de clés RSA à l'aide de la clé generate-asymmetric-pair commande et de l'`wrap`attribut définis sur. `true`

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Cet exemple montre comment utiliser la key wrap rsa-ae commande à l'aide d'une clé publique RSA dont la valeur `wrap` d'attribut est définie sur. `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=="
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<PAYLOAD_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé de charge utile.

Obligatoire : oui

<PATH>

Chemin d'accès au fichier binaire dans lequel les données clés encapsulées seront enregistrées.

Obligatoire : non

<WRAPPING_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé d'encapsulation.

Obligatoire : oui

<MGF>

Spécifie la fonction de génération de masques.

> **Note**
>
> La fonction de hachage de la fonction de génération de masque doit correspondre à la fonction de hachage du mécanisme de signature.

Valeurs valides

- mgf1-sha1
- mgf1-sha225
- mgf1-sha255
- mgf1-sha385
- mgf1-sha512

Obligatoire : oui

Rubriques en relation

- étui pour clés
- déballage des clés

porte-clés RSA-OAEP

La key wrap rsa-oaep commande encapsule une clé de charge utile à l'aide d'une clé publique RSA sur le HSM et du mécanisme d'encapsulation. `RSA-OAEP` L'`extractable`attribut de la clé de charge utile doit être défini sur. `true`

Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur cryptographique (CU) qui a créé la clé, peut encapsuler la clé. Les utilisateurs qui partagent la clé peuvent l'utiliser dans des opérations cryptographiques.

Pour utiliser la key wrap rsa-oaep commande, vous devez d'abord disposer d'une clé RSA dans votre AWS CloudHSM cluster. Vous pouvez générer une paire de clés RSA à l'aide de la clé generate-asymmetric-pair commande et de l'`wrap`attribut définis sur. `true`

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Cet exemple montre comment utiliser la key wrap rsa-oaep commande à l'aide d'une clé publique RSA dont la valeur `wrap` d'attribut est définie sur. `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=="
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<PAYLOAD_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé de charge utile.

Obligatoire : oui

<PATH>

Chemin d'accès au fichier binaire dans lequel les données clés encapsulées seront enregistrées.

Obligatoire : non

<WRAPPING_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé d'encapsulation.

Obligatoire : oui

<MGF>

Spécifie la fonction de génération de masques.

> **Note**
>
> La fonction de hachage de la fonction de génération de masque doit correspondre à la fonction de hachage du mécanisme de signature.

Valeurs valides

- mgf1-sha1
- mgf1-sha225
- mgf1-sha255
- mgf1-sha385
- mgf1-sha512

Obligatoire : oui

Rubriques en relation

- étui pour clés
- déballage des clés

porte-clés rsa-pkcs

La key wrap rsa-pkcs commande encapsule une clé de charge utile à l'aide d'une clé publique RSA sur le HSM et du mécanisme d'encapsulation. `RSA-PKCS` L'`extractable`attribut de la clé de charge utile doit être défini sur. `true`

Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur cryptographique (CU) qui a créé la clé, peut encapsuler la clé. Les utilisateurs qui partagent la clé peuvent l'utiliser dans des opérations cryptographiques.

Pour utiliser la key wrap rsa-pkcs commande, vous devez d'abord disposer d'une clé RSA dans votre AWS CloudHSM cluster. Vous pouvez générer une paire de clés RSA à l'aide de la clé generate-asymmetric-pair commande et de l'`wrap`attribut définis sur. `true`

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Prérequis

- Pour exécuter cette commande, vous devez être connecté en tant que CU.

Syntaxe

```
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
```

Exemple

Cet exemple montre comment utiliser la key wrap rsa-pkcs commande à l'aide d'une clé publique RSA.

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=="
  }
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<PAYLOAD_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste séparée par des espaces d'attributs clés sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé de charge utile.

Obligatoire : oui

<PATH>

Chemin d'accès au fichier binaire dans lequel les données clés encapsulées seront enregistrées.

Obligatoire : non

<WRAPPING_FILTER>

Référence clé (par exemple,`key-reference=0xabc`) ou liste d'attributs clés séparés par des espaces sous la forme de `attr.KEY_ATTRIBUTE_NAME=KEY_ATTRIBUTE_VALUE` pour sélectionner une clé d'encapsulation.

Obligatoire : oui

Rubriques en relation

- étui pour clés
- déballage des clés

login

Vous pouvez utiliser la commande login dans la CLI CloudHSM pour vous connecter et vous déconnecter de chaque HSM dans un cluster.

> **Note**
>
> Si vous dépassez cinq tentatives de connexion incorrectes, votre compte est verrouillé. Pour déverrouiller le compte, un administrateur doit réinitialiser votre mot de passe à l'aide de la commande user change-password dans cloudhsm_cli.

Pour résoudre les problèmes de connexion et de déconnexion

Si vous avez plusieurs HSM dans votre cluster, vous pouvez autoriser des tentatives de connexion incorrectes supplémentaires avant que votre compte soit verrouillé. Cela est dû au fait que le client CloudHSM équilibre la charge sur plusieurs HSM. Par conséquent, la tentative de connexion peut ne pas commencer sur le même HSM à chaque fois. Si vous testez cette fonctionnalité, nous vous recommandons de le faire sur un cluster avec un seul HSM actif.

Si vous avez créé votre cluster avant février 2018, votre compte est verrouillé après 20 tentatives de connexion incorrectes.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Administrateur non activé
- Administrateur
- Utilisateur de chiffrement (CU)

Syntaxe

```
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')
```

Exemple

Example

Cette commande se connecte à tous les HSM dans un cluster avec les informations d'identification d'un utilisateur administrateur nommé `admin1`.

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<USERNAME>

Spécifie un nom convivial pour l'utilisateur. La longueur maximale est de 31 caractères. Le seul caractère spécial autorisé est un trait de soulignement (_). Le nom d'utilisateur n'est pas sensible à la casse dans cette commande, il est toujours affiché en minuscules.

Obligatoire : oui

<ROLE>

Spécifie le rôle attribué à cet utilisateur. Ce paramètre est obligatoire. Les valeurs valides sont admin, crypto-user.

Pour obtenir le rôle de l'utilisateur, utilisez la commande user list. Pour plus d'informations sur les types d'utilisateur sur un HSM, consultez Comprendre les utilisateurs HSM.

<PASSWORD>

Spécifie le mot de passe de l'utilisateur qui se connecte aux HSM.

Rubriques en relation

- Mise en route avec la CLI CloudHSM
- Activation du cluster

connexion mfa-token-sign

Utilisez la commande login mfa-token-sign dans le journal de la CLI CloudHSM AWS CloudHSM pour vous connecter à l'aide de l'authentification multifactorielle. Pour utiliser cette commande, vous devez d'abord configurer la MFA pour la CLI CloudHSM.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Administrateur
- Utilisateur de chiffrement (CU)

Syntaxe

```
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
```

Exemple

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"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<TOKEN>

Chemin de fichier dans lequel le fichier de jeton non signé sera écrit.

Obligatoire : oui

Rubriques en relation

- Mise en route avec la CLI CloudHSM
- Activation du cluster
- Utilisation de la CLI CloudHSM pour gérer la MFA

logout

Vous pouvez utiliser la commande logout dans la CLI CloudHSM pour vous connecter et vous déconnecter de chaque HSM dans un cluster.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Administrateur
- Utilisateur de chiffrement (CU)

Syntaxe

```
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
```

Exemple

Example

Cette commande vous déconnecte de tous les HSM d'un cluster.

```
aws-cloudhsm > logout
{
  "error_code": 0,
  "data": "Logout successful"
}
```

Rubriques en relation

- Mise en route avec la CLI CloudHSM
- Activation du cluster

utilisateur

user est une catégorie parent pour un groupe de commandes qui, lorsqu'elles sont combinées à la catégorie parent, créent une commande spécifique aux utilisateurs. Actuellement, la catégorie utilisateur comprend les commandes suivantes :

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

user change-mfa

Actuellement, cette catégorie comprend la sous-commande suivante :

- user change-mfa token-sign

user change-mfa token-sign

Utilisez la commande user change-mfa de la CLI CloudHSM pour mettre à jour la configuration de l'authentification multifactorielle (MFA) d'un compte utilisateur. N'importe quel compte utilisateur peut exécuter cette commande. Les comptes dotés du rôle d'administrateur peuvent exécuter cette commande pour d'autres utilisateurs.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Administrateur
- Utilisateur de chiffrement

Syntaxe

Actuellement, il n'existe qu'une seule stratégie multifactorielle disponible pour les utilisateurs : Token Sign.

```
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)
```

La stratégie Token Sign demande un fichier de jetons dans lequel écrire des jetons non signés.

```
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')
```

Exemple

Cette commande écrira un jeton non signé par HSM de votre cluster dans le fichier spécifié par `token`. Lorsque vous y êtes invité, signez les jetons du fichier.

Example : écrivez un jeton non signé par HSM dans votre cluster

```
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"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<ROLE>

Spécifie le rôle attribué au compte utilisateur. Ce paramètre est obligatoire. Pour plus d'informations sur les types d'utilisateur sur un HSM, consultez Comprendre les utilisateurs HSM.

Valeurs valides

- Administrateur : les administrateurs peuvent gérer les utilisateurs, mais ils ne peuvent pas gérer les clés.
- CU : les utilisateurs de chiffrement peuvent créer et gérer des clés, et utiliser ces clés dans des opérations de chiffrement.

<USERNAME>

Spécifie un nom convivial pour l'utilisateur. La longueur maximale est de 31 caractères. Le seul caractère spécial autorisé est un trait de soulignement (_).

Vous ne pouvez pas modifier le nom d'un utilisateur après l'avoir créé. Dans les commandes de la CLI CloudHSM, le rôle et le mot de passe sont sensibles à la casse, mais le nom d'utilisateur ne l'est pas.

Obligatoire : oui

<CHANGE_PASSWORD>

Spécifie le nouveau mot de passe en texte brut de l'utilisateur dont la MFA est enregistrée/désenregistrée.

Obligatoire : oui

<TOKEN>

Chemin de fichier dans lequel le fichier de jeton non signé sera écrit.

Obligatoire : oui

<APPROVAL>

Spécifie le chemin d'accès à un fichier de jeton de quorum signé pour approuver l'opération. Obligatoire uniquement si la valeur du quorum du service utilisateur du quorum est supérieure à 1.

<DEREGISTER>

Désenregistre la clé publique MFA, si elle est présente.

<CHANGE-QUORUM>

Modifie la clé publique du quorum ainsi que la clé MFA.

Rubriques en relation

- Comprendre l'authentification à deux facteurs pour les utilisateurs HSM

user change-password

Utilisez la user change-password commande de la CLI CloudHSM pour modifier le mot de passe d'un utilisateur AWS CloudHSM existant dans votre cluster. Pour activer la MFA pour un utilisateur, utilisez la commande `user change-mfa`.

Tout utilisateur peut modifier son propre mot de passe. En outre, les utilisateurs dotés du rôle d'administrateur peuvent modifier le mot de passe d'un autre utilisateur du cluster. Vous n'avez pas besoin d'entrer le mot de passe actuel pour effectuer la modification.

> **Note**
>
> Vous ne pouvez pas modifier le mot de passe d'un utilisateur qui est actuellement connecté au cluster.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Administrateur
- Utilisateur de chiffrement (CU)

Syntaxe

> **Note**
>
> Pour activer l'authentification multifactorielle (MFA) pour un utilisateur, utilisez la commande user change-mfa.

```
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')
```

Exemple

Les exemples suivants montrent comment utiliser user change-password pour réinitialiser le mot de passe de l'utilisateur actuel ou de tout autre utilisateur dans votre cluster.

Example : Modifier votre mot de passe

Tout utilisateur du cluster peut utiliser user change-password pour modifier son propre mot de passe.

La sortie suivante montre que Bob est actuellement connecté en tant qu'utilisateur de chiffrement (CU).

```
aws-cloudhsm > user change-password --username bob --role crypto-user
Enter password:
```

```
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "bob",
    "role": "crypto-user"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<APPROVAL>

Spécifie le chemin d'accès à un fichier de jeton de quorum signé pour approuver l'opération. Obligatoire uniquement si la valeur du quorum du service utilisateur du quorum est supérieure à 1.

<DEREGISTER-MFA>

Désenregistre la clé publique MFA, si elle est présente.

<DEREGISTER-QUORUM>

Désenregistrez la clé publique Quorum, si elle est présente.

<PASSWORD>

Spécifie le nouveau mot de passe de l'utilisateur en texte brut.

Obligatoire : oui

<ROLE>

Spécifie le rôle attribué au compte utilisateur. Ce paramètre est obligatoire. Pour plus d'informations sur les types d'utilisateur sur un HSM, consultez Comprendre les utilisateurs HSM.

Valeurs valides

- Administrateur : les administrateurs peuvent gérer les utilisateurs, mais ils ne peuvent pas gérer les clés.
- CU : les utilisateurs de chiffrement peuvent créer et gérer des clés, et utiliser ces clés dans des opérations de chiffrement.

<USERNAME>

Spécifie un nom convivial pour l'utilisateur. La longueur maximale est de 31 caractères. Le seul caractère spécial autorisé est un trait de soulignement (_).

Vous ne pouvez pas modifier le nom d'un utilisateur après l'avoir créé. Dans les commandes de la CLI CloudHSM, le rôle et le mot de passe sont sensibles à la casse, mais le nom d'utilisateur ne l'est pas.

Obligatoire : oui

Rubriques en relation

- Répertorier les utilisateurs
- Créer un utilisateur
- Supprimer un utilisateur

user change-quorum

user change-quorum est une catégorie parent pour un groupe de commandes qui, lorsqu'elles sont combinées à la catégorie parent, créent une commande spécifique à la modification du quorum des utilisateurs.

user change-quorum est utilisé pour enregistrer l'authentification du quorum des utilisateurs à l'aide d'une stratégie de quorum spécifiée. À partir du SDK 5.8.0, une seule stratégie de quorum est disponible pour les utilisateurs, comme indiqué ci-dessous.

Actuellement, cette catégorie comprend la catégorie et la sous-commande suivantes :

- token-sign
  - register

user change-quorum token-sign

user change-quorum token-sign est une catégorie parent pour les commandes qui, combinées à cette catégorie parent, créent une commande spécifique aux opérations de quorum token-sign.

Actuellement, cette catégorie comprend les commandes suivantes :

- register

user change-quorum token-sign register

Utilisez la commande user change-quorum token-sign register de la CLI CloudHSM pour enregistrer la stratégie de quorum token-sign pour un utilisateur administrateur.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Administrateur

Syntaxe

```
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')
```

Exemple

Example

Pour exécuter cette commande, vous devez être connecté sous le nom d'utilisateur pour lequel vous souhaitez register quorum token-sign.

```
aws-cloudhsm > login --username admin1 --role admin
Enter password:
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
```

```
}
```

La commande user change-quorum token-sign register enregistrera votre clé publique auprès du HSM. Par conséquent, vous serez qualifié d'approbateur de quorum pour les opérations nécessitant un quorum qui requièrent qu'un utilisateur obtienne des signatures de quorum pour atteindre le seuil de quorum requis.

```
aws-cloudhsm > user change-quorum token-sign register \
    --public-key /home/mypemfile \
    --signed-token /home/mysignedtoken
{
  "error_code": 0,
  "data": {
    "username": "admin1",
    "role": "admin"
  }
}
```

Vous pouvez maintenant exécuter la commande user list et confirmer que quorum token-sign a été enregistré pour cet utilisateur.

```
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"
      }
    ]
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<PUBLIC-KEY>

Chemin d'accès au fichier PEM à clé publique.

Obligatoire : oui

<SIGNED-TOKEN>

Chemin de fichier avec jeton signé par clé privée de l'utilisateur.

Obligatoire : oui

Rubriques en relation

- Utilisation de la CLI CloudHSM pour gérer l'authentification par quorum
- Utilisation de l'authentification par quorum pour les administrateurs : première configuration
- Modifier la valeur minimale du quorum pour les administrateurs
- Noms et types de services prenant en charge l'authentification par quorum

user create

La user create commande de la CLI CloudHSM crée un utilisateur AWS CloudHSM dans votre cluster. Seuls les comptes utilisateurs dotés du rôle d'administrateur peuvent exécuter cette commande.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Administrateur

Prérequis

Pour exécuter cette commande, vous devez être connecté en tant qu'utilisateur administrateur

Syntaxe

```
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')
```

Exemple

Ces exemples montrent comment utiliser user create pour créer de nouveaux utilisateurs dans vos HSM.

Example : Créer un utilisateur de chiffrement

Cet exemple crée un compte dans votre AWS CloudHSM cluster avec le rôle d'utilisateur cryptographique.

```
aws-cloudhsm > user create --username alice --role crypto-user
Enter password:
Confirm password:
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<USERNAME>

Spécifie un nom convivial pour l'utilisateur. La longueur maximale est de 31 caractères. Le seul caractère spécial autorisé est un trait de soulignement (_). Le nom d'utilisateur n'est pas sensible à la casse dans cette commande, il est toujours affiché en minuscules.

Obligatoire : oui

<ROLE>

Spécifie le rôle attribué à cet utilisateur. Ce paramètre est obligatoire. Les valeurs valides sont admin, crypto-user.

Pour obtenir le rôle de l'utilisateur, utilisez la commande user list. Pour plus d'informations sur les types d'utilisateur sur un HSM, consultez Comprendre les utilisateurs HSM.

<PASSWORD>

Spécifie le mot de passe de l'utilisateur qui se connecte aux HSM.

Obligatoire : oui

<APPROVAL>

Spécifie le chemin d'accès à un fichier de jeton de quorum signé pour approuver l'opération. Obligatoire uniquement si la valeur du quorum du service utilisateur du quorum est supérieure à 1.

Rubriques en relation

- user list
- user delete
- Modifier un mot de passe utilisateur

Supprimer un utilisateur

La user delete commande de la CLI CloudHSM supprime un utilisateur de votre cluster. AWS CloudHSM Seuls les comptes utilisateurs dotés du rôle d'administrateur peuvent exécuter cette commande. Vous ne pouvez pas supprimer un utilisateur actuellement connecté à un HSM.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Administrateur

Prérequis

- Vous ne pouvez pas supprimer les comptes utilisateurs qui possèdent des clés.
- Votre compte utilisateur doit avoir le rôle d'administrateur pour exécuter cette commande.

Syntaxe

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
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
```

Exemple

```
aws-cloudhsm > user delete --username alice --role crypto-user
{
  "error_code": 0,
  "data": {
    "username": "alice",
    "role": "crypto-user"
  }
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<USERNAME>

Spécifie un nom convivial pour l'utilisateur. La longueur maximale est de 31 caractères. Le seul caractère spécial autorisé est un trait de soulignement (_). Le nom d'utilisateur n'est pas sensible à la casse dans cette commande, il est toujours affiché en minuscules.

Obligatoire : oui

<ROLE>

Spécifie le rôle attribué à cet utilisateur. Ce paramètre est obligatoire. Les valeurs valides sont admin, crypto-user.

Pour obtenir le rôle de l'utilisateur, utilisez la commande user list. Pour plus d'informations sur les types d'utilisateur sur un HSM, consultez Comprendre les utilisateurs HSM.

Obligatoire : oui

<APPROVAL>

Spécifie le chemin d'accès à un fichier de jeton de quorum signé pour approuver l'opération. Obligatoire uniquement si la valeur du quorum du service utilisateur du quorum est supérieure à 1.

Obligatoire : oui

Rubriques en relation

- Répertorier les utilisateurs
- Créer un utilisateur
- Modifier un mot de passe utilisateur

Répertorier les utilisateurs

La commande user list de la CLI CloudHSM répertorie les comptes utilisateur présents dans votre cluster CloudHSM. Vous n'avez pas besoin d'être connecté à la CLI CloudHSM pour exécuter cette commande.

> **Note**
>
> Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration utilisés par le AWS CloudHSM client et les outils de ligne de commande. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Tous les utilisateurs. Vous n'avez pas besoin d'être connecté pour exécuter cette commande.

Syntaxe

```
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
```

Exemple

Cette commande répertorie les utilisateurs présents dans votre cluster CloudHSM.

```
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"
      }
    ]
  }
}
```

La sortie inclut les attributs utilisateur suivants :

- Username : affiche le nom convivial défini pour l'utilisateur. Le nom d'utilisateur est toujours affiché en minuscules.
- Role : détermine les opérations que l'utilisateur peut effectuer sur le HSM.
- Locked : indique si ce compte utilisateur a été verrouillé.
- MFA : indique les mécanismes d'authentification multifactorielle pris en charge pour ce compte utilisateur.
- Cluster coverage : indique la disponibilité de ce compte utilisateur à l'échelle du cluster.

Rubriques en relation

- listUsers dans key_mgmt_util
- Créer un utilisateur
- Supprimer un utilisateur

- Modifier un mot de passe utilisateur

quorum

quorum est une catégorie parent pour un groupe de commandes qui, lorsqu'il est combiné avec quorum, crée une commande spécifique à l'authentification par quorum ou aux opérations M sur N. Actuellement, cette catégorie comprend la sous-catégorie token-sign qui comprend ses propres commandes. Cliquez sur le lien ci-dessous pour plus d'informations.

- token-sign

Services d'administrateur : l'authentification par quorum est utilisée pour les services dotés de privilèges d'administrateur tels que la création d'utilisateurs, la suppression d'utilisateurs, la modification des mots de passe des utilisateurs, la définition des valeurs de quorum et la désactivation des fonctionnalités de quorum et de MFA.

Chaque type de service est ensuite décomposé en un nom de service éligible, qui contient un ensemble spécifique d'opérations de service prises en charge par le quorum qui peuvent être effectuées.

Nom du service	Type de service	Opérations de service
utilisateur	Administrateur	• créer un utilisateur • supprimer un utilisateur • modifier un mot de passe utilisateur • modifier la MFA d'un utilisateur
quorum	Administrateur	• signe du jeton quorum set-quorum-value

Rubriques en relation

- Utilisation de l'authentification par quorum pour les administrateurs : première configuration
- Utilisation de la CLI CloudHSM pour gérer l'authentification par quorum (contrôle d'accès M sur N)

quorum token-sign

quorum token-sign est une catégorie pour un groupe de commandes qui, lorsqu'elles sont combinées avec quorum token-sign, créent une commande spécifique à l'authentification par quorum ou aux opérations M sur N.

Actuellement, cette catégorie comprend les commandes suivantes :

- supprimer
- generate
- liste
- list-quorum-values
- list-timeouts
- set-quorum-value
- set-timeout

quorum token-sign delete

Utilisez la commande quorum token-sign delete de la CLI CloudHSM pour supprimer un ou plusieurs jetons pour un service autorisé par quorum.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Administrateur

Syntaxe

```
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')
```

Exemple

L'exemple suivant montre comment la commande quorum token-sign delete de la CLI CloudHSM peut être utilisée pour supprimer un ou plusieurs jetons pour un service autorisé par quorum.

Example : Supprimer un ou plusieurs jetons pour un service autorisé par quorum

```
aws-cloudhsm > quorum token-sign delete --scope all
{
  "error_code": 0,
  "data": "Deletion of quorum token(s) successful"
}
```

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<SCOPE>

Étendue dans laquelle les jetons seront supprimés dans le AWS CloudHSM cluster.

Valeurs valides

- Utilisateur : Utilisé pour supprimer uniquement les jetons appartenant à l'utilisateur connecté.
- Tout : Utilisé pour supprimer tous les jetons du AWS CloudHSM cluster.

Rubriques en relation

- Répertorier les utilisateurs

- Créer un utilisateur
- Supprimer un utilisateur

quorum token-sign generate

Utilisez la commande quorum token-sign generate de la CLI CloudHSM pour générer un jeton pour un service autorisé par quorum.

L'obtention d'un jeton actif par utilisateur et par service sur un cluster HSM est limitée pour l'utilisateur des services et le quorum.

> **Note**
>
> Seuls les administrateurs peuvent générer un jeton de service.

Services d'administrateur : l'authentification par quorum est utilisée pour les services dotés de privilèges d'administrateur tels que la création d'utilisateurs, la suppression d'utilisateurs, la modification des mots de passe des utilisateurs, la définition des valeurs de quorum et la désactivation des fonctionnalités de quorum et de MFA.

Chaque type de service est ensuite décomposé en un nom de service éligible, qui contient un ensemble spécifique d'opérations de service prises en charge par le quorum qui peuvent être effectuées.

Nom du service	Type de service	Opérations de service
utilisateur	Administrateur	• créer un utilisateur • supprimer un utilisateur • modifier un mot de passe utilisateur • modifier la MFA d'un utilisateur
quorum	Administrateur	• signe du jeton quorum set-quorum-value

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Administrateur
- Utilisateur de chiffrement (CU)

Syntaxe

```
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
```

Exemple

Cette commande écrira un jeton non signé par HSM de votre cluster dans le fichier spécifié par `token`.

Example : écrivez un jeton non signé par HSM dans votre cluster

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

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<SERVICE>

Spécifie le service autorisé par le quorum pour lequel générer un jeton. Ce paramètre est obligatoire.

Valeurs valides

- utilisateur : service de gestion des utilisateurs utilisé pour exécuter les opérations de gestion des utilisateurs autorisés par quorum.
- quorum : service de gestion du quorum utilisé pour définir les valeurs de quorum autorisées pour tout service autorisé par quorum.
- enregistrement : génère un jeton non signé à utiliser lors de l'enregistrement d'une clé publique pour l'autorisation du quorum.

Obligatoire : oui

<TOKEN>

Chemin de fichier dans lequel le fichier de jeton non signé sera écrit.

Obligatoire : oui

Rubriques en relation

- Noms et types de services prenant en charge l'authentification par quorum

liste quorum token-sign

Utilisez la quorum token-sign list commande de la CLI CloudHSM pour répertorier tous les jetons de quorum tokensign présents dans votre cluster. AWS CloudHSM

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Administrateur
- Utilisateur de chiffrement (CU)

Syntaxe

```
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
```

Exemple

Cette commande listera tous les jetons de signature présents dans votre AWS CloudHSM cluster.

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"
      }
    ]
  }
}
```

Rubriques en relation

- quorum token-sign generate

signe du jeton quorum list-quorum-values

Utilisez la quorum token-sign list-quorum-values commande de la CLI CloudHSM pour répertorier les valeurs de quorum définies AWS CloudHSM dans votre cluster.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Tous les utilisateurs. Vous n'avez pas besoin d'être connecté pour exécuter cette commande.

Syntaxe

```
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
```

Exemple

Cette commande répertorie les valeurs de quorum définies dans votre AWS CloudHSM cluster pour chaque service.

Example

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

Rubriques en relation

- Noms et types de services prenant en charge l'authentification par quorum

quorum token-sign list-timeouts

Utilisez la commande quorum token-sign list-timeouts de la CLI CloudHSM pour obtenir le délai d'expiration du jeton en secondes pour tous les types de jetons.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Tous les utilisateurs. Vous n'avez pas besoin d'être connecté pour exécuter cette commande.

Syntaxe

```
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
```

Exemple

Example

```
aws-cloudhsm > quorum token-sign list-timeouts
{
  "error_code": 0,
  "data": {
    "generated": 600,
    "approved": 600
  }
}
```

La sortie comprend la ligne suivante :

- generated : délai d'expiration en secondes pour qu'un jeton généré soit approuvé.
- approved : délai d'expiration en secondes pour qu'un jeton approuvé soit utilisé pour exécuter une opération autorisée par quorum.

Rubriques en relation

- quorum token-sign set-timeout

signe du jeton quorum set-quorum-value

Utilisez la commande quorum token-sign set-quorum-value de la CLI CloudHSM pour définir une nouvelle valeur de quorum pour un service autorisé par quorum.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Administrateur

Syntaxe

```
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')
```

Exemple

Example

Dans l'exemple suivant, cette commande écrit un jeton non signé par HSM de votre cluster dans le fichier spécifié par jeton. Lorsque vous y êtes invité, signez les jetons du fichier.

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

Vous pouvez ensuite exécuter la commande list-quorum-values pour confirmer que la valeur du quorum pour le service de gestion du quorum a été définie :

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

Arguments

<CLUSTER_ID>

ID du cluster sur lequel exécuter cette opération.

Obligatoire : si plusieurs clusters ont été configurés.

<APPROVAL>

Le chemin du fichier de jeton signé à approuver sur le HSM.

<SERVICE>

Spécifie le service autorisé par le quorum pour lequel générer un jeton. Ce paramètre est obligatoire. Pour plus d'informations sur les types et les noms de services, consultez la section Noms des services et types prenant en charge l'authentification par quorum.

Valeurs valides

- user : le service de gestion des utilisateurs. Service utilisé pour exécuter les opérations de gestion des utilisateurs autorisés par quorum.
- quorum : le service de gestion du quorum. Service utilisé pour définir les valeurs de quorum autorisées pour tout service autorisé par quorum.
- registration : génère un jeton non signé à utiliser pour enregistrer une clé publique pour l'autorisation du quorum.

Obligatoire : oui

<VALUE>

Spécifie la valeur du quorum à définir. La valeur maximale du quorum est de huit (8).

Obligatoire : oui

Rubriques en relation

- signe du jeton quorum list-quorum-values
- Noms et types de services prenant en charge l'authentification par quorum

quorum token-sign set-timeout

Utilisez la commande quorum token-sign set-timeout de la CLI CloudHSM pour définir le délai d'expiration du jeton en secondes pour chaque type de jeton.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Administrateur

Syntaxe

```
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')
```

Exemple

Les exemples suivants montrent comment utiliser la commande quorum token-sign set-timeout pour définir le délai d'expiration du jeton.

```
aws-cloudhsm > quorum token-sign set-timeout --generated 900
```

```
{
  "error_code": 0,
  "data": "Set token timeout successful"
}
```

Rubriques en relation

- quorum token-sign list-timeouts

Utilitaire de gestion CloudHSM (CMU)

L'outil de ligne de commande cloudhsm_mgmt_util aide les responsables de chiffrement à gérer les utilisateurs dans les HSM. Il inclut des outils permettant de créer, supprimer et répertorier des utilisateurs, et de modifier des mots de passe utilisateur.

KMU et CMU font partie de la suite du SDK client 3.

cloudhsm_mgmt_util inclut également des commandes qui permettent aux utilisateurs de chiffrement (CU) de partager des clés, ainsi que d'obtenir ou de définir des attributs de clé. Ces commandes viennent compléter les commandes de gestion des clés de l'outil de gestion de clé primaire, key_mgmt_util.

Pour une mise en route rapide, consultez Gestion des clusters clonés. Pour plus d'informations sur les commandes cloudhsm_mgmt_util et des exemples d'utilisation des commandes, consultez Référence de la commande cloudhsm_mgmt_util.

Rubriques

- Plateformes prises en charge pour l'utilitaire AWS CloudHSM de gestion
- Mise en route avec l'Utilitaire de gestion CloudHSM (CMU)
- Installation et configuration du AWS CloudHSM client (Linux)
- Installation et configuration du AWS CloudHSM client (Windows)
- Référence de la commande cloudhsm_mgmt_util

Plateformes prises en charge pour l'utilitaire AWS CloudHSM de gestion

Prise en charge de Linux

- 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

Prise en charge de Windows

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2
- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Mise en route avec l'Utilitaire de gestion CloudHSM (CMU)

L'Utilitaire de gestion CloudHSM (CMU) vous permet de gérer les utilisateurs du module de sécurité matérielle (HSM). Utilisez cette rubrique pour démarrer avec les tâches de base de gestion des utilisateurs HSM, telles que la création d'utilisateurs, la création de listes des utilisateurs et la connexion de l'utilitaire CMU au cluster.

1. Pour utiliser l'utilitaire CMU, vous devez d'abord utiliser l'outil de configuration pour mettre à jour la configuration de l'utilitaire CMU avec le paramètre `--cmu` et une adresse IP provenant de l'un des HSM de votre cluster. Procédez ainsi chaque fois que vous utilisez l'utilitaire CMU pour vous assurer que vous gérez les utilisateurs HSM sur chaque HSM du cluster.

Linux

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

Windows

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

2. Entrez la commande suivante pour démarrer la CLI en mode interactif.

   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
   ```

   La sortie doit ressembler à ce qui suit en fonction du nombre de HSM que vous avez.

   ```
   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.
   ```

   L'invite devient `aws-cloudhsm>` lorsque cloudhsm_mgmt_util est en cours d'exécution.

3. Utilisez la commande loginHSM pour vous connecter au cluster. Tout utilisateur de n'importe quel type peut utiliser cette commande pour se connecter au cluster.

La commande de l'exemple suivant se connecte à admin, qui est le responsable de chiffrement (CO) par défaut. Vous définissez le mot de passe de l'utilisateur lors de l'activation du cluster. Vous pouvez utiliser le paramètre `-hpswd` pour masquer votre mot de passe.

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

Le système vous invite à entrer votre mot de passe. Vous entrez le mot de passe, le système le masque et le résultat indique que la commande a réussi et que vous vous êtes connecté à tous les HSM du cluster.

```
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. Utilisez listUsers pour répertorier tous les utilisateurs du cluster.

```
aws-cloudhsm>listUsers
```

L'utilitaire CMU répertorie tous les utilisateurs du cluster.

```


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. Utilisez createUser pour créer un utilisateur CU nommé **example_user** dont le mot de passe est **password1**.

   Vous utilisez des utilisateurs CU dans vos applications pour effectuer des opérations de chiffrement et de gestion des clés. Vous pouvez créer des utilisateurs CU car à l'étape 3, vous vous êtes connecté en tant qu'utilisateur CO. Seuls les utilisateurs CO peuvent effectuer des tâches de gestion des utilisateurs avec l'utilitaire CMU, telles que la création et la suppression d'utilisateurs et la modification des mots de passe des autres utilisateurs.

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

   L'utilitaire CMU vous invite à réaliser l'opération de création d'utilisateurs.

   ```
   
   ************************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. Pour créer l'utilisateur CU **example_user**, saisissez **y**.
7. Utilisez listUsers pour répertorier tous les utilisateurs du cluster.

   ```
   aws-cloudhsm>listUsers
   ```

L'utilitaire CMU répertorie tous les utilisateurs du cluster, y compris le nouvel utilisateur CU que vous venez de créer.

```
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. Utilisez la commande logoutHSM pour vous déconnecter des HSM.

```
aws-cloudhsm>logoutHSM
```

```
logoutHSM success on server 0(10.0.2.9)
logoutHSM success on server 1(10.0.3.11)
logoutHSM success on server 2(10.0.1.12)
```

9. Utilisez la commande quit pour arrêter cloudhsm_mgmt_util.

```
aws-cloudhsm>quit


disconnecting from servers, please wait...
```

Installation et configuration du AWS CloudHSM client (Linux)

Pour interagir avec le HSM de votre AWS CloudHSM cluster, vous avez besoin du logiciel AWS CloudHSM client pour Linux. Nous vous conseillons de l'installer sur l'instance client Linux EC2 créée précédemment. Vous pouvez également installer un client si vous utilisez Windows. Pour plus d'informations, consultez Installation et configuration du AWS CloudHSM client (Windows).

Tâches

- Installation du AWS CloudHSM client et des outils de ligne de commande
- Modification de la configuration du client

Installation du AWS CloudHSM client et des outils de ligne de commande

Connectez-vous à votre instance cliente et exécutez les commandes suivantes pour télécharger et installer le AWS CloudHSM client et les outils de ligne de commande.

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
```

Modification de la configuration du client

Avant de pouvoir utiliser le AWS CloudHSM client pour vous connecter à votre cluster, vous devez modifier la configuration du client.

Pour modifier la configuration du logiciel client

1. Si vous installez le SDK client 3 sur cloudhsm_mgmt_util, procédez comme suit pour vous assurer que tous les nœuds du cluster sont synchronisés.

   a. Exécutez configure -a ***<IP of one of the HSMs>***.

   b. Redémarrez le service client.

   c. Exécutez config -m.

2. Copiez votre certificat d'émission— celui que vous avez utilisé pour signer le certificat du cluster—à l'emplacement suivant sur l'instance client : `/opt/cloudhsm/etc/customerCA.crt`. Vous avez besoin d'autorisations d'utilisateur racine sur l'instance de votre client pour copier votre certificat à cet emplacement.

3. Utilisez la commande de configuration suivante pour mettre à jour les fichiers de configuration du AWS CloudHSM client et des outils de ligne de commande, en spécifiant l'adresse IP du HSM de votre cluster. Pour obtenir l'adresse IP du HSM, consultez votre cluster dans la AWS CloudHSM console ou exécutez la commande describe-clusters CLI. Dans la sortie de la commande, l'adresse IP du HSM est la valeur du champ `EniIp`. Si vous avez plusieurs HSM, choisissez l'adresse IP de l'un des HSM, peu importe lequel.

```
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. Accédez à Activation du cluster.

Installation et configuration du AWS CloudHSM client (Windows)

Pour utiliser un HSM dans votre AWS CloudHSM cluster sous Windows, vous avez besoin du logiciel AWS CloudHSM client pour Windows. Nous vous conseillons de l'installer sur l'instance Windows Server que vous avez créée précédemment.

Pour installer (ou mettre à jour) le client et les outils de ligne de commande Windows les plus récents

1. Connectez-vous à votre instance Windows Server.
2. Téléchargez le programme d'installation AWSCloudHSMClient -latest.msi.
3. Si vous installez le SDK client 3 sur cloudhsm_mgmt_util, procédez comme suit pour vous assurer que tous les nœuds du cluster sont synchronisés.
   a. Exécutez configure -a ***<IP of one of the HSMs>***.
   b. Redémarrez le service client.
   c. Exécutez config -m.
4. Accédez à votre emplacement de téléchargement et exécutez le programme d'installation (AWSCloudHSMClient-latest.msi) avec des privilèges d'administrateur.
5. Suivez les instructions du programme d'installation, puis choisissez Fermer une fois le programme d'installation terminé.
6. Copiez votre certificat d'émission auto-signé, celui que vous avez utilisé pour signer le certificat du cluster, dans le dossier `C:\ProgramData\Amazon\CloudHSM`.

7. Exécutez la commande suivante pour mettre à jour vos fichiers de configuration. Veillez à arrêter et démarrer le client au cours de la reconfiguration si vous le mettez à jour :

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

8. Accédez à Activation du cluster.

Remarques :

- Si vous mettez à jour le client, les fichiers de configuration existants des installations précédentes ne sont pas remplacés.
- Le programme d'installation du AWS CloudHSM client pour Windows enregistre automatiquement l'API de chiffrement : Next Generation (CNG) et le fournisseur de stockage de clés (KSP). Pour désinstaller le client, exécutez à nouveau le programme d'installation et suivez les instructions de désinstallation.
- Vous pouvez également installer le client Linux si vous utilisez Linux. Pour plus d'informations, voir Installation et configuration du AWS CloudHSM client (Linux).

Référence de la commande cloudhsm_mgmt_util

L'outil de ligne de commande cloudhsm_mgmt_util aide les responsables de chiffrement à gérer les utilisateurs dans les HSM. Il inclut également des commandes qui permettent aux utilisateurs de chiffrement (CU) de partager des clés, ainsi que d'obtenir ou de définir des attributs de clé. Ces commandes viennent compléter les commandes de gestion des clés primaires de l'outil de ligne de commande key_mgmt_util.

Pour une mise en route rapide, consultez Gestion des clusters clonés.

Avant d'exécuter une commande cloudhsm_mgmt_util, vous devez démarrer cloudhsm_mgmt_util et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Pour afficher la liste des commandes cloudhsm_mgmt_util, exécutez la commande suivante :

```
aws-cloudhsm> help
```

Pour obtenir la syntaxe d'une commande cloudhsm_mgmt_util, exécutez la commande suivante :

```
aws-cloudhsm> help <command-name>
```

> **Note**
>
> Utilisez la syntaxe conformément à la documentation. Bien que l'aide intégrée au logiciel puisse fournir des options supplémentaires, celles-ci ne doivent pas être considérées comme prises en charge et ne doivent pas être utilisées dans le code de production.

Pour exécuter une commande, saisissez le nom de la commande ou une partie suffisante du nom pour la distinguer des noms des autres commandes cloudhsm_mgmt_util.

Par exemple, pour obtenir la liste des utilisateurs sur les HSM, tapez listUsers ou listU.

```
aws-cloudhsm> listUsers
```

Pour terminer votre session cloudhsm_mgmt_util, exécutez la commande suivante :

```
aws-cloudhsm> quit
```

Pour obtenir de l'aide sur l'interprétation des attributs de clés, consultez le Référence des attributs de clé.

Les rubriques suivantes décrivent les commandes dans cloudhsm_mgmt_util.

> **Note**
>
> Certaines commandes de key_mgmt_util et cloudhsm_mgmt_util portent le même nom. Cependant, les commandes ont généralement une syntaxe différente, une sortie différente et des fonctionnalités légèrement différentes.

Command	Description	Type d'utilisateur
changePswd	Modifie les mots de passe des utilisateurs sur les HSM. Tout utilisateur peut modifier son propre mot de passe. Les	CO

Command	Description	Type d'utilisateur
	responsables de chiffrement peuvent modifier le mot de passe de chacun.	
createUser	Crée des utilisateurs de tous les types sur les HSM.	CO
deleteUser	Supprime des utilisateurs de tous les types sur les HSM.	CO
findAllKeys	Obtient les clés qu'un utilisateur possède ou partage. Obtient également un hachage des données de propriété et de partage des clés pour toutes les clés sur chaque HSM.	CO, AU
getAttribute	Obtient une valeur d'attribut pour une AWS CloudHSM clé et l'écrit dans un fichier ou une sortie standard (sortie standard).	CU
getCert	Permet d'obtenir le certificat d'un HSM et l'enregistre dans un format de certification souhaitée.	Tout.
getHSMInfo	Obtient des informations sur le matériel sur lequel un HSM est en cours d'exécution.	Tout. La connexion n'est pas requise.
getKeyInfo	Obtient les propriétaires, les utilisateurs partagés et le statut d'authentification par quorum d'une clé.	Tout. La connexion n'est pas requise.

Command	Description	Type d'utilisateur
info	Obtient des informations sur un HSM, y compris l'adresse IP, le nom d'hôte, le port et l'utilisateur actuel.	Tout. La connexion n'est pas requise.
listUsers	Obtient les utilisateurs figurant dans chaque HSM, leurs types et leurs ID d'utilisateur, ainsi que d'autres attributs.	Tout. La connexion n'est pas requise.
loginHSM et logoutHSM	Connexion à un HSM et déconnexion d'un HSM.	Tout.
quit	Quitte cloudhsm_mgmt_util.	Tout. La connexion n'est pas requise.
serveur	Entre en mode de serveur sur un HSM et le quitte.	Tout.
registerQuorumPubClé	Associe un utilisateur HSM à une paire de clés RSA-2048 asymétrique.	CO
setAttribute	Modifie les valeurs des attributs d'étiquette, de chiffrement, de déchiffre ment, d'encapsulage et de désencapsulage d'une clé existante.	CU
shareKey	Partage une clé existante avec d'autres utilisateurs.	CU
syncKey	Synchronise une clé entre les clusters clonés. AWS CloudHSM	CU, CO

Command	Description	Type d'utilisateur
syncUser	Synchronise un utilisateur entre les clusters clonés. AWS CloudHSM	CO

changePswd

La commande changePswd dans cloudhsm_mgmt_util modifie le mot de passe d'un utilisateur existant sur les HSM du cluster.

Tout utilisateur peut modifier son propre mot de passe. En outre, les responsables de chiffrement (CO et PCO) peuvent modifier le mot de passe d'un autre CO ou d'un utilisateur de chiffrement (CU). Vous n'avez pas besoin d'entrer le mot de passe actuel pour effectuer la modification.

> **Note**
>
> Vous ne pouvez pas modifier le mot de passe d'un utilisateur actuellement connecté au AWS CloudHSM client ou à key_mgmt_util.

Pour résoudre les problèmes liés à ChangePswd

Avant d'exécuter une commande de CMU, vous devez démarrer l'utilitaire CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Responsables de chiffrement (CO)
- Utilisateurs de chiffrement (CU)

Syntaxe

Entrez les arguments dans l'ordre indiqué dans le diagramme de syntaxe. Utilisez le paramètre `-hpswd` pour masquer votre mot de passe. Pour activer l'authentification à deux facteurs (2FA) pour un utilisateur CO, utilisez le paramètre `-2fa` et incluez un chemin de fichier. Pour plus d'informations, consultez the section called "Arguments".

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

Exemples

Les exemples suivants montrent comment utiliser changePassword pour réinitialiser le mot de passe de l'utilisateur actuel ou de tout autre utilisateur dans vos instances HSM.

Example : Modifier votre mot de passe

Tout utilisateur des HSM peut utiliser changePswd pour modifier son propre mot de passe. Avant de modifier le mot de passe, utilisez info pour obtenir plus d'informations sur chaque HSM dans le cluster, y compris le nom d'utilisateur et le type de l'utilisateur connecté.

La sortie suivante montre que Bob est actuellement connecté en tant qu'utilisateur de chiffrement (CU).

```
        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)'
```

Pour modifier son mot de passe, Bob exécute la commande changePswd suivie du type d'utilisateur, du nom d'utilisateur et d'un nouveau mot de passe.

```
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 : Modifier le mot de passe d'un autre utilisateur

Vous devez être un CO ou un PCO pour modifier le mot de passe d'un autre CO, d'un CU ou sur les HSM. Avant de modifier le mot de passe d'un autre utilisateur, utilisez la commande info pour confirmer que votre type d'utilisateur est soit CO soit PCO.

La sortie suivante confirme qu'Alice, qui est un CO, est actuellement connectée.

```
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 souhaite réinitialiser le mot de passe d'un autre utilisateur, John. Avant de modifier le mot de passe, elle utilise la commande listUsers pour vérifier le type d'utilisateur de John.

La sortie suivante répertorie John en tant qu'utilisateur CO.

```
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
```

Pour modifier le mot de passe, Alice exécute la commande changePswd suivie du type d'utilisateur, du nom d'utilisateur et d'un nouveau mot de passe pour John.

```
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
```

Arguments

Entrez les arguments dans l'ordre indiqué dans le diagramme de syntaxe. Utilisez le paramètre `-hpswd` pour masquer votre mot de passe. Pour activer l'authentification à deux facteurs pour un utilisateur CO, utilisez le paramètre `-2fa` et incluez un chemin de fichier. Pour de plus amples informations sur l'utilisation de l'authentification à deux facteurs, consultez Utilisation de la CMU pour gérer l'authentification à deux facteurs.

```
changePswd <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<type-utilisateur>

Spécifie le type actuel de l'utilisateur dont vous modifiez le mot de passe. Vous ne pouvez pas utiliser changePswd pour modifier le type d'utilisateur.

Les valeurs valides sont `CO`, `CU`, `PCO` et `PRECO`.

Pour obtenir le type d'utilisateur, utilisez listUsers. Pour plus d'informations sur les types d'utilisateur sur un HSM, consultez Comprendre les utilisateurs HSM.

Obligatoire : oui

<nom-utilisateur>

Spécifie le nom convivial de l'utilisateur. Ce paramètre n'est pas sensible à la casse. Vous ne pouvez pas utiliser changePswd pour modifier le nom d'utilisateur.

Obligatoire : oui

<password | -hpswd >

Spécifie un nouveau mot de passe pour l'utilisateur. Entrez une chaîne de 7 à 32 caractères. Cette valeur est sensible à la casse. Le mot de passe s'affiche en texte brut lorsque vous le tapez. Pour masquer votre mot de passe, utilisez le paramètre `-hpswd` à la place du mot de passe et suivez les instructions.

Obligatoire : oui

[-2 fa] </path/to/authdata>

Spécifie l'activation de l'authentification à deux facteurs pour cet utilisateur CO. Pour obtenir les données nécessaires à la configuration de l'authentification à deux facteurs, incluez un chemin vers un emplacement dans le système de fichiers avec un nom de fichier après le paramètre `-2fa`. Pour plus d'informations sur l'utilisation de l'authentification à deux facteurs, consultez Utilisation de la CMU pour gérer l'authentification à deux facteurs.

Obligatoire : non

Rubriques en relation

- info
- listUsers
- createUser
- deleteUser

createUser

La commande createUser de cloudhsm_mgmt_util crée un utilisateur sur les HSM. Seuls les responsables de chiffrement (CO et PRECO) peuvent exécuter cette commande. Lorsque la commande réussit, elle crée l'utilisateur dans tous les HSM du cluster.

Pour résoudre les problèmes liés à createUser

Si votre configuration HSM est inexacte, l'utilisateur peut ne pas être créé sur tous les HSM. Pour ajouter l'utilisateur dans tous les HSM où il est manquant, utilisez les commandes syncUser ou createUser uniquement sur les HSM où cet utilisateur est manquant. Pour éviter les erreurs de configuration, exécutez l'outil configure avec l'option `-m`.

Avant d'exécuter une commande de CMU, vous devez démarrer l'utilitaire CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Responsables de chiffrement (CO, PRECO)

Syntaxe

Entrez les arguments dans l'ordre indiqué dans le diagramme de syntaxe. Utilisez le paramètre `-hpswd` pour masquer votre mot de passe. Pour créer un utilisateur CO avec une authentification à deux facteurs (2FA), utilisez le paramètre `-2fa` et incluez un chemin de fichier. Pour plus d'informations, consultez the section called "Arguments".

```
createUser <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

Exemples

Ces exemples montrent comment utiliser createUser pour créer de nouveaux utilisateurs dans vos HSM.

Example : Créer un responsable de chiffrement

Cet exemple crée un responsable de chiffrement (CO) sur les HSM d'un cluster. La première commande utilise loginHSM pour vous connecter au HSM en tant que responsable de chiffrement.

```
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)
```

La deuxième commande utilise la commande createUser pour créer `alice`, une nouvelle responsable de chiffrement dans le HSM.

Le message d'avertissement explique que la commande crée des utilisateurs sur tous les HSM du cluster. Toutefois, si la commande échoue sur des HSM, l'utilisateur n'existera pas sur ces HSM. Pour continuer, tapez `y`.

La sortie indique que le nouvel utilisateur a été créé sur les trois HSM du cluster.

```
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
```

Une fois la commande terminée, `alice` a les mêmes autorisations sur le HSM que l'utilisateur responsable de chiffrement `admin`, y compris celle de modifier le mot de passe de n'importe quel utilisateur sur les HSM.

La dernière commande utilise la commande listUsers pour vérifier qu'`alice` existe sur les trois HSM du cluster. La sortie indique également que `alice` se voit affecter l'ID d'utilisateur `3`.. Vous utilisez l'ID utilisateur pour vous identifier `alice` dans d'autres commandes, telles que 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 : Créer un utilisateur de chiffrement

Cet exemple crée un utilisateur de chiffrement (CU), bob, sur le HSM. Les utilisateurs de chiffrement peuvent créer et gérer des clés, mais ils ne peuvent pas gérer les utilisateurs.

Une fois que vous tapez y pour répondre au message d'avertissement, la sortie indique que bob a été créé sur les trois HSM du cluster. Le nouvel utilisateur de chiffrement peut se connecter au HSM pour créer et gérer des clés.

La commande a utilisé la valeur de mot de passe `defaultPassword`. Ultérieurement, bob ou un responsable de chiffrement quelconque peut utiliser la commande changePswd pour modifier son mot de passe.

```
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
```

Arguments

Entrez les arguments dans l'ordre indiqué dans le diagramme de syntaxe. Utilisez le paramètre `-hpswd` pour masquer votre mot de passe. Pour créer un utilisateur CO avec 2FA activé, utilisez le paramètre `-2fa` et incluez un chemin de fichier. Pour plus d'informations sur l'authentification à deux facteurs, consultez Utilisation de la CMU pour gérer l'authentification à deux facteurs.

```
createUser <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<type-utilisateur>

Spécifie le type d'utilisateur. Ce paramètre est obligatoire.

Pour plus d'informations sur les types d'utilisateur sur un HSM, consultez Comprendre les utilisateurs HSM.

Valeurs valides :

- CO : les responsables de chiffrement peuvent gérer les utilisateurs, mais ne peuvent pas gérer les clés.
- CU : les utilisateurs de chiffrement peuvent créer et gérer des clés, ainsi qu'utiliser ces clés dans des opérations de chiffrement.

L'utilisateur PRECO est converti en CO lorsque vous attribuez un mot de passe lors de l'activation des HSM.

Obligatoire : oui

<nom-utilisateur>

Spécifie un nom convivial pour l'utilisateur. La longueur maximale est de 31 caractères. Le seul caractère spécial autorisé est un trait de soulignement (_).

Vous ne pouvez pas modifier le nom d'un utilisateur après l'avoir créé. Dans les commandes cloudhsm_mgmt_util, le type d'utilisateur et le mot de passe sont sensibles à la casse, mais le nom d'utilisateur ne l'est pas.

Obligatoire : oui

<password | -hpswd >

Spécifie un mot de passe pour l'utilisateur. Entrez une chaîne de 7 à 32 caractères. Cette valeur est sensible à la casse. Le mot de passe s'affiche en texte brut lorsque vous le tapez. Pour

masquer votre mot de passe, utilisez le paramètre `-hpswd` à la place du mot de passe et suivez les instructions.

Pour changer un mot de passe d'utilisateur, utilisez changePswd. Tout utilisateur HSM peut modifier son propre mot de passe, mais les utilisateurs CO peuvent changer le mot de passe de n'importe quel utilisateur (de tout type) sur les HSM.

Obligatoire : oui

[-2fa </path/to/authdata>]

Spécifie la création d'un utilisateur CO avec l'authentification à deux facteurs activée. Pour obtenir les données nécessaires à la configuration de l'authentification 2FA, incluez un chemin vers un emplacement dans le système de fichiers avec un nom de fichier après le paramètre `-2fa`. Pour plus d'informations sur la configuration et l'utilisation de l'authentification 2FA, consultez Utilisation de la CMU pour gérer l'authentification à deux facteurs.

Obligatoire : non

Rubriques en relation

- listUsers
- deleteUser
- syncUser
- changePswd

deleteUser

La commande deleteUser contenue dans cloudhsm-mgmt_util supprime un utilisateur des modules de sécurité matériels (HSM). Seuls les responsables de chiffrement (CO) peuvent exécuter cette commande. Vous ne pouvez pas supprimer un utilisateur actuellement connecté à un HSM. Pour plus d'informations sur la suppression d'utilisateurs, consultez Comment supprimer des utilisateurs HSM.

> **Tip**
>
> Vous ne pouvez pas supprimer les utilisateurs de chiffrement (CU) qui possèdent des clés.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- CO

Syntaxe

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
deleteUser <user-type> <user-name>
```

Exemple

Cet exemple supprime un responsable de chiffrement (CO) des HSM d'un cluster. La première commande utilise listUsers pour établir la liste de tous les utilisateurs des HSM.

La sortie indique que l'utilisateur `3`, `alice`est un responsable de chiffrement sur les HSM.

```
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
```

La deuxième commande utilise la commande deleteUser pour supprimer `alice` des HSM.

La sortie indique que la commande a abouti sur les trois HSM du cluster.

```
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)
```

La dernière commande utilise la commande listUsers pour vérifier que `alice` a été supprimée des trois HSM du cluster.

```
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
```

Arguments

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
deleteUser <user-type> <user-name>
```

<type-utilisateur>

Spécifie le type d'utilisateur. Ce paramètre est obligatoire.

> **Tip**
>
> Vous ne pouvez pas supprimer les utilisateurs de chiffrement (CU) qui possèdent des clés.

Les valeurs valides sont CO, CU.

Pour obtenir le type d'utilisateur, utilisez listUsers. Pour plus d'informations sur les types d'utilisateur sur un HSM, consultez Comprendre les utilisateurs HSM.

Obligatoire : oui

<nom-utilisateur>

Spécifie un nom convivial pour l'utilisateur. La longueur maximale est de 31 caractères. Le seul caractère spécial autorisé est un trait de soulignement (_).

Vous ne pouvez pas modifier le nom d'un utilisateur après l'avoir créé. Dans les commandes cloudhsm_mgmt_util, le type d'utilisateur et le mot de passe sont sensibles à la casse, mais le nom d'utilisateur ne l'est pas.

Obligatoire : oui

Rubriques en relation

- listUsers
- createUser
- syncUser
- changePswd

findAllKeys

La commande de l'outil findAllKeys dans cloudhsm_mgmt_util permet d'obtenir les clés qui appartiennent ou sont partagées par un utilisateur de chiffrement (CU) spécifié. Elle renvoie également un hachage des données utilisateur sur chaque HSM. Vous pouvez utiliser le hachage pour déterminer d'un coup d'œil si les utilisateurs, la propriété de clé et les données de partage de clé sont identiques sur tous les HSM du cluster. Dans la sortie, les clés détenues par l'utilisateur sont annotées par `(o)` et les clés partagées sont annotées par `(s)`.

findAllKeys renvoie les clés publiques uniquement lorsque l'utilisateur de chiffrement spécifié est propriétaire de la clé, même si tous les utilisateurs de chiffrement du HSM peuvent utiliser n'importe quelle clé publique. Ce comportement est différent de celui de findKey dans key_mgmt_util, qui renvoie les clés publiques pour tous les utilisateurs CU.

Seuls les responsables de chiffrement (CO et PCO) et les utilisateurs de l'application (AU) peuvent exécuter cette commande. Les utilisateurs de chiffrement (CU) peuvent exécuter les commandes suivantes :

- listUsers pour rechercher tous les utilisateurs
- findKey dans key_mgmt_util pour trouver les clés qu'ils peuvent utiliser
- getKeyInfodans key_mgmt_util pour trouver le propriétaire et les utilisateurs partagés d'une clé particulière qu'ils possèdent ou partagent

Avant d'exécuter une commande de CMU, vous devez démarrer l'utilitaire CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Responsables de chiffrement (CO, PCO)
- Utilisateurs de l'appliance (AU)

Syntaxe

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

Exemples

Ces exemples illustrent comment utiliser `findAllKeys` pour rechercher toutes les clés d'un utilisateur et obtenir un hachage des informations utilisateur de la clé sur chaque HSM.

Example : Rechercher les clés pour un utilisateur de chiffrement

Cet exemple utilise findAllKeys pour rechercher les clés dans les HSM qui appartiennent et sont partagées par l'utilisateur 4. La commande utilise la valeur `0` pour le deuxième argument afin de supprimer la valeur de hachage. Le nom de fichier optionnel n'étant pas spécifié, la commande écrit sur stdout (sortie standard).

La sortie montre que l'utilisateur 4 peut utiliser 6 clés : 8, 9, 17, 262162, 19 et 31. La sortie utilise un `(s)` pour indiquer les clés qui sont explicitement partagées par l'utilisateur. Les clés que l'utilisateur possède sont indiquées par un `(o)` et comprennent des clés symétriques et privées que l'utilisateur ne partage pas, et des clés publiques qui sont disponibles pour tous les utilisateurs du chiffrement.

```
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 : Vérifier que les données utilisateur sont synchronisées

Cet exemple utilise findAllKeys pour vérifier que tous les HSM du cluster contiennent les mêmes utilisateurs, propriétés de clé et valeurs de partage de clé. Pour ce faire, elle obtient un hachage des données utilisateur de clé sur chaque HSM et compare les valeurs de hachage.

Pour obtenir le hachage de clé, la commande utilise la valeur 1 dans le deuxième argument. Le nom de fichier optionnel n'étant pas spécifié, la commande écrit le hachage de clé dans stdout.

L'exemple spécifie l'utilisateur 6, mais la valeur de hachage sera identique pour n'importe quel utilisateur possédant ou partageant une ou plusieurs clés sur les HSM. Si l'utilisateur spécifié ne possède ou ne partage aucune clé, comme c'est le cas d'un responsable de chiffrement, la commande ne renvoie pas de valeur de hachage.

La sortie indique que le hachage de clé est identique pour tous les HSM du cluster. Si l'un des HSM a des utilisateurs, des propriétaires de clé ou des utilisateurs partagés différents, les valeurs de hachage de clé ne seront pas égales.

```
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)
```

Cette commande montre que la valeur de hachage représente les données utilisateur pour tous les clés sur le HSM. Elle utilise findAllKeys pour l'utilisateur 3. Contrairement à l'utilisateur 6, qui possède ou partage seulement 3 clés, l'utilisateur 3 possède ou partages 17 clés, mais la valeur de hachage de clé est la même.

```
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)
```

Arguments

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
findAllKeys <user id> <key hash (0/1)> [<output file>]
```

<ID utilisateur>

Permet d'obtenir toutes les clés que l'utilisateur spécifié possède ou partage. Saisissez l'ID d'un utilisateur sur les HSM. Pour rechercher l'ID de tous les utilisateurs, utilisez listUsers.

Tous les ID utilisateur sont valides, mais `findAllKeys` retourne uniquement les clés des utilisateurs de chiffrement (CU).

Obligatoire : oui

<hachage de clé>

Inclut (`1`) ou exclut (`0`) un hachage de la propriété utilisateur et des données de partage pour toutes les clés de chaque HSM.

Lorsque l'argument `user id` représente un utilisateur qui possède ou partage des clés, le hachage de clé est renseigné. La valeur de hachage de clé est identique pour tous les utilisateurs qui possèdent ou partagent des clés sur le HSM, même s'ils possèdent et partagent des clés différentes. Toutefois, lorsque l'argument `user id` représente un utilisateur qui ne possède ou ne partage pas de clé, comme un responsable de chiffrement, la valeur de hachage n'est pas renseignée.

Obligatoire : oui

<fichier de sortie>

Écrit la sortie sur le fichier spécifié.

Obligatoire : non

Par défaut : Stdout

Rubriques en relation

- changePswd
- deleteUser
- listUsers
- syncUser
- findKey dans key_mgmt_util

- getKeyInfodans key_mgmt_util

getAttribute

La commande getAttribute dans cloudhsm_mgmt_util permet d'obtenir une valeur d'attribut pour une clé à partir de tous les HSM du cluster et les écrit dans stdout (sortie standard) ou dans un fichier. Seuls les utilisateurs de chiffrement (CU) peuvent exécuter cette commande.

Les attributs de clé sont les propriétés d'une clé. Ils incluent des caractéristiques telles que le type de clé, la classe, l'étiquette et l'ID, ainsi que les valeurs qui représentent les actions que vous pouvez exécuter sur la clé, comme le chiffrement, le déchiffrement, l'encapsulation, la signature et la vérification.

Vous ne pouvez utiliser getAttribute que sur les clés que vous possédez et les clés que vous partagez. Vous pouvez exécuter cette commande ou la commande getAttribute dans key_mgmt_util, qui écrit une ou toutes les valeurs d'attribut d'une clé dans un fichier.

Pour obtenir une liste d'attributs et les constantes qui les représentent, utilisez la commande listAttributes. Pour modifier les valeurs d'attribut des clés existantes, utilisez setAttribute dans key_mgmt_util et setAttribute dans cloudhsm_mgmt_util. Pour obtenir de l'aide sur l'interprétation des attributs de clés, consultez le Référence des attributs de clé.

Avant d'exécuter une commande de CMU, vous devez démarrer l'utilitaire CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Syntaxe

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
getAttribute <key handle> <attribute id> [<filename>]
```

Exemple

Cet exemple obtient la valeur de l'attribut extractible d'une clé des modules HSM. Vous pouvez utiliser une commande comme celle-ci pour déterminer si vous pouvez exporter une clé à partir des modules HSM.

La première commande utilise listAttributes pour rechercher la constante qui représente l'attribut extractible. La sortie indique que la constante de `OBJ_ATTR_EXTRACTABLE` est `354`. Vous pouvez également trouver ces informations avec des descriptions des attributs et leurs valeurs dans le Référence des attributs de clé.

```
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
```

La deuxième commande utilise getAttribute pour obtenir la valeur de l'attribut extractible de la clé avec le handle de clé 262170 dans les modules HSM. Pour spécifier l'attribut extractible, la commande utilise 354, soit la constante qui représente l'attribut extractible. Comme la commande ne permet pas de spécifier un nom de fichier, getAttribute écrit la sortie dans stdout.

La sortie indique que la valeur de l'attribut extractible est « 1 » sur la totalité des HSM. Cette valeur indique que le propriétaire de la clé peut l'exporter. Lorsque la valeur est 0 (0x0), elle ne peut pas être exportée à partir des modules HSM. Vous définissez la valeur de l'attribut extractible lorsque vous créez une clé, mais vous ne pouvez pas la modifier.

```
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
```

Arguments

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
getAttribute <key handle> <attribute id> [<filename>]
```

<handle-clé>

Spécifie le handle de clé de la clé cible. Vous pouvez spécifier une seule clé dans chaque commande. Pour obtenir le handle d'une clé, utilisez findKey dans key_mgmt_util.

Vous devez posséder la clé spécifiée ou elle doit être partagé avec vous. Pour trouver les utilisateurs d'une clé, utilisez getKeyInfokey_mgmt_util.

Obligatoire : oui

<ID attribut>

Identifie l'attribut. Entrez une constante qui représente un attribut, ou `512`, qui représente tous les attributs. Par exemple, pour obtenir le type de clé, tapez `256`, qui correspond à la constante de l'attribut `OBJ_ATTR_KEY_TYPE`.

Pour obtenir la liste des attributs et de leurs constantes, utilisez listAttributes. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

Obligatoire : oui

<nom de fichier>

Écrit la sortie sur le fichier spécifié. Saisissez un chemin de fichier.

Si le fichier spécifié existe, getAttribute remplace le fichier sans avertissement.

Obligatoire : non

Par défaut : Stdout

Rubriques en relation

- getAttribute dans key_mgmt_util
- listAttributes
- setAttribute dans cloudhsm_mgmt_util
- setAttribute dans key_mgmt_util
- Référence des attributs de clé

getCert

Avec la commande getCert de cloudhsm_mgmt_util, vous pouvez récupérer les certificats d'un HSM particulier dans un cluster. Lorsque vous exécutez la commande, vous désignez le type de certificat à extraire. Pour ce faire, vous devez utiliser l'un des nombres entiers correspondants décrits dans la section Arguments ci-dessous. Pour plus d'informations sur le rôle de chacun de ces certificats, consultez Vérification de l'identité du HSM.

Avant d'exécuter une commande de CMU, vous devez démarrer l'utilitaire CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Tous les utilisateurs.

Prérequis

Avant de commencer, vous devez saisir le mode serveur sur le HSM cible. Pour plus d'informations, consultez server.

Syntaxe

Pour utiliser la commande getCert une fois en mode serveur :

```
server> getCert <file-name> <certificate-type>
```

Exemple

D'abord, entrez le mode serveur. Cette commande entre en mode serveur sur un HSM avec le numéro de serveur `0`.

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

Ensuite, utilisez la commande getCert. Dans cet exemple, nous utilisons `/tmp/P0.crt` comme nom de fichier sous lequel le certificat sera enregistré et `4` (Customer Root Certificate) en tant que type de certification souhaité :

```
server0> getCert /tmp/P0.crt 4
```

```
getCert Success
```

Arguments

```
getCert <file-name> <certificate-type>
```

<nom-fichier>

Spécifie le nom du fichier sous lequel le certificat est enregistré.

Obligatoire : oui

<type-certificat>

Nombre entier qui spécifie le type de certificat à extraire. Les nombres entiers et leurs types de certificats correspondants sont les suivants :

- 1 – Certificat racine du fabricant
- 2 – Certificat du fabricant du matériel
- 4 – Certificat racine du client
- 8 – Certificat du cluster (signé par le certificat racine du client)
- 16 – Certificat du cluster (chaîné au certificat racine du fabricant)

Obligatoire : oui

Rubriques en relation

- serveur

getHSMInfo

La commande getHSMInfo dans cloudhsm_mgmt_util gets permet d'obtenir des informations sur le matériel sur lequel chaque HSM s'exécute, y compris le modèle, le numéro de série, l'état FIPS, la mémoire, la température et les numéros de version du matériel et du microprogramme. Ces informations incluent également l'ID de serveur que cloudhsm_mgmt_util utilise pour faire référence au HSM.

Avant d'exécuter une commande de l'utilitaire CMU, vous devez démarrer l'utilitaire CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Tous les utilisateurs. Vous n'avez pas besoin d'être connecté pour exécuter cette commande.

Syntaxe

Cette commande n'a pas de paramètres.

```
getHSMInfo
```

Exemple

Cet exemple utilise getHSMInfo pour obtenir des informations sur les HSM figurant dans le cluster.

```
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

...
```

Rubriques en relation

- info

getKeyInfo

La commande getKeyInfo de key_mgmt_util renvoie les ID utilisateur HSM des utilisateurs qui peuvent utiliser la clé, y compris le propriétaire et les utilisateurs de chiffrement (CU) avec lesquelles la clé est partagée. Lorsque l'authentification par quorum est activée sur une clé, getKeyInfo renvoie également le nombre d'utilisateurs qui doivent approuver les opérations de chiffrement qui utilisent la clé. Vous ne pouvez exécuter getKeyInfo que sur les clés que vous possédez et sur les clés que vous partagez.

Lorsque vous exécutez getKeyInfo sur les clés publiques, getKeyInfo renvoie uniquement le propriétaire de la clé, même si tous les utilisateurs du HSM peuvent utiliser la clé publique. Pour rechercher les ID d'utilisateur HSM de tous les utilisateurs de vos HSM, utilisez listUsers. Pour trouver les clés pour un utilisateur particulier, utilisez findKey -u dans key_mgmt_util. Les responsables de la cryptographie peuvent l'utiliser findAllKeysdans cloudhsm-mgmt_util.

Vous possédez les clés que vous créez. Vous pouvez partager une clé avec d'autres utilisateurs lorsque vous la créez. Ensuite, pour partager ou annuler le partage d'une clé existante, utilisez shareKey dans cloudhsm_mgmt_util.

Avant d'exécuter une commande de CMU, vous devez démarrer l'utilitaire CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Syntaxe

```
getKeyInfo -k <key-handle> [<output file>]
```

Exemples

Ces exemples montrent comment utiliser getKeyInfo pour obtenir des informations sur les utilisateurs d'une clé.

Example : Obtention des utilisateurs pour une clé asymétrique

Cette commande obtient les utilisateurs qui peuvent utiliser la clé AES (asymétrique) avec le handle de clé 262162. La sortie indique que l'utilisateur 3 possède la clé et la partage avec les utilisateurs 4 et 6.

Seuls les utilisateurs 3, 4 et 6 peuvent exécuter getKeyInfo sur la clé 262162.

```
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 : Obtention des utilisateurs pour une paire de clés symétrique

Ces commandes utilisent getKeyInfo pour obtenir les utilisateurs qui peuvent utiliser les clés d'une paire de clés ECC (symétrique). La clé publique dispose du handle de clé 262179. La clé privée dispose du handle de clé 262177.

Lorsque vous exécutez getKeyInfo sur la clé privée (262177), la commande renvoie le propriétaire de la clé (3) et les utilisateurs de chiffrement (CU) 4 avec lesquels la clé est partagée.

```
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
```

Lorsque vous exécutez getKeyInfo sur la clé publique (262179), la commande renvoie uniquement le propriétaire de la clé, l'utilisateur 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
```

Pour confirmer que l'utilisateur 4 peut utiliser la clé publique (et toutes les clés publiques sur le HSM), utilisez le paramètre `-u` de findKey dans key_mgmt_util.

La sortie indique que l'utilisateur 4 peut utiliser la clé publique (262179) et la clé privée (262177) dans la paire de clés. L'utilisateur 4 peut également utiliser toutes les autres clés publiques et clés privées qu'il a créées ou qu'il a partagées.

```
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 : Obtention de la valeur d'authentification par quorum (m_value) pour une clé

Cet exemple montre comment obtenir la valeur `m_value` pour une clé. La valeur m_value correspond au nombre d'utilisateurs du quorum qui doivent approuver toutes les opérations de chiffrement qui utilisent la clé et toutes les opérations de partage et d'annulation de partage.

Lorsque l'authentification par quorum est activée sur une clé, un quorum d'utilisateurs doit approuver toutes les opérations de chiffrement qui utilisent la clé. Pour activer l'authentification par quorum et définir la taille de quorum, utilisez le paramètre `-m_value` lorsque vous créez la clé.

Cette commande permet genSymKeyde créer une clé AES 256 bits partagée avec l'utilisateur 4. Elle utilise le paramètre `m_value` pour activer l'authentification par quorum et définir la taille du

quorum sur deux utilisateurs. Le nombre d'utilisateurs doit être suffisamment grand pour fournir les approbations requises.

La sortie montre que la commande a créé la clé 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
```

Cette commande utilise getKeyInfo dans cloudhsm_mgmt_util pour obtenir des informations sur les utilisateurs de la clé 10. La sortie indique que la clé est détenue par l'utilisateur 3 et partagée avec l'utilisateur 4. Elle indique également qu'un quorum de deux utilisateurs doit approuver chaque opération de chiffrement qui utilise la clé.

```
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
```

Arguments

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
getKeyInfo -k <key-handle> <output file>
```

<handle-clé>

Spécifie le handle d'une clé du module HSM. Entrez le handle de clé d'une clé qui vous appartient ou que vous partagez. Ce paramètre est obligatoire.

Obligatoire : oui

<fichier de sortie>

Écrit la sortie sur le fichier spécifié, à la place de stdout. Si le fichier existe, la commande le remplace sans avertissement.

Obligatoire : non

Par défaut : stdout

Rubriques en relation

- getKeyInfodans key_mgmt_util
- findKey dans key_mgmt_util
- findAllKeysdans cloudhs_mgmt_util
- listUsers
- shareKey

info

La commande info dans cloudhsm_mgmt_util permet d'obtenir des informations sur chaque HSM du cluster, y compris le nom d'hôte, le port, l'adresse IP et les nom et type de l'utilisateur qui est connecté à cloudhsm_mgmt_util dans le HSM.

Avant d'exécuter une commande de CMU, vous devez démarrer l'utilitaire CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Tous les utilisateurs. Vous n'avez pas besoin d'être connecté pour exécuter cette commande.

Syntaxe

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
info server <server ID>
```

Exemple

Cet exemple utilise info pour obtenir des informations sur un HSM figurant dans le cluster. La commande utilise 0 pour faire référence au premier HSM dans le cluster. La sortie indique l'adresse IP, le port, le type et le nom de l'utilisateur actuel.

```
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)'
```

Arguments

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
info server <server ID>
```

<ID serveur>

Spécifie l'ID de serveur du HSM. Les HSM se voient attribuer des nombres ordinaux qui représentent l'ordre dans lequel ils sont ajoutés dans le cluster, en commençant par 0. Pour trouver l'ID de serveur d'un HSM, utilisez getHSMInfo.

Obligatoire : oui

Rubriques en relation

- getHSMInfo
- loginHSM et logoutHSM

listAttributes

La listAttributes commande contenue dans cloudhs_mgmt_util répertorie les attributs d'une AWS CloudHSM clé et les constantes qui les représentent. Vous utilisez ces constantes pour identifier les attributs dans les commandes getAttribute et setAttribute.

Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Tous les utilisateurs. Vous n'avez pas besoin d'être connecté pour exécuter cette commande.

Syntaxe

```
listAttributes [-h]
```

Exemple

Cette commande répertorie les attributs de clé que vous pouvez obtenir et modifier dans key_mgmt_util et les constantes qui les représentent. Pour obtenir de l'aide sur l'interprétation des attributs de clés, consultez le Référence des attributs de clé. Pour représenter tous les attributs, utilisez 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
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

Rubriques en relation

- getAttribute
- setAttribute
- Référence des attributs de clé

listUsers

La commande listUsers dans cloudhsm_mgmt_util permet d'obtenir les utilisateurs de chaque module HSM, ainsi que leur type d'utilisateur et d'autres attributs. Tous les types d'utilisateur peuvent exécuter cette commande. Vous n'avez même pas besoin d'être connecté à cloudhsm_mgmt_util pour exécuter cette commande.

Avant d'exécuter une commande de l'utilitaire CMU, vous devez démarrer l'utilitaire CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Tous les utilisateurs. Vous n'avez pas besoin d'être connecté pour exécuter cette commande.

Syntaxe

Cette commande n'a pas de paramètres.

```
listUsers
```

Exemple

Cette commande répertorie les utilisateurs de chaque module HSM du cluster et affiche leurs attributs. Vous pouvez utiliser l'attribut `User ID` pour identifier les utilisateurs dans d'autres commandes, telles que deleteUser, changePswd et findAllKeys.

```
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
```

La sortie inclut les attributs utilisateur suivants :

- User ID : identifie l'utilisateur dans les commandes key_mgmt_util et cloudhsm_mgmt_util.
- User type : détermine les opérations que l'utilisateur peut effectuer sur le HSM.
- User Name : affiche le nom convivial défini pour l'utilisateur.
- MofnPubKey: indique si l'utilisateur a enregistré une paire de clés pour signer les jetons d'authentification du quorum.
- LoginFailureCnt: indique le nombre de fois où l'utilisateur s'est connecté sans succès.

- 2FA : indique que l'utilisateur a activé l'authentification multi-facteurs.

Rubriques en relation

- listUsers dans key_mgmt_util
- createUser
- deleteUser
- changePswd

loginHSM et logoutHSM

Vous pouvez utiliser les commandes loginHSM et logoutHSM dans cloudhsm_mgmt_util pour vous connecter et vous déconnecter de chaque HSM dans un cluster. Tous les utilisateurs de tous les types peuvent utiliser ces commandes.

> **Note**
>
> Si vous dépassez cinq tentatives de connexion incorrectes, votre compte est verrouillé. Pour déverrouiller le compte, un responsable de chiffrement (CO) doit réinitialiser votre mot de passe à l'aide de la commande changePswd dans cloudhsm_mgmt_util.

Pour résoudre les problèmes liés à loginHSM et logoutHSM

Avant d'exécuter ces commandes cloudhsm_mgmt_util, vous devez démarrer cloudhsm_mgmt_util.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration utilisés par le AWS CloudHSM client et les outils de ligne de commande. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Si vous avez plusieurs HSM dans votre cluster, vous pouvez autoriser des tentatives de connexion incorrectes supplémentaires avant que votre compte soit verrouillé. Cela est dû au fait que le client CloudHSM équilibre la charge sur plusieurs HSM. Par conséquent, la tentative de connexion peut ne pas commencer sur le même HSM à chaque fois. Si vous testez cette fonctionnalité, nous vous recommandons de le faire sur un cluster avec un seul HSM actif.

Si vous avez créé votre cluster avant février 2018, votre compte est verrouillé après 20 tentatives de connexion incorrectes.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Responsable du pré-chiffrement (PRECO)
- Responsable de chiffrement (CO)
- Utilisateur de chiffrement (CU)

Syntaxe

Entrez les arguments dans l'ordre indiqué dans le diagramme de syntaxe. Utilisez le paramètre `-hpswd` pour masquer votre mot de passe. Pour vous connecter avec l'authentification à deux facteurs (2FA), utilisez le paramètre `-2fa` et incluez un chemin de fichier. Pour plus d'informations, consultez the section called "Arguments".

```
loginHSM <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

```
logoutHSM
```

Exemples

Ces exemples montrent comment utiliser loginHSM et logoutHSM pour vous connecter et vous déconnecter de tous les HSM dans un cluster.

Example : Se connecter aux HSM dans un cluster

Cette commande se connecte à tous les HSM dans un cluster avec les informations d'identification d'un utilisateur CO nommé `admin` et le mot de passe `co12345`. Le résultat montre que la commande a été exécutée et que l'utilisateur s'est connecté aux HSM (qui, dans ce cas, sont `server 0` et `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 : Se connecter avec un mot de passe masqué

Cette commande est identique à l'exemple ci-dessus, sauf que cette fois, vous spécifiez que le système doit masquer le mot de passe.

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

Le système vous invite à saisir votre mot de passe. Vous entrez le mot de passe, le système le masque et le résultat indique que la commande a réussi et que vous vous êtes connecté aux HSM.

```
Enter password:

loginHSM success on server 0(10.0.2.9)
loginHSM success on server 1(10.0.3.11)

aws-cloudhsm>
```

Example : Se déconnecter d'un HSM

Cette commande se déconnecte des HSM auxquels vous êtes connecté (qui, dans ce cas, sont `server 0` et `server 1`). Le résultat montre que la commande a été exécutée et que l'utilisateur s'est déconnecté des HSM.

```
aws-cloudhsm>logoutHSM

logoutHSM success on server 0(10.0.2.9)
logoutHSM success on server 1(10.0.3.11)
```

Arguments

Entrez les arguments dans l'ordre indiqué dans le diagramme de syntaxe. Utilisez le paramètre `-hpswd` pour masquer votre mot de passe. Pour vous connecter avec l'authentification à deux facteurs (2FA), utilisez le paramètre `-2fa` et incluez un chemin de fichier. Pour de plus amples informations sur l'utilisation de l'authentification à deux facteurs, consultez Utilisation de la CMU pour gérer l'authentification à deux facteurs.

```
loginHSM <user-type> <user-name> <password |-hpswd> [-2fa </path/to/authdata>]
```

<type d'utilisateur>

Spécifie le type d'utilisateur qui se connecte aux HSM. Pour plus d'informations, consultez Type d'utilisateur ci-dessus.

Obligatoire : oui

<nom d'utilisateur>

Spécifie le nom d'utilisateur de l'utilisateur qui se connecte aux HSM.

Obligatoire : oui

<password | -hpswd >

Spécifie le mot de passe de l'utilisateur qui se connecte aux HSM. Pour masquer votre mot de passe, utilisez le paramètre `-hpswd` à la place du mot de passe et suivez les instructions qui s'affichent.

Obligatoire : oui

[-2fa] </path/to/authdata>

Spécifie que le système doit utiliser un deuxième facteur pour authentifier cet utilisateur CO compatible avec 2FA. Pour obtenir les données nécessaires à la connexion avec 2FA, incluez un chemin vers un emplacement dans le système de fichiers avec un nom de fichier après le paramètre `-2fa`. Pour de plus amples informations sur l'utilisation de l'authentification à deux facteurs, consultez Utilisation de la CMU pour gérer l'authentification à deux facteurs.

Obligatoire : non

Rubriques en relation

- Se lancer avec cloudhsm_mgmt_util
- Activation du cluster

registerQuorumPubClé

La commande registerQuorumPubKey dans cloudhsm_mgmt_util associe les utilisateurs de module de sécurité matérielle (HSM) à des paires de clés RSA-2048 asymétriques. Une fois que vous avez associé des utilisateurs HSM à des clés, ces utilisateurs peuvent utiliser la clé privée pour approuver les demandes de quorum et le cluster peut utiliser la clé publique enregistrée pour vérifier que la signature provient de l'utilisateur. Pour plus d'informations sur l'authentification par quorum, consultez Gestion de l'authentification par quorum (contrôle d'accès M sur N).

> **Tip**
>
> Dans la AWS CloudHSM documentation, l'authentification par quorum est parfois appelée M of N (MoFN), ce qui signifie un minimum de M approbateurs sur un nombre total de N approbateurs.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Responsables de chiffrement (CO)

Syntaxe

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-token> <public-key>
```

Exemples

Cet exemple montre comment utiliser registerQuorumPubKey pour enregistrer les responsables de chiffrement (CO) en tant qu'approbateurs sur les demandes d'authentification par quorum. Pour exécuter cette commande, vous devez disposer d'une paire de clés RSA-2048 asymétrique, d'un jeton signé et d'un jeton non signé. Pour plus d'informations sur les autres conditions requises, consultez the section called "Arguments".

Example : Enregistrez un utilisateur HSM pour l'authentification par quorum

Cet exemple enregistre un CO nommé `quorum_officer` en tant qu'approbateur pour l'authentification par quorum.

```
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)
```

La dernière commande utilise la commande listUsers pour vérifier que `quorum_officer` est enregistré en tant qu'utilisateur MofN.

```
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
```

Arguments

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
registerQuorumPubKey <user-type> <user-name> <registration-token> <signed-registration-
token> <public-key>
```

<type-utilisateur>

Spécifie le type d'utilisateur. Ce paramètre est obligatoire.

Pour plus d'informations sur les types d'utilisateur sur un HSM, consultez Comprendre les utilisateurs HSM.

Valeurs valides :

- CO : les responsables de chiffrement peuvent gérer les utilisateurs, mais ne peuvent pas gérer les clés.

Obligatoire : oui

<nom-utilisateur>

Spécifie un nom convivial pour l'utilisateur. La longueur maximale est de 31 caractères. Le seul caractère spécial autorisé est un trait de soulignement (_).

Vous ne pouvez pas modifier le nom d'un utilisateur après l'avoir créé. Dans les commandes cloudhsm_mgmt_util, le type d'utilisateur et le mot de passe sont sensibles à la casse, mais le nom d'utilisateur ne l'est pas.

Obligatoire : oui

<registration-token>

Spécifie le chemin d'accès à un fichier contenant un jeton d'enregistrement non signé. Peut avoir n'importe quelle donnée aléatoire d'une taille de fichier maximale de 245 octets. Pour plus d'informations sur la création d'un jeton d'enregistrement non signé, voir Créer et signer un jeton d'enregistrement.

Obligatoire : oui

<signed-registration-token>

Spécifie le chemin d'accès à un fichier contenant le hachage signé par le mécanisme SHA256_PKCS du jeton registration-token. Pour plus d'informations, voir Créer et signer un jeton d'enregistrement.

Obligatoire : oui

<public-key>

Spécifie le chemin d'accès à un fichier contenant la clé publique d'une paire de clés asymétrique RSA-2048. Utilisez la clé privée pour signer le jeton d'enregistrement. Pour plus d'informations, consultez Créer une paire de clés RSA.

Obligatoire : oui

> **Note**
>
> Le cluster utilise la même clé pour l'authentification par quorum et pour l'authentification à deux facteurs (2FA). Cela signifie que vous ne pouvez pas faire pivoter une clé de quorum

pour un utilisateur pour lequel l'authentification à deux facteurs est activée à l'aide de registerQuorumPubKey. Pour faire pivoter la clé, vous devez utiliser changePswd. Pour plus d'informations sur l'utilisation de l'authentification par quorum et de l'authentification 2FA, consultez Authentification par quorum et 2FA.

Rubriques en relation

- Création d'une paire de clés RSA
- Création et signature d'un jeton d'enregistrement
- Enregistrement d'une clé publique avec le HSM
- Gestion de l'authentification par quorum (contrôle d'accès M sur N)
- Authentification par quorum et 2FA
- listUsers

serveur

Normalement, lorsque vous émettez une commande dans cloudhsm_mgmt_util, la commande s'applique à tous les HSM du cluster désigné (mode global). Toutefois, il peut y avoir des cas où vous devez émettre des commandes pour un seul HSM. Par exemple, dans le cas où une synchronisation automatique échoue, vous devrez peut-être synchroniser les clés et les utilisateurs sur un HSM afin de maintenir la cohérence dans le cluster. Vous pouvez utiliser la commande server de cloudhsm_mgmt_util pour entrer en mode serveur et interagir directement avec une instance HSM donnée.

Lorsque vous aurez validé l'ouverture, l'invite de commande `aws-cloudhsm>` est remplacée par l'invite de commande `server>`.

Pour quitter le mode serveur, utilisez la commande `exit`. Une fois sorti, vous serez renvoyé à l'invite de commande cloudhsm_mgmt_util command.

Avant d'exécuter n'importe quelle commande cloudhsm_mgmt_util, vous devez démarrer cloudhsm_mgmt_util.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Tous les utilisateurs.

Prérequis

Afin d'entrer en mode de serveur, vous devez d'abord connaître le nombre de serveurs du HSM cible. Les numéros de serveurs sont répertoriés dans la sortie de trace générée par cloudhsm_mgmt_util au début. Ils sont attribués dans l'ordre d'affichage des HSM dans le fichier de configuration. Pour cet exemple, nous supposons que `server 0` est le serveur qui correspond au HSM souhaité.

Syntaxe

Pour démarrer le mode serveur :

```
server <server-number>
```

Pour quitter le mode serveur :

```
server> exit
```

Exemple

Cette commande entre en mode serveur sur un HSM avec le numéro de serveur `0`.

```
aws-cloudhsm> server 0

Server is in 'E2' mode...
```

Pour quitter le mode serveur, utilisez la commande exit.

```
server0> exit
```

Arguments

```
server <server-number>
```

<numéro-serveur>

Spécifie le numéro de serveur du HSM cible.

Obligatoire : oui

Il n'y a pas d'arguments pour la commande `exit`.

Rubriques en relation

- syncKey
- createUser
- deleteUser

setAttribute

La commande setAttribute de cloudhsm_mgmt_util modifie la valeur de l'étiquette, chiffre et déchiffre, encapsule et désencapsule les attributs d'une clé dans les HSM. Vous pouvez également utiliser la commande setAttribute de key_mgmt_util pour convertir une clé de session en clé persistante. Vous pouvez uniquement modifier les attributs des clés qui vous appartiennent.

Avant d'exécuter une commande de CMU, vous devez démarrer l'utilitaire CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Syntaxe

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
setAttribute <key handle> <attribute id>
```

Exemple

Cet exemple montre comment désactiver la fonction de déchiffrement d'une clé symétrique. Vous pouvez utiliser une commande comme celle-ci pour configurer une clé d'encapsulage, pour encapsuler et désencapsuler d'autres clés, mais pas pour chiffrer ou déchiffrer les données.

La première étape consiste à créer la clé d'encapsulage. Cette commande est utilisée genSymKeydans key_mgmt_util pour générer une clé symétrique AES 256 bits. La sortie indique que la nouvelle clé a le handle de clé 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
```

Ensuite, nous voulons confirmer la valeur actuelle de l'attribut de déchiffrement. Pour obtenir l'ID de l'attribut de déchiffrement, utilisez listAttributes. La sortie indique que la constante qui représente l'attribut OBJ_ATTR_DECRYPT est 261. Pour obtenir de l'aide sur l'interprétation des attributs de clés, consultez le Référence des attributs de clé.

```
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
```

Pour obtenir la valeur actuelle de l'attribut de déchiffrement pour la clé 14, la commande suivante utilise getAttribute dans cloudhsm_mgmt_util.

La sortie indique que la valeur de l'attribut de déchiffrement est « vrai » (1) sur les deux HSM du cluster.

```
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
```

Cette commande utilise setAttribute pour modifier la valeur de l'attribut de déchiffrement (attribut 261) de la clé 14 en 0. Cela permet de désactiver la fonction de déchiffrement sur la clé.

La sortie indique que la commande a abouti sur les deux HSM du cluster.

```
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)
```

La dernière commande répète la commande getAttribute. Là encore, elle obtient l'attribut de déchiffrement (attribut 261) de la clé 14.

Cette fois, la sortie indique que la valeur de l'attribut de déchiffrement est « faux » (0) sur les deux HSM du cluster.

```
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
```

Arguments

```
setAttribute <key handle> <attribute id>
```

<handle-clé>

Spécifie le handle de clé d'une clé symétrique qui vous appartient. Vous pouvez spécifier une seule clé dans chaque commande. Pour obtenir le handle d'une clé, utilisez findKey dans key_mgmt_util. Pour rechercher les utilisateurs d'une clé, utilisez getKeyInfo.

Obligatoire : oui

<ID attribut>

Spécifie la constante qui représente l'attribut que vous souhaitez modifier. Vous pouvez spécifier un seul attribut dans chaque commande. Pour obtenir les attributs et leurs valeurs entières, utilisez listAttributes. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

Valeurs valides :

- 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.

Obligatoire : oui

Rubriques en relation

- setAttribute dans key_mgmt_util
- getAttribute
- listAttributes
- Référence des attributs de clé

quit

La commande quit de cloudhsm_mgmt_util permet de quitter cloudhsm_mgmt_util. Tous les utilisateurs de tous les types peuvent utiliser cette commande.

Avant d'exécuter n'importe quelle commande cloudhsm_mgmt_util, démarrez cloudhsm_mgmt_util.

Type utilisateur

Les utilisateurs suivants peuvent exécuter cette commande.

- Tous les utilisateurs. Vous n'avez pas besoin d'être connecté pour exécuter cette commande.

Syntaxe

```
quit
```

Exemple

Cette commande permet de quitter cloudhsm_mgmt_util. Après avoir réussi, vous êtes renvoyé à la ligne de commande standard. Cette commande n'a pas de paramètres de sortie.

```
aws-cloudhsm> quit

disconnecting from servers, please wait...
```

Rubriques en relation

- Se lancer avec cloudhsm_mgmt_util

shareKey

La commande shareKey de cloudhsm_mgmt_util permet de partager et d'annuler le partage des clés que vous détenez avec d'autres utilisateurs de chiffrement. Seul le propriétaire de la clé peut partager et annuler le partage d'une clé. Vous pouvez également partager une clé au moment de sa création.

Les utilisateurs qui partagent la clé peut l'utiliser dans des opérations de chiffrement, mais ils ne peuvent pas la supprimer, l'exporter, la partager ou en annuler le partage, ni en modifier les attributs. Lorsque l'authentification par quorum est activée sur une clé, le quorum doit approuver toutes les opérations qui partagent ou annulent le partage de la clé.

Avant d'exécuter une commande de CMU, vous devez démarrer l'utilitaire de CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Utilisateurs de chiffrement (CU)

Syntaxe

Comme cette commandes n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

Type d'utilisateur : utilisateur de chiffrement

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

Exemple

Les exemples suivants montrent comment utiliser shareKey pour partager et annuler le partage des clés que vous possédez avec d'autres utilisateurs de chiffrement.

Example : Partager une clé

Cet exemple utilise shareKey pour partager une clé privée ECC qui appartient à l'utilisateur actuel avec un autre utilisateur de chiffrement sur les HSM. Des clés publiques sont disponibles pour tous les utilisateurs du HSM, vous ne pouvez donc pas les partager ou en annuler le partage.

La première commande permet getKeyInfod'obtenir les informations utilisateur pour la clé262177, une clé privée ECC sur les HSM.

La sortie indique que la clé 262177 appartient à l'utilisateur 3, mais qu'elle n'est pas partagée.

```
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
```

Cette commande utilise shareKey pour partager la clé 262177 avec l'utilisateur 4, un autre utilisateur de chiffrement des HSM. L'argument final utilise la valeur 1 pour indiquer une opération de partage.

La sortie indique que l'opération a abouti sur les deux HSM du cluster.

```
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)
```

Pour vérifier que l'opération a abouti, l'exemple répète la première commande getKeyInfo.

La sortie indique que la clé 262177 est désormais partagée avec l'utilisateur 4.

```
aws-cloudhsm>getKeyInfo 262177

Key Info on server 0(10.0.3.10):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
Key Info on server 1(10.0.3.6):

        Token/Flash Key,

        Owned by user 3

        also, shared to following 1 user(s):

                4
```

Example : Annuler le partage d'une clé

Cet exemple annule le partage d'une de chiffrement symétrique, c'est-à-dire qu'il supprime un utilisateur de chiffrement de la liste des utilisateurs partagés pour la clé.

Cette commande utilise shareKey pour supprimer l'utilisateur 4 de la liste des utilisateurs partagés pour la clé 6. L'argument final utilise la valeur `0` pour indiquer une opération d'annulation du partage.

La sortie indique que la commande a abouti sur les deux HSM. Par conséquent, l'utilisateur 4 ne peut plus utiliser la clé 6 dans des opérations de chiffrement.

```
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)
```

Arguments

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
shareKey <key handle> <user id> <(share/unshare key?) 1/0>
```

<handle-clé>

Spécifie le handle de clé d'une clé symétrique qui vous appartient. Vous pouvez spécifier une seule clé dans chaque commande. Pour obtenir le handle d'une clé, utilisez findKey dans key_mgmt_util. Pour vérifier que vous possédez une clé, utilisez getKeyInfo.

Obligatoire : oui

<ID utilisateur>

Indique l'ID utilisateur de l'utilisateur de chiffrement avec qui vous partagez ou annulez le partage de la clé. Pour obtenir l'ID d'un utilisateur, utilisez listUsers.

Obligatoire : oui

<partage 1 ou annulation du partage 0>

Pour partager la clé avec l'utilisateur spécifié, tapez 1. Pour annuler le partage de la clé, c'est-à-dire pour supprimer l'utilisateur spécifié de la liste des utilisateurs partagés pour la clé, tapez `0`.

Obligatoire : oui

Rubriques en relation

- getKeyInfo

syncKey

Vous pouvez utiliser la commande syncKey dans cloudhsm_mgmt_util pour synchroniser manuellement des clés entre des instances HSM au sein d'un cluster ou entre des clusters clonés. En général, vous n'avez pas besoin d'utiliser cette commande, car les instances de HSM au sein d'un cluster synchronisent les clés automatiquement. Toutefois, la synchronisation de clés entre des clusters clonés doit être réalisée manuellement. Les clusters clonés sont généralement créés dans différentes AWS régions afin de simplifier les processus de mise à l'échelle mondiale et de reprise après sinistre.

Vous ne pouvez pas utiliser syncKey pour synchroniser des clés entre des clusters arbitraires : l'un des clusters doit avoir été créé à partir d'une sauvegarde de l'autre. En outre, les deux clusters doivent avoir des informations d'identification de CU et de CO cohérentes pour que l'opération réussisse. Pour plus d'informations, consultez Utilisateurs HSM.

Pour l'utilisersyncKey, vous devez d'abord créer un fichier de AWS CloudHSM configuration qui spécifie un HSM provenant du cluster source et un autre provenant du cluster de destination. Cela permet à cloudhsm_mgmt_util de se connecter aux deux instances HSM. Utilisez ce fichier de configuration pour démarrer cloudhsm_mgmt_util. Ensuite, connectez-vous avec les informations d'identification d'un CO ou d'un CU qui possède les clés à synchroniser.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Responsables de chiffrement (CO)
- Utilisateurs de chiffrement (CU)

> **Note**
>
> Les CO peuvent utiliser syncKey sur toutes les clés, tandis que les CU peuvent uniquement utiliser cette commande sur les clés qui leur appartiennent. Pour plus d'informations, consultez the section called "Comprendre les utilisateurs HSM".

Prérequis

Avant de commencer, vous devez connaître le `key handle` de la clé sur le HSM source à synchroniser avec le HSM de destination. Pour trouver le `key handle`, utilisez la commande listUsers pour répertorier tous les identifiants des utilisateurs nommés. Utilisez ensuite la findAllKeyscommande pour rechercher toutes les clés appartenant à un utilisateur en particulier.

Vous devez également connaître les `server IDs` affectés aux HSM source et de destination, qui sont affichés dans la sortie de suivi renvoyée par cloudhsm_mgmt_util au lancement. Ils sont attribués dans l'ordre d'affichage des HSM dans le fichier de configuration.

Suivez les instructions fournies dans la section Utilisation de l'utilitaire CMU entre des clusters clonés et initialisez cloudhsm_mgmt_util avec le nouveau fichier de configuration. Ensuite, activez le mode serveur sur le HSM source en exécutant la commande server.

Syntaxe

> **Note**
>
> Pour exécuter syncKey, commencez par activer le mode serveur sur le HSM qui contient la clé à synchroniser.

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

Type d'utilisateur : utilisateur de chiffrement

```
syncKey <key handle> <destination hsm>
```

Exemple

Exécutez la commande server pour vous connecter au HSM source et activez le mode serveur. Pour cet exemple, nous supposons que `server 0` est le HSM source.

```
aws-cloudhsm> server 0
```

À présent, exécutez la commande syncKey. Dans cet exemple, nous supposons que la clé `261251` doit être synchronisée avec `server 1`.

```
aws-cloudhsm> syncKey 261251 1
syncKey success
```

Arguments

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
syncKey <key handle> <destination hsm>
```

<handle de clé>

Spécifie le handle de clé de la clé à synchroniser. Vous pouvez spécifier une seule clé dans chaque commande. Pour obtenir le descripteur d'une clé, utilisez-le findAllKeyslorsque vous êtes connecté à un serveur HSM.

Obligatoire : oui

<hsm de destination>

Spécifie le numéro du serveur avec lequel vous synchronisez une clé.

Obligatoire : oui

Rubriques en relation

- listUsers
- findAllKeys
- describe-clusters dans la CLI
- serveur

syncUser

Vous pouvez utiliser la syncUser commande contenue dans cloudhsm-mgmt_util pour synchroniser manuellement les utilisateurs de chiffrement (CU) ou les agents de chiffrement (CO) entre les instances HSM d'un cluster ou entre les clusters clonés. AWS CloudHSM ne synchronise pas automatiquement les utilisateurs. En général, vous gérez des utilisateurs en mode global afin que tous les HSM d'un cluster soient actualisés ensemble. Vous devrez peut-être utiliser syncUser si un HSM est désynchronisé accidentellement (par exemple, en raison d'un changement de mot de passe) ou si vous souhaitez effectuer une rotation des informations d'identification utilisateur entre des clusters clonés. Les clusters clonés sont généralement créés dans différentes AWS régions afin de simplifier les processus de mise à l'échelle mondiale et de reprise après sinistre.

Avant d'exécuter une commande de CMU, vous devez démarrer l'utilitaire CMU et vous connecter au HSM. Veillez à ce que le type d'utilisateur du compte que vous utilisez pour vous connecter puisse exécuter les commandes que vous prévoyez d'utiliser.

Si vous ajoutez ou supprimez des HSM, mettez à jour les fichiers de configuration de l'utilitaire CMU. Sinon, les modifications que vous apportez peuvent ne pas être effectives sur tous les HSM du cluster.

Type utilisateur

Les types d'utilisateur suivants peuvent exécuter cette commande.

- Responsables de chiffrement (CO)

Prérequis

Avant de commencer, vous devez connaître l'`user ID` de l'utilisateur sur le HSM source à synchroniser avec le HSM de destination. Pour trouver l'`user ID`, utilisez la commande listUsers pour répertorier tous les utilisateurs sur les HSM d'un cluster.

Vous devez également connaître les `server ID` affectés aux HSM source et de destination, qui sont affichés dans la sortie de suivi renvoyée par cloudhsm_mgmt_util au lancement. Ils sont attribués dans l'ordre d'affichage des HSM dans le fichier de configuration.

Si vous synchronisez des HSM entre des clusters clonés, suivez les instructions dans Utilisation de l'utilitaire CMU entre des clusters clonés et initialisez cloudhsm_mgmt_util avec le nouveau fichier de configuration.

Lorsque vous êtes prêt à exécuter syncUser, activez le mode serveur sur le HSM source en exécutant la commande server.

Syntaxe

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
syncUser <user ID> <server ID>
```

Exemple

Exécutez la commande server pour vous connecter au HSM source et activez le mode serveur. Pour cet exemple, nous supposons que `server 0` est le HSM source.

```
aws-cloudhsm> server 0
```

Maintenant, exécutez la commande syncUser. Pour cet exemple, nous supposons que l'utilisateur 6 est l'utilisateur à synchroniser, et `server 1` est le HSM de destination.

```
server 0> syncUser 6 1
ExtractMaskedObject: 0x0 !
InsertMaskedObject: 0x0 !
syncUser success
```

Arguments

Comme cette commande n'a pas de paramètres nommés, vous devez entrer les arguments dans l'ordre spécifié dans le diagramme de syntaxe.

```
syncUser <user ID> <server ID>
```

<ID utilisateur>

Spécifie l'ID de l'utilisateur à synchroniser. Vous pouvez spécifier un seul utilisateur dans chaque commande. Pour obtenir l'ID d'un utilisateur, utilisez listUsers.

Obligatoire : oui

<ID serveur>

Spécifie le numéro du serveur du HSM avec lequel vous synchronisez un utilisateur.

Obligatoire : oui

Rubriques en relation

- listUsers
- describe-clusters dans la CLI
- serveur

Utilitaire de gestion des clés (KMU)

L'Utilitaire de gestion des clés (KMU) est un outil de ligne de commande qui aide les utilisateurs de chiffrement (CU) à gérer les clés des modules de sécurité matériels (HSM). Il inclut plusieurs commandes qui génèrent, suppriment, importent et exportent des clés, obtiennent et définissent des attributs, recherchent des clés et effectuent des opérations de chiffrement.

KMU et CMU font partie de la suite du SDK client 3.

Pour une mise en route rapide, consultez Mise en route avec key_mgmt_util. Pour obtenir des informations détaillées sur ces commandes, consultez Référence de commande key_mgmt_util. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

Pour utiliser l'outil key_mgmt_util si vous utilisez Linux, connectez-vous à votre instance client, puis consultez Installation et configuration du AWS CloudHSM client (Linux). Si vous utilisez Windows, consultez Installation et configuration du AWS CloudHSM client (Windows).

Rubriques

- Mise en route avec key_mgmt_util
- Installation et configuration du AWS CloudHSM client (Linux)
- Installation et configuration du AWS CloudHSM client (Windows)
- Référence de commande key_mgmt_util

Mise en route avec key_mgmt_util

AWS CloudHSM inclut deux outils de ligne de commande avec le logiciel AWS CloudHSM client. L'outil cloudhsm_mgmt_util inclut les commandes pour gérer les utilisateurs HSM. L'outil

key_mgmt_util inclut les commandes de gestion des clés. Pour démarrer avec l'outil de ligne de commande key_mgmt_util, consultez les rubriques suivantes.

Rubriques

- Configuration de key_mgmt_util
- Utilisation de base de l'outil key_mgmt_util

Si vous rencontrez un message d'erreur ou des résultats inattendus pour une commande, consultez les rubriques Résolution des problèmes AWS CloudHSM pour obtenir de l'aide. Pour plus d'informations sur les commandes key_mgmt_util, consultez Référence de commande key_mgmt_util.

Configuration de key_mgmt_util

Effectuez la configuration suivante avant d'utiliser l'outil key_mgmt_util.

Démarrez le AWS CloudHSM client

Avant d'utiliser key_mgmt_util, vous devez démarrer le client. AWS CloudHSM Le client est un démon qui établit une communication end-to-end cryptée avec les HSM de votre cluster. L'outil key_mgmt_util utilise la connexion du client pour communiquer avec les HSM de votre cluster. Sans cela, l'outil key_mgmt_util ne fonctionne pas.

Pour démarrer le AWS CloudHSM client

Utilisez la commande suivante pour démarrer le AWS CloudHSM client.

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

- Pour le Client Windows version 1.1.2 et ultérieure :

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Pour les clients Windows version 1.1.1 et antérieure :

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Démarrage de l'outil key_mgmt_util

Après avoir démarré le AWS CloudHSM client, utilisez la commande suivante pour démarrer 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
```

L'invite devient `Command:` lorsque l'outil key_mgmt_util est en cours d'exécution.

Si la commande échoue, par exemple en renvoyant un message `Daemon socket connection error`, essayez d'effectuer la mise à jour de votre fichier de configuration.

Utilisation de base de l'outil key_mgmt_util

Consultez les rubriques suivantes pour connaître le fonctionnement de base de l'outil key_mgmt_util.

Rubriques

Connectez-vous aux HSM

Utilisez la commande loginHSM pour vous connecter aux HSM. La commande suivante se connecte en tant qu'utilisateur de chiffrement (CU) nommé `example_user`. La sortie indique une connexion réussie pour les trois HSM du cluster.

```
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
```

L'exemple suivant montre la syntaxe de la commande loginHSM.

```
Command:  loginHSM -u <USER TYPE> -s <USERNAME> -p <PASSWORD>
```

Déconnectez-vous des HSM

Utilisez la commande logoutHSM pour vous déconnecter des HSM.

```
Command:  logoutHSM
Cfm3LogoutHSM returned: 0x00 : HSM Return: SUCCESS

Cluster Error Status
Node id 0 and err state 0x00000000 : HSM Return: SUCCESS
Node id 1 and err state 0x00000000 : HSM Return: SUCCESS
Node id 2 and err state 0x00000000 : HSM Return: SUCCESS
```

Arrêt de l'outil key_mgmt_util

Pour arrêter l'outil exit, utilisez la commande key_mgmt_util.

```
Command:  exit
```

Installation et configuration du AWS CloudHSM client (Linux)

Pour interagir avec le HSM de votre AWS CloudHSM cluster, vous avez besoin du logiciel AWS CloudHSM client pour Linux. Nous vous conseillons de l'installer sur l'instance client Linux EC2 créée précédemment. Vous pouvez également installer un client si vous utilisez Windows. Pour plus d'informations, consultez Installation et configuration du AWS CloudHSM client (Windows).

Tâches

- Installation du AWS CloudHSM client et des outils de ligne de commande
- Modification de la configuration du client

Installation du AWS CloudHSM client et des outils de ligne de commande

Connectez-vous à votre instance cliente et exécutez les commandes suivantes pour télécharger et installer le AWS CloudHSM client et les outils de ligne de commande.

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
```

Modification de la configuration du client

Avant de pouvoir utiliser le AWS CloudHSM client pour vous connecter à votre cluster, vous devez modifier la configuration du client.

Pour modifier la configuration du logiciel client

1. Copiez votre certificat d'émission celui que vous avez utilisé pour signer le certificat du cluster à l'emplacement suivant sur l'instance client : `/opt/cloudhsm/etc/customerCA.crt`. Vous avez besoin d'autorisations d'utilisateur racine sur l'instance de votre client pour copier votre certificat à cet emplacement.
2. Utilisez la commande de configuration suivante pour mettre à jour les fichiers de configuration du AWS CloudHSM client et des outils de ligne de commande, en spécifiant l'adresse IP du HSM de votre cluster. Pour obtenir l'adresse IP du HSM, consultez votre cluster dans la AWS CloudHSM console ou exécutez la commande describe-clusters CLI. Dans la sortie de la commande, l'adresse IP du HSM est la valeur du champ `EniIp`. Si vous avez plusieurs HSM, choisissez l'adresse IP de l'un des HSM, peu importe lequel.

   ```
   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. Accédez à Activation du cluster.

Installation et configuration du AWS CloudHSM client (Windows)

Pour utiliser un HSM dans votre AWS CloudHSM cluster sous Windows, vous avez besoin du logiciel AWS CloudHSM client pour Windows. Nous vous conseillons de l'installer sur l'instance Windows Server que vous avez créée précédemment.

Pour installer (ou mettre à jour) le client et les outils de ligne de commande Windows les plus récents

1. Connectez-vous à votre instance Windows Server.
2. Téléchargez la dernière version (AWSCloudHSMClient-latest.msi) depuis la page des téléchargements.
3. Accédez à votre emplacement de téléchargement et exécutez le programme d'installation (AWSCloudHSMClient-latest.msi) avec des privilèges d'administrateur.
4. Suivez les instructions du programme d'installation, puis choisissez Fermer une fois le programme d'installation terminé.
5. Copiez votre certificat d'émission auto-signé, celui que vous avez utilisé pour signer le certificat du cluster, dans le dossier `C:\ProgramData\Amazon\CloudHSM`.
6. Exécutez la commande suivante pour mettre à jour vos fichiers de configuration. Veillez à arrêter et démarrer le client au cours de la reconfiguration si vous le mettez à jour :

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

7. Accédez à Activation du cluster.

Remarques :

- Si vous mettez à jour le client, les fichiers de configuration existants des installations précédentes ne sont pas remplacés.
- Le programme d'installation du AWS CloudHSM client pour Windows enregistre automatiquement l'API de chiffrement : Next Generation (CNG) et le fournisseur de stockage de clés (KSP). Pour désinstaller le client, exécutez à nouveau le programme d'installation et suivez les instructions de désinstallation.
- Vous pouvez également installer le client Linux si vous utilisez Linux. Pour plus d'informations, voir Installation et configuration du AWS CloudHSM client (Linux).

Référence de commande key_mgmt_util

L'outil de ligne de commande key_mgmt_util vous aide à gérer des clés dans les HSM de votre cluster, y compris à créer, supprimer et rechercher des clés et leurs attributs. Il inclut plusieurs commandes, chacune d'entre elles étant décrite en détail dans cette rubrique.

Pour une mise en route rapide, consultez Mise en route avec key_mgmt_util. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé. Pour plus d'informations sur l'outil de ligne de commande cloudhsm_mgmt_util qui inclut des commandes pour gérer le HSM et les utilisateurs de votre cluster, consultez Utilitaire de gestion CloudHSM (CMU).

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Pour répertorier toutes les commandes key_mgmt_util, tapez :

```
Command: help
```

Pour obtenir de l'aide sur une commande key_mgmt_util, tapez :

```
Command: <command-name> -h
```

Pour terminer votre session key_mgmt_util, tapez :

```
Command: exit
```

Les rubriques suivantes décrivent les commandes dans key_mgmt_util.

> **Note**
>
> Certaines commandes de key_mgmt_util et cloudhsm_mgmt_util portent le même nom. Cependant, les commandes ont généralement une syntaxe différente, une sortie différente et des fonctionnalités légèrement différentes.

Command	Description
aesWrapUnwrap	Chiffre et déchiffre le contenu d'une clé dans un fichier.

Command	Description
deleteKey	Supprime une clé des HSM.
Error2String	Permet d'obtenir l'erreur qui correspond à un code d'erreur hexadécimal key_mgmt_util.
exit	Quitte le key_mgmt_util.
exportPrivateKey	Exporte une copie d'une clé privée depuis un HSM vers un fichier sur disque.
exportPubKey	Exporte une copie d'une clé publique d'un HSM vers un fichier.
exSymKey	Exporte une copie en texte brut d'une clé symétrique à partir des HSM vers un fichier.
extractMaskedObject	Extrait une clé depuis un HSM sous la forme d'un fichier d'objets masqués.
findKey	Recherche des clés par valeur d'attribut de clé.
findSingleKey	Vérifie qu'une clé existe sur tous les HSM du cluster.
GendSA KeyPair	Génère une paire de clés Digital Signing Algorithm dans vos HSM.
GèneCC KeyPair	Génère une paire de clés Elliptic Curve Cryptography (ECC) dans vos HSM.
Genre RSA KeyPair	Génère une paire de clés asymétrique RSA dans vos HSM.
genSymKey	Génère une clé symétrique dans vos HSM.
getAttribute	Permet d'obtenir les valeurs d'attribut pour une clé AWS CloudHSM et de les écrire dans un fichier.

Command	Description
getCaviumPrivClé	Crée une version de format PEM factice d'une clé privée et l'exporte vers un fichier.
getCert	Extrait les certificats de partitions d'un HSM et les enregistre dans un fichier.
getKeyInfo	Permet d'obtenir les ID HSM des utilisateurs qui peuvent utiliser la clé. Si la clé est contrôlée par quorum, elle permet d'obtenir le nombre d'utilisateurs dans le quorum.
help	Affiche les informations d'aide sur les commandes disponibles dans key_mgmt_util.
importPrivateKey	Importe une clé privée dans un HSM.
importPubKey	Importe une clé publique dans un HSM.
imSymKey	Importe une copie en texte brut d'une clé symétrique à partir d'un fichier dans le HSM.
insertMaskedObject	Insère un objet masqué à partir d'un fichier sur le disque dans un HSM contenu dans le cluster lié vers le cluster d'origine de l'objet. Les clusters liés sont tous les clusters générés à partir d'une sauvegarde du cluster d'origine.
???	Détermine si un fichier contient une clé privée réelle ou une fausse clé PEM.
listAttributes	Répertorie les attributs d'une AWS CloudHSM clé et les constantes qui les représentent.

Command	Description
listUsers	Permet d'obtenir la liste des utilisateurs des HSM, leur type et leur ID, ainsi que d'autres attributs.
loginHSM et logoutHSM	Permet la connexion au HSM et la déconnexion du HSM dans un cluster.
setAttribute	Convertit une clé de session en clé persistante.
sign	Génère une signature pour un fichier à l'aide d'une clé privée sélectionnée.
unWrapKey	Importe une clé encapsulée (chiffrée) à partir d'un fichier dans les HSM.
verify	Vérifie si une clé spécifique a été utilisée pour signer un fichier donné.
wrapKey	Exporte une copie chiffrée d'une clé à partir du HSM vers un fichier.

aesWrapUnwrap

La commande aesWrapUnwrap chiffre ou déchiffre le contenu d'un fichier sur le disque. Cette commande est conçue pour encapsuler ou désencapsuler les clés de chiffrement, mais vous pouvez l'utiliser sur n'importe quel fichier qui contient moins de 4 Ko (4 096 octets) de données.

aesWrapUnwrap utilise l'algorithme AES Key Wrap. Il utilise une clé AES sur le HSM en tant que clé d'encapsulage ou de désencapsulage. Ensuite, il écrit le résultat dans un autre fichier sur le disque.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
aesWrapUnwrap -h

aesWrapUnwrap -m <wrap-unwrap mode>
```

```
                -f <file-to-wrap-unwrap>
                -w <wrapping-key-handle>
                [-i <wrapping-IV>]
                [-out <output-file>]
```

Exemples

Ces exemples illustrent comment utiliser aesWrapUnwrap pour chiffrer et déchiffrer une clé de chiffrement dans un fichier.

Example : Encapsuler une clé de chiffrement

Cette commande utilise aesWrapUnwrap pour encapsuler une clé symétrique Triple DES qui a été exportée à partir du HSM en texte brut dans le fichier `3DES.key`. Vous pouvez utiliser une commande similaire pour encapsuler n'importe quelle clé enregistrée dans un fichier.

La commande utilise le paramètre `-m` avec la valeur 1 pour indiquer le mode d'encapsulage. Elle utilise le paramètre `-w` pour spécifier une clé AES dans le HSM (handle de clé 6) comme clé d'encapsulage. Elle écrit la clé encapsulée obtenue dans le fichier `3DES.key.wrapped`.

La sortie indique que la commande a réussi et que l'opération a utilisé la valeur initiale par défaut, qui est préférable.

```
 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 : Désencapsuler une clé de chiffrement

Cet exemple montre comment utiliser aesWrapUnwrap pour désencapsuler (déchiffrer) une clé encapsulée (chiffrée) dans un fichier. Vous pouvez effectuer une opération comme celle-ci avant

d'importer une clé sur le HSM. Par exemple, si vous essayez d'utiliser la imSymKeycommande pour importer une clé chiffrée, elle renvoie une erreur car la clé chiffrée n'a pas le format requis pour une clé en texte brut de ce type.

La commande désencapsule la clé dans le fichier `3DES.key.wrapped` et écrit le texte brut dans le fichier `3DES.key.unwrapped`. La commande utilise le paramètre `-m` avec la valeur `0` pour indiquer le mode de désencapsulage. Elle utilise le paramètre `-w` pour spécifier une clé AES dans le HSM (handle de clé 6) comme clé d'encapsulage. Elle écrit la clé encapsulée obtenue dans le fichier `3DES.key.unwrapped`.

```
 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
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-m

Spécifie le mode. Pour encapsuler (chiffrer) le contenu du fichier, tapez 1 ; pour désencapsuler (déchiffrer) le contenu du fichier, tapez `0`.

Obligatoire : oui

-f

Spécifie le fichier à encapsuler. Entrez un fichier contenant moins de 4 Ko (4 096 octets) de données. Cette opération est conçue pour encapsuler et désencapsuler les clés de chiffrement.

Obligatoire : oui

-s, sem

Spécifie la clé d'encapsulage. Entrez le handle d'une clé AES sur le HSM. Ce paramètre est obligatoire. Pour trouver des handles de clé, utilisez la commande findKey.

Pour créer une clé d'encapsulation, genSymKeyutilisez-la pour générer une clé AES (type 31).

Obligatoire : oui

-i

Spécifie une autre valeur initiale (IV) pour l'algorithme. Utilisez la valeur par défaut, sauf si des conditions particulières nécessitent une alternative.

Par défaut: `0xA6A6A6A6A6A6A6A6`. La valeur par défaut est définie dans les spécifications de l'algorithme AES Key Wrap.

Obligatoire : non

-out

Spécifie un autre nom pour le fichier de sortie qui contient la clé encapsulée ou désencapsulée. La valeur par défaut est `wrapped_key` (pour les opérations d'encapsulage) et `unwrapped_key` (pour les opérations de désencapsulage) dans le répertoire local.

Si le fichier existe, la commande aesWrapUnwrap le remplace sans avertissement. Si la commande échoue, aesWrapUnwrap crée un fichier de sortie sans contenu.

Par défaut : pour l'encapsulage : `wrapped_key`. Pour le désencapsulage : `unwrapped_key`.

Obligatoire : non

Rubriques en relation

- exSymKey
- imSymKey
- unWrapKey
- wrapKey

deleteKey

La commande deleteKey de key_mgmt_util permet de supprimer une clé du HSM. Vous pouvez supprimer une seule clé à la fois. La suppression d'une clé dans une paire de clés n'a aucun effet sur l'autre clé de la paire.

Seul le propriétaire de la clé peut la supprimer. Les utilisateurs qui partagent la clé peut l'utiliser dans des opérations de chiffrement, mais pas la supprimer.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
deleteKey -h

deleteKey -k
```

Exemples

Ces exemples montrent comment utiliser deleteKey pour supprimer des clés de vos HSM.

Example : Supprimer une clé

Cette commande supprime la clé avec le handle de clé 6. Lorsque la commande aboutit, deleteKey renvoie des messages de réussite depuis chaque HSM du cluster.

```
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 : Supprimer une clé (échec)

Lorsque la commande échoue car aucune clé ne dispose du handle spécifié, deleteKey renvoie un message d'erreur signalant un handle d'objet non valide.

```
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
```

Lorsque la commande échoue car l'utilisateur actuel n'est pas le propriétaire de la clé, la commande renvoie une erreur d'accès refusé.

```
Command:  deleteKey -k 262152

Cfm3DeleteKey returned: 0xc6 : HSM Error: Key Access is denied.
```

Paramètres

-h

Affiche l'aide de la ligne de commande pour la commande.

Obligatoire : oui

-k

Spécifie le handle de clé de la clé à supprimer. Pour rechercher les handles de clé dans le HSM, utilisez findKey.

Obligatoire : oui

Rubriques en relation

- findKey

Error2String

La commande d'assistance Error2String de key_mgmt_util renvoie l'erreur qui correspond à un code d'erreur hexadécimal key_mgmt_util. Vous pouvez utiliser cette commande pour dépanner vos commandes et vos scripts.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
Error2String -h

Error2String -r <response-code>
```

Exemples

Ces exemples montrent comment utiliser Error2String pour obtenir la chaîne d'erreur pour un code d'erreur key_mgmt_util.

Example : Obtenir une description de l'erreur

Cette commande permet d'obtenir la description de l'erreur pour le code d'erreur `0xdb`. La description explique qu'une tentative de connexion à key_mgmt_util a échoué, car le type de l'utilisateur est incorrect. Seuls les utilisateurs de chiffrement (CU) peuvent se connecter à key_mgmt_util.

```
        Command:  Error2String -r 0xdb

        Error Code db maps to HSM Error: Invalid User Type.
```

Example : Trouver le code d'erreur

Cet exemple montre où trouver le code d'erreur dans une erreur key_mgmt_util. Le code d'erreur, `0xc6`, s'affiche après la chaîne : `Cfm3`*`command-name`* `returned:` .

Dans cet exemple, getKeyInfoindique que l'utilisateur actuel (utilisateur 4) peut utiliser la clé dans des opérations cryptographiques. Toutefois, lorsque l'utilisateur essaie d'utiliser deleteKey pour supprimer la clé, la commande renvoie le code d'erreur `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
```

Si l'erreur `0xc6` est signalée, vous pouvez utiliser une commande Error2String comme celle-ci pour rechercher l'erreur. Dans ce cas, la commande `deleteKey` a échoué avec une erreur d'accès refusé car la clé est partagée avec l'utilisateur actuel, mais appartient à un autre utilisateur. Seuls les propriétaires de clé sont autorisés à supprimer une clé.

```
        Command:  Error2String -r 0xa8

        Error Code c6 maps to HSM Error: Key Access is denied
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-r

Spécifie un code d'erreur hexadécimal. L'indicateur hexadécimal `0x` est obligatoire.

Obligatoire : oui

exit

La commande exit contenue dans key_mgmt_util quitte key_mgmt_util. Lorsque vous quittez, vous êtes renvoyé à la ligne de commande standard.

Avant d'exécuter n'importe quelle commande key_mgmt_util, vous devez démarrer l'outil key_mgmt_util.

Syntaxe

```
exit
```

Paramètres

Il n'existe aucun paramètre pour cette commande.

Rubriques en relation

- Démarrage de l'outil key_mgmt_util

exportPrivateKey

La commande exportPrivateKey dans key_mgmt_util exporte une clé privée asymétrique d'un HSM dans un fichier. Le HSM n'autorise pas l'exportation directe des clés en texte clair. La commande encapsule la clé privée à l'aide d'une clé d'encapsulage AES que vous spécifiez, déchiffre les octets encapsulés et copie la clé privée en texte clair dans un fichier.

La commande exportPrivateKey ne supprime pas la clé du HSM, ne modifie pas ses attributs de clé, ni ne vous empêche d'utiliser la clé dans d'autres opérations de chiffrement. Vous pouvez exporter la même clé plusieurs fois.

Vous pouvez uniquement exporter les clés privées qui ont la valeur d'attribut `OBJ_ATTR_EXTRACTABLE1`. Vous devez spécifier une clé d'encapsulage AES dotée de la valeur d'attribut `OBJ_ATTR_WRAP` et`OBJ_ATTR_DECRYPT 1`. Pour obtenir les attributs d'une clé, utilisez la commande getAttribute.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
exportPrivateKey -h

exportPrivateKey -k <private-key-handle
                 -w <wrapping-key-handle>
                 -out <key-file>
                 [-m <wrapping-mechanism>]
                 [-wk <wrapping-key-file>]
```

Exemples

Cet exemple montre comment utiliser exportPrivateKey pour exporter une clé privée hors d'un HSM.

Example : Exporter une clé privée

Cette commande exporte une clé privée avec le handle 15 à l'aide d'une clé d'encapsulage avec handle 16 vers un fichier PEM appelé `exportKey.pem`. Lorsque la commande aboutit, exportPrivateKey renvoie un message de réussite.

```
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
```

Paramètres

Cette commande accepte les paramètres suivants :

`-h`

Affiche l'aide de la ligne de commande pour la commande.

Obligatoire : oui

`-k`

Spécifie le handle de clé de la clé privée à exporter.

Obligatoire : oui

`-w`

Spécifie le handle de clé de la clé d'encapsulage. Ce paramètre est obligatoire. Pour trouver des handles de clé, utilisez la commande findKey.

Pour déterminer si une clé peut être utilisée comme clé d'encapsulage, utilisez getAttribute pour obtenir la valeur de l'attribut `OBJ_ATTR_WRAP` (262). Pour créer une clé d'encapsulage, utilisez genSymKey pour créer une clé AES (type 31).

Si vous utilisez le paramètre `-wk` pour spécifier une clé de désencapsulage externe, la clé d'encapsulage `-w` est utilisée pour encapsuler la clé lors de l'exportation, mais pas pour la désencapsuler.

Obligatoire : oui

`-out`

Spécifie le nom du fichier dans lequel la clé privée exportée sera écrite.

Obligatoire : oui

`-m`

Spécifie le mécanisme d'encapsulage pour encapsuler la clé privée exporté. La seule valeur valide est `4`, qui représente le `NIST_AES_WRAP mechanism.`.

Par défaut :`4` (`NIST_AES_WRAP`)

Obligatoire : non

`-wk`

Spécifie la clé à utiliser pour désencapsuler la clé en cours d'exportation. Entrez le chemin et le nom d'un fichier qui contient une clé AES en texte brut.

Lorsque vous incluez ce paramètre, exportPrivateKey utilise la clé dans le fichier `-w` pour encapsuler la clé en cours d'exportation, puis la clé spécifiée par le paramètre `-wk` pour la désencapsuler.

Par défaut : Utilise la clé d'encapsulage spécifiée dans le paramètre `-w` pour encapsuler et désencapsuler.

Obligatoire : non

Rubriques en relation

- importPrivateKey
- wrapKey
- unWrapKey
- genSymKey

exportPubKey

La commande exportPubKey dans key_mgmt_util exporte une clé publique dans un HSM vers un fichier. Vous pouvez l'utiliser pour exporter des clés publiques que vous générez dans un HSM. Vous pouvez également utiliser cette commande pour exporter les clés publiques qui ont été importées dans un HSM, telles que celles importées avec la commande importPubKey.

L'opération exportPubKey copie les clés dans un fichier que vous spécifiez. Mais elle ne supprime pas la clé du HSM, ne modifie pas ses attributs clés, ni vous empêche d'utiliser la clé dans d'autres opérations de chiffrement. Vous pouvez exporter la même clé plusieurs fois.

Vous pouvez uniquement exporter les clés publiques dont la valeur `OBJ_ATTR_EXTRACTABLE` égale 1. Pour obtenir les attributs d'une clé, utilisez la commande getAttribute.

Avant d'exécuter des commandes `key_mgmt_util`, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
exportPubKey -h

exportPubKey -k <public-key-handle
             -out <key-file>
```

Exemples

Cet exemple montre comment utiliser exportPubKey pour exporter une clé publique à partir d'un HSM.

Example : Exporter une clé publique

Cette commande exporte une clé publique avec handle `10` dans un fichier appelé `public.pem`. Lorsque la commande aboutit, exportPubKey renvoie un message de réussite.

```
Command: exportPubKey -k 10 -out public.pem

PEM formatted public key is written to public.pem

Cfm3ExportPubKey returned: 0x00 : HSM Return: SUCCESS
```

Paramètres

Cette commande accepte les paramètres suivants :

`-h`

Affiche l'aide de la ligne de commande pour la commande.

Obligatoire : oui

`-k`

Spécifie le handle de clé de la clé publique à exporter.

Obligatoire : oui

`-out`

Spécifie le nom du fichier dans lequel la clé publique exportée sera écrite.

Obligatoire : oui

Rubriques en relation

- importPubKey
- Générez des clés

exSymKey

La commande exSymKey de l'outil key_mgmt_util permet d'exporter une copie en texte brut d'une clé symétrique à partir du HSM et de l'enregistrer dans un fichier sur disque. Pour exporter une copie chiffrée (encapsulée) d'une clé, utilisez wrapKey. Pour importer une clé en texte brut, comme celles qui sont `exSymKey` exportées, utilisez imSymKey.

Au cours du processus d'exportation, exSymKey utilise une clé AES que vous spécifiez (la clé d'encapsulage) pour encapsuler (chiffrer), puis désencapsuler (déchiffrer) la clé à exporter. Toutefois, le résultat de l'opération d'exportation est une clé en texte brut (décapsulée) sur disque.

Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur de chiffrement ayant créé la clé, peut l'exporter. Les utilisateurs qui partagent la clé peut l'utiliser dans des opérations de chiffrement, mais ne peuvent pas l'exporter.

L'opération exSymKey copie les éléments de clé dans un fichier que vous spécifiez ; elle ne supprime pas la clé du HSM, ne modifie pas ses attributs de clés ou ne vous empêche pas de l'utiliser dans des opérations de chiffrement. Vous pouvez exporter la même clé plusieurs fois.

exSymKey exporte uniquement des clés symétriques. Pour exporter des clés publiques, utilisez exportPubKey. Pour exporter des clés privées, utilisez exportPrivateKey.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
exSymKey -h

exSymKey -k <key-to-export>
         -w <wrapping-key>
         -out <key-file>
         [-m 4]
         [-wk <unwrapping-key-file> ]
```

Exemples

Ces exemples montrent comment utiliser exSymKey pour exporter des clés symétriques qui vous appartiennent à partir de vos HSM.

Example : Exporter une clé symétrique 3DES

Cette commande permet d'exporter une clé symétrique Triple DES (3DES) (poignée de clé 7). Elle utilise une clé AES existante (handle de clé 6) dans le HSM en tant que clé d'encapsulage. Ensuite, elle écrit le texte brut de la clé 3DES dans le fichier `3DES.key`.

La sortie indique que la clé 7 (clé 3DES) a été correctement encapsulée et désencapsulée, puis écrite dans le fichier `3DES.key`.

> **Warning**
>
> Bien que la sortie indique qu'une « clé symétrique encapsulée » a été écrite dans le fichier de sortie, ce dernier contient une clé en texte brut (désencapsulée).

```
        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 : Exportation avec une clé d'encapsulage de session unique

Cet exemple montre comment utiliser une clé qui existe uniquement dans la session en tant que clé d'encapsulage. La clé à exporter étant encapsulée, immédiatement désencapsulée, et fournie sous forme de texte brut, il n'est pas nécessaire de conserver la clé d'encapsulage.

Cette série de commandes permet d'exporter une clé AES avec le handle de clé 8 à partir du HSM. Elle utilise une clé de session AES créé spécialement à cet effet.

La première commande permet genSymKeyde créer une clé AES 256 bits. Elle utilise le paramètre `-sess` pour créer une clé qui existe uniquement dans la session en cours.

La sortie montre que le HSM crée la clé 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
```

Ensuite, l'exemple vérifie que la clé à exporter (clé 8) est une clé symétrique extractible. Il vérifie également que la clé de chiffrement (clé 262168) est une clé AES qui existe uniquement dans la session. Vous pouvez utiliser la commande findKey, mais cet exemple exporte les attributs des deux clés vers des fichiers, puis utilise `grep` pour rechercher des valeurs d'attribut pertinents dans le fichier.

Ces commandes utilisent `getAttribute` avec une valeur `-a` de 512 (tous) pour obtenir tous les attributs pour les clés 8 et 262168. Pour plus d'informations sur les attributs de clé, consultez the section called “Référence des attributs de clé”.

```
getAttribute -o 8 -a 512 -out attributes/attr_8
getAttribute -o 262168 -a 512 -out attributes/attr_262168
```

Ces commandes utilisent `grep` pour vérifier les attributs de la clé à exporter (clé 8) et la clé chiffrement de session unique (clé 262168).

```
    // Verify that the key to be exported is a symmetric key.
    $  grep -A 1 "OBJ_ATTR_CLASS" attributes/attr_8
    OBJ_ATTR_CLASS
    0x04

    // Verify that the key to be exported is extractable.
    $  grep -A 1 "OBJ_ATTR_KEY_TYPE" attributes/attr_8
    OBJ_ATTR_EXTRACTABLE
    0x00000001

    // Verify that the wrapping key is an AES key
    $  grep -A 1 "OBJ_ATTR_KEY_TYPE" attributes/attr_262168
    OBJ_ATTR_KEY_TYPE
    0x1f

    // Verify that the wrapping key is a session key
    $  grep -A 1 "OBJ_ATTR_TOKEN" attributes/attr_262168
    OBJ_ATTR_TOKEN
    0x00

    // Verify that the wrapping key can be used for wrapping
     $  grep -A 1 "OBJ_ATTR_WRAP" attributes/attr_262168
    OBJ_ATTR_WRAP
    0x00000001
```

Enfin, nous utilisons une commande exSymKey pour exporter la clé 8 à l'aide de la clé de session (clé 262168) en tant que clé d'encapsulage.

Lorsque la session prend fin, la clé 262168 n'existe plus.

```
        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 : Utiliser une clé de désencapsulage externe

Cet exemple montre comment utiliser une clé de désencapsulage externe pour exporter une clé à partir du HSM.

Lorsque vous exportez une clé à partir du HSM, vous spécifiez une clé AES sur le HSM qui sera utilisée en tant que clé d'encapsulage. Par défaut, cette clé d'encapsulage est utilisée pour encapsuler et désencapsuler la clé à exporter. Toutefois, vous pouvez utiliser le paramètre `-wk` pour indiquer à exSymKey d'utiliser une clé externe dans un fichier sur disque pour le désencapsulage. Dans ce cas, la clé spécifiée par le paramètre `-w` encapsule la clé cible, et la clé contenue dans le fichier et spécifiée par le paramètre `-wk` désencapsule la clé.

La clé d'encapsulage devant être une clé AES, donc symétrique, la clé d'encapsulage sur le HSM et la clé de désencapsulage sur le disque doivent avoir les mêmes éléments de clé. Pour ce faire, vous devez importer dans le HSM ou exporter du HSM la clé d'encapsulage avant l'opération d'exportation.

Cet exemple montre comment créer une clé en dehors du HSM et l'importer dans le HSM. Il utilise la copie interne de la clé pour encapsuler une clé symétrique en train d'être exportée, et la copie de la clé dans le fichier pour la désencapsuler.

La première commande utilise OpenSSL pour générer une clé AES 256 bits. Elle enregistrer la clé dans le fichier `aes256-forImport.key`. La commande OpenSSL ne renvoie pas de sortie, mais vous pouvez utiliser plusieurs commandes pour confirmer sa réussite. Cet exemple utilise l'outil wc (wordcount), qui confirme que le fichier contient 32 octets de données.

```
$  openssl rand -out keys/aes256-forImport.key 32

$ wc keys/aes256-forImport.key
 0  2 32 keys/aes256-forImport.key
```

Cette commande utilise la imSymKeycommande pour importer la clé AES du `aes256-forImport.key` fichier vers le HSM. Lorsque la commande aboutit, la clé existe dans le HSM avec le handle de clé 262167 et dans le fichier `aes256-forImport.key`.

```
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
```

Cette commande utilise la clé dans une opération d'exportation. La commande utilise exSymKey pour exporter une clé 21, une clé AES 192 bits. Pour encapsuler la clé, elle utilise la clé 262167, qui est la copie ayant été importée dans le HSM. Pour désencapsuler la clé, elle utilise les mêmes éléments de clé dans le fichier `aes256-forImport.key`. Lorsque la commande aboutit, la clé 21 est exportée vers le fichier `aes192_h21.key`.

```
        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"
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-k

Spécifie le handle de clé de la clé à exporter. Ce paramètre est obligatoire. Entrez le handle de clé d'une clé symétrique qui vous appartient. Ce paramètre est obligatoire. Pour trouver des handles de clé, utilisez la commande findKey.

Pour vérifier qu'une clé peut être exportée, utilisez la commande getAttribute pour obtenir la valeur de l'attribut `OBJ_ATTR_EXTRACTABLE`, représentée par la constante `354`. De même, vous pouvez exporter uniquement les clés qui vous appartiennent. Pour trouver le propriétaire d'une clé, utilisez la getKeyInfocommande.

Obligatoire : oui

-s, sem

Spécifie le handle de clé de la clé d'encapsulage. Ce paramètre est obligatoire. Pour trouver des handles de clé, utilisez la commande findKey.

Une clé d'encapsulage est une clé du HSM utilisée pour chiffrer (encapsuler), puis déchiffrer (désencapsuler) la clé à exporter. Seules les clés AES peut être utilisées en tant que clés d'encapsulage.

Vous pouvez utiliser n'importe quelle clé AES (quelle que soit sa taille) comme clé d'encapsulage. Puisque les clés d'encapsulage encapsulent, puis désencapsulent immédiatement la clé cible, vous pouvez utiliser des clés AES de session unique en tant que clé d'encapsulage. Pour déterminer si une clé peut être utilisée comme clé d'encapsulage, utilisez getAttribute pour obtenir la valeur de l'attribut `OBJ_ATTR_WRAP`, représentée par la constante `262`. Pour créer une clé d'encapsulation, genSymKeyutilisez-la pour créer une clé AES (type 31).

Si vous utilisez le paramètre `-wk` pour spécifier une clé de désencapsulage externe, la clé d'encapsulage `-w` est utilisée pour encapsuler, mais pas pour désencapsuler, la clé lors de l'exportation.

> **Note**
>
> Le de clé 4 représente une clé interne non prise en charge. Nous vous recommandons d'utiliser une clé AES que vous créez et gérez comme clé d'encapsulage.

Obligatoire : oui

-out

Spécifie le chemin et le nom du fichier de sortie. Lorsque la commande aboutit, ce fichier contient la clé exportée en texte brut. Si le fichier existe déjà, la commande le remplace sans avertissement.

Obligatoire : oui

-m

Spécifie le mécanisme d'encapsulage. La seule valeur valide est `4`, qui représente le mécanisme `NIST_AES_WRAP`.

Obligatoire : non

Par défaut: 4

-wk

Utilisez la clé AES dans le fichier spécifié pour désencapsuler la clé en cours d'exportation. Entrez le chemin et le nom d'un fichier qui contient une clé AES en texte brut.

Lorsque vous incluez ce paramètre, `exSymKey` utilise la clé dans le HSM spécifiée par le paramètre `-w` pour encapsuler la clé en cours d'exportation, puis la clé spécifiée dans le fichier `-wk` pour la désencapsuler. La valeur des paramètres `-w` et `-wk` doivent correspondre à la même clé en texte brut.

Obligatoire : non

Par défaut : utilisez la clé d'encapsulage sur le HSM pour procéder au désencapsulage.

Rubriques en relation

- genSymKey
- imSymKey
- wrapKey

extractMaskedObject

La commande extractMaskedObject dans key_mgmt_util extrait une clé à partir d'un HSM et l'enregistre dans un fichier sous la forme d'un objet masqué. Les objets masqués sont des objets clonés qui ne peuvent être utilisés qu'après leur réinsertion dans le cluster d'origine à l'aide de la commande insertMaskedObject. Vous pouvez insérer un objet masqué uniquement dans le cluster à partir duquel il a été généré, ou un clone de ce cluster. Cela inclut toutes les versions clonées du cluster générées en copiant une sauvegarde entre les régions et en utilisant cette sauvegarde pour créer un nouveau cluster.

Les objets masqués sont un moyen efficace pour décharger et synchroniser les clés, y compris les clés non extractibles (c'est-à-dire, les clés qui ont une valeur `OBJ_ATTR_EXTRACTABLE` de `0`). Ainsi, les clés peuvent être synchronisées en toute sécurité entre les clusters associés dans différentes régions sans qu'il soit nécessaire de mettre à jour le fichier de AWS CloudHSM configuration.

> **Important**
>
> Lors de l'insertion, les objets masqués sont déchiffrés et obtiennent un handle de clé qui est différent du handle de clé de la clé d'origine. Un objet masqué inclut toutes les métadonnées associées à la clé d'origine, y compris les attributs, la propriété et le partage d'informations, et les paramètres de quorum. Si vous avez besoin de synchroniser des clés entre les clusters dans une application, utilisez syncKey dans cloudhsm_mgmt_util.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM. La commande extractMaskedObject peut être utilisée par l'utilisateur CU qui possède la clé ou un responsable CO.

Syntaxe

```
extractMaskedObject -h

extractMaskedObject -o <object-handle>
                    -out <object-file>
```

Exemples

Cet exemple montre comment utiliser extractMaskedObject pour extraire une clé d'un HSM en tant qu'objet masqué.

Example : Extraire un objet masqué

Cette commande extrait un objet masqué hors d'un HSM à partir d'une clé avec handle `524295` et l'enregistre sous forme d'un fichier appelé `maskedObj`. Lorsque la commande aboutit, extractMaskedObject renvoie un message de réussite.

```
Command: extractMaskedObject -o 524295 -out maskedObj

Object was masked and written to file "maskedObj"

        Cfm3ExtractMaskedObject returned: 0x00 : HSM Return: SUCCESS
```

Paramètres

Cette commande accepte les paramètres suivants :

`-h`

Affiche l'aide de la ligne de commande pour la commande.

Obligatoire : oui

`-o`

Spécifie le handle de la clé à extraire en tant qu'objet masqué.

Obligatoire : oui

`-out`

Spécifie le nom du fichier dans lequel l'objet masqué sera enregistré.

Obligatoire : oui

Rubriques en relation

- insertMaskedObject
- syncKey
- Copie de sauvegardes dans plusieurs régions
- Création d'un AWS CloudHSM cluster à partir d'une sauvegarde précédente

findKey

Utilisez la commande findKey dans key_mgmt_util pour rechercher des clés en utilisant les valeurs des attributs de clé. Lorsqu'une clé correspond à tous les critères définis, findKey renvoie le handle de clé. Sans paramètre, findKey renvoie les handles de clé pour toutes les clés que vous pouvez utiliser dans le HSM. Pour rechercher les valeurs d'attribut d'une clé particulière, utilisez getAttribute.

Comme toutes les commandes key_mgmt_util, findKey est spécifique à chaque utilisateur. Elle renvoie uniquement les clés que l'utilisateur actuel peut utiliser dans des opérations de chiffrement. Cela comprend les clés qui appartiennent à l'utilisateur actuel, ainsi que celles qu'il partage.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
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>]
```

Exemples

Ces exemples montrent comment utiliser findKey pour rechercher et identifier des clés dans vos HSM.

Example : Rechercher toutes les clés

Cette commande recherche toutes les clés pour l'utilisateur actuel dans le HSM. La sortie inclut les clés que l'utilisateur possède et partage, ainsi que toutes les clés publiques des HSM.

Pour obtenir les attributs d'une clé avec un handle de clé particulier, utilisez getAttribute. Pour déterminer si l'utilisateur actuel possède ou partage une clé particulière, utilisez getKeyInfoou findAllKeysdans cloudhs_mgmt_util.

```
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 : Rechercher des clés par type, par utilisateur et par session

Cette commande recherche les clés AES persistantes que l'utilisateur actuel et l'utilisateur 3 peuvent utiliser. (L'utilisateur 3 peut être en mesure d'utiliser d'autres clés que l'utilisateur actuel ne peut pas voir.)

```
Command: findKey -t 31 -sess 0 -u 3
```

Example : Rechercher des clés par classe et par étiquette

Cette commande recherche toutes les clés pour l'utilisateur actuel avec l'étiquette `2018-sept`.

```
Command: findKey -c 2 -l 2018-sept
```

Example : Rechercher des clés RSA par module

Cette commande recherche les clés RSA (type 0) pour l'utilisateur actuel qui ont été créées à l'aide du module dans le fichier `m4.txt`.

```
Command: findKey -t 0 -m m4.txt
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-t

Recherche les clés du type spécifié. Entrez la constante qui représente la classe de clé. Par exemple, pour rechercher les clés 3DES, tapez `-t 21`.

Valeurs valides :

- 0 : RSA
- 1 : DSA
- 3 : EC
- 16 : GENERIC_SECRET
- 18 : RC4
- 21 : Triple DES (3DES)
- 31 : AES

Obligatoire : non

-c

Recherche les clés de la classe spécifiée. Entrez la constante qui représente la classe de clé. Par exemple, pour rechercher les clés publiques, tapez `-c 2`.

Valeurs valides pour chaque type de clé :

- 2 : Publique. Cette classe contient les clés publiques des paires de clés publique-privée.
- 3 : Privée. Cette classe contient les clés privées des paires de clés publique-privée.
- 4 : Secrète. Cette classe contient toutes les clés symétriques.

Obligatoire : non

-l

Recherche les clés avec l'étiquette spécifiée. Saisissez l'étiquette exacte. Vous ne pouvez pas utiliser de caractère générique ou d'expression régulière dans la valeur `--l`.

Obligatoire : non

-id

Recherche la clé avec l'ID spécifié. Tapez la chaîne d'ID exacte. Vous ne pouvez pas utiliser de caractère générique ou d'expression régulière dans la valeur `-id`.

Obligatoire : non

-sess

Recherche les clés par statut de session. Pour trouver les clés valides uniquement dans la session en cours, tapez `1`. Pour rechercher les clés persistantes, tapez `0`.

Obligatoire : non

-u

Recherche les clés partagées par les utilisateurs spécifiés et l'utilisateur actuel. Saisissez une liste séparée par des virgules d'ID utilisateur HSM, par exemple, `-u 3` ou `-u 4,7`. Pour rechercher les ID utilisateur sur un HSM, utilisez listUsers.

Lorsque vous spécifiez un ID utilisateur, findKey renvoie les clés pour cet utilisateur. Lorsque vous spécifiez plusieurs ID utilisateur, findKey renvoie les clés que tous les utilisateurs spécifiés peuvent utiliser.

Dans la mesure où findKey renvoie uniquement les clés que l'utilisateur actuel peut utiliser, les résultats de -u correspondent toujours à tout ou partie des clés de l'utilisateur actuel. Pour obtenir toutes les clés détenues ou partagées avec n'importe quel utilisateur, les responsables de la cryptographie (CoS) peuvent les utiliser findAllKeysdans cloudhs_mgmt_util.

Obligatoire : non

-m

Recherche les clés créées à l'aide du module RSA dans le fichier spécifié. Tapez le chemin d'accès au fichier qui stocke le module.

-m spécifie le fichier binaire contenant le module RSA avec lequel correspondre (facultatif).

Obligatoire : non

-kcv

Recherche les clés avec la valeur de contrôle de clé spécifiée.

La valeur de contrôle de clé (KCV) est un hachage ou une somme de contrôle sur 3 octets d'une clé qui est générée lorsque le HSM importe ou génère une clé. Vous pouvez également calculer un KCV en dehors du HSM, par exemple après avoir exporté une clé. Vous pouvez ensuite comparer les valeurs KCV pour confirmer l'identité et l'intégrité de la clé. Pour obtenir le KCV d'une clé, utilisez getAttribute.

AWS CloudHSM utilise la méthode standard suivante pour générer une valeur de contrôle clé :

- Clés symétriques : les 3 premiers octets du résultat du chiffrement d'un bloc zéro avec la clé.
- Paires de clés asymétriques : les 3 premiers octets du hachage SHA-1 de la clé publique.
- Clés HMAC : la KCV pour les clés HMAC n'est pas prise en charge pour le moment.

Obligatoire : non

Sortie

La sortie findKey répertorie le nombre total de clés correspondantes et leurs handles de clé.

```
        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
```

Rubriques en relation

- findSingleKey
- getKeyInfo
- getAttribute
- findAllKeysdans cloudhs_mgmt_util
- Référence des attributs de clé

findSingleKey

La commande findSingleKey dans l'outil key_mgmt_util permet de vérifier qu'une clé existe sur tous les HSM du cluster.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
findSingleKey -h

findSingleKey -k <key-handle>
```

Exemple

Example

Cette commande vérifie que la clé 252136 existe sur les trois HSM du cluster.

```
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
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-k

Spécifie le handle d'une clé du module HSM. Ce paramètre est obligatoire.

Pour trouver des handles de clé, utilisez la commande findKey.

Obligatoire : oui

Rubriques en relation

- findKey
- getKeyInfo
- getAttribute

GendSA KeyPair

La commande genDSAKeyPair de l'outil key_mgmt_util génère une paire de clés Digital Signing Algorithm (DSA) dans vos HSM. Vous devez spécifier la longueur du module ; la commande génère la valeur du module. Vous pouvez également attribuer un ID, partager la clé avec d'autres utilisateurs HSM, créer des clés non extractibles, et créer des clés qui expirent lorsque la session se termine. Lorsque la commande aboutit, elle renvoie les handles de clé attribués par le HSM aux clés publiques et privées. Vous pouvez utiliser les handles de clé pour identifier les clés pour d'autres commandes.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

> **Tip**
>
> Pour rechercher les attributs d'une clé que vous avez créée, tels que le type, la longueur, l'étiquette et l'ID, utilisez getAttribute. Pour trouver les clés d'un utilisateur en particulier, utilisez getKeyInfo. Pour rechercher des clés en fonction de leurs valeurs d'attribut, utilisez findKey.

Syntaxe

```
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]
```

Exemples

Ces exemples montrent comment utiliser genDSAKeyPair pour créer une paire de clés DSA.

Example : Créer une paire de clés DSA

Cette commande crée une paire de clés DSA avec une étiquette DSA. La sortie montre que le handle de la clé publique est 19 et celui de la clé privée, 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 : Créer une paire de clés DSA de session unique

Cette commande crée une paire de clés DSA valable uniquement dans la session en cours. La commande affecte l'ID unique `DSA_temp_pair` en plus de l'étiquette requise (non unique). Vous pouvez créer une paire de clés de ce type pour signer et vérifier un jeton de session unique. La sortie montre que le handle de la clé publique est 12 et celui de la clé privée, 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
```

Pour confirmer que la paire de clés existe seulement dans la session, utilisez le paramètre `-sess` de findKey avec la valeur 1 (vrai).

```
  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 : Créer une paire de clés DSA partagée, non extractible

Cette commande crée une paire de clés DSA. La clé privée est partagé avec trois autres utilisateurs, et ne peut pas être exportée à partir du HSM. Les clés publiques peuvent être utilisées par n'importe quel utilisateur et peuvent toujours être extraites.

```
        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 : Créer une paire de clés contrôlée par quorum

Cette commande crée une paire de clés DSA avec l'étiquette `DSA-mV2`. La commande utilise le paramètre `-u` pour partager la clé privée avec les utilisateurs 4 et 6. Elle utilise le paramètre `-m_value` pour demander un quorum d'au moins deux approbations pour les opérations de chiffrement qui utilisent la clé privée. La commande utilise également le paramètre `-attest` pour vérifier l'intégrité du microprogramme sur lequel la paire de clés est générée.

La sortie montre que la commande génère une clé publique avec le handle de clé 12 et une clé privée avec le handle de clé 17, mais aussi que le contrôle d'attestation sur le microprogramme du cluster a abouti.

```
        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
```

Cette commande utilise getKeyInfola clé privée (descripteur de clé17). La sortie confirme que la clé appartient à l'utilisateur actuel (utilisateur 3) et qu'elle est partagée avec les utilisateurs 4 et 6 (et aucun autre). La sortie montre également que l'authentification du quorum est activée et que la taille du quorum est deux.

```
        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
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-m

Indique la longueur du module en bits. La seule valeur valide est `2048`.

Obligatoire : oui

-l

Spécifie l'étiquette définie par l'utilisateur pour la paire de clés. Saisissez une chaîne. La même étiquette s'applique aux deux clés de la paire. La taille maximale autorisée pour `label` est de 127 caractères.

Vous pouvez utiliser n'importe quelle phrase qui vous aide à identifier la clé. Comme il n'est pas nécessaire que l'étiquette soit unique, vous pouvez l'utiliser pour regrouper et classer des clés.

Obligatoire : oui

-id

Spécifie un identifiant défini par l'utilisateur pour la paire de clés. Saisissez une chaîne unique dans le cluster. La valeur par défaut est une chaîne vide. L'ID que vous spécifiez s'applique aux deux clés de la paire.

Valeur par défaut : pas de valeur d'ID.

Obligatoire : non

-min_srv

Spécifie le nombre minimal de HSM sur lesquels la clé est synchronisée avant que la valeur du paramètre `-timeout` n'expire. Si la clé n'est pas synchronisée sur le nombre spécifié de serveurs dans le temps imparti, elle n'est pas créée.

AWS CloudHSM synchronise automatiquement chaque clé avec chaque HSM du cluster. Pour accélérer le processus, définissez la valeur de `min_srv` sur une valeur inférieure au nombre de HSM dans le cluster et définissez une valeur de délai d'expiration peu élevée. Toutefois, notez que certaines demandes peuvent ne pas générer une clé.

Valeur par défaut : 1

Obligatoire : non

-m_value

Spécifie le nombre d'utilisateurs qui doivent approuver les opérations cryptographiques qui utilisent la clé privée dans la paire. Entrez une valeur comprise entre `0` et 8.

Ce paramètre définit une exigence d'authentification par quorum pour la clé privée. La valeur par défaut, `0`, désactive la fonctionnalité d'authentification par quorum pour la clé. Lorsque l'authentification par quorum est activée, le nombre spécifié d'utilisateurs doit signer un jeton pour approuver les opérations cryptographiques utilisant la clé privée et les opérations qui partagent ou annulent le partage de la clé privée.

Pour trouver le `m_value` code d'une clé, utilisez getKeyInfo.

Ce paramètre est valide uniquement quand le paramètre `-u` de la commande partage la paire de clés avec suffisamment d'utilisateurs pour satisfaire l'exigence `m_value`.

Par défaut : 0

Obligatoire : non

-nex

Rend la clé privée non extractible. La clé privée générée ne peut pas être exportée depuis le HSM. Les clés publiques sont toujours extractibles.

Par défaut : les clés publiques et privées de la paire de clés sont extractibles.

Obligatoire : non

-sess

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Pour remplacer une clé de session par une clé persistante (jeton), utilisez setAttribute.

Par défaut : la clé est persistante.

Obligatoire : non

-timeout

Spécifie la durée (en secondes) pendant laquelle la commande attend qu'une clé soit synchronisée avec le nombre de HSM spécifié par le paramètre `min_srv`.

Ce paramètre n'est valide que lorsque le paramètre `min_srv` est également utilisé dans la commande.

Par défaut : aucun délai d'expiration. La commande attend indéfiniment et ne renvoie des résultats que lorsque la clé est synchronisée avec le nombre minimum de serveurs.

Obligatoire : non

-u

Partage la clé privée de la paire avec les utilisateurs spécifiés. Ce paramètre autorise les autres utilisateurs de chiffrement (CU) HSM à utiliser la clé privée dans les opérations cryptographiques. Les clés publiques peuvent être utilisées par n'importe quel utilisateur sans partage.

Saisissez une liste séparée par des virgules d'ID utilisateur HSM, par exemple `-u 5,6`. N'incluez pas l'ID utilisateur HSM de l'utilisateur actuel. Pour trouver les ID utilisateur HSM des CU sur le HSM, utilisez listUsers. Pour partager ou annuler le partage d'une clé existante, utilisez shareKey dans cloudhsm_mgmt_util.

Par défaut : seul l'utilisateur actuel peut utiliser la clé privée.

Obligatoire : non

-attest

Exécute un contrôle d'intégrité qui vérifie que le microprogramme sur lequel le cluster est exécuté n'a pas été altéré.

Par défaut : aucune vérification d'attestation.

Obligatoire : non

Rubriques en relation

- Genre RSA KeyPair
- genSymKey
- GèneCC KeyPair

GèneCC KeyPair

La commande `genECCKeyPair` de l'outil key_mgmt_util génère une paire de clés Elliptic Curve Cryptography (ECC) dans vos HSM. Lorsque vous exécutez la commande `genECCKeyPair`, vous devez spécifier l'identificateur de courbe elliptique et une étiquette pour la paire de clés. Vous pouvez également partager la clé privée avec d'autres utilisateurs CU, créer des clés non extractibles, des clés contrôlées par quorum et des clés qui expirent lorsque la session se termine. Lorsque la commande aboutit, elle renvoie les handles de clé attribués par le HSM aux clés ECC publiques et privées. Vous pouvez utiliser les handles de clé pour identifier les clés pour d'autres commandes.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

> **Tip**
>
> Pour rechercher les attributs d'une clé que vous avez créée, tels que le type, la longueur, l'étiquette et l'ID, utilisez getAttribute. Pour trouver les clés d'un utilisateur en particulier, utilisez getKeyInfo. Pour rechercher des clés en fonction de leurs valeurs d'attribut, utilisez findKey.

Syntaxe

```
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]
```

Exemples

Ces exemples montrent comment utiliser genECCKeyPair pour créer des paires de clés ECC dans vos HSM.

Example : Création et examen d'une paire de clés ECC

Cette commande utilise une courbe elliptique NID_secp384r1 et une étiquette `ecc14` pour créer une paire de clés ECC. La sortie montre que le handle de la clé privée est 262177 et celui de la clé publique, 262179. La même étiquette s'applique aux deux clés.

```
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
```

Après avoir généré la clé, vous pouvez examiner ses attributs. Utilisez getAttribute pour écrire tous les attributs (représentés par la constante 512) de la nouvelle clé privée ECC sur le fichier `attr_262177`.

```
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
```

Ensuite, utilisez la commande `cat` pour afficher le contenu du fichier d'attribut `attr_262177`. La sortie indique que la clé est une clé privée de courbe elliptique qui peut être utilisée pour la signature, mais pas pour le chiffrement, le déchiffrement, l'encapsulage, le désencapsulage ou la vérification. La clé est permanente et exportable.

```
$  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 Utilisation d'une courbe ECC non valide

Cette commande tente de créer une paire de clés ECC en utilisant une courbe NID_X9_62_prime192v1. Dans la mesure où cette courbe elliptique n'est pas valide pour les HSM en mode FIPS, la commande échoue. Le message indique qu'un serveur du cluster est indisponible, mais cela n'indique généralement pas un problème avec les HSM du cluster.

```
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
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-i

Spécifie l'identifiant de la courbe elliptique. Saisissez un identifiant.

Valeurs valides :

- 2 : NID_X9_62_prime256v1
- 14 : NID_SECP384R1
- 16 : NID_secp256k1

Obligatoire : oui

-l

Spécifie l'étiquette définie par l'utilisateur pour la paire de clés. Saisissez une chaîne. La même étiquette s'applique aux deux clés de la paire. La taille maximale autorisée pour `label` est de 127 caractères.

Vous pouvez utiliser n'importe quelle phrase qui vous aide à identifier la clé. Comme il n'est pas nécessaire que l'étiquette soit unique, vous pouvez l'utiliser pour regrouper et classer des clés.

Obligatoire : oui

-id

Spécifie un identifiant défini par l'utilisateur pour la paire de clés. Saisissez une chaîne unique dans le cluster. La valeur par défaut est une chaîne vide. L'ID que vous spécifiez s'applique aux deux clés de la paire.

Valeur par défaut : pas de valeur d'ID.

Obligatoire : non

-min_srv

Spécifie le nombre minimal de HSM sur lesquels la clé est synchronisée avant que la valeur du paramètre `-timeout` n'expire. Si la clé n'est pas synchronisée sur le nombre spécifié de serveurs dans le temps imparti, elle n'est pas créée.

AWS CloudHSM synchronise automatiquement chaque clé avec chaque HSM du cluster. Pour accélérer le processus, définissez la valeur de `min_srv` sur une valeur inférieure au nombre de HSM dans le cluster et définissez une valeur de délai d'expiration peu élevée. Toutefois, notez que certaines demandes peuvent ne pas générer une clé.

Valeur par défaut : 1

Obligatoire : non

-m_value

Spécifie le nombre d'utilisateurs qui doivent approuver les opérations cryptographiques qui utilisent la clé privée dans la paire. Entrez une valeur comprise entre `0` et 8.

Ce paramètre définit une exigence d'authentification par quorum pour la clé privée. La valeur par défaut, `0`, désactive la fonctionnalité d'authentification par quorum pour la clé. Lorsque l'authentification par quorum est activée, le nombre spécifié d'utilisateurs doit signer un jeton pour

approuver les opérations cryptographiques utilisant la clé privée et les opérations qui partagent ou annulent le partage de la clé privée.

Pour trouver le `m_value` code d'une clé, utilisez getKeyInfo.

Ce paramètre est valide uniquement quand le paramètre `-u` de la commande partage la paire de clés avec suffisamment d'utilisateurs pour satisfaire l'exigence `m_value`.

Par défaut : 0

Obligatoire : non

-nex

Rend la clé privée non extractible. La clé privée générée ne peut pas être exportée depuis le HSM. Les clés publiques sont toujours extractibles.

Par défaut : les clés publiques et privées de la paire de clés sont extractibles.

Obligatoire : non

-sess

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Pour remplacer une clé de session par une clé persistante (jeton), utilisez setAttribute.

Par défaut : la clé est persistante.

Obligatoire : non

-timeout

Spécifie la durée (en secondes) pendant laquelle la commande attend qu'une clé soit synchronisée avec le nombre de HSM spécifié par le paramètre `min_srv`.

Ce paramètre n'est valide que lorsque le paramètre `min_srv` est également utilisé dans la commande.

Par défaut : aucun délai d'expiration. La commande attend indéfiniment et ne renvoie des résultats que lorsque la clé est synchronisée avec le nombre minimum de serveurs.

Obligatoire : non

-u

Partage la clé privée de la paire avec les utilisateurs spécifiés. Ce paramètre autorise les autres utilisateurs de chiffrement (CU) HSM à utiliser la clé privée dans les opérations cryptographiques. Les clés publiques peuvent être utilisées par n'importe quel utilisateur sans partage.

Saisissez une liste séparée par des virgules d'ID utilisateur HSM, par exemple `-u 5,6`. N'incluez pas l'ID utilisateur HSM de l'utilisateur actuel. Pour trouver les ID utilisateur HSM des CU sur le HSM, utilisez listUsers. Pour partager ou annuler le partage d'une clé existante, utilisez shareKey dans cloudhsm_mgmt_util.

Par défaut : seul l'utilisateur actuel peut utiliser la clé privée.

Obligatoire : non

-attest

Exécute un contrôle d'intégrité qui vérifie que le microprogramme sur lequel le cluster est exécuté n'a pas été altéré.

Par défaut : aucune vérification d'attestation.

Obligatoire : non

Rubriques en relation

- genSymKey
- Genre RSA KeyPair
- GendSA KeyPair

Genre RSA KeyPair

La commande genRSAKeyPair de l'outil key_mgmt_util génère une paire de clés asymétriques RSA. Vous spécifiez le type de clé, une longueur de module et un exposant public. La commande génère un module de la longueur spécifiée et crée la paire de clés. Vous pouvez attribuer un ID, partager

la clé avec d'autres utilisateurs HSM, créer des clés non extractibles et créer des clés qui expirent lorsque la session se termine. Lorsque la commande réussit, elle renvoie un handle de clé que le HSM attribue à la clé. Vous pouvez utiliser le handle de clé pour identifier la clé auprès d'autres commandes.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

> **Tip**
>
> Pour rechercher les attributs d'une clé que vous avez créée, tels que le type, la longueur, l'étiquette et l'ID, utilisez getAttribute. Pour trouver les clés d'un utilisateur en particulier, utilisez getKeyInfo. Pour rechercher des clés en fonction de leurs valeurs d'attribut, utilisez findKey.

Syntaxe

```
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]
```

Exemples

Ces exemples montrent comment utiliser genRSAKeyPair pour créer des paires de clés asymétriques dans vos HSM.

Example : Création et examen d'une paire de clés RSA

Cette commande crée une paire de clés RSA avec un module 2048 bits et l'exposant 65537. La sortie montre que le handle de la clé publique est `2100177` et celui de la clé privée, `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
```

La commande suivante utilise getAttribute pour obtenir les attributs de la clé publique que vous venez de créer. Elle écrit la sortie dans le fichier `attr_2100177`. Elle est suivie d'une commande cat permettant d'obtenir le contenu du fichier d'attribut. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

Les valeurs hexadécimales obtenues confirment qu'il s'agit d'une clé publique (`OBJ_ATTR_CLASS 0x02`) avec le type RSA (`OBJ_ATTR_KEY_TYPE 0x00`). Vous pouvez utiliser cette clé publique pour chiffrer (`OBJ_ATTR_ENCRYPT 0x01`), mais pas pour déchiffrer (`OBJ_ATTR_DECRYPT 0x00`). Les résultats incluent également la longueur de la clé (512, `0x200`), le module, la longueur du module (2048, `0x800`) et l'exposant public (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 : Génération d'une paire de clés RSA partagée

Cette commande génère une paire de clés RSA et partage la clé privée de l'utilisateur 4, un autre CU du HSM. Elle utilise le paramètre `m_value` pour demander au moins deux approbations avant que la clé privée de la paire ne puisse être utilisée dans une opération de chiffrement. Lorsque vous utilisez le paramètre `m_value`, vous devez également utiliser `-u` dans la commande et la valeur `m_value` ne peut pas dépasser le nombre total d'utilisateurs (nombre de valeurs dans `-u` + propriétaire).

```
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
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-m

Indique la longueur du module en bits. La valeur minimale est de 2048.

Obligatoire : oui

-e

Spécifie l' exposant public. La valeur doit être un entier impair supérieur ou égal à 65 537.

Obligatoire : oui

-l

Spécifie l'étiquette définie par l'utilisateur pour la paire de clés. Saisissez une chaîne. La même étiquette s'applique aux deux clés de la paire. La taille maximale autorisée pour `label` est de 127 caractères.

Vous pouvez utiliser n'importe quelle phrase qui vous aide à identifier la clé. Comme il n'est pas nécessaire que l'étiquette soit unique, vous pouvez l'utiliser pour regrouper et classer des clés.

Obligatoire : oui

-id

Spécifie un identifiant défini par l'utilisateur pour la paire de clés. Saisissez une chaîne unique dans le cluster. La valeur par défaut est une chaîne vide. L'ID que vous spécifiez s'applique aux deux clés de la paire.

Valeur par défaut : pas de valeur d'ID.

Obligatoire : non

-min_srv

Spécifie le nombre minimal de HSM sur lesquels la clé est synchronisée avant que la valeur du paramètre `-timeout` n'expire. Si la clé n'est pas synchronisée sur le nombre spécifié de serveurs dans le temps imparti, elle n'est pas créée.

AWS CloudHSM synchronise automatiquement chaque clé avec chaque HSM du cluster. Pour accélérer le processus, définissez la valeur de `min_srv` sur une valeur inférieure au nombre de HSM dans le cluster et définissez une valeur de délai d'expiration peu élevée. Toutefois, notez que certaines demandes peuvent ne pas générer une clé.

Valeur par défaut : 1

Obligatoire : non

-m_value

Spécifie le nombre d'utilisateurs qui doivent approuver les opérations cryptographiques qui utilisent la clé privée dans la paire. Entrez une valeur comprise entre `0` et 8.

Ce paramètre définit une exigence d'authentification par quorum pour la clé privée. La valeur par défaut, `0`, désactive la fonctionnalité d'authentification par quorum pour la clé. Lorsque l'authentification par quorum est activée, le nombre spécifié d'utilisateurs doit signer un jeton pour

approuver les opérations cryptographiques utilisant la clé privée et les opérations qui partagent ou annulent le partage de la clé privée.

Pour trouver le `m_value` code d'une clé, utilisez getKeyInfo.

Ce paramètre est valide uniquement quand le paramètre `-u` de la commande partage la paire de clés avec suffisamment d'utilisateurs pour satisfaire l'exigence `m_value`.

Par défaut : 0

Obligatoire : non

-nex

Rend la clé privée non extractible. La clé privée générée ne peut pas être exportée depuis le HSM. Les clés publiques sont toujours extractibles.

Par défaut : les clés publiques et privées de la paire de clés sont extractibles.

Obligatoire : non

-sess

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Pour remplacer une clé de session par une clé persistante (jeton), utilisez setAttribute.

Par défaut : la clé est persistante.

Obligatoire : non

-timeout

Spécifie la durée (en secondes) pendant laquelle la commande attend qu'une clé soit synchronisée avec le nombre de HSM spécifié par le paramètre `min_srv`.

Ce paramètre n'est valide que lorsque le paramètre `min_srv` est également utilisé dans la commande.

Par défaut : aucun délai d'expiration. La commande attend indéfiniment et ne renvoie des résultats que lorsque la clé est synchronisée avec le nombre minimum de serveurs.

Obligatoire : non

-u

Partage la clé privée de la paire avec les utilisateurs spécifiés. Ce paramètre autorise les autres utilisateurs de chiffrement (CU) HSM à utiliser la clé privée dans les opérations cryptographiques. Les clés publiques peuvent être utilisées par n'importe quel utilisateur sans partage.

Saisissez une liste séparée par des virgules d'ID utilisateur HSM, par exemple `-u 5,6`. N'incluez pas l'ID utilisateur HSM de l'utilisateur actuel. Pour trouver les ID utilisateur HSM des CU sur le HSM, utilisez listUsers. Pour partager ou annuler le partage d'une clé existante, utilisez shareKey dans cloudhsm_mgmt_util.

Par défaut : seul l'utilisateur actuel peut utiliser la clé privée.

Obligatoire : non

-attest

Exécute un contrôle d'intégrité qui vérifie que le microprogramme sur lequel le cluster est exécuté n'a pas été altéré.

Par défaut : aucune vérification d'attestation.

Obligatoire : non

Rubriques en relation

- genSymKey
- GendSA KeyPair
- GèneCC KeyPair

genSymKey

La commande genSymKey de l'outil key_mgmt_util génère une clé symétrique dans vos HSM. Vous pouvez spécifier le type et la taille de la clé, attribuer un ID et une étiquette, et partager la clé avec d'autres utilisateurs HSM. Vous pouvez également créer des clés non extractibles et des clés qui expirent lorsque la session se termine. Lorsque la commande réussit, elle renvoie un handle de

clé que le HSM attribue à la clé. Vous pouvez utiliser le handle de clé pour identifier la clé auprès d'autres commandes.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
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]
```

Exemples

Ces exemples montrent comment utiliser genSymKey pour créer des clés symétriques dans vos HSM.

> **Tip**
>
> Pour utiliser les clés que vous créez avec ces exemples pour les opérations HMAC, vous devez définir `OBJ_ATTR_SIGN` et `OBJ_ATTR_VERIFY` sur `TRUE` après avoir généré la clé. Pour définir ces valeurs, utilisez setAttribute dans l'Utilitaire de gestion CloudHSM (CMU). Pour de plus amples informations, veuillez consulter la commande setAttribute.

Example : Générer une clé AES

Cette commande crée une clé AES de 256 bits avec une étiquette `aes256`. La sortie indique que le handle de clé de la nouvelle clé est 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 : Créer une clé de session

Cette commande crée une clé AES de 192 bits non extractible, valide uniquement dans la session en cours. Vous pouvez créer une clé de ce type pour encapsuler (puis immédiatement désencapsuler) une clé en cours d'exportation.

```
Command: genSymKey -t 31 -s 24 -l tmpAES -id wrap01 -nex -sess
```

Example : Renvoyer rapidement

Cette commande crée une clé générique de 512 octets avec une étiquette de `IT_test_key`. La commande n'attend pas que la clé soit synchronisée sur tous les HSM dans le cluster. Au lieu de cela, elle est renvoyée dès que la clé est créée sur un HSM quelconque (`-min_srv 1`) ou après 1 seconde (`-timeout 1`), selon l'événement qui intervient le plus tôt. Si la clé n'est pas synchronisée sur le nombre minimal spécifié de HSM avant l'expiration du délai, elle n'est pas générée. Vous pouvez utiliser une commande comme celle-ci dans un script qui crée de nombreuses clés, tel que la boucle `for` dans l'exemple suivant.

```
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 : Créer une clé générique autorisée par quorum

Cette commande crée une clé secrète générique de 2 048 bits avec l'étiquette `generic-mV2`. La commande utilise le paramètre `-u` pour partager la clé avec un autre utilisateur de chiffrement, l'utilisateur 6. Elle utilise le paramètre `-m_value` pour exiger un quorum d'au moins deux approbations pour les opérations cryptographiques qui utilisent la clé. La commande utilise également le paramètre `-attest` pour vérifier l'intégrité du microprogramme sur lequel la clé est générée.

La sortie montre que la commande a généré une clé avec un handle de clé 9 et que le contrôle d'attestation sur le microprogramme du cluster a abouti.

```
                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 : Créer et examiner une clé

Cette commande crée une clé Triple DES avec une étiquette `3DES_shared` et un ID de `IT-02`. La clé peut être utilisée par l'utilisateur actuel et les utilisateurs 4 et 5. La commande échoue si l'ID n'est pas unique dans le cluster ou si l'utilisateur actuel est l'utilisateur 4 ou 5.

La sortie indique que la nouvelle clé a le handle de clé 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
```

Pour vérifier que la nouvelle clé 3DES appartient à l'utilisateur actuel et qu'elle est partagée avec les utilisateurs 4 et 5, utilisez getKeyInfo. La commande utilise le handle qui a été attribué à la nouvelle clé (`Key Handle: 7`).

La sortie confirme que la clé est détenue par l'utilisateur 3 et partagée avec les utilisateurs 4 et 5.

```
Command:  getKeyInfo -k 7
```

```
        Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

        Owned by user 3

        also, shared to following 2 user(s):

                4, 5
```

Pour confirmer les autres propriétés de la clé, utilisez getAttribute. La première commande utilise `getAttribute` pour obtenir tous les attributs (`-a 512`) du handle de clé 7 (`-o 7`). Elle les écrit dans le fichier `attr_7`. La seconde commande utilise `cat` pour obtenir le contenu du fichier `attr_7`.

Cette commande confirme que la clé 7 est une clé symétrique (`OBJ_ATTR_CLASS 0x04`) 3DES (`OBJ_ATTR_KEY_TYPE 0x15`) de 192 bits (`OBJ_ATTR_VALUE_LEN 0x00000018` ou 24 octets) avec l'étiquette `3DES_shared` (`OBJ_ATTR_LABEL 3DES_shared`) et l'ID `IT_02` (`OBJ_ATTR_ID IT-02`). La clé est persistante (`OBJ_ATTR_TOKEN 0x01`) et extractible (`OBJ_ATTR_EXTRACTABLE 0x01`), et elle peut être utilisée pour le chiffrement, le déchiffrement et l'encapsulage.

> **Tip**
>
> Pour rechercher les attributs d'une clé que vous avez créée, tels que le type, la longueur, l'étiquette et l'ID, utilisez getAttribute. Pour trouver les clés d'un utilisateur en particulier, utilisez getKeyInfo. Pour rechercher des clés en fonction de leurs valeurs d'attribut, utilisez findKey.

Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

```
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

Pour utiliser les clés que vous créez avec ces exemples pour les opérations HMAC, vous devez définir `OBJ_ATTR_SIGN` et `OBJ_ATTR_VERIFY` sur `TRUE` après avoir généré la clé. Pour définir ces valeurs, utilisez setAttribute dans l'utilitaire CMU. Pour de plus amples informations, consultez setAttribute .

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-t

Spécifie le type de la clé symétrique. Entrez la constante qui représente le type de clé. Par exemple, pour créer une clé AES, tapez `-t 31`.

Valeurs valides :

- 16 : GENERIC_SECRET. Une clé secrète générique est un tableau d'octets qui n'est pas conforme à une norme particulière, telle que les exigences pour une clé AES.
- 18 : RC4. Les clés RC4 ne sont pas valides sur les HSM en mode FIPS.
- 21 : Triple DES (3DES). Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.
- 31 : AES

Obligatoire : oui

-s

Spécifie la taille de la clé en octets. Par exemple, pour créer une clé de 192 bits, tapez `24`.

Valeurs valides pour chaque type de clé :

- AES : 16 (128 bits), 24 (192 bits), 32 (256 bits)
- 3DES : 24 (192 bits)
- Secret générique : <3584 (28672 bits)

Obligatoire : oui

-l

Spécifie l'étiquette définie par l'utilisateur pour la clé. Saisissez une chaîne.

Vous pouvez utiliser n'importe quelle phrase qui vous aide à identifier la clé. Comme il n'est pas nécessaire que l'étiquette soit unique, vous pouvez l'utiliser pour regrouper et classer des clés.

Obligatoire : oui

-attest

Exécute un contrôle d'intégrité qui vérifie que le microprogramme sur lequel le cluster est exécuté n'a pas été altéré.

Par défaut : aucune vérification d'attestation.

Obligatoire : non

-id

Spécifie un identifiant défini par l'utilisateur pour la clé. Saisissez une chaîne unique dans le cluster. La valeur par défaut est une chaîne vide.

Valeur par défaut : pas de valeur d'ID.

Obligatoire : non

-min_srv

Spécifie le nombre minimal de HSM sur lesquels la clé est synchronisée avant que la valeur du paramètre `-timeout` n'expire. Si la clé n'est pas synchronisée sur le nombre spécifié de serveurs dans le temps imparti, elle n'est pas créée.

AWS CloudHSM synchronise automatiquement chaque clé avec chaque HSM du cluster. Pour accélérer le processus, définissez la valeur de `min_srv` sur une valeur inférieure au nombre de HSM dans le cluster et définissez une valeur de délai d'expiration peu élevée. Toutefois, notez que certaines demandes peuvent ne pas générer une clé.

Valeur par défaut : 1

Obligatoire : non

-m_value

Spécifie le nombre d'utilisateurs qui doivent approuver les opérations cryptographiques qui utilisent la clé. Entrez une valeur comprise entre `0` et 8.

Ce paramètre définit une exigence d'authentification par quorum pour la clé. La valeur par défaut, `0`, désactive la fonctionnalité d'authentification par quorum pour la clé. Lorsque l'authentification par quorum est activée, le nombre d'utilisateurs spécifié doivent signer un jeton pour approuver

les opérations de chiffrement qui utilisent la clé, et les opérations qui partagent la clé ou annulent son partage.

Pour trouver le `m_value` code d'une clé, utilisez getKeyInfo.

Ce paramètre est valide uniquement quand le paramètre `-u` de la commande partage la clé avec suffisamment d'utilisateurs pour satisfaire l'exigence `m_value`.

Par défaut : 0

Obligatoire : non

-nex

Rend la clé non extractible. La clé générée ne peut pas être exportée depuis le HSM.

Par défaut : la clé est extractible.

Obligatoire : non

-sess

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Pour remplacer une clé de session par une clé persistante (jeton), utilisez setAttribute.

Par défaut : la clé est persistante.

Obligatoire : non

-timeout

Spécifie la durée (en secondes) pendant laquelle la commande attend qu'une clé soit synchronisée avec le nombre de HSM spécifié par le paramètre `min_srv`.

Ce paramètre n'est valide que lorsque le paramètre `min_srv` est également utilisé dans la commande.

Par défaut : aucun délai d'expiration. La commande attend indéfiniment et ne renvoie des résultats que lorsque la clé est synchronisée avec le nombre minimum de serveurs.

Obligatoire : non

-u

Pour partager la clé avec les utilisateurs spécifiés. Ce paramètre autorise les autres utilisateurs de chiffrement (CU) HSM à utiliser cette clé dans les opérations cryptographiques.

Saisissez une liste séparée par des virgules d'ID utilisateur HSM, par exemple `-u 5,6`. N'incluez pas l'ID utilisateur HSM de l'utilisateur actuel. Pour trouver les ID utilisateur HSM des CU sur le HSM, utilisez listUsers. Pour partager ou annuler le partage d'une clé existante, utilisez shareKey dans cloudhsm_mgmt_util.

Par défaut : seul l'utilisateur actuel peut utiliser la clé.

Obligatoire : non

Rubriques en relation

- exSymKey
- Genre RSA KeyPair
- GendSA KeyPair
- GèneCC KeyPair
- setAttribute

getAttribute

La getAttribute commande contenue dans key_mgmt_util écrit une ou toutes les valeurs d'attribut d'une clé dans un fichier. AWS CloudHSM Si l'attribut que vous spécifiez n'existe pas pour le type de clé, tel que le module d'une clé AES, getAttribute renvoie une erreur.

Les attributs de clé sont les propriétés d'une clé. Ils incluent des caractéristiques telles que le type de clé, la classe, l'étiquette et l'ID, ainsi que les valeurs qui représentent les actions que vous pouvez exécuter avec la clé, comme le chiffrement, le déchiffrement, l'encapsulation, la signature et la vérification.

Vous ne pouvez utiliser getAttribute que sur les clés que vous possédez et les clés que vous partagez. Vous pouvez exécuter cette commande ou la commande getAttribute de key_mgmt_util

pour obtenir une valeur d'attribut d'une clé à partir de tous les HSM d'un cluster et l'écrire dans stdout ou dans un fichier.

Pour obtenir une liste d'attributs et les constantes qui les représentent, utilisez la commande listAttributes. Pour modifier les valeurs d'attribut des clés existantes, utilisez setAttribute dans key_mgmt_util et setAttribute dans cloudhsm_mgmt_util. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
getAttribute -h

getAttribute -o <key handle>
             -a <attribute constant>
             -out <file>
```

Exemples

Ces exemples illustrent comment utiliser getAttribute pour obtenir les attributs de clés dans vos HSM.

Example : Obtenir le type de clé

Cet exemple obtient le type de la clé, par exemple une clé AES, 3DES ou générique, ou une paire de clés RSA ou de courbe elliptique.

La première commande exécute listAttributes, qui permet d'obtenir les attributs d'une clé et les constantes qui les représentent. La sortie indique que la constante pour le type de clé est 256. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

```
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
```

La deuxième commande exécute getAttribute. Elle demande le type de clé (attribut 256) pour le handle de clé 524296 et l'écrit dans le fichier `attribute.txt`.

```
Command: getAttribute -o 524296 -a 256 -out attribute.txt
Attributes dumped into attribute.txt file
```

La commande finale permet d'obtenir le contenu du fichier de clé. La sortie révèle que le type de clé est 0x15 ou 21, ce qui correspond à une clé Triple DES (3DES). Pour obtenir les définitions des valeurs de classe et de type, consultez Référence des attributs de clé.

```
$  cat attribute.txt
OBJ_ATTR_KEY_TYPE
0x00000015
```

Example : Obtenir tous les attributs d'une clé

Cette commande permet d'obtenir tous les attributs de la clé avec le handle de clé 6 et les écrit dans le fichier `attr_6`. Elle utilise la valeur d'attribut 512, qui représente tous les attributs.

```
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>
```

Cette commande indique le contenu d'un exemple de fichier d'attribut avec toutes les valeurs d'attribut. Parmi les valeurs, elle signale que la clé est une clé AES de 256 bits avec pour ID `test_01` et pour étiquette `aes256`. La clé est extractible et persistante, et ne correspond donc pas à une clé de simple session. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

```
$  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
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-o

Spécifie le handle de clé de la clé cible. Vous pouvez spécifier une seule clé dans chaque commande. Pour obtenir le handle d'une clé, utilisez findKey.

Vous devez également être propriétaire de la clé spécifiée ou elle doit être partagée avec vous. Pour rechercher les utilisateurs d'une clé, utilisez getKeyInfo.

Obligatoire : oui

-a

Identifie l'attribut. Entrez une constante qui représente un attribut, ou `512`, qui représente tous les attributs. Par exemple, pour obtenir le type de clé, tapez `256`, qui correspond à la constante de l'attribut `OBJ_ATTR_KEY_TYPE`.

Pour obtenir la liste des attributs et de leurs constantes, utilisez listAttributes. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

Obligatoire : oui

-out

Écrit la sortie sur le fichier spécifié. Entrez un chemin de fichier. Vous ne pouvez pas écrire la sortie dans `stdout`.

Si le fichier spécifié existe, getAttribute remplace le fichier sans avertissement.

Obligatoire : oui

Rubriques en relation

- getAttribute dans cloudhsm_mgmt_util
- listAttributes

- setAttribute
- findKey
- Référence des attributs de clé

getCaviumPrivClé

Le getCaviumPrivraccourci clavier de key_mgmt_util exporte une clé privée depuis un HSM au faux format PEM. Le fichier PEM factice, qui au lieu de contenir la clé privée réelle référence la clé privée dans le HSM, peut ensuite être utilisé pour établir le déchargement SSL/TLS à partir de votre serveur web vers AWS CloudHSM. Pour plus d'informations, consultez la section Déchargement SSL/TLS sur Linux.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
getCaviumPrivKey -h

getCaviumPrivKey -k <private-key-handle
                 -out <fake-PEM-file>
```

Exemples

Cet exemple montre comment utiliser getCaviumPrivKey pour exporter une clé privée au format PEM factice.

Example : Exporter un fichier factice PEM

Cette commande crée et exporte une version factice PEM d'une clé privée avec handle 15 et l'enregistre dans un fichier appelé `cavKey.pem`. Lorsque la commande aboutit, exportPrivateKey renvoie un message de réussite.

```
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
```

Paramètres

Cette commande accepte les paramètres suivants :

`-h`

Affiche l'aide de la ligne de commande pour la commande.

Obligatoire : oui

`-k`

Spécifie le handle de clé de la clé privée à exporter au format PEM factice.

Obligatoire : oui

`-out`

Spécifie le nom du fichier dans lequel la clé PEM factice sera écrite.

Obligatoire : oui

Rubriques en relation

- importPrivateKey
- Déchargement SSL/TLS sur Linux

getCert

La commande getCert dans key_mgmt_util extrait les certificats de partition d'un HSM et les enregistre dans un fichier. Lorsque vous exécutez la commande, vous désignez le type de certificat à extraire. Pour ce faire, vous devez utiliser l'un des nombres entiers correspondants, comme décrit dans la section Paramètres ci-après. Pour plus d'informations sur le rôle de chacun de ces certificats, consultez Vérification de l'identité du HSM.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
getCert -h

getCert -f <file-name>
```

```
        -t <certificate-type>
```

Exemple

Cet exemple montre comment utiliser getCert pour extraire le certificat racine du client d'un cluster et l'enregistrer en tant que fichier.

Example : Récupérer un certificat racine du client

Cette commande exporte un certificat racine du client (représenté par le nombre entier 4) et l'enregistre dans un fichier appelé `userRoot.crt`. Lorsque la commande aboutit, getCert renvoie un message de réussite.

```
Command: getCert -f userRoot.crt -s 4

Cfm3GetCert() returned 0 :HSM Return: SUCCESS
```

Paramètres

Cette commande accepte les paramètres suivants :

`-h`

Affiche l'aide de la ligne de commande pour la commande.

Obligatoire : oui

`-f`

Spécifie le nom du fichier dans lequel le certificat extrait sera enregistré.

Obligatoire : oui

-s

Nombre entier qui spécifie le type de certificat de partition à extraire. Les nombres entiers et leurs types de certificats correspondants sont les suivants :

- 1 : Certificat racine du fabricant
- 2 : Certificat du fabricant du matériel
- 4 : Certificat racine du client
- 8 : Certificat du cluster (signé par le certificat racine du client)
- 16 : Certificat du cluster (chaîné au certificat racine du fabricant)

Obligatoire : oui

Rubriques en relation

- Vérification de l'identité du HSM
- getCert (dans cloudhsm_mgmt_util)

getKeyInfo

La commande getKeyInfo de key_mgmt_util renvoie les ID utilisateur HSM des utilisateurs qui peuvent utiliser la clé, y compris le propriétaire et les utilisateurs de chiffrement (CU) avec lesquelles la clé est partagée. Lorsque l'authentification par quorum est activée sur une clé, getKeyInfo renvoie également le nombre d'utilisateurs qui doivent approuver les opérations de chiffrement qui utilisent la clé. Vous ne pouvez exécuter getKeyInfo que sur les clés que vous possédez et sur les clés que vous partagez.

Lorsque vous exécutez getKeyInfo sur les clés publiques, getKeyInfo renvoie uniquement le propriétaire de la clé, même si tous les utilisateurs du HSM peuvent utiliser la clé publique. Pour rechercher les ID d'utilisateur HSM de tous les utilisateurs de vos HSM, utilisez listUsers. Pour trouver les clés pour un utilisateur particulier, utilisez findKey `-u`.

Vous possédez les clés que vous créez. Vous pouvez partager une clé avec d'autres utilisateurs lorsque vous la créez. Ensuite, pour partager ou annuler le partage d'une clé existante, utilisez shareKey dans cloudhsm_mgmt_util.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
getKeyInfo -h

getKeyInfo -k <key-handle>
```

Exemples

Ces exemples montrent comment utiliser getKeyInfo pour obtenir des informations sur les utilisateurs d'une clé.

Example : Obtenez les utilisateurs pour une clé symétrique

Cette commande obtient les utilisateurs qui peuvent utiliser la clé AES (symétrique) avec le handle de clé 9. La sortie indique que l'utilisateur 3 possède la clé et l'a partagée avec l'utilisateur 4.

```
Command:  getKeyInfo -k 9

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 1 user(s):

                4
```

Example : Obtenir les utilisateurs pour une paire de clés asymétrique

Ces commandes utilisent getKeyInfo pour obtenir les utilisateurs qui peuvent utiliser les clés d'une paire de clés RSA (asymétrique). La clé publique dispose du handle de clé 21. La clé privée dispose du handle de clé 20.

Lorsque vous exécutez getKeyInfo sur la clé privée (20), cela renvoie le propriétaire de la clé (3) et les utilisateurs de chiffrement (CU) 4 et 5 avec lesquels la clé est partagée.

```
Command:  getKeyInfo -k 20

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS

       Owned by user 3

       also, shared to following 2 user(s):

                4
                5
```

Lorsque vous exécutez getKeyInfo sur la clé publique (21), cela renvoie uniquement le propriétaire de la clé (3).

```
Command:  getKeyInfo -k 21

       Cfm3GetKey returned: 0x00 : HSM Return: SUCCESS
```

```
        Owned by user 3
```

Pour confirmer que l'utilisateur 4 peut utiliser la clé publique (et toutes les clés publiques sur le HSM), utilisez le paramètre `-u` de findKey.

La sortie indique que l'utilisateur 4 peut utiliser la clé publique (21) et la clé privée (20) dans la paire de clés. L'utilisateur 4 peut également utiliser toutes les autres clés publiques et clés privées qu'il a créées ou qu'il a partagées.

```
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 : Obtention de la valeur d'authentification par quorum (m_value) pour une clé

Cet exemple montre comment obtenir la valeur `m_value` pour une clé, c'est-à-dire le nombre d'utilisateurs dans le quorum qui doivent approuver toutes les opérations cryptographiques qui utilisent la clé.

Lorsque l'authentification par quorum est activée sur une clé, un quorum d'utilisateurs doit approuver toutes les opérations de chiffrement qui utilisent la clé. Pour activer l'authentification par quorum et définir la taille de quorum, utilisez le paramètre `-m_value` lorsque vous créez la clé.

Cette commande utilise GenRSA KeyPair pour créer une paire de clés RSA partagée avec l'utilisateur 4. Elle utilise le paramètre `m_value` pour activer l'authentification par quorum sur la clé privée de la paire et définir la taille de quorum sur deux utilisateurs. Le nombre d'utilisateurs doit être suffisamment grand pour fournir les approbations requises.

La sortie indique que la commande a créé la clé publique 27 et la clé privée 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
```

Cette commande utilise getKeyInfo pour obtenir des informations sur les utilisateurs de la clé privée. La sortie indique que la clé est détenue par l'utilisateur 3 et partagée avec l'utilisateur 4. Elle indique également qu'un quorum de deux utilisateurs doit approuver chaque opération de chiffrement qui utilise la clé.

```
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
```

Paramètres

-h

Affiche l'aide de la ligne de commande pour la commande.

Obligatoire : oui

-k

Spécifie le handle d'une clé du module HSM. Entrez le handle de clé d'une clé qui vous appartient ou que vous partagez. Ce paramètre est obligatoire.

Pour trouver des handles de clé, utilisez la commande findKey.

Obligatoire : oui

Rubriques en relation

- getKeyInfodans cloudhs_mgmt_util
- listUsers

- findKey
- findAllKeysdans cloudhs_mgmt_util

aide

La commande help dans key_mgmt_util affiche les informations sur toutes les commandes key_mgmt_util disponibles.

Avant d'exécuter help, vous devez démarrer key_mgmt_util.

Syntaxe

```
help
```

Exemple

Cet exemple illustre la sortie de la commande `help`.

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
```

Paramètres

Il n'existe aucun paramètre pour cette commande.

Rubriques en relation

- loginHSM et logoutHSM

importPrivateKey

La commande importPrivateKey contenue dans key_mgmt_util importe une clé privée asymétrique depuis un fichier vers un HSM. Le HSM n'autorise pas l'importation directe de clés en texte clair. La commande chiffre la clé privée à l'aide d'une clé d'encapsulage AES que vous spécifiez et désencapsule la clé dans le HSM. Si vous essayez d'associer une AWS CloudHSM clé à un certificat, consultez cette rubrique.

> **Note**
>
> Vous ne pouvez pas importer une clé PEM protégée par mot de passe à l'aide d'une clé symétrique ou privée.

Vous devez spécifier une clé d'encapsulage AES dotée d'attribut `OBJ_ATTR_UNWRAP` et `OBJ_ATTR_ENCRYPT` d'une valeur `1`. Pour obtenir les attributs d'une clé, utilisez la commande getAttribute.

> **Note**
>
> Cette commande n'offre pas la possibilité de marquer la clé importée comme non exportable.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
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]
```

Exemples

Cet exemple montre comment utiliser importPrivateKey pour importer une clé privée dans un HSM.

Example : Importer une clé privée

Cette commande importe la clé privée depuis un fichier nommé `rsa2048.key` avec l'étiquette `rsa2048-imported` et une clé d'encapsulage avec handle `524299`. Lorsque la commande réussit, importPrivateKey renvoie un handle de clé pour la clé importée et un message de réussite.

```
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
```

Paramètres

Cette commande accepte les paramètres suivants :

`-h`

Affiche l'aide de la ligne de commande pour la commande.

Obligatoire : oui

`-l`

Spécifie l'étiquette définie par l'utilisateur pour la clé privée.

Obligatoire : oui

`-f`

Spécifie le nom de fichier de la clé à importer.

Obligatoire : oui

`-w`

Spécifie le handle de clé de la clé d'encapsulage. Ce paramètre est obligatoire. Pour trouver des handles de clé, utilisez la commande findKey.

Pour déterminer si une clé peut être utilisée comme clé d'encapsulage, utilisez getAttribute pour obtenir la valeur de l'attribut `OBJ_ATTR_WRAP` (262). Pour créer une clé d'encapsulage, utilisez genSymKey pour créer une clé AES (type 31).

Si vous utilisez le paramètre `-wk` pour spécifier une clé de désencapsulage externe, la clé d'encapsulage `-w` est utilisée pour encapsuler la clé lors de l'import, mais pas pour la désencapsuler.

Obligatoire : oui

`-sess`

Spécifie la clé importée en tant que clé de session.

Par défaut, la clé importée est conservée en tant que clé persistante (jeton) dans le cluster.

Obligatoire : non

`-id`

Spécifie l'ID de la clé à importer.

Valeur par défaut : pas de valeur d'ID.

Obligatoire : non

`-m_value`

Spécifie le nombre d'utilisateurs qui doivent approuver les opérations cryptographiques qui utilisent la clé importée. Saisissez une valeur comprise entre **`0`** et **`8`**.

Ce paramètre est valide uniquement quand le paramètre `-u` de la commande partage la clé avec suffisamment d'utilisateurs pour satisfaire l'exigence `m_value`.

Par défaut : 0

Obligatoire : non

`-min_srv`

Spécifie le nombre minimal de HSM sur lesquels la clé importée est synchronisée avant que la valeur du paramètre `-timeout` n'expire. Si la clé n'est pas synchronisée sur le nombre spécifié de serveurs dans le temps imparti, elle n'est pas créée.

AWS CloudHSM synchronise automatiquement chaque clé avec chaque HSM du cluster. Pour accélérer le processus, définissez la valeur de `min_srv` sur une valeur inférieure au nombre de HSM dans le cluster et définissez une valeur de délai d'expiration peu élevée. Toutefois, notez que certaines demandes peuvent ne pas générer une clé.

Valeur par défaut : 1

Obligatoire : non

`-timout`

Spécifie le délai d'attente (en secondes) pour la synchronisation de la clé sur les HSM lorsque le paramètre `min-serv` est inclus. Si aucun nombre n'est spécifié, l'interrogation continue indéfiniment.

Par défaut : pas de limite

Obligatoire : non

`-u`

Spécifie la liste des utilisateurs avec lesquels partager la clé privée importée. Ce paramètre autorise les autres utilisateurs de chiffrement (CU) du HSM à utiliser la clé importée pour les opérations cryptographiques.

Saisissez une liste d'ID utilisateur HSM séparés par des virgules, par exemple, `-u 5,6`. N'incluez pas l'ID utilisateur HSM de l'utilisateur actuel. Pour trouver les ID utilisateur HSM des CU sur le HSM, utilisez listUsers.

Par défaut, seul l'utilisateur actuel peut utiliser la clé importée.

Obligatoire : non

`-wk`

Spécifie la clé à utiliser pour encapsuler la clé en cours d'importation. Entrez le chemin et le nom d'un fichier qui contient une clé AES en texte brut.

Lorsque vous incluez ce paramètre, importPrivateKey utilise la clé dans le fichier `-wk` pour encapsuler la clé en cours d'importation. Il utilise également la clé spécifiée par le paramètre `-w` pour la désencapsuler.

Par défaut : Utilise la clé d'encapsulage spécifiée dans le paramètre `-w` pour encapsuler et désencapsuler.

Obligatoire : non

`-attest`

Effectue un contrôle d'attestation sur la réponse de microprogramme pour s'assurer que le microprogramme sur lequel le cluster s'exécute n'a pas été compromis.

Obligatoire : non

Rubriques en relation

- wrapKey
- unWrapKey
- genSymKey
- exportPrivateKey

importPubKey

La commande importPubKey dans key_mgmt_util importe une clé publique au format PEM dans un HSM. Vous pouvez l'utiliser pour importer des clés publiques générées hors du HSM. Vous pouvez également utiliser la commande pour importer des clés exportées depuis un HSM, telles que celles exportées par la exportPubKeycommande.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
importPubKey -h

importPubKey -l <label>
             -f <key-file>
             [-sess]
             [-id <key-id>]
             [min_srv <minimum-number-of-servers>]
             [-timeout <number-of-seconds>]
```

Exemples

Cet exemple montre comment utiliser importPubKey pour importer une clé publique dans un HSM.

Example : Importer une clé publique

Cette commande importe une clé publique depuis un fichier nommé `public.pem` avec l'étiquette `importedPublicKey`. Lorsque la commande réussit, importPubKey renvoie un handle de clé pour la clé importée et un message de réussite.

```
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
```

Paramètres

Cette commande accepte les paramètres suivants :

-h

Affiche l'aide de la ligne de commande pour la commande.

Obligatoire : oui

-l

Spécifie une étiquette définie par l'utilisateur pour la clé publique.

Obligatoire : oui

-f

Spécifie le nom de fichier de la clé à importer.

Obligatoire : oui

-sess

Désigne la clé importée en tant que clé de session.

Par défaut, la clé importée est conservée en tant que clé persistante (jeton) dans le cluster.

Obligatoire : non

-id

Spécifie l'ID de la clé à importer.

Valeur par défaut : pas de valeur d'ID.

Obligatoire : non

`-min_srv`

Spécifie le nombre minimal de HSM sur lesquels la clé importée est synchronisée avant que la valeur du paramètre `-timeout` n'expire. Si la clé n'est pas synchronisée sur le nombre spécifié de serveurs dans le temps imparti, elle n'est pas créée.

AWS CloudHSM synchronise automatiquement chaque clé avec chaque HSM du cluster. Pour accélérer le processus, définissez la valeur de `min_srv` sur une valeur inférieure au nombre de HSM dans le cluster et définissez une valeur de délai d'expiration peu élevée. Toutefois, notez que certaines demandes peuvent ne pas générer une clé.

Valeur par défaut : 1

Obligatoire : non

`-timeout`

Spécifie le délai d'attente (en secondes) pour la synchronisation de la clé sur les HSM lorsque le paramètre `min-serv` est inclus. Si aucun nombre n'est spécifié, l'interrogation continue indéfiniment.

Par défaut : pas de limite

Obligatoire : non

Rubriques en relation

- exportPubKey
- Générez des clés

imSymKey

La commande imSymKey de l'outil key_mgmt_util importe la copie en texte brut d'une clé symétrique depuis un fichier vers le HSM. Vous pouvez l'utiliser pour importer des clés générées par n'importe quelle méthode en dehors du HSM et des clés exportées depuis un HSM, telles que les clés que la exSymKeycommande écrit dans un fichier.

Pendant le processus d'importation, imSymKey utilise une clé AES que vous sélectionnez (la clé d'encapsulage) pour encapsuler (chiffrer), puis désencapsuler (déchiffrer) la clé à importer. Cependant, imSymKey fonctionne uniquement sur les fichiers qui contiennent des clés en texte brut. Pour exporter et importer des clés chiffrées, utilisez le WrapKey et unWrapKeyles commandes.

De plus, la commande imSymKey importe uniquement les clés symétriques. Pour importer les clés publiques, utilisez importPubKey. Pour importer des clés privées, utilisez importPrivateKeyou WrapKey.

> **Note**
>
> Vous ne pouvez pas importer une clé PEM protégée par mot de passe à l'aide d'une clé symétrique ou privée.

Le fonctionnement des clés importées est très similaire à celui des clés générées dans le HSM. Toutefois, la valeur de l'attribut OBJ_ATTR_LOCAL est zéro, ce qui indique qu'il n'a pas été généré en local. Vous pouvez utiliser la commande suivante pour partager une clé symétrique que vous importez. Vous pouvez utiliser la commande `shareKey` dans cloudhsm_mgmt_util pour partager la clé après qu'elle a été importée.

```
imSymKey -l aesShared -t 31 -f kms.key -w 3296 -u 5
```

Après avoir importé une clé, n'oubliez pas de marquer ou de supprimer le fichier de clé. Cette commande ne vous empêche pas d'importer plusieurs fois les mêmes éléments de clés. Le résultat, plusieurs clés avec des handles de clé distincts et les mêmes éléments de clé, complique le suivi de l'utilisation des éléments de clé et évite un dépassement des limites de chiffrement.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
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>]
```

Exemples

Ces exemples montrent comment utiliser imSymKey pour importer des clés symétriques dans vos HSM.

Example : Importation d'une clé symétrique AES

Cet exemple utilise imSymKey pour importer une clé symétrique AES dans les HSM.

La première commande utilise OpenSSL pour générer une clé symétrique AES 256 bits aléatoire. Elle enregistre la clé dans le fichier `aes256.key`.

```
$  openssl rand -out aes256-forImport.key 32
```

La deuxième commande utilise imSymKey pour importer la clé AES depuis le fichier `aes256.key` vers les HSM. Elle utilise la clé 20, clé AES du HSM, comme clé d'encapsulage et spécifie l'étiquette `imported`. Contrairement à l'ID, l'étiquette n'a pas besoin d'être unique dans le cluster. La valeur du paramètre `-t` (type) est `31`, qui correspond à AES.

La sortie indique que la clé du fichier a été encapsulée et désencapsulée, puis importée dans le HSM, où le handle de clé 262180 lui a été attribué.

```
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
```

La commande suivante utilise getAttribute pour obtenir l'attribut OBJ_ATTR_LOCAL (attribut 355) de la clé nouvellement importée et l'écrit dans le fichier `attr_262180`.

```
Command:  getAttribute -o 262180 -a 355 -out attributes/attr_262180
Attributes dumped into attributes/attr_262180_imported file

        Cfm3GetAttribute returned: 0x00 : HSM Return: SUCCESS
```

Lorsque vous examinez le fichier d'attribut, vous pouvez voir que l'attribut `OBJ_ATTR_LOCAL` a pour valeur zéro, ce qui indique que les clés n'ont pas été générées dans le HSM.

```
$  cat attributes/attr_262180_local
OBJ_ATTR_LOCAL
0x00000000
```

Example : Déplacement d'une clé symétrique entre les clusters

Cet exemple montre comment utiliser exSymKeyet imSymKey déplacer une clé AES en texte brut entre des clusters. Vous pouvez utiliser un processus comme celui-ci pour créer un encapsulage AES qui existe sur les HSM des deux clusters. Une fois que la clé d'encapsulation partagée est en place, vous pouvez utiliser WrapKey unWrapKeypour déplacer les clés chiffrées entre les clusters.

L'utilisateur CU qui effectue cette opération doit avoir l'autorisation de se connecter aux HSM des deux clusters.

La première commande permet d'exSymKeyexporter la clé 14, une clé AES 32 bits, du cluster 1 vers le `aes.key` fichier. Elle utilise la clé 6, clé AES des HSM du cluster 1, en tant que clé d'encapsulage.

```
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"
```

L'utilisateur se connecte ensuite à key_mgmt_util du cluster 2 et exécute une commande imSymKey pour importer la clé depuis le fichier `aes.key` vers les HSM du cluster 2. La commande utilise la clé 252152, clé AES des HSM du cluster 2, comme clé d'encapsulage.

Étant donné que les clés d'encapsulation qui exSymKeypeuvent être imSymKey utilisées encapsulent et désencapsulent immédiatement les clés cibles, les clés d'encapsulation des différents clusters ne doivent pas nécessairement être les mêmes.

La sortie montre que la clé a été importée avec succès dans le cluster 2 et que le handle de clé 21 lui a été attribué.

```
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
```

Pour prouver que la clé 14 du cluster 1 et la clé 21 du cluster 2 ont les mêmes éléments de clé, recherchez la valeur de contrôle de clé (KCV) de chaque clé. Si les valeurs KCV sont identiques, les éléments de clé sont les mêmes.

La commande suivante utilise getAttribute du cluster 1 pour écrire la valeur de l'attribut KCV (attribut 371) de la clé 14 dans le fichier `attr_14_kcv`. Elle utilise ensuite une commande cat pour obtenir le contenu du fichier `attr_14_kcv`.

```
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
```

Cette commande similaire utilise getAttribute du cluster 2 pour écrire la valeur de l'attribut KCV (attribut 371) de la clé 21 dans le fichier `attr_21_kcv`. Elle utilise ensuite une commande cat pour obtenir le contenu du fichier `attr_21_kcv`.

```
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
```

La sortie montre que les valeurs KCV des deux clés sont identiques, ce qui prouve que les éléments de clé sont les mêmes.

Dans la mesure où les mêmes éléments de clé existent dans les HSM des deux clusters, vous pouvez désormais partager les clés chiffrées entre les clusters sans jamais exposer la clé en texte brut. Par exemple, vous pouvez utiliser la commande `wrapKey` avec la clé d'encapsulage 14 pour exporter une clé chiffrée depuis le cluster 1, puis utiliser `unWrapKey` avec la clé d'encapsulage 21 pour importer la clé chiffrée dans le cluster 2.

Example : Importation d'une clé de session

Cette commande utilise les paramètres `-sess` de imSymKey pour importer une clé DES triple 192 bits, valable uniquement pendant la session en cours.

La commande utilise le paramètre `-f` pour spécifier le fichier contenant la clé à importer, le paramètre `-t` pour spécifier le type de clé et le paramètre `-w` pour spécifier la clé d'encapsulage. Elle utilise le paramètre `-l` pour spécifier une étiquette qui catégorise la clé et le paramètre `-id` pour créer un identifiant descriptif, mais unique, de la clé. Elle utilise également le paramètre `-attest` pour vérifier le microprogramme qui importe la clé.

La sortie montre que la clé a été encapsulée et désencapsulée avec succès, puis importée dans le HSM ; enfin, le handle de clé 37 lui a été attribué. De plus, le contrôle d'attestation a été passé avec succès, ce qui indique que le microprogramme n'a pas été falsifié.

```
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
```

Ensuite, vous pouvez utiliser les commandes getAttribute ou findKey pour vérifier les attributs de la clé nouvellement importée. La commande suivante utilise findKey pour vérifier que la clé 37 a le type, l'étiquette et l'ID spécifiés par la commande, et qu'il s'agit d'une clé de session. Comme affiché sur la ligne 5 de la sortie, findKey rapporte que la seule clé qui correspond à tous les attributs est la clé 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
```

Paramètres

-attest

Exécute un contrôle d'intégrité qui vérifie que le microprogramme sur lequel le cluster est exécuté n'a pas été altéré.

Par défaut : aucune vérification d'attestation.

Obligatoire : non

-f

Spécifie le fichier qui contient la clé à importer.

Le fichier doit contenir la copie en texte brut d'une clé AES ou Triple DES de la longueur spécifiée. Les clés RC4 et DES ne sont pas valides sur les HSM en mode FIPS.

- AES : 16, 24 ou 32 octets
- Triple DES (3DES) : 24 octets

Obligatoire : oui

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-id

Spécifie un identifiant défini par l'utilisateur pour la clé. Saisissez une chaîne unique dans le cluster. La valeur par défaut est une chaîne vide.

Valeur par défaut : pas de valeur d'ID.

Obligatoire : non

-l

Spécifie l'étiquette définie par l'utilisateur pour la clé. Saisissez une chaîne.

Vous pouvez utiliser n'importe quelle phrase qui vous aide à identifier la clé. Comme il n'est pas nécessaire que l'étiquette soit unique, vous pouvez l'utiliser pour regrouper et classer des clés.

Obligatoire : oui

-min_srv

Spécifie le nombre minimal de HSM sur lesquels la clé est synchronisée avant que la valeur du paramètre `-timeout` n'expire. Si la clé n'est pas synchronisée sur le nombre spécifié de serveurs dans le temps imparti, elle n'est pas créée.

AWS CloudHSM synchronise automatiquement chaque clé avec chaque HSM du cluster. Pour accélérer le processus, définissez la valeur de `min_srv` sur une valeur inférieure au nombre de

HSM dans le cluster et définissez une valeur de délai d'expiration peu élevée. Toutefois, notez que certaines demandes peuvent ne pas générer une clé.

Valeur par défaut : 1

Obligatoire : non

-sess

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Pour remplacer une clé de session par une clé persistante (jeton), utilisez setAttribute.

Par défaut : la clé est persistante.

Obligatoire : non

-timeout

Spécifie la durée (en secondes) pendant laquelle la commande attend qu'une clé soit synchronisée avec le nombre de HSM spécifié par le paramètre `min_srv`.

Ce paramètre n'est valide que lorsque le paramètre `min_srv` est également utilisé dans la commande.

Par défaut : aucun délai d'expiration. La commande attend indéfiniment et ne renvoie des résultats que lorsque la clé est synchronisée avec le nombre minimum de serveurs.

Obligatoire : non

-t

Spécifie le type de la clé symétrique. Entrez la constante qui représente le type de clé. Par exemple, pour créer une clé AES, entrez `-t 31`.

Valeurs valides :

- 21 : Triple DES (3DES).

- 31 : AES

Obligatoire : oui

-u

Partage la clé que vous importez avec les utilisateurs spécifiés. Ce paramètre autorise les autres utilisateurs de chiffrement (CU) HSM à utiliser cette clé dans les opérations cryptographiques.

Tapez un ID ou une liste séparée par des virgules d'ID utilisateur HSM, par exemple, `-u 5,6`. N'incluez pas l'ID utilisateur HSM de l'utilisateur actuel. Pour trouver un ID, vous pouvez utiliser la commande listUsers dans l'outil de ligne de commande cloudhsm_mgmt_util ou la commande listUsers dans l'outil de ligne de commande key_mgmt_util.

Obligatoire : non

-s, sem

Spécifie le handle de clé de la clé d'encapsulage. Ce paramètre est obligatoire. Pour trouver des handles de clé, utilisez la commande findKey.

Une clé d'encapsulage est une clé du HSM utilisée pour chiffrer (« encapsuler »), puis déchiffrer (« désencapsuler ») la clé pendant le processus d'importation. Seules les clés AES peut être utilisées en tant que clés d'encapsulage.

Vous pouvez utiliser n'importe quelle clé AES (quelle que soit sa taille) comme clé d'encapsulage. Puisque les clés d'encapsulage encapsulent, puis désencapsulent immédiatement la clé cible, vous pouvez utiliser des clés AES de session unique en tant que clé d'encapsulage. Pour déterminer si une clé peut être utilisée comme clé d'encapsulage, utilisez getAttribute pour obtenir la valeur de l'attribut `OBJ_ATTR_WRAP` (262). Pour créer une clé d'encapsulation, genSymKeyutilisez-la pour créer une clé AES (type 31).

Si vous utilisez le paramètre `-wk` pour spécifier une clé d'encapsulage externe, la clé d'encapsulage `-w` est utilisée pour désencapsuler, mais pas pour encapsuler, la clé importée.

> **Note**
>
> Clé 4 est une clé interne non prise en charge. Nous vous recommandons d'utiliser une clé AES que vous créez et gérez comme clé d'encapsulage.

Obligatoire : oui

-wk

Utilisez la clé AES du fichier spécifié pour encapsuler la clé en cours d'importation. Entrez le chemin et le nom d'un fichier qui contient une clé AES en texte brut.

Lorsque vous incluez ce paramètre, imSymKey utilise la clé du fichier `-wk` pour encapsuler la clé en cours d'importation, ainsi que la clé du HSM spécifié par le paramètre `-w` pour la désencapsuler. La valeur des paramètres `-w` et `-wk` doivent correspondre à la même clé en texte brut.

Par défaut : utilisez la clé d'encapsulage sur le HSM pour procéder au désencapsulage.

Obligatoire : non

Rubriques en relation

- genSymKey
- exSymKey
- wrapKey
- unWrapKey
- exportPrivateKey
- exportPubKey

insertMaskedObject

La commande insertMaskedObject de key_mgmt_util insère un objet masqué à partir d'un fichier dans un HSM désigné. Les objets masqués sont des objets clonés qui sont extraits d'un HSM à l'aide de la commande extractMaskedObject. Ils peuvent uniquement être utilisés après réinsertion dans leur cluster d'origine. Vous pouvez insérer un objet masqué uniquement dans le cluster à partir duquel il a été généré, ou un clone de ce cluster. Cela inclut toutes les versions clonées du cluster original générées en copiant une sauvegarde entre les régions et en utilisant cette sauvegarde pour créer un nouveau cluster.

Les objets masqués sont un moyen efficace pour décharger et synchroniser les clés, y compris les clés non extractibles (c'est-à-dire, les clés qui ont une valeur `OBJ_ATTR_EXTRACTABLE` de `0`). Ainsi, les clés peuvent être synchronisées en toute sécurité entre les clusters associés dans différentes régions sans qu'il soit nécessaire de mettre à jour le fichier de AWS CloudHSM configuration.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
insertMaskedObject -h

insertMaskedObject -f <filename>
                   [-min_srv <minimum-number-of-servers>]
                   [-timeout <number-of-seconds>]
```

Exemples

Cet exemple montre comment utiliser insertMaskedObject pour insérer un fichier d'objet masqué dans un HSM.

Example : Insérer un objet masqué

Cette commande insère un objet masqué dans un HSM à partir d'un fichier nommé `maskedObj`. Lorsque la commande réussit, insertMaskedObject renvoie un handle de clé pour la clé déchiffré à partir de l'objet masqué et un message de réussite.

```
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
```

Paramètres

Cette commande accepte les paramètres suivants :

`-h`

Affiche l'aide de la ligne de commande pour la commande.

Obligatoire : oui

`-f`

Spécifie le nom de fichier de l'objet masqué à insérer.

Obligatoire : oui

`-min_srv`

Spécifie le nombre minimal de serveurs sur lesquels l'objet masqué inséré est synchronisée avant que la valeur du paramètre `-timeout` n'expire. Si l'objet n'est pas synchronisé sur le nombre spécifié de serveurs dans le temps imparti, il n'est pas inséré.

Valeur par défaut : 1

Obligatoire : non

`-timeout`

Spécifie le délai d'attente (en secondes) pour la synchronisation de la clé sur les serveurs lorsque le paramètre `min-serv` est inclus. Si aucun nombre n'est spécifié, l'interrogation continue indéfiniment.

Par défaut : pas de limite

Obligatoire : non

Rubriques en relation

- extractMaskedObject
- syncKey
- Copie de sauvegardes dans plusieurs régions
- Création d'un AWS CloudHSM cluster à partir d'une sauvegarde précédente

IsValidKeyHandlefile

La IsValidKeyHandlefile commande dans key_mgmt_util est utilisée pour savoir si un fichier clé contient une vraie clé privée ou une fausse clé RSA PEM. Au lieu de contenir la clé privée réelle, un fichier PEM factice référence la clé privée dans le HSM. Ce fichier peut être utilisé pour établir le déchargement SSL/TLS à partir de votre serveur web vers le AWS CloudHSM. Pour plus d'informations, consultez la section Déchargement SSL/TLS sur Linux.

> **Note**
>
> IsValidKeyHandlefilene fonctionne que pour les clés RSA.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
IsValidKeyHandlefile -h

IsValidKeyHandlefile -f <rsa-private-key-file>
```

Exemples

Ces exemples montrent comment utiliser IsValidKeyHandlefile pour déterminer si un fichier de clé spécifique contient les véritables clés ou des clés PEM factices.

Example : Valider une clé privée réelle

Cette commande confirme que le fichier appelé `privateKey.pem` contient de véritables clés.

```
Command: IsValidKeyHandlefile -f privateKey.pem

Input key file has real private key
```

Example : Invalider une clé PEM factice

Cette commande confirme que le fichier appelé `caviumKey.pem` contient une clé PEM factice conçue à partir du handle de clé 15.

```
Command: IsValidKeyHandlefile -f caviumKey.pem

Input file has invalid key handle: 15
```

Paramètres

Cette commande accepte les paramètres suivants :

`-h`

Affiche l'aide de la ligne de commande pour la commande.

Obligatoire : oui

`-f`

Spécifie le fichier de clé privée RSA dont le contenu clé doit être vérifié.

Obligatoire : oui

Rubriques en relation

- getCaviumPrivClé
- Déchargement SSL/TLS sur Linux

listAttributes

La listAttributes commande contenue dans key_mgmt_util répertorie les attributs d'une AWS CloudHSM clé et les constantes qui les représentent. Vous utilisez ces constantes pour identifier les attributs dans les commandes getAttribute et setAttribute. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

Cette commande n'a pas de paramètres.

```
listAttributes
```

Exemple

Cette commande répertorie les attributs de clé que vous pouvez obtenir et modifier dans key_mgmt_util et les constantes qui les représentent. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

Pour représenter tous les attributs dans la commande getAttribute de key_mgmt_util, utilisez 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
```

Rubriques en relation

- listAttributes dans cloudhsm_mgmt_util
- getAttribute
- setAttribute
- Référence des attributs de clé

listUsers

La commande listUsers de key_mgmt_util permet d'obtenir la liste des utilisateurs des HSM, ainsi que leur type d'utilisateur et d'autres attributs.

Dans key_mgmt_util, listUsers renvoie une sortie qui représente tous les HSM du cluster, même s'ils ne sont pas cohérents. Pour obtenir des informations sur les utilisateurs de chaque HSM, utilisez la commande listUsers dans cloudhsm_mgmt_util.

Les commandes utilisateur contenues dans key_mgmt_util listUsers et getKeyInfokey_mgmt_util sont des commandes en lecture seule que les utilisateurs cryptographiques (CU) sont autorisés à

exécuter. Les autres commandes de gestion des utilisateurs font partie de cloudhsm_mgmt_util. Elles sont exécutées par des responsables de chiffrement ayant des autorisations de gestion des utilisateurs.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
listUsers

listUsers -h
```

Exemple

Cette commande répertorie les utilisateurs des HSM du cluster et leurs attributs. Vous pouvez utiliser l'`User ID`attribut pour identifier les utilisateurs dans d'autres commandes, telles que FindKey, GetAttribute et. 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
```

La sortie inclut les attributs utilisateur suivants :

- User ID : identifie l'utilisateur dans les commandes key_mgmt_util et cloudhsm_mgmt_util.

- User type : détermine les opérations que l'utilisateur peut effectuer sur le HSM.
- User Name : affiche le nom convivial défini pour l'utilisateur.
- MofnPubKey: indique si l'utilisateur a enregistré une paire de clés pour signer les jetons d'authentification du quorum.
- LoginFailureCnt: indique le nombre de fois où l'utilisateur s'est connecté sans succès.
- 2FA : indique que l'utilisateur a activé l'authentification multi-facteurs.

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

Rubriques en relation

- listUsers dans cloudhsm_mgmt_util
- findKey
- getAttribute
- getKeyInfo

loginHSM et logoutHSM

Vous pouvez utiliser les commandes loginHSM et logoutHSM dans key_mgmt_util pour vous connecter et vous déconnecter de chaque HSM dans un cluster. Une fois connecté aux HSM, vous pouvez utiliser key_mgmt_util pour effectuer plusieurs opérations de gestion des clés, y compris la synchronisation, la génération et l'encapsulage des clés publiques et privées.

Avant d'exécuter n'importe quelle commande key_mgmt_util, vous devez démarrer l'outil key_mgmt_util. Pour gérer les clés avec key_mgmt_util, vous devez vous connecter aux HSM en tant qu'utilisateur de chiffrement (CU).

> **Note**
>
> Si vous dépassez cinq tentatives de connexion incorrectes, votre compte est verrouillé. Si vous avez créé votre cluster avant février 2018, votre compte est verrouillé après 20

tentatives de connexion incorrectes. Pour déverrouiller le compte, un responsable de chiffrement (CO) doit réinitialiser votre mot de passe à l'aide de la commande changePswd dans cloudhsm_mgmt_util.
Si vous avez plusieurs HSM dans votre cluster, vous pouvez autoriser des tentatives de connexion incorrectes supplémentaires avant que votre compte soit verrouillé. Cela est dû au fait que le client CloudHSM équilibre la charge sur plusieurs HSM. Par conséquent, la tentative de connexion peut ne pas commencer sur le même HSM à chaque fois. Si vous testez cette fonctionnalité, nous vous recommandons de le faire sur un cluster avec un seul HSM actif.

Syntaxe

```
loginHSM -h

loginHSM -u <user type>
         { -p | -hpswd } <password>
         -s <username>
```

Exemple

Cet exemple montre comment se connecter et se déconnecter aux HSM d'un cluster avec les commandes `logoutHSM` et `loginHSM`.

Example : se connecter aux HSM

Cette commande se connecte aux HSM en tant qu'utilisateur de chiffrement (CU) avec le nom d'utilisateur `example_user` et le mot de passe `aws`. Le résultat indique que vous vous êtes connecté à tous les HSM du cluster.

```
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 : Se connecter avec un mot de passe masqué

Cette commande est identique à l'exemple ci-dessus, sauf que cette fois, vous spécifiez que le système doit masquer le mot de passe.

```
Command:  loginHSM -u CU -s example_user -hpswd
```

Le système vous invite à saisir votre mot de passe. Vous entrez le mot de passe, le système le masque et le résultat indique que la commande a réussi et que vous vous êtes connecté aux 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 : Déconnexion des HSM

Cette commande permet la déconnexion des HSM. Le résultat indique que vous vous êtes déconnecté de tous les HSM du cluster.

```
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
```

Paramètres

-h

Affiche l'aide concernant cette commande.

-u

Spécifie le type d'utilisateur de connexion. Vous devez vous connecter en tant qu'utilisateur de chiffrement afin d'utiliser key_mgmt_util.

Obligatoire : oui

-s

Spécifie le nom d'utilisateur de connexion.

Obligatoire : oui

{ -p | -hpswd }

Spécifiez le mot de passe de connexion avec `-p`. Le mot de passe s'affiche en texte brut lorsque vous le tapez. Pour masquer votre mot de passe, utilisez le paramètre `-hpswd` facultatif à la place de `-p` et suivez les instructions.

Obligatoire : oui

Rubriques en relation

- exit

setAttribute

La commande setAttribute dans key_mgmt_util convertit une clé valide uniquement dans la session en cours en une clé permanente qui existe jusqu'à ce que vous la supprimiez. Pour ce faire, la valeur de l'attribut de jeton de la clé (`OBJ_ATTR_TOKEN`), false (`0`), est remplacée par true (`1`). Vous pouvez uniquement modifier les attributs des clés qui vous appartiennent.

Vous pouvez également utiliser la commande setAttribute dans cloudhsm_mgmt_util pour modifier les attributs d'étiquette, d'encapsulage, de désencapsulage, de chiffrement et de déchiffrement.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
setAttribute -h
```

```
setAttribute -o <object handle>
             -a 1
```

Exemple

Cet exemple montre comment convertir une clé de session en une clé permanente.

La première commande utilise le `-sess` paramètre de genSymKeypour créer une clé AES 192 bits valide uniquement dans la session en cours. La sortie indique que le handle de clé de la nouvelle clé de session est 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
```

Cette commande utilise findKey pour trouver les clés de session de la session en cours. La sortie vérifie que la clé 262154 est une clé de session.

```
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
```

Cette commande utilise setAttribute pour convertir la clé 262154 de clé de session en clé permanente. Pour ce faire, la valeur de l'attribut de jeton (`OBJ_ATTR_TOKEN`) de la clé est modifié de false (`0`) à true (1). Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

La commande utilise le paramètre `-o` pour spécifier le handle de clé (`262154`) et le paramètre `-a` pour spécifier la constante qui représente l'attribut de jeton (`1`). Lorsque vous exécutez la commande, vous êtes invité à entrer une valeur pour l'attribut de jeton. La seule valeur valide est 1 (true), soit la valeur d'une clé permanente.

```
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
```

Pour confirmer que la clé `262154` est désormais permanente, la commande utilise findKey pour rechercher les clés de session (`-sess 1`) et les clés permanentes (`-sess 0`). Cette fois, la commande ne recherche pas de clés de session, mais renvoie `262154` dans la liste des clés permanentes.

```
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
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-o

Spécifie le handle de clé de la clé cible. Vous pouvez spécifier une seule clé dans chaque commande. Pour obtenir le handle d'une clé, utilisez findKey.

Obligatoire : oui

-a

Spécifie la constante qui représente l'attribut que vous souhaitez modifier. La seule valeur valide est 1, qui représente l'attribut de jeton `OBJ_ATTR_TOKEN`.

Pour obtenir les attributs et leurs valeurs entières, utilisez listAttributes.

Obligatoire : oui

Rubriques en relation

- setAttribute dans cloudhsm_mgmt_util
- getAttribute
- listAttributes
- Référence des attributs de clé

sign

La commande sign dans key_mgmt_util utilise une clé privée sélectionnée pour générer une signature pour un fichier.

Afin d'utiliser sign, vous devez tout d'abord disposer d'une clé privée dans votre HSM. Vous pouvez générer une clé privée avec les commandes genSymKey, genRSAKeyPair, ou genECCKeyPair.

Vous pouvez également en importer une avec la commande importPrivateKey. Pour plus d'informations, consultez Générez des clés.

La commande sign utilise un mécanisme de signature désigné par l'utilisateur représenté par un nombre entier pour signer un fichier message. Pour obtenir la liste des mécanismes de signature possibles, consultez Paramètres.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
sign -h

sign -f <file name>
     -k <private key handle>
     -m <signature mechanism>
     -out <signed file name>
```

Exemple

Cet exemple montre comment utiliser sign pour signer un fichier.

Example : Signer un fichier

Cette commande signe un fichier nommé `messageFile` avec une clé privée avec handle `266309`. Elle utilise le mécanisme de signature `SHA256_RSA_PKCS` (1) et enregistre le fichier signature résultant en tant que `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
```

Paramètres

Cette commande accepte les paramètres suivants :

`-f`

Le nom du fichier à signer.

Obligatoire : oui

`-k`

Le handle de clé privée à utiliser pour la signature.

Obligatoire : oui

`-m`

Un nombre entier qui représente le mécanisme de signature à utiliser pour la signature. Les mécanismes possibles correspondent aux nombres entiers suivants :

Mécanisme de signature	Nombre entier correspondant
`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

Mécanisme de signature	Nombre entier correspondant
ECDSA_SHA384	18
ECDSA_SHA512	19

Obligatoire : oui

-out

Le nom du fichier dans lequel le fichier signé sera enregistré.

Obligatoire : oui

Rubriques en relation

- verify
- importPrivateKey
- Genre RSA KeyPair
- GèneCC KeyPair
- genSymKey
- Générez des clés

unWrapKey

La commande unWrapKey de l'outil key_mgmt_util importe une clé symétrique (encapsulée) chiffrée ou privée à partir d'un fichier dans le HSM. Elle a été conçue pour importer des clés chiffrées qui ont été encapsulées par la commande wrapKey dans key_mgmt_util, mais elle peut également être utilisée pour désencapsuler des clés qui ont été encapsulées au moyen d'autres outils. Toutefois, dans ces cas, nous vous recommandons d'utiliser les bibliothèques de logiciels PKCS # 11 ou JCE pour désencapsuler la clé.

Les clés importées fonctionnent comme les clés générées par AWS CloudHSM. Cependant, la valeur de leur attribut OBJ_ATTR_LOCAL est zéro, ce qui indique qu'elle n'a pas été générée localement.

Après avoir importé une clé, veillez à marquer ou à supprimer le fichier de clé. Cette commande ne vous empêche pas d'importer plusieurs fois les mêmes éléments de clés. Le résultat, plusieurs clés

avec des handles de clé distincts et les mêmes éléments de clé, complique le suivi de l'utilisation des éléments de clé et évite un dépassement des limites de chiffrement.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
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>]
```

Exemple

Ces exemples montrent comment utiliser unWrapKey pour importer une clé encapsulée à partir d'un fichier dans les HSM. Dans le premier exemple, nous désencapsulons une clé qui a été encapsulée avec la commande wrapKey key_mgmt_util et qui, par conséquent, comporte un en-tête. Dans le deuxième exemple, nous désencapsulons une clé qui a été encapsulée en dehors de key_mgmt_util et qui, par conséquent, ne présente pas d'en-tête.

Example : Désencapsuler une clé (avec en-tête)

Cette commande importe une copie encapsulée d'une clé symétrique 3DES dans un HSM. La clé est désencapsulée avec une clé AES dotée de l'étiquette 6, qui est identique, du point de vue du chiffrement à celle utilisée pour encapsuler la clé 3DES. La sortie montre que la clé dans le fichier a été désencapsulée et importée, et que le handle de la clé importée est 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 : Désencapsuler une clé (sans en-tête))

Cette commande importe une copie encapsulée d'une clé symétrique 3DES dans un HSM. La clé est désencapsulée avec une clé AES dotée de l'étiquette 6, qui est identique, du point de vue du chiffrement à celle utilisée pour encapsuler la clé 3DES. Étant donné que cette clé 3DES n'a pas été encapsulée avec key_mgmt_util, le paramètre `noheader` est spécifié, ainsi que ses paramètres associés obligatoires : une étiquette de clé (`unwrapped3DES`), la classe de clé (`4`) et le type de clé (`21`). La sortie montre que la clé dans le fichier a été désencapsulée et importée, et que le handle de la clé importée est 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
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-f

Chemin et nom du fichier qui contient la clé encapsulée.

Obligatoire : oui

-s, sem

Spécifie la clé d'encapsulage. Entrez le handle d'une clé AES ou RSA sur le HSM. Ce paramètre est obligatoire. Pour trouver des handles de clé, utilisez la commande findKey.

Pour créer une clé d'encapsulation, genSymKeyutilisez-la pour générer une clé AES (type 31) ou genRSA KeyPair pour générer une paire de clés RSA (type 0). Si vous utilisez une paire de clés RSA, veillez à encapsuler la clé avec l'une des clés et à la désencapsuler avec l'autre. Pour déterminer si une clé peut être utilisée comme clé d'encapsulage, utilisez getAttribute pour obtenir la valeur de l'attribut `OBJ_ATTR_WRAP`, représentée par la constante `262`.

Obligatoire : oui

-sess

Crée une clé qui existe uniquement dans la session en cours. La clé ne peut pas être récupérée une fois la session terminée.

Utilisez ce paramètre lorsque vous n'avez besoin que brièvement d'une clé, par exemple une clé d'encapsulage qui chiffre, puis déchiffre rapidement, une autre clé. N'utilisez pas de clé de session pour chiffrer les données que vous pourriez avoir besoin de déchiffrer après la fin de la session.

Pour remplacer une clé de session par une clé persistante (jeton), utilisez setAttribute.

Par défaut : la clé est persistante.

Obligatoire : non

-min_srv

Spécifie le nombre minimal de HSM sur lesquels la clé est synchronisée avant que la valeur du paramètre `-timeout` n'expire. Si la clé n'est pas synchronisée sur le nombre spécifié de serveurs dans le temps imparti, elle n'est pas créée.

AWS CloudHSM synchronise automatiquement chaque clé avec chaque HSM du cluster. Pour accélérer le processus, définissez la valeur de `min_srv` sur une valeur inférieure au nombre de HSM dans le cluster et définissez une valeur de délai d'expiration peu élevée. Toutefois, notez que certaines demandes peuvent ne pas générer une clé.

Valeur par défaut : 1

Obligatoire : non

-timeout

Spécifie la durée (en secondes) pendant laquelle la commande attend qu'une clé soit synchronisée avec le nombre de HSM spécifié par le paramètre `min_srv`.

Ce paramètre n'est valide que lorsque le paramètre `min_srv` est également utilisé dans la commande.

Par défaut : aucun délai d'expiration. La commande attend indéfiniment et ne renvoie des résultats que lorsque la clé est synchronisée avec le nombre minimum de serveurs.

Obligatoire : non

-attest

Exécute un contrôle d'intégrité qui vérifie que le microprogramme sur lequel le cluster est exécuté n'a pas été altéré.

Par défaut : aucune vérification d'attestation.

Obligatoire : non

-nex

Rend la clé non extractible. La clé générée ne peut pas être exportée depuis le HSM.

Par défaut : la clé est extractible.

Obligatoire : non

-m

Valeur représentant le mécanisme d'encapsulage. CloudHSM prend en charge les mécanismes suivants :

Mécanisme	Valeur
`AES_KEY_WRAP_PAD_PKCS5`	4
`NIST_AES_WRAP_NO_PAD`	5
`NIST_AES_WRAP_PAD`	6
`RSA_AES`	7
`RSA_OAEP` (pour connaître la taille maximale des données, consultez la note plus loin dans cette section)	8
`AES_GCM`	10
`CLOUDHSM_AES_GCM`	11
`RSA_PKCS` (pour connaître la taille maximale des données, consultez la note plus loin dans cette section) Voir la note 1 ci-dessous pour un changement à venir.	12

Obligatoire : oui

> **Note**
>
> Lorsque vous utilisez le mécanisme `RSA_OAEP` d'encapsulage, la taille de clé maximale que vous pouvez encapsuler est déterminée par le module de la clé RSA et la longueur du hachage spécifié comme suit : Taille de clé maximale = modulusLengthIn octets- (2* hashLengthIn octets) -2.

Lorsque vous utilisez le mécanisme d'encapsulage RSA_PKCS, la taille maximale de clé que vous pouvez encapsuler est déterminée par le module de la clé RSA comme suit : Taille maximale de clé = (octets -11). modulusLengthIn

-t

Algorithme de hachage	Valeur
SHA1	2
SHA256	3
SHA384	4
SHA512	5
SHA224 (valable pour les mécanismes RSA_AES et RSA_OAEP)	6

Obligatoire : non

-noheader

Si vous désencapsulez une clé qui a été encapsulée en dehors de key_mgmt_util, vous devez spécifier ce paramètre et tous les autres paramètres associés.

Obligatoire : non

> **Note**
>
> Si vous spécifiez ce paramètre, vous devez également spécifier les `-noheader` paramètres suivants :

- -l

  Spécifie l'étiquette à ajouter à la clé désencapsulée.

  Obligatoire : oui

- -kc

  Spécifie la classe de la clé à désencapsuler. Les valeurs acceptables sont les suivantes :

  3 = clé privée d'une paire de clés publique-privée

  4 = clé secrète (symétrique)

  Obligatoire : oui

- -kt

  Spécifie le type de clé à désencapsuler. Les valeurs acceptables sont les suivantes :

  0 = `RSA`

  1 = `DSA`

  3 = `ECC`

  16 = `GENERIC_SECRET`

  21 = `DES3`

  31 = `AES`

  Obligatoire : oui

Vous pouvez également éventuellement préciser les paramètres `-noheader` suivants :

- -id

  L'ID à ajouter à la clé désencapsulée.

  Obligatoire : non

- -i

  Le vecteur d'initialisation de désencapsulage (IV) à utiliserdéballer vecteur d'initialisation (IV) à utiliser.

  Obligatoire : non

[1] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

Rubriques en relation

- wrapKey
- exSymKey
- imSymKey

vérifier

La commande verify dans key_mgmt_util confirme si un fichier a été signé ou non par une clé spécifique. Pour cela, la commande verify compare un fichier signé à un fichier source et analyse s'ils sont apparentés sur le plan cryptographique en fonction d'une clé publique donnée et du mécanisme de signature. Les fichiers peuvent être connectés à l' AWS CloudHSM aide de sign cette opération.

Les mécanismes de signature sont représentés par les nombres entiers répertoriés dans la section Paramètres.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
verify -h

verify -f <message-file>
       -s <signature-file>
       -k <public-key-handle>
       -m <signature-mechanism>
```

Exemple

Ces exemples montrent comment utiliser verify pour vérifier si une clé publique spécifique a été utilisée pour signer un fichier donné.

Example : Vérifier la signature d'un fichier

Cette commande tente de vérifier si un fichier nommé `hardwarCert.crt` a été signé par clé publique 262276 à l'aide du mécanisme de signature `SHA256_RSA_PKCS` pour générer le fichier

signé `hardwareCertSigned`. Vu que les paramètres donnés présentent une vraie relation de signature, la commande renvoie un message de réussite.

```
Command: verify -f hardwareCert.crt -s hardwareCertSigned -k 262276 -m 1

Signature verification successful

Cfm3Verify returned: 0x00 : HSM Return: SUCCESS
```

Example : Prouver une fausse relation de signature

Cette commande vérifie si un fichier nommé `hardwareCert.crt` a été signé par clé publique 262276 à l'aide du mécanisme de signature `SHA256_RSA_PKCS` pour générer le fichier signé `userCertSigned`. Vu que les paramètres donnés ne présentent pas de vraie relation de signature, la commande renvoie un message d'erreur.

```
Command: verify -f hardwarecert.crt -s usercertsigned -k 262276 -m 1
Cfm3Verify returned: 0x1b

CSP Error: ERR_BAD_PKCS_DATA
```

Paramètres

Cette commande accepte les paramètres suivants :

`-f`

Le nom du fichier message d'origine.

Obligatoire : oui

`-s`

Le nom du fichier signé.

Obligatoire : oui

`-k`

Le handle de la clé publique que l'on pense être utilisé pour signer le fichier.

Obligatoire : oui

-m

Un nombre entier qui représente le mécanisme de signature proposé utilisé pour signer le fichier. Les mécanismes possibles correspondent aux nombres entiers suivants :

Mécanisme de signature	Nombre entier correspondant
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

Obligatoire : oui

Rubriques en relation

- sign
- getCert
- Générez des clés

wrapKey

La commande wrapKey de key_mgmt_util exporte une copie chiffrée d'une clé symétrique ou privée depuis le module HSM vers un fichier. Lorsque vous exécutez wrapKey, vous spécifiez la clé à exporter, une clé sur le module HSM pour chiffrer (encapsuler) la clé à exporter et le fichier de sortie.

La commande `wrapKey` écrit la clé chiffrée dans un fichier que vous spécifiez, mais elle ne supprime pas la clé du HSM ou ne vous empêche pas de l'utiliser dans des opérations de chiffrement. Vous pouvez exporter la même clé plusieurs fois.

Seul le propriétaire d'une clé, c'est-à-dire l'utilisateur de chiffrement (CU) ayant créé la clé, peut l'exporter. Les utilisateurs qui partagent la clé peut l'utiliser dans des opérations de chiffrement, mais ne peuvent pas l'exporter.

Pour réimporter la clé chiffrée dans le HSM, utilisez unWrapKey. Pour exporter une clé en texte brut depuis un HSM, utilisez exSymKeyou selon exportPrivateKeyle cas. La aesWrapUnwrapcommande ne peut pas déchiffrer (déballer) les clés chiffréeswrapKey.

Avant d'exécuter une commande key_mgmt_util, vous devez démarrer key_mgmt_util et vous connecter au HSM en tant qu'utilisateur de chiffrement (CU).

Syntaxe

```
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>>]
```

Exemple

Example

Cette commande permet d'exporter une clé symétrique Triple DES (3DES) 192 bits (handle de clé 7). Elle utilise une clé AES 256 bits dans le module HSM (handle de clé 14) pour encapsuler la clé 7. Ensuite, elle écrit la clé 3DES chiffrée dans le fichier `3DES-encrypted.key`.

La sortie indique que la clé 7 (clé 3DES) a été correctement encapsulée et écrite dans le fichier spécifié. La clé chiffrée est longue de 307 octets.

```
        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
```

Paramètres

-h

Affiche l'aide concernant la commande.

Obligatoire : oui

-k

Poignée de clé de la clé que vous souhaitez exporter. Saisissez le handle de clé d'une clé symétrique ou privée qui vous appartient. Pour trouver des handles de clé, utilisez la commande findKey.

Pour vérifier qu'une clé peut être exportée, utilisez la commande getAttribute pour obtenir la valeur de l'attribut `OBJ_ATTR_EXTRACTABLE`, représentée par la constante `354`. Pour obtenir de l'aide sur l'interprétation des attributs de clé, consultez le Référence des attributs de clé.

De même, vous pouvez exporter uniquement les clés qui vous appartiennent. Pour trouver le propriétaire d'une clé, utilisez la getKeyInfocommande.

Obligatoire : oui

-s, sem

Spécifie la clé d'encapsulage. Entrez le handle d'une clé AES ou RSA sur le HSM. Ce paramètre est obligatoire. Pour trouver des handles de clé, utilisez la commande findKey.

Pour créer une clé d'encapsulation, genSymKeyutilisez-la pour générer une clé AES (type 31) ou genRSA KeyPair pour générer une paire de clés RSA (type 0). Si vous utilisez une paire de clés RSA, veillez à encapsuler la clé avec l'une des clés et à la désencapsuler avec l'autre. Pour déterminer si une clé peut être utilisée comme clé d'encapsulage, utilisez getAttribute pour obtenir la valeur de l'attribut `OBJ_ATTR_WRAP`, représentée par la constante `262`.

Obligatoire : oui

-out

Chemin et nom du fichier de sortie. Lorsque la commande aboutit, ce fichier contient une copie chiffrée de la clé exportée. Si le fichier existe déjà, la commande le remplace sans avertissement.

Obligatoire : oui

-m

Valeur représentant le mécanisme d'encapsulage. CloudHSM prend en charge les mécanismes suivants :

Mécanisme	Valeur
`AES_KEY_WRAP_PAD_PKCS5`	4
`NIST_AES_WRAP_NO_PAD`	5
`NIST_AES_WRAP_PAD`	6
`RSA_AES`	7
`RSA_OAEP` (pour connaître la taille maximale des données, consultez la note plus loin dans cette section)	8

Mécanisme	Valeur
AES_GCM	10
CLOUDHSM_AES_GCM	11
RSA_PKCS (pour connaître la taille maximale des données, consultez la note plus loin dans cette section) Voir la note 1 ci-dessous pour un changement à venir.	12

Obligatoire : oui

> **Note**
>
> Lorsque vous utilisez le mécanisme RSA_OAEP d'encapsulage, la taille de clé maximale que vous pouvez encapsuler est déterminée par le module de la clé RSA et la longueur du hachage spécifié comme suit : Taille maximale de la clé = (modulusLengthInoctets-2* octets-2). hashLengthIn
> Lorsque vous utilisez le mécanisme d'encapsulage RSA_PKCS, la taille maximale de clé que vous pouvez encapsuler est déterminée par le module de la clé RSA comme suit : Taille maximale de la clé = (octets -11). modulusLengthIn

-t

Valeur représentant l'algorithme de hachage. CloudHSM prend en charge les algorithmes suivants :

Algorithme de hachage	Valeur
SHA1	2
SHA256	3
SHA384	4
SHA512	5

Algorithme de hachage	Valeur
SHA224 (valable pour les mécanismes RSA_AES et RSA_OAEP)	6

Obligatoire : non

-aad

Nom de fichier contenant AAD.

> **Note**
>
> Valide uniquement pour les mécanismes AES_GCM et CLOUDHSM_AES_GCM.

Obligatoire : non

-noheader

Omet l'en-tête qui spécifie les attributs de clés spécifiques à CloudHSM. Utilisez ce paramètre uniquement si vous prévoyez de désencapsuler la clé à l'aide d'outils en dehors de key_mgmt_util.

Obligatoire : non

-i

Vecteur d'initialisation (IV) (valeur hexadécimale).

> **Note**
>
> Valide uniquement lorsqu'il est transmis avec le paramètre -noheader pour les mécanismes CLOUDHSM_AES_KEY_WRAP et NIST_AES_WRAP.

Obligatoire : non

-iv_file

Fichier dans lequel vous voulez écrire la valeur IV obtenue en réponse.

> **Note**
>
> Valide uniquement lorsqu'il est transmis avec le paramètre `-noheader` pour le mécanisme `AES_GCM`.

Obligatoire : non

-tag_size

Taille de l'étiquette à enregistrer avec le blob encapsulé.

> **Note**
>
> Valide uniquement lorsqu'il est transmis avec le paramètre `-noheader` pour les mécanismes `AES_GCM` et `CLOUDHSM_AES_GCM`. La taille minimale de la balise est de huit.

Obligatoire : non

[1] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

Rubriques en relation

- exSymKey
- imSymKey
- unWrapKey

Référence des attributs de clé

Les commandes key_mgmt_util utilisent des constantes pour représenter les attributs des clés dans un HSM. Cette rubrique peut vous aider à identifier les attributs, à trouver les constantes qui les représentent dans les commandes et à comprendre leurs valeurs.

Vous définissez les attributs d'une clé lors de sa création. Pour modifier l'attribut de jeton qui indique si une clé est persistante ou n'existe que dans la session, utilisez la commande setAttribute de key_mgmt_util. Pour modifier les attributs d'étiquette, d'encapsulage, de désencapsulage, de chiffrement ou de déchiffrement, utilisez la commande `setAttribute` de cloudhsm_mgmt_util.

Pour obtenir la liste des attributs et de leurs constantes, utilisez listAttributes. Pour obtenir les valeurs d'attribut d'une clé, utilisez getAttribute.

Le tableau suivant répertorie les attributs de clé, leurs constantes et leurs valeurs valides.

Attribut	Constant	Valeurs
OBJ_ATTR_ALL	512	Représente tous les attributs.
OBJ_ATTR_ALWAYS_SENSITIVE	357	0 : False. 1 : True.
OBJ_ATTR_CLASS	0	2 : Clé publique dans une paire de clés publique-privée. 3 : Clé privée dans une paire de clés publique-privée. 4 : Clé secrète (symétrique).
OBJ_ATTR_DECRYPT	261	0 : False. 1 : True. La clé peut être utilisée pour déchiffrer les données.
OBJ_ATTR_DERIVE	268	0 : False. 1 : True. La fonction dérive la clé.
OBJ_ATTR_DESTROYABLE	370	0 : False. 1 : True.
OBJ_ATTR_ENCRYPT	260	0 : False. 1 : True. La clé peut être utilisée pour chiffrer les données.

Attribut	Constant	Valeurs
OBJ_ATTR_EXTRACTABLE	354	0 : False. 1 : True. La clé peut être exportée à partir des HSM.
OBJ_ATTR_ID	258	Chaîne définie par l'utilisateur. Elle doit être unique dans le cluster. La valeur par défaut est une chaîne vide.
OBJ_ATTR_KCV	371	Valeur de contrôle de clé de la clé. Pour plus d'informations, consultez Informations supplémentaires.
OBJ_ATTR_KEY_TYPE	256	0 : RSA. 1 : DSA. 3 : EC. 16 : Secret générique. 18 : RC4. 21 : Triple DES (3DES). 31 : AES.
OBJ_ATTR_LABEL	3	Chaîne définie par l'utilisateur. Elle n'est pas tenue d'être unique dans le cluster.
OBJ_ATTR_LOCAL	355	0. Faux. La clé a été importée dans les HSM. 1 : True.

Attribut	Constant	Valeurs
OBJ_ATTR_MODULUS	288	Module qui a été utilisé pour générer une paire de clés RSA. Pour les clés EC, cette valeur représente le codage DER de la valeur ECPoint ANSI X9.62 « Q » au format hexadécimal. Pour les autres types de clé, cet attribut n'existe pas.
OBJ_ATTR_MODULUS_BITS	289	Longueur du module qui a été utilisé pour générer une paire de clés RSA. Pour les clés EC, cela représente l'ID de la courbe elliptique utilisée pour générer la clé. Pour les autres types de clé, cet attribut n'existe pas.
OBJ_ATTR_NEVER_EXTRACTABLE	356	0 : False. 1 : True. La clé ne peut être exportée à partir des HSM.
OBJ_ATTR_PUBLIC_EXPONENT	290	Exposant public utilisé pour générer une paire de clés RSA. Pour les autres types de clé, cet attribut n'existe pas.

Attribut	Constant	Valeurs
OBJ_ATTR_PRIVATE	2	0 : False. 1 : True. Cet attribut indique si les utilisateurs non authentifiés peuvent répertorier les attributs de la clé. Étant donné que le fournisseur CloudHSM PKCS#11 ne prend actuellement pas en charge les sessions publiques, cet attribut est défini sur 1 pour toutes les clés (y compris les clés publiques dans une paire de clés publique-privée).
OBJ_ATTR_SENSITIVE	259	0 : False. Clé publique dans une paire de clés publique-privée. 1 : True.
OBJ_ATTR_SIGN	264	0 : False. 1 : True. La clé peut être utilisée pour la signature (clés privées).
OBJ_ATTR_TOKEN	1	0 : False. Clé de session. 1 : True. Clé persistante.
OBJ_ATTR_TRUSTED	134	0 : False. 1 : True.

Attribut	Constant	Valeurs
OBJ_ATTR_UNWRAP	263	0 : False. 1 : True. La clé peut être utilisée pour déchiffrer les clés.
OBJ_ATTR_UNWRAP_TEMPLATE	1073742354	Les valeurs doivent utiliser le modèle d'attribut appliqué à toute clé désencapsulée à l'aide de cette clé d'encapsulage.
OBJ_ATTR_VALUE_LEN	353	Longueur de clé en octets
OBJ_ATTR_VERIFY	266	0 : False. 1 : True. La clé peut être utilisée pour la vérification (clés publiques).
OBJ_ATTR_WRAP	262	0 : False. 1 : True. La clé peut être utilisée pour chiffrer les clés.
OBJ_ATTR_WRAP_TEMPLATE	1073742353	Les valeurs doivent utiliser le modèle d'attribut pour correspondre à la clé encapsulée à l'aide de cette clé d'encapsulage.
OBJ_ATTR_WRAP_WITH_TRUSTED	528	0 : False. 1 : True.

Informations supplémentaires

Valeur de contrôle de clé (kcv)

La valeur de contrôle de clé (KCV) est un hachage ou une somme de contrôle sur 3 octets d'une clé qui est générée lorsque le HSM importe ou génère une clé. Vous pouvez également calculer un KCV en dehors du HSM, par exemple après avoir exporté une clé. Vous pouvez ensuite comparer les valeurs KCV pour confirmer l'identité et l'intégrité de la clé. Pour obtenir le KCV d'une clé, utilisez getAttribute.

AWS CloudHSM utilise la méthode standard suivante pour générer une valeur de contrôle clé :

- Clés symétriques : les 3 premiers octets du résultat du chiffrement d'un bloc zéro avec la clé.
- Paires de clés asymétriques : les 3 premiers octets du hachage SHA-1 de la clé publique.
- Clés HMAC : la KCV pour les clés HMAC n'est pas prise en charge pour le moment.

AWS CloudHSM Kits de développement logiciel pour clients

Utilisez un SDK client pour décharger les opérations cryptographiques des applications basées sur des plateformes ou des langages vers des modules de sécurité matériels (HSM).

AWS CloudHSM propose deux versions principales, et le SDK client 5 est la plus récente. Il offre de nombreux avantages par rapport au SDK client 3 (la série précédente). Pour plus d'informations, consultez Avantages du SDK client 5. Pour plus d'informations sur les plateformes prises en charge, veuillez consulter Plateformes prises en charge par le SDK client 5.

Pour plus d'informations sur l'utilisation du SDK client 3, veuillez consulter SDK client précédent (SDK client 3).

the section called "Bibliothèque PKCS #11"

PKCS #11 est une norme pour effectuer des opérations cryptographiques sur des modules HSM (Hardware Security Modules). AWS CloudHSM propose des implémentations de la bibliothèque PKCS #11 conformes à la version 2.40 de PKCS #11.

the section called "OpenSSL Dynamic Engine"

Le moteur dynamique AWS CloudHSM OpenSSL vous permet de décharger des opérations cryptographiques vers votre cluster CloudHSM via l'API OpenSSL.

the section called "Fournisseur JCE"

Le fournisseur AWS CloudHSM JCE est conforme à l'architecture cryptographique Java (JCA). Le fournisseur vous permet d'effectuer des opérations cryptographiques sur le HSM.

the section called "Fournisseurs KSP et CNG"

Le AWS CloudHSM client pour Windows inclut les fournisseurs CNG et KSP. Actuellement, seul le SDK client 3 prend en charge les fournisseurs CNG et KSP.

Plateformes prises en charge par le SDK client 5

Le support de base est différent pour chaque version du SDK AWS CloudHSM client. Le support de plateforme pour les composants d'un SDK correspond généralement au support de base, mais pas toujours. Pour déterminer le support de plateforme pour un composant donné, assurez-vous d'abord que la plateforme de votre choix apparaît dans la section de base du SDK, puis recherchez les exclusions ou toute autre information pertinente dans la section du composant.

AWS CloudHSM ne prend en charge que les systèmes d'exploitation 64 bits.

Le support de la plateforme évolue au fil du temps. Les versions antérieures du SDK du client CloudHSM peuvent ne pas prendre en charge tous les systèmes d'exploitation répertoriés ici. Utilisez les notes de mise à jour pour déterminer le système d'exploitation compatible avec les versions précédentes du SDK du client CloudHSM. Pour plus d'informations, consultez Téléchargements pour AWS CloudHSM le SDK client.

Pour connaître les plateformes prises en charge par le précédent SDK client, voir Plateformes prises en charge par le SDK client 3

Le SDK client 5 ne nécessite pas de démon client.

Support Linux pour le SDK client 5

Plateformes prises en charge	Architecture x86_64	Architecture ARM
Amazon Linux 2	Oui	Oui
Amazon Linux 2023	Oui	Oui
CentOS 7 (7,8+)	Oui	Non
Red Hat Enterprise Linux 7 (7.8+)	Oui	Non
Red Hat Enterprise Linux 8 (8.3 ou version ultérieure)	Oui	Non
Red Hat Enterprise Linux 9 (version 9.2+)	Oui	Oui
Ubuntu 20.04 LTS	Oui	Non
Ubuntu 22.04 LTS	Oui	Oui

Remarque : le SDK 5.4.2 est la dernière version à prendre en charge la plate-forme CentOS 8. Pour plus d'informations, veuillez consulter le site Web CentOS.

Support Windows pour le SDK client 5

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Support pour architecture sans serveur pour le SDK client 5

- AWS Lambda
- Docker/ECS

Composants pris en charge

Bibliothèque PKCS #11

La bibliothèque PKCS #11 est un composant multiplateforme compatible avec le support de base du SDK client 5 pour Linux et Windows. Pour plus d'informations, consultez the section called "Support Linux pour le SDK client 5" et the section called "Support Windows pour le SDK client 5".

OpenSSL Dynamic Engine

Le moteur dynamique OpenSSL est un composant uniquement pour Linux qui nécessite OpenSSL 1.0.2, 1.1.1 ou 3.x.

Fournisseur JCE

Le fournisseur JCE est un SDK Java compatible avec OpenJDK 8, OpenJDK 11, OpenJDK 17 et OpenJDK 21 sur toutes les plateformes prises en charge.

Avantages du SDK client 5

Comparé au SDK client 3, le SDK client 5 est plus facile à gérer, et offre une configurabilité supérieure ainsi qu'une fiabilité accrue. Le SDK client 5 apporte également des avantages clés supplémentaires au SDK client 3.

Conçu pour une architecture sans serveur

> Le SDK client 5 ne nécessite pas de démon client, vous n'avez donc plus besoin de gérer un service d'arrière-plan. Cela aide les utilisateurs de plusieurs manières significatives :

- Cela simplifie le processus de démarrage des applications. Pour vous lancer avec CloudHSM, il vous suffit de configurer le SDK avant de démarrer votre application.
- Vous n'avez pas besoin d'un processus permanent, ce qui facilite l'intégration avec des composants sans serveur tels que Lambda et Elastic Container Service (ECS).

Meilleures intégrations tierces et portabilité facilitée

Le SDK client 5 suit de près les spécifications JCE et facilite la portabilité entre les différents fournisseurs JCE et améliore les intégrations tierces

Expérience utilisateur et configurabilité améliorées

Le SDK client 5 améliore la lisibilité des messages de journal et fournit des exceptions et des mécanismes de gestion des erreurs plus clairs, ce qui facilite considérablement le triage en libre-service pour les utilisateurs. Le SDK 5 propose également diverses configurations, répertoriées sur la page de l'outil de configuration.

Support de plateforme plus large

Le SDK client 5 offre une meilleure prise en charge des plateformes d'exploitation modernes. Cela inclut la prise en charge des technologies ARM et une meilleure prise en charge de JCE, PKCS #11 et OpenSSL. Pour plus d'informations, reportez-vous à la section Plateformes prises en charge.

Fonctionnalités et mécanismes supplémentaires

Le SDK client 5 inclut des fonctionnalités et des mécanismes supplémentaires qui ne sont pas disponibles dans le SDK client 3, et le SDK client 5 continuera d'ajouter d'autres mécanismes à l'avenir.

Migration du SDK client 3 vers le SDK client 5

Pour obtenir des instructions détaillées sur la migration du SDK client 3 vers le SDK client 5, reportez-vous aux instructions de migration de chaque SDK client individuel :

- Migrez votre bibliothèque PKCS #11 du SDK client 3 vers le SDK client 5
- Migrez votre moteur dynamique OpenSSL du SDK client 3 vers le SDK client 5

- Migrez votre fournisseur JCE du SDK client 3 vers le SDK client 5
- Migrer de la CMU et de la KMU du SDK client 3 vers la CLI CloudHSM du SDK client 5

Pour les fonctionnalités ou les cas d'utilisation qui ne sont pas pris en charge par la CLI CloudHSM, contactez le support.

> **Note**
>
> La bibliothèque PKCS #11 du SDK client 5 est désormais prise en charge sur les plateformes Windows. Il peut gérer la plupart des cas d'utilisation dans lesquels les fournisseurs de GNC et de KSP peuvent et doivent être considérés comme un remplacement. KSP n'est actuellement disponible que dans le SDK client 3.

Bibliothèque PKCS #11

PKCS #11 est une norme pour effectuer des opérations cryptographiques sur des modules HSM (Hardware Security Modules). AWS CloudHSM propose des implémentations de la bibliothèque PKCS #11 conformes à la version 2.40 de PKCS #11.

Pour plus d'informations sur l'amorçage, veuillez consulter Connexion au cluster. Pour le dépannage, voirProblèmes connus pour la bibliothèque PKCS#11.

Pour plus d'informations sur l'utilisation du SDK client 3, veuillez consulter SDK client précédent (SDK client 3).

Rubriques

- Installation du SDK client 5 pour la bibliothèque PKCS #11
- Bibliothèque PKCS #11
- Types de clé pris en charge
- Mécanismes pris en charge
- Opérations d'API prises en charge
- Attributs de clé pris en charge
- Exemples de code pour la bibliothèque PKCS #11
- Migrez votre bibliothèque PKCS #11 du SDK client 3 vers le SDK client 5
- Configurations avancées pour PKCS #11

Installation du SDK client 5 pour la bibliothèque PKCS #11

Cette rubrique fournit des instructions pour installer la dernière version de la bibliothèque PKCS #11 de la série de versions 5 du SDK client. Pour plus d'informations sur le SDK client ou la bibliothèque PKCS #11, consultez la section Utilisation du SDK client et la bibliothèque PKCS #11.

Installation

Avec le SDK client 5, il n'est pas nécessaire d'installer ou d'exécuter un démon client.

Pour exécuter un seul cluster HSM avec le SDK client 5, vous devez d'abord gérer les paramètres de durabilité des clés client en définissant `disable_key_availability_check` sur `True`. Pour plus d'informations, veuillez consulter les sections Synchronisation des clés et outil de configuration du SDK client 5.

Pour plus d'informations sur la bibliothèque PKCS #11 dans le SDK client 5, consultez la section bibliothèque PKCS #11.

> **Note**
>
> Pour exécuter un seul cluster HSM avec le SDK client 5, vous devez d'abord gérer les paramètres de durabilité des clés client en définissant `disable_key_availability_check` sur `True`. Pour plus d'informations, veuillez consulter les sections Synchronisation des clés et outil de configuration du SDK client 5.

Pour installer e configurer la bibliothèque PKCS #11

1. Utilisez les commandes suivantes pour télécharger et installer la bibliothèque PKCS #11.

   Amazon Linux 2

   Installez la bibliothèque PKCS #11 pour Amazon Linux 2 sur une architecture X86_64 :

   ```
   $ 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
   ```

   Installez la bibliothèque PKCS #11 pour Amazon Linux 2 sur l'architecture ARM64 :

```
$ 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

Installez la bibliothèque PKCS #11 pour Amazon Linux 2023 sur une architecture X86_64 :

```
$ 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
```

Installez la bibliothèque PKCS #11 pour Amazon Linux 2023 sur une architecture ARM64 :

```
$ 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+)

Installez la bibliothèque PKCS #11 pour CentOS 7.8+ sur l'architecture X86_64 :

```
$ 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+)

Installez la bibliothèque PKCS #11 pour RHEL 7 sur l'architecture X86_64 :

```
$ 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+)

Installez la bibliothèque PKCS #11 pour RHEL 8 sur l'architecture X86_64 :

```
$ 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+)

Installez la bibliothèque PKCS #11 pour RHEL 9 sur l'architecture X86_64 :

```
$ 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
```

Installez la bibliothèque PKCS #11 pour RHEL 9 sur l'architecture ARM64 :

```
$ 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

Installez la bibliothèque PKCS #11 pour Ubuntu 20.04 LTS sur l'architecture X86_64 :

```
$ 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

Installez la bibliothèque PKCS #11 pour Ubuntu 22.04 LTS sur l'architecture X86_64 :

```
$ 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
```

Installez la bibliothèque PKCS #11 pour Ubuntu 22.04 LTS sur l'architecture ARM64 :

```
$ 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

Installez la bibliothèque PKCS #11 pour Windows Server 2016 sur une architecture X86_64 :

1. Téléchargez la bibliothèque PKCS #11 pour le SDK client 5.
2. Exécutez le programme d'installation de la bibliothèque PKCS #11 (AWSCloudHSMPKCS11-latest.msi) avec les privilèges d'administrateur Windows.

Windows Server 2019

Installez la bibliothèque PKCS #11 pour Windows Server 2019 sur une architecture X86_64 :

1. Téléchargez la bibliothèque PKCS #11 pour le SDK client 5.
2. Exécutez le programme d'installation de la bibliothèque PKCS #11 (AWSCloudHSMPKCS11-latest.msi) avec les privilèges d'administrateur Windows.

2. Utilisez l'outil de configuration pour spécifier l'emplacement du certificat émetteur. Pour obtenir des instructions, veuillez consulter Spécifier l'emplacement du certificat émetteur.
3. Pour vous connecter à votre cluster, consultez Amorcez le SDK client.
4. Vous trouverez les fichiers de bibliothèque PKCS #11 dans les emplacements suivants :

- Binaires, scripts de configuration et fichiers journaux Linux :

```
/opt/cloudhsm
```

Binaires Windows :

```
C:\ProgramFiles\Amazon\CloudHSM
```

Scripts de configuration et fichiers journaux Windows :

```
C:\ProgramData\Amazon\CloudHSM
```

Bibliothèque PKCS #11

Lorsque vous utilisez PKCS # 11 avec , votre application s'exécute en tant qu'utilisateur de chiffrement (CU) dans vos HSM. Votre application peut afficher et gérer uniquement les clés que le CU possède et partage. Vous pouvez utiliser un CU existant dans votre HSM ou créer un nouveau CU pour votre application. Pour plus d'informations sur la gestion des CU, consultez le sections Gestion des utilisateurs HSM avec la CLI CloudHSM et Gestion des utilisateurs HSM avec l'utilitaire de gestion CloudHSM (CMU)

Pour spécifier le CU pour PKCS #11, utilisez le paramètre de code PIN PKCS #11 fonction C_Login. En AWS CloudHSM effet, le paramètre pin a le format suivant :

```
<CU_user_name>:<password>
```

Par exemple, la commande suivante définit le code PIN de bibliothèque PKCS #11 sur le CU avec le nom d'utilisateur `CryptoUser` et le mot de passe `CUPassword123!`.

```
CryptoUser:CUPassword123!
```

Types de clé pris en charge

La bibliothèque PKCS #11 prend en charge les Types de clé suivants.

Type de clé	Description
AES	Générez des clés AES de 128, 192 et 256 bits.
Triple DES (3DES, DESede)	Générez des clés Triple DES 192 bits. Voir la note 1 ci-dessous pour un changement à venir.
EC	Générez des clés avec les courbes secp224r1 (P-224), secp256r1 (P-256), secp256k1 (Blockchain), secp384r1 (P-384) et secp521r1 (P-521).
GENERIC_SECRET	Générez des secrets génériques de 1 à 800 octets.
RSA	Générez des clés RSA de 2 048 bits à 4 096 bits, par incréments de 256 bits.

[1] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

Mécanismes pris en charge

La bibliothèque PKCS #11 est conforme à la version 2.40 de la spécification PKCS #11. Pour appeler une fonction de chiffrement utilisant PKCS #11, appelez une fonction à l'aide d'un mécanisme donné. Les sections suivantes résument les combinaisons de fonctions et de mécanismes prises en charge par AWS CloudHSM.

La bibliothèque PKCS #11 prend en charge les algorithmes suivants :

- Chiffrement et déchiffrement : AES-CBC, AES-CTR, AES-ECB, AES-GCM, DES3-CBC, DES3-ECB, RSA-OAEP et RSA-PKCS
- Signature et vérification : RSA, HMAC et ECDSA ; avec et sans hachage
- Hachage/résumé : SHA1, SHA224, SHA256, SHA384 et SHA512
- Encapsulage de clés – AES Key Wrap,[1] AES-GCM, RSA-AES et RSA-OAEP

Rubriques

Génération de fonctions de clé et de paire de clés

La bibliothèque AWS CloudHSM logicielle de la bibliothèque PKCS #11 vous permet d'utiliser les mécanismes suivants pour les fonctions Generate Key et Key Pair.

- `CKM_RSA_PKCS_KEY_PAIR_GEN`
- `CKM_RSA_X9_31_KEY_PAIR_GEN` – Le fonctionnement de ce mécanisme est identique au mécanisme `CKM_RSA_PKCS_KEY_PAIR_GEN`, mais offre de meilleures garanties pour la génération de p et q.
- `CKM_EC_KEY_PAIR_GEN`
- `CKM_GENERIC_SECRET_KEY_GEN`
- `CKM_AES_KEY_GEN`
- `CKM_DES3_KEY_GEN`— le changement à venir est indiqué dans la note de bas de page 5.

Fonctions de signature et de vérification

La bibliothèque AWS CloudHSM logicielle de la bibliothèque PKCS #11 vous permet d'utiliser les mécanismes suivants pour les fonctions de signature et de vérification. Avec le SDK client 5, les données sont hachées localement dans le logiciel. Cela signifie qu'il n'y a aucune limite quant à la taille des données pouvant être hachées par le SDK.

Avec le SDK client 5, le hachage RSA et ECDSA est effectué localement, il n'y a donc aucune limite de données. Avec HMAC, il existe une limite de données. Pour plus d'informations, consultez la note de bas de page 2.

RSA

- CKM_RSA_X_509
- CKM_RSA_PKCS - Opérations à une seule partie uniquement.
- CKM_RSA_PKCS_PSS - Opérations à une seule partie uniquement.
- 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 - Opérations à une seule partie uniquement.
- 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`

Fonctions de récupération de signature et de récupération de vérification

Le SDK client 5 ne prend pas en charge les fonctions de récupération de signature et de récupération de vérification.

Fonctions de résumé

La bibliothèque AWS CloudHSM logicielle de la bibliothèque PKCS #11 vous permet d'utiliser les mécanismes suivants pour les fonctions Digest. Avec le SDK client 5, les données sont hachées localement dans le logiciel. Cela signifie qu'il n'y a aucune limite quant à la taille des données pouvant être hachées par le SDK.

- `CKM_SHA_1`
- `CKM_SHA224`
- `CKM_SHA256`
- `CKM_SHA384`
- `CKM_SHA512`

Fonctions de chiffrement et de déchiffrement

La bibliothèque AWS CloudHSM logicielle de la bibliothèque PKCS #11 vous permet d'utiliser les mécanismes suivants pour les fonctions de chiffrement et de déchiffrement.

- `CKM_RSA_X_509`
- `CKM_RSA_PKCS` - Opérations à une seule partie uniquement. Modification à venir répertoriée dans la note de bas de page 5.
- `CKM_RSA_PKCS_OAEP` - Opérations à une seule partie uniquement.
- `CKM_AES_ECB`

- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_DES3_CBC— le changement à venir est indiqué dans la note de bas de page 5.
- CKM_DES3_ECB— le changement à venir est indiqué dans la note de bas de page 5.
- CKM_DES3_CBC_PAD— le changement à venir est indiqué dans la note de bas de page 5.
- CKM_AES_GCM [1, 2]
- CKM_CLOUDHSM_AES_GCM[3]

Fonctions de dérivation de clé

La bibliothèque AWS CloudHSM logicielle de la bibliothèque PKCS #11 vous permet d'utiliser les mécanismes suivants pour les fonctions Derive.

- CKM_SP800_108_COUNTER_KDF

Fonctions d'encapsulage et de désencapsulage

La bibliothèque AWS CloudHSM logicielle de la bibliothèque PKCS #11 vous permet d'utiliser les mécanismes suivants pour les fonctions Wrap et Unwrap.

Pour plus d'informations sur l'encapsulage de clé AES, voir Encapsulage de clé AES.

- CKM_RSA_PKCS - Opérations à une seule partie uniquement. Une modification à venir est répertoriée dans la note de bas de page 5.
- 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]

Taille maximale des données pour chaque mécanisme

Le tableau suivant répertorie la taille maximale des données définie pour chaque mécanisme :

Taille maximale des jeux de données

Mécanisme	Taille maximale des données en octets
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

Annotations du mécanisme

- [1] Lorsque vous procédez au chiffrement AES GCM, le HSM n'accepte pas de données du vecteur d'initialisation (VI) de l'application. Vous devez utiliser un vecteur d'initialisation qu'il génère. Le vecteur d'initialisation 12 octets fourni par le HSM est écrit dans la référence en mémoire vers lequel pointe l'élément pIV des paramètres CK_GCM_PARAMS de la structure que vous fournissez. Pour éviter toute confusion de l'utilisateur, le kit SDK PKCS#11 version 1.1.1 et ultérieure s'assure que cet élément pIV pointe vers une mémoire tampon mise à zéro lorsque le chiffrement AES-GCM est initialisé.
- [2] Lors de l'utilisation de données avec l'un des mécanismes suivants, si la mémoire tampon des données dépasse la taille maximale des données, l'opération génère une erreur. Pour ces mécanismes, tout le traitement des données doit avoir lieu à l'intérieur du HSM. Pour plus d'informations sur les ensembles de tailles de données maximales pour chaque mécanisme, reportez-vous àTaille maximale des données pour chaque mécanisme.

- [3] Mécanisme défini par le fournisseur. Afin d'utiliser les mécanismes définis par le fournisseur CloudHSM, les applications PKCS #11 doivent inclure `/opt/cloudhsm/include/pkcs11t.h` lors de la compilation.

  `CKM_CLOUDHSM_AES_GCM` : Ce mécanisme propriétaire est une alternative plus sûre par programme à la norme `CKM_AES_GCM`. Il ajoute le IV généré par le HSM au chiffrement au lieu de l'écrire dans la structure `CK_GCM_PARAMS` fournie lors de l'initialisation du chiffrement. Vous pouvez utiliser ce mécanisme avec les fonctions `C_Encrypt`, `C_WrapKey`, `C_Decrypt` et `C_UnwrapKey`. Lors de l'utilisation de ce mécanisme, la variable PiV dans la structure `CK_GCM_PARAMS` doit être définie sur `NULL`. Lors de l'utilisation de ce mécanisme avec `C_Decrypt` et `C_UnwrapKey`, le IV doit être ajouté au texte chiffré qui est en cours de désencapsulage.

  `CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD` : Encapsulage des clés AES avec remplissage PKCS #5

  `CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD` : Encapsulage des clés AES avec remplissage à l'aide de zéros

- [4] Les `CK_MECHANISM_TYPE` et `CK_RSA_PKCS_MGF_TYPE` suivants sont pris en charge en tant que `CK_RSA_PKCS_OAEP_PARAMS` pour `CKM_RSA_PKCS_OAEP`:
  - `CKM_SHA_1` utilisant `CKG_MGF1_SHA1`
  - `CKM_SHA224` utilisant `CKG_MGF1_SHA224`
  - `CKM_SHA256` utilisant `CKG_MGF1_SHA256`
  - `CKM_SHA384` utilisant `CKM_MGF1_SHA384`
  - `CKM_SHA512` utilisant `CKM_MGF1_SHA512`
- [5] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

Opérations d'API prises en charge

La bibliothèque PKCS #11 prend en charge les opérations d'API PKCS #11 suivantes.

- `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`

Attributs de clé pris en charge

Un objet de clé peut être une clé publique, privée ou secrète. Les actions autorisées sur un objet de clé sont spécifiées via des attributs. Les attributs sont définis lorsque l'objet de clé est créé. Lorsque vous utilisez la bibliothèque SDK PKCS #11, des valeurs par défaut sont attribuées, comme indiqué par la norme PKCS #11.

AWS CloudHSM ne prend pas en charge tous les attributs répertoriés dans la spécification PKCS #11. Nous nous conformons à la spécification pour tous les attributs que nous prenons en charge. Ces attributs sont indiqués dans les tableaux respectifs.

Les fonctions cryptographiques telles que `C_CreateObject`, `C_GenerateKey`, `C_GenerateKeyPair`, `C_UnwrapKey` et `C_DeriveKey` qui créent, modifient ou copient des objets utilisent un modèle d'attribut en tant que paramètre. Pour plus d'informations sur la transmission d'un modèle d'attributs lors de la création d'un objet, consultez les exemples dans Génération de clés par la bibliothèque PKCS #11.

Interprétation du tableau d'attributs de la bibliothèque PKCS #11

Le tableau de la bibliothèque PKCS #11 contient une liste d'attributs qui diffèrent en fonction des types de clé. Il indique si un attribut donné est pris en charge pour un type de clé particulier lors de l'utilisation d'une fonction cryptographique spécifique avec AWS CloudHSM.

Légende :

- ✓ indique que CloudHSM prend en charge l'attribut pour le type de clé spécifique.
- ✖ indique que CloudHSM ne prend pas en charge l'attribut pour le type de clé spécifique.
- R indique que la valeur de l'attribut est définie en lecture seule pour le type de clé spécifique.
- S indique que l'attribut ne peut pas être lu par `GetAttributeValue` car sensible.
- Une cellule vide dans la colonne Default Value (Valeur par défaut) indique qu'il n'y a aucune valeur par défaut attribuée à l'attribut.

GenerateKeyPair

Attribut	Type de clé				Valeur par défaut
	EC privée	EC publique	RSA privée	RSA publique	
`CKA_CLASS`	✓	✓	✓	✓	
`CKA_KEY_TYPE`	✓	✓	✓	✓	
`CKA_LABEL`	✓	✓	✓	✓	
`CKA_ID`	✓	✓	✓	✓	
`CKA_LOCAL`	R	R	R	R	True

Attribut	Type de clé				Valeur par défaut
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
CKA_SIGN	✓	✖	✓	✖	False
CKA_SIGN_RECOVER	✖	✖	✖	✖	
CKA_VERIFY	✖	✓	✖	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	
CKA_WRAP	✖	✓	✖	✓	False
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	

Attribut	Type de clé				Valeur par défaut
CKA_TRUSTED	✖	✓	✖	✓	False
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	False
CKA_UNWRAP	✓	✖	✓	✖	False
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	✖	✖	✖	✖	

Attribut	Type de clé				Valeur par défaut
CKA_PRIME_2	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	✓[2]	
CKA_EC_PARAMS	✖	✓[2]	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	
CKA_VALUE_LEN	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	

GenerateKey

Attribut	Type de clé			Valeur par défaut
	AES	DES3	Secrète générique	
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✓	✓	✖	False
CKA_DECRYPT	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	True

Attribut	Type de clé			Valeur par défaut
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	True
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	False
CKA_WRAP_TEMPLATE	✓	✓	✖	
CKA_TRUSTED	✓	✓	✖	False
CKA_WRAP_WITH_TRUSTED	✓	✓	✓	False
CKA_UNWRAP	✓	✓	✖	False
CKA_UNWRAP_TEMPLATE	✓	✓	✖	
CKA_SENSITIVE	✓	✓	✓	True
CKA_ALWAYS_SENSITIVE	✖	✖	✖	

Attribut	Type de clé			Valeur par défaut
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	
CKA_EXPONENT_2	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	
CKA_PUBLIC_EXPONENT	✖	✖	✖	

Attribut	Type de clé			Valeur par défaut
CKA_EC_PARAMS	✖	✖	✖	
CKA_EC_POINT	✖	✖	✖	
CKA_VALUE	✖	✖	✖	
CKA_VALUE_LEN	✓[2]	✖	✓[2]	
CKA_CHECK_VALUE	R	R	R	

CreateObject

Attribut	Type de clé							Valeur par défaut
	EC privée	EC publique	RSA privée	RSA publique	AES	DES3	Secrète générique	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	✓	✓	

Attribut	Type de clé							Valeur par défaut
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
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	✖	✖	False
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	✖	✖	

Attribut	Type de clé							Valeur par défaut
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	
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	R	R	R	
CKA_MODULUS	✖	✖	✓[2]	✓[2]	✖	✖	✖	

Attribut	Type de clé							Valeur par défaut
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]	✖	✖	✖	
CKA_EC_PARAMS	✓[2]	✓[2]	✖	✖	✖	✖	✖	
CKA_EC_POINT	✖	✓[2]	✖	✖	✖	✖	✖	
CKA_VALUE	✓[2]	✖	✖	✖	✓[2]	✓[2]	✓[2]	

Attribut	Type de clé							Valeur par défaut
CKA_VALUE_LEN	✖	✖	✖	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	R	R	R	

UnwrapKey

Attribut	Type de clé					Valeur par défaut
	EC privée	RSA privée	AES	DES3	Secrète générique	
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]	
CKA_LABEL	✓	✓	✓	✓	✓	
CKA_ID	✓	✓	✓	✓	✓	
CKA_LOCAL	R	R	R	R	R	False
CKA_TOKEN	✓	✓	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True

Attribut	Type de clé					Valeur par défaut
CKA_ENCRYPT	✖	✖	✓	✓	✖	False
CKA_DECRYPT	✖	✓	✓	✓	✖	False
CKA_DERIVE	✓	✓	✓	✓	✓	False
CKA_MODIFIABLE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	True
CKA_DESTROYABLE	✓	✓	✓	✓	✓	True
CKA_SIGN	✓	✓	✓	✓	✓	False
CKA_SIGN_RECOVER	✖	✖	✖	✖	✖	False
CKA_VERIFY	✖	✖	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	
CKA_WRAP	✖	✖	✓	✓	✖	False
CKA_UNWRAP	✖	✓	✓	✓	✖	False
CKA_SENSITIVE	✓	✓	✓	✓	✓	True

Attribut	Type de clé					Valeur par défaut
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	✖	✖	✖	✖	✖	
CKA_PRIVATE_EXPONENT	✖	✖	✖	✖	✖	

Attribut	Type de clé					Valeur par défaut
CKA_PUBLIC_EXPONENT		✖	✖	✖	✖	✖	
CKA_EC_PARAMS		✖	✖	✖	✖	✖	
CKA_EC_POINT		✖	✖	✖	✖	✖	
CKA_VALUE		✖	✖	✖	✖	✖	
CKA_VALUE_LEN		✖	✖	✖	✖	✖	
CKA_CHECK_VALUE		R	R	R	R	R	

DeriveKey

Attribut	Type de clé				Valeur par défaut
		AES	DES3	Secrète générique	
CKA_CLASS		✓[2]	✓[2]	✓[2]	
CKA_KEY_TYPE		✓[2]	✓[2]	✓[2]	
CKA_LABEL		✓	✓	✓	

Attribut	Type de clé			Valeur par défaut
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

Attribut	Type de clé			Valeur par défaut
CKA_UNWRAP	✓	✓	✖	False
CKA_SENSITIVE	R	R	R	True
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	

Attribut	Type de clé			Valeur par défaut
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

Attribut	Type de clé						
	EC privée	EC publique	RSA privée	RSA publique	AES	DES3	Secrète générique
CKA_CLASS	✓	✓	✓	✓	✓	✓	✓

Attribut	Type de clé						
CKA_KEY_TYPE	✓	✓	✓	✓	✓	✓	✓
CKA_LABEL	✓	✓	✓	✓	✓	✓	✓
CKA_ID	✓	✓	✓	✓	✓	✓	✓
CKA_LOCAL	✓	✓	✓	✓	✓	✓	✓
CKA_TOKEN	✓	✓	✓	✓	✓	✓	✓
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]	✓[1]
CKA_ENCRYPT	✖	✖	✖	✓	✓	✓	✖
CKA_DECRYPT	✖	✖	✓	✖	✓	✓	✖
CKA_DERIVE	✓	✓	✓	✓	✓	✓	✓
CKA_MODIFIABLE	✓	✓	✓	✓	✓	✓	✓
CKA_DESTROYABLE	✓	✓	✓	✓	✓	✓	✓
CKA_SIGN	✓	✖	✓	✖	✓	✓	✓
CKA_SIGN_RECOVER	✖	✖	✓	✖	✖	✖	✖

Attribut	Type de clé						
CKA_VERIFY	✖	✓	✖	✓	✓	✓	✓
CKA_VERIFY_RECOVER	✖	✖	✖	✓	✖	✖	✖
CKA_WRAP	✖	✖	✖	✓	✓	✓	✖
CKA_WRAP_TEMPLATE	✖	✓	✖	✓	✓	✓	✖
CKA_TRUSTED	✖	✓	✖	✓	✓	✓	✓
CKA_WRAP_WITH_TRUSTED	✓	✖	✓	✖	✓	✓	✓
CKA_UNWRAP	✖	✖	✓	✖	✓	✓	✖
CKA_UNWRAP_TEMPLATE	✓	✖	✓	✖	✓	✓	✖
CKA_SENSITIVE	✓	✖	✓	✖	✓	✓	✓
CKA_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓
CKA_NEVER_EXTRACTABLE	✓	✖	✓	✖	✓	✓	✓

Attribut	Type de clé						
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	R	R
CKA_MODULUS	✖	✖	✓	✓	✖	✖	✖
CKA_MODULUS_BITS	✖	✖	✖	✓	✖	✖	✖
CKA_PRIME_1	✖	✖	S	✖	✖	✖	✖
CKA_PRIME_2	✖	✖	S	✖	✖	✖	✖
CKA_COEFFICIENT	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_1	✖	✖	S	✖	✖	✖	✖
CKA_EXPONENT_2	✖	✖	S	✖	✖	✖	✖
CKA_PRIVATE_EXPONENT	✖	✖	S	✖	✖	✖	✖
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖

Attribut	Type de clé						
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
CKA_VALUE	S	✖	✖	✖	✓	✓	✓
CKA_VALUE_LEN	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	✖

Annotations d'attributs

- [1] Cet attribut est partiellement pris en charge par le micrologiciel et doit être explicitement défini sur la valeur par défaut.
- [2] Attribut obligatoire.

Modification d'attributs

Certains attributs d'un objet peuvent être modifiés une fois que l'objet a été créé, tandis que d'autres ne le peuvent pas. Pour modifier des attributs, utilisez la commande setAttribute à partir de cloudhsm_mgmt_util. Vous pouvez également obtenir une liste des attributs et des constantes qui les représentent en utilisant la commande listAttribute à partir de cloudhsm_mgmt_util.

La liste suivante contient les attributs que vous pouvez modifier après la création de l'objet :

- `CKA_LABEL`
- `CKA_TOKEN`

> **Note**
>
> La modification est autorisée uniquement pour changer une clé de session en une clé de jeton. Utilisez la commande setAttribute à partir de key_mgmt_util pour modifier la valeur de l'attribut.

- CKA_ENCRYPT
- CKA_DECRYPT
- CKA_SIGN
- CKA_VERIFY
- CKA_WRAP
- CKA_UNWRAP
- CKA_LABEL
- CKA_SENSITIVE
- CKA_DERIVE

> **Note**
>
> Cet attribut prend en charge la dérivation de clé. Il doit être `False` pour toutes les clés publiques et ne peut pas être défini sur `True`. Pour les clés privées EC et secrètes, il peut être défini sur `True` ou `False`.

- CKA_TRUSTED

> **Note**
>
> Cet attribut peut être défini sur `True` ou `False` par le responsable du chiffrement uniquement.

- CKA_WRAP_WITH_TRUSTED

> **Note**
>
> Appliquez cet attribut à une clé de données exportable pour indiquer que vous ne pouvez encapsuler cette clé qu'avec des clés marquées comme CKA_TRUSTED. Une fois CKA_WRAP_WITH_TRUSTED défini sur true, l'attribut passe en lecture seule et vous ne pouvez ni le modifier ni le supprimer.

Interprétation des codes d'erreur

Si vous spécifiez dans le modèle un attribut qui n'est pas pris en charge par une clé spécifique, une erreur se produit. Le tableau suivant contient des codes d'erreur qui sont générés lorsque vous violez des spécifications :

Code d'erreur	Description
`CKR_TEMPLATE_INCONSISTENT`	Vous recevez cette erreur lorsque vous spécifiez, dans le modèle d'attributs, un attribut conforme à la spécification PKCS #11 mais non pris en charge par CloudHSM.
`CKR_ATTRIBUTE_TYPE_INVALID`	Vous recevez cette erreur lorsque vous récupérez la valeur d'un attribut conforme à la spécification PKCS #11 mais non pris en charge par CloudHSM.
`CKR_ATTRIBUTE_INCOMPLETE`	Vous recevez cette erreur lorsque vous ne spécifiez pas l'attribut obligatoire dans le modèle d'attributs.
`CKR_ATTRIBUTE_READ_ONLY`	Vous recevez cette erreur lorsque vous spécifiez un attribut en lecture seule dans le modèle d'attributs.

Exemples de code pour la bibliothèque PKCS #11

Les exemples de code ci-dessous vous GitHub montrent comment accomplir des tâches de base à l'aide de la bibliothèque PKCS #11.

Prérequis

Avant d'exécuter les exemples, effectuez les étapes suivantes pour configurer votre environnement :

- Installez et configurez la bibliothèque PKCS #11 pour le SDK client 5.
- Configurez un utilisateur de chiffrement(CU). Votre application utilise ce compte HSM pour exécuter les exemples de code sur le HSM.

Exemples de code

Des exemples de code pour la bibliothèque AWS CloudHSM logicielle de PKCS #11 sont disponibles sur GitHub. Ce référentiel contient des exemples sur la façon d'effectuer des opérations courantes à l'aide de PKCS #11, y compris le chiffrement, le déchiffrement, la signature et la vérification.

- Générer des clés (AES, RSA, EC)
- Afficher les attributs des clés
- Chiffrer et déchiffrer les données avec AES GCM
- Chiffrer et déchiffrer les données avec AES_CTR
- Chiffrer et déchiffrer les données avec 3DES
- Signer et vérifier les données avec RSA
- Dériver des clés à l'aide de HMAC KDF
- Encapsuler et désencapsuler les clés avec AES en utilisant le remplissage PKCS #5
- Encapsuler et désencapsuler les clés avec AES sans remplissage
- Encapsuler et désencapsuler les clés avec AES à l'aide du remplissage avec des zéros
- Encapsuler et désencapsuler les clés avec AES-GCM
- Encapsuler et désencapsuler les clés avec RSA

Migrez votre bibliothèque PKCS #11 du SDK client 3 vers le SDK client 5

Utilisez cette rubrique pour migrer votre bibliothèque PKCS #11 du SDK client 3 vers le SDK client 5. Pour connaître les avantages de la migration, voirAvantages du SDK client 5.

Dans AWS CloudHSM, les applications client exécutent des opérations cryptographiques à l'aide du kit de développement logiciel (SDK) AWS CloudHSM client. Le SDK client 5 est le SDK principal auquel de nouvelles fonctionnalités et un support de plateforme continuent d'être ajoutés.

Pour consulter les instructions de migration pour tous les fournisseurs, voirMigration du SDK client 3 vers le SDK client 5.

Préparez-vous en faisant face aux changements les plus importants

Passez en revue ces modifications majeures et mettez à jour votre application dans votre environnement de développement en conséquence.

Les mécanismes d'enroulement ont changé

Mécanisme du SDK client 3	Mécanisme équivalent du SDK client 5
`CKM_AES_KEY_WRAP`	`CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD`
`CKM_AES_KEY_WRAP_PAD`	`CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD`
`CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD`	`CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD`
`CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD`	`CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD`
`CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD`	`CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD`

ECDH

Dans le SDK client 3, vous pouvez utiliser ECDH et spécifier un KDF. Cette fonctionnalité n'est actuellement pas disponible dans le SDK client 5. Si votre application a besoin de cette fonctionnalité, veuillez contacter l'assistance.

Les poignées de touches sont désormais spécifiques à la session

Pour utiliser correctement les descripteurs de clé dans le SDK client 5, vous devez obtenir des descripteurs de clé chaque fois que vous exécutez une application. Si vous avez des applications existantes qui utiliseront les mêmes descripteurs de clé au cours de différentes sessions, vous devez modifier votre code pour obtenir le descripteur de clé à chaque exécution de l'application. Pour plus d'informations sur la récupération des descripteurs de touches, consultez cet exemple AWS CloudHSM PKCS #11. Cette modification est conforme à la spécification PKCS #11 2.40.

Migrer vers le SDK client 5

Suivez les instructions de cette section pour migrer du SDK client 3 vers le SDK client 5.

> **Note**
>
> Amazon Linux, Ubuntu 16.04, Ubuntu 18.04, CentOS 6, CentOS 8 et RHEL 6 ne sont actuellement pas pris en charge avec le SDK client 5. Si vous utilisez actuellement l'une de ces plateformes avec le SDK client 3, vous devrez en choisir une autre lors de la migration vers le SDK client 5.

1. Désinstallez la bibliothèque PKCS #11 pour le SDK client 3.

   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. Désinstallez le démon client pour le SDK client 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**
>
> Les configurations personnalisées doivent être réactivées.

3. Installez la bibliothèque PKCS #11 du SDK client en suivant les étapes décrites dans. Installation du SDK client 5 pour la bibliothèque PKCS #11
4. Le SDK client 5 introduit un nouveau format de fichier de configuration et un outil d'amorçage en ligne de commande. Pour démarrer votre bibliothèque PKCS #11 du SDK client 5, suivez les instructions répertoriées dans le guide de l'utilisateur ci-dessous. Amorcez le SDK client
5. Dans votre environnement de développement, testez votre application. Mettez à jour votre code existant pour corriger les modifications importantes avant votre migration finale.

Rubriques en relation

- Les meilleures pratiques pour AWS CloudHSM

Configurations avancées pour PKCS #11

Le fournisseur AWS CloudHSM PKCS #11 inclut la configuration avancée suivante, qui ne fait pas partie des configurations générales utilisées par la plupart des clients. Ces configurations fournissent des fonctionnalités supplémentaires.

- Connexion à plusieurs emplacements avec PKCS #11
- Configuration de nouvelle tentative pour PKCS #11

Connexion à plusieurs emplacements avec PKCS #11

Un emplacement unique dans la bibliothèque PKCS #11 du SDK client 5 représente une connexion unique à un cluster dans AWS CloudHSM. Avec le SDK client 5, vous pouvez configurer votre bibliothèque PKCS11 pour autoriser plusieurs emplacements à connecter les utilisateurs à plusieurs clusters CloudHSM à partir d'une seule application PKCS #11.

Suivez les instructions de cette rubrique pour que votre application utilise la fonctionnalité d'emplacements multiples pour se connecter à plusieurs clusters.

Rubriques

- Prérequis pour plusieurs emplacements
- Configuration de la bibliothèque PKCS #11 pour la fonctionnalité d'emplacements multiples
- configure-pkcs11 add-cluster
- configure-pkcs11 remove-cluster

Prérequis pour plusieurs emplacements

- Au moins deux AWS CloudHSM clusters auxquels vous souhaitez vous connecter, ainsi que leurs certificats de cluster.
- Une instance EC2 avec des groupes de sécurité correctement configurés pour se connecter à tous les clusters ci-dessus. Pour plus d'informations sur la configuration d'un cluster et de l'instance client, reportez-vous à la section Mise en route avec AWS CloudHSM.
- Pour configurer la fonctionnalité d'emplacements multiples, vous devez avoir déjà téléchargé et installé la bibliothèque PKCS #11. Si vous ne l'avez pas déjà fait, consultez les instructions figurant dans ???.

Configuration de la bibliothèque PKCS #11 pour la fonctionnalité d'emplacements multiples

Pour configurer votre bibliothèque PKCS #11 pour la fonctionnalité d'emplacements multiples, procédez comme suit :

1. Identifiez les clusters auxquels vous souhaitez vous connecter à l'aide de la fonctionnalité d'emplacements multiples.
2. Ajoutez ces clusters à votre configuration PKCS #11 en suivant les instructions de ???

3. La prochaine fois que votre application PKCS #11 s'exécutera, elle disposera d'une fonctionnalité d'emplacements multiples.

configure-pkcs11 add-cluster

Lorsque vous vous connectez à plusieurs emplacements avec PKCS #11, utilisez la commande configure-pkcs11 add-cluster pour ajouter un cluster à votre configuration.

Syntaxe

```
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]
```

Exemples

Ajoutez un cluster à l'aide du paramètre **`cluster-id`**

Example

Utilisez le paramètre configure-pkcs11 add-cluster ainsi que le paramètre `cluster-id` pour ajouter un cluster (avec l'ID de `cluster-1234567`) à votre configuration.

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**
>
> Si l'utilisation de configure-pkcs11 add-cluster avec le paramètre `cluster-id` n'entraîne pas l'ajout du cluster, reportez-vous à l'exemple suivant pour une version plus longue de cette commande qui nécessite également des paramètres `--region` et `--endpoint` pour identifier le cluster ajouté. Si, par exemple, la région du cluster est différente de celle configurée par défaut dans votre interface de ligne de commande AWS, vous devez utiliser le paramètre `--region` pour utiliser la bonne région. En outre, il est possible de spécifier le point de terminaison d' AWS CloudHSM API à utiliser pour l'appel, ce qui peut être nécessaire pour diverses configurations réseau, telles que l'utilisation de points de terminaison d'interface VPC qui n'utilisent pas le nom d'hôte DNS par défaut pour. AWS CloudHSM

Ajouter un cluster à l'aide des paramètres **`cluster-id`**, **`endpoint`** et **`region`**

Example

Utilisez les paramètres configure-pkcs11 add-cluster ainsi que `cluster-id`, `endpoint` et `region` pour ajouter un cluster (avec l'ID de `cluster-1234567`) à votre configuration.

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
```

Pour plus d'informations sur les paramètres `--cluster-id`, `--region` et `--endpoint`, consultez the section called “Paramètres”.

Paramètres

--cluster-id ***<Cluster ID>***

Effectue un appel `DescribeClusters` pour rechercher toutes les adresses IP de l'interface réseau Elastic (ENI) du HSM du cluster associées à l'ID du cluster. Le système ajoute les adresses IP ENI aux fichiers AWS CloudHSM de configuration.

> **Note**
>
> Si vous utilisez le `--cluster-id` paramètre à partir d'une instance EC2 au sein d'un VPC qui n'a pas accès à l'Internet public, vous devez créer un point de terminaison VPC d'interface auquel vous connecter. AWS CloudHSM Pour plus d'informations sur les points de terminaison d'un VPC, veuillez consulter ???.

Obligatoire : oui

--endpoint ***<Endpoint>***

Spécifiez le point de terminaison de l' AWS CloudHSM API utilisé pour effectuer l'`DescribeClusters`appel. Vous devez définir cette option en combinaison avec `--cluster-id`.

Obligatoire : non

-- hsm-ca-cert <HsmCA Certificate Filepath>

Spécifie le chemin du fichier vers le certificat HSM CA.

Obligatoire : non

--region ***<Region>***

Spécifiez la région de votre cluster. Vous devez définir cette option en combinaison avec `--cluster-id`.

Si vous ne fournissez pas le paramètre `--region`, le système choisit la région en essayant de lire les variables d'environnement `AWS_DEFAULT_REGION` ou `AWS_REGION`. Si ces variables ne sont pas définies, le système vérifie la région associée à votre profil dans votre fichier AWS Config (généralement `~/.aws/config`), sauf si vous avez spécifié un autre fichier dans la variable d'environnement `AWS_CONFIG_FILE`. Si aucune des options ci-dessus n'est définie, le système utilise par défaut la région `us-east-1`.

Obligatoire : non

-- server-client-cert-file <Client Certificate Filepath>

Chemin d'accès au certificat client utilisé pour l'authentification mutuelle client-serveur TLS.

N'utilisez cette option que si vous ne souhaitez pas utiliser la clé par défaut et le certificat SSL/TLS inclus dans le SDK client 5. Vous devez définir cette option en combinaison avec `--server-client-key-file`.

Obligatoire : non

-- server-client-key-file <Client Key Filepath>

Chemin d'accès à la clé client utilisée pour l'authentification mutuelle client-serveur TLS.

N'utilisez cette option que si vous ne souhaitez pas utiliser la clé par défaut et le certificat SSL/TLS inclus dans le SDK client 5. Vous devez définir cette option en combinaison avec `--server-client-cert-file`.

Obligatoire : non

configure-pkcs11 remove-cluster

Lorsque vous vous connectez à plusieurs emplacements avec PKCS #11, utilisez la commande configure-pkcs11 remove-cluster pour supprimer un cluster des emplacements PKCS #11 disponibles.

Syntaxe

```
configure-pkcs11 remove-cluster [OPTIONS]
        --cluster-id <CLUSTER ID>
        [-h, --help]
```

Exemples

Supprimer un cluster à l'aide du paramètre **`cluster-id`**

Example

Utilisez le paramètre configure-pkcs11 remove-cluster ainsi que le paramètre `cluster-id` pour supprimer un cluster (avec l'ID de `cluster-1234567`) de votre configuration.

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
```

Pour plus d'informations sur le paramètre `--cluster-id`, consultez the section called "Paramètres".

Paramètre

--cluster-id *<Cluster ID>*

L'ID du cluster à supprimer de la configuration

Obligatoire : oui

Commandes de nouvelle tentative pour PKCS #11

Le SDK client 5.8.0 et versions ultérieures disposent d'une stratégie de relance automatique intégrée qui permet de réessayer les opérations limitées par HSM du côté client. Lorsqu'un HSM limite les opérations parce qu'il est trop occupé à effectuer les opérations précédentes et qu'il ne peut pas traiter plus de demandes, les SDK clients tentent de retenter les opérations limitées jusqu'à 3 fois tout en reculant de manière exponentielle. Cette stratégie de nouvelle tentative automatique peut être réglée sur l'un des deux modes suivants : désactivé et standard.

- désactivé : le SDK client n'exécutera aucune stratégie de nouvelle tentative pour les opérations limitées effectuées par le HSM.
- standard : il s'agit du mode par défaut pour le SDK client 5.8.0 et versions ultérieures. Dans ce mode, les SDK clients réessaieront automatiquement les opérations limitées en reculant de manière exponentielle.

Pour plus d'informations, consultez Limitation du HSM.

Définir des commandes de nouvelle tentative sur le mode désactivé

Linux

Pour définir les commandes de nouvelle tentative sur off pour le SDK client 5 sous Linux

- Vous pouvez utiliser la commande suivante pour définir une nouvelle tentative de configuration sur le mode off :

```
$ sudo /opt/cloudhsm/bin/configure-pkcs11 --default-retry-mode off
```

Windows

Pour définir les commandes de nouvelle tentative sur off pour le SDK client 5 sous Windows

- Vous pouvez utiliser la commande suivante pour définir une nouvelle tentative de configuration sur le modeoff :

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-pkcs11.exe --default-retry-
mode off
```

OpenSSL Dynamic Engine

Le moteur dynamique AWS CloudHSM OpenSSL vous permet de décharger des opérations cryptographiques vers votre cluster CloudHSM via l'API OpenSSL.

AWS CloudHSM fournit un moteur dynamique OpenSSL, que vous pouvez consulter dans. Déchargement SSL/TLS sur Linux Pour un exemple d'utilisation AWS CloudHSM avec OpenSSL, consultez ce blog sur la sécurité AWS. Pour plus d'informations sur les plateformes prises en charge pour les SDK, veuillez consulter the section called "Plateformes prises en charge". Pour le dépannage, voirProblèmes connus pour le moteur dynamique OpenSSL.

Pour plus d'informations sur l'utilisation du SDK client 3, veuillez consulter SDK client précédent (SDK client 3).

Pour plus d'informations, consultez les rubriques ci-dessous.

Rubriques

Installation du moteur dynamique OpenSSL

> **Note**
>
> Pour exécuter un seul cluster HSM avec le SDK client 5, vous devez d'abord gérer les paramètres de durabilité des clés client en définissant `disable_key_availability_check` sur `True`. Pour plus d'informations, veuillez consulter les sections Synchronisation des clés et outil de configuration du SDK client 5.

Pour installer et configurer le moteur dynamique OpenSSL

1. Utilisez les commandes suivantes pour télécharger et installer le moteur OpenSSL.

   Amazon Linux 2

   Installez le moteur dynamique OpenSSL pour Amazon Linux 2 sur une architecture x86_64 :

   ```
   $ 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
   ```

   Installez le moteur dynamique OpenSSL pour Amazon Linux 2 sur une architecture ARM64 :

   ```
   $ 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

Installez le moteur dynamique OpenSSL pour Amazon Linux 2023 sur une architecture x86_64 :

```
$ 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
```

Installez le moteur dynamique OpenSSL pour Amazon Linux 2023 sur l'architecture ARM64 :

```
$ 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+)

Installez le moteur dynamique OpenSSL pour CentOS 7 sur une architecture x86_64 :

```
$ 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+)

Installez le moteur dynamique OpenSSL pour RHEL 7 sur une architecture x86_64 :

```
$ 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+)

Installez le moteur dynamique OpenSSL pour RHEL 8 sur une architecture x86_64 :

```
$ 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+)

Installez le moteur dynamique OpenSSL pour RHEL 9 sur une architecture x86_64 :

```
$ 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
```

Installez le moteur dynamique OpenSSL pour RHEL 9 sur l'architecture ARM64 :

```
$ 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

Installez le moteur dynamique OpenSSL pour Ubuntu 20.04 LTS sur une architecture x86_64 :

```
$ 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

Installez le moteur dynamique OpenSSL pour Ubuntu 22.04 LTS sur une architecture x86_64 :

```
$ 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
```

Installez le moteur dynamique OpenSSL pour Ubuntu 22.04 LTS sur l'architecture ARM64 :

```
$ 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
```

Vous avez installé la bibliothèque partagée pour le moteur dynamique à l'adresse `/opt/cloudhsm/lib/libcloudhsm_openssl_engine.so`.

2. Démarrez le SDK client 5. Pour plus d'informations sur le démarrage, consultez Amorcez le SDK client.
3. Définissez une variable d'environnement avec les informations d'identification d'un utilisateur de chiffrement (CU). Pour plus d'informations sur la création de CU, consultez Utilisation du CMU pour gérer les utilisateurs.

```
$ export CLOUDHSM_PIN=<HSM user name>:<password>
```

> **Note**
>
> Le SDK client 5 introduit la variable d'environnement `CLOUDHSM_PIN` permettant de stocker les informations d'identification du CU. Dans le SDK client 3, vous stockez les informations d'identification CU dans la variable d'environnement `n3fips_password`. Le SDK client 5 prend en charge les deux variables d'environnement, mais nous vous recommandons d'utiliser `CLOUDHSM_PIN`.

4. Connectez votre installation de moteur dynamique OpenSSL au cluster. Pour plus d'informations, consultez Connexion à un Cluster.
5. Démarrez le SDK client 5. Pour plus d'informations, consultez the section called "Amorcez le SDK client".

Vérifiez le moteur dynamique OpenSSL pour le SDK client 5

Utilisez la commande suivante pour vérifier votre installation de moteur dynamique OpenSSL.

```
$ openssl engine -t cloudhsm
```

Le résultat suivant vérifie votre configuration :

```
(cloudhsm) CloudHSM OpenSSL Engine
     [ available ]
```

Types de clés du moteur dynamique OpenSSL

Le moteur dynamique AWS CloudHSM OpenSSL prend en charge les types de clés suivants.

Type de clé	Description
EC	Signature/vérification ECDSA pour les Types de clé P-256, P-384 et secp256k1. Pour générer des clés EC interopérables avec le moteur OpenSSL, consultez key generate-file.
RSA	Génération de clés RSA pour les clés 2048, 3072 et 4096 bits. Signation/vérification RSA. La vérification est déchargée vers le logiciel OpenSSL.

Mécanismes du moteur dynamique OpenSSL

Apprenez à utiliser les mécanismes du AWS CloudHSM moteur dynamique OpenSSL.

Fonctions de signature et de vérification

Le moteur dynamique AWS CloudHSM OpenSSL vous permet d'utiliser les mécanismes suivants pour les fonctions de signature et de vérification.

Avec le SDK client 5, les données sont hachées localement dans le logiciel. Cela signifie qu'il n'y a aucune limite quant à la taille des données pouvant être hachées.

Types de signature RSA

- SHA1withRSA
- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA

Types de signature ECDSA

- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

Migrez votre moteur dynamique OpenSSL du SDK client 3 vers le SDK client 5

Utilisez cette rubrique pour migrer votre moteur dynamique OpenSSL du SDK client 3 vers le SDK client 5. Pour connaître les avantages de la migration, voirAvantages du SDK client 5.

Dans AWS CloudHSM, les applications client exécutent des opérations cryptographiques à l'aide du kit de développement logiciel (SDK) AWS CloudHSM client. Le SDK client 5 est le SDK principal auquel de nouvelles fonctionnalités et un support de plateforme continuent d'être ajoutés.

> **Note**
>
> La génération de nombres aléatoires n'est actuellement pas prise en charge dans le SDK client 5 avec OpenSSL Dynamic Engine.

Pour consulter les instructions de migration pour tous les fournisseurs, voirMigration du SDK client 3 vers le SDK client 5.

Migrer vers le SDK client 5

Suivez les instructions de cette section pour migrer du SDK client 3 vers le SDK client 5.

> **Note**
>
> Amazon Linux, Ubuntu 16.04, Ubuntu 18.04, CentOS 6, CentOS 8 et RHEL 6 ne sont actuellement pas pris en charge avec le SDK client 5. Si vous utilisez actuellement l'une de ces plateformes avec le SDK client 3, vous devrez en choisir une autre lors de la migration vers le SDK client 5.

1. Désinstallez le moteur dynamique OpenSSL pour le SDK client 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. Désinstallez le démon client pour le SDK client 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**
   >
   > Les configurations personnalisées doivent être réactivées.

3. Installez le SDK client OpenSSL Dynamic Engine en suivant les étapes décrites dans. Installation du moteur dynamique OpenSSL
4. Le SDK client 5 introduit un nouveau format de fichier de configuration et un outil d'amorçage en ligne de commande. Pour démarrer votre moteur dynamique OpenSSL Client SDK 5, suivez les instructions répertoriées dans le guide de l'utilisateur ci-dessous. Amorcez le SDK client
5. Dans votre environnement de développement, testez votre application. Mettez à jour votre code existant pour corriger les modifications importantes avant votre migration finale.

Rubriques en relation

- Les meilleures pratiques pour AWS CloudHSM

Configurations avancées pour OpenSSL

Le fournisseur AWS CloudHSM OpenSSL inclut la configuration avancée suivante, qui ne fait pas partie des configurations générales utilisées par la plupart des clients. Ces configurations fournissent des fonctionnalités supplémentaires.

- Commandes de nouvelle tentative pour OpenSSL

Commandes de nouvelle tentative pour OpenSSL

Le SDK client 5.8.0 et versions ultérieures disposent d'une stratégie de relance automatique intégrée qui permet de réessayer les opérations limitées par HSM du côté client. Lorsqu'un HSM limite les opérations parce qu'il est trop occupé à effectuer les opérations précédentes et qu'il ne peut pas traiter plus de demandes, les SDK clients tentent de retenter les opérations limitées jusqu'à 3 fois tout en reculant de manière exponentielle. Cette stratégie de nouvelle tentative automatique peut être réglée sur l'un des deux modes suivants : désactivé et standard.

- désactivé : le SDK client n'exécutera aucune stratégie de nouvelle tentative pour les opérations limitées effectuées par le HSM.
- standard : il s'agit du mode par défaut pour le SDK client 5.8.0 et versions ultérieures. Dans ce mode, les SDK clients réessaieront automatiquement les opérations limitées en reculant de manière exponentielle.

Pour plus d'informations, consultez Limitation du HSM.

Définir des commandes de nouvelle tentative sur le mode désactivé

Linux

Pour définir les commandes de nouvelle tentative sur off pour le SDK client 5 sous Linux

- Vous pouvez utiliser la commande suivante pour configurer des commandes de nouvelle tentative en mode off :

```
$ sudo /opt/cloudhsm/bin/configure-dyn --default-retry-mode off
```

Windows

Pour définir les commandes de nouvelle tentative sur off pour le SDK client 5 sous Windows

- Vous pouvez utiliser la commande suivante pour configurer des commandes de nouvelle tentative en mode off :

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-dyn.exe --default-retry-mode
 off
```

Fournisseur JCE

Le fournisseur AWS CloudHSM JCE est une implémentation de fournisseur créée à partir du framework de fournisseur Java Cryptographic Extension (JCE). Le JCE vous permet d'effectuer des opérations cryptographiques à l'aide du kit de développement Java (JDK). Dans ce guide, le fournisseur AWS CloudHSM JCE est parfois appelé fournisseur JCE. Utilisez le fournisseur JCE et le JDK pour décharger les opérations cryptographiques vers le HSM. Pour le dépannage, voirProblèmes connus pour le kit SDK JCE.

Pour plus d'informations sur l'utilisation du SDK client 3, consultez SDK client précédent (SDK client 3).

Rubriques

- Installation et utilisation du fournisseur AWS CloudHSM JCE pour le SDK client 5
- Types de clé pris en charge
- Mécanismes pris en charge
- Attributs de clé Java pris en charge
- Exemples de code pour la bibliothèque de AWS CloudHSM logiciels pour Java
- AWS CloudHSM Javadocs, fournisseur JCE
- Utilisation de la classe AWS CloudHSM KeyStore Java
- Migrez votre fournisseur JCE du SDK client 3 vers le SDK client 5
- Configurations avancées pour JCE

Installation et utilisation du fournisseur AWS CloudHSM JCE pour le SDK client 5

Le fournisseur JCE est compatible avec OpenJDK 8, OpenJDK 11, OpenJDK 17 et OpenJDK 21. Vous pouvez télécharger les deux depuis le site Web d'OpenJDK.

> **Note**
>
> Pour exécuter un seul cluster HSM avec le SDK client 5, vous devez d'abord gérer les paramètres de durabilité des clés client en définissant `disable_key_availability_check` sur `True`. Pour plus d'informations, veuillez consulter les sections Synchronisation des clés et outil de configuration du SDK client 5.

Rubriques

- Installation du fournisseur JCE
- Fournir des informations d'identification au fournisseur JCE
- Principes de base de la gestion des clés chez le fournisseur JCE

Installation du fournisseur JCE

1. Utilisez les commandes suivantes pour télécharger et installer le fournisseur JCE.

   Amazon Linux 2

   Installez le fournisseur JCE pour Amazon Linux 2 sur une architecture x86_64 :

   ```
   $ 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
   ```

   Installez le fournisseur JCE pour Amazon Linux 2 sur une architecture ARM64 :

   ```
   $ 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

Installez le fournisseur JCE pour Amazon Linux 2023 sur une architecture x86_64 :

```
$ 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
```

Installez le fournisseur JCE pour Amazon Linux 2023 sur l'architecture ARM64 :

```
$ 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+)

Installez le fournisseur JCE pour CentOS 7 sur une architecture x86_64 :

```
$ 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+)

Installez le fournisseur JCE pour RHEL 7 sur une architecture x86_64 :

```
$ 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+)

Installez le fournisseur JCE pour RHEL 8 sur une architecture x86_64 :

```
$ 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+)

Installez le fournisseur JCE pour RHEL 9 (9.2+) sur une architecture x86_64 :

```
$ 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
```

Installez le fournisseur JCE pour RHEL 9 (9.2+) sur l'architecture ARM64 :

```
$ 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

Installez le fournisseur JCE pour Ubuntu 20.04 LTS sur une architecture x86_64 :

```
$ 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

Installez le fournisseur JCE pour Ubuntu 22.04 LTS sur une architecture x86_64 :

```
$ 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
```

Installez le fournisseur JCE pour Ubuntu 22.04 LTS sur l'architecture ARM64 :

```
$ 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

Installez le fournisseur JCE pour Windows Server 2016 sur une architecture x86_64, ouvrez-le en PowerShell tant qu'administrateur et exécutez la commande suivante :

```
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

Installez le fournisseur JCE pour Windows Server 2019 sur une architecture x86_64, ouvrez-le en PowerShell tant qu'administrateur et exécutez la commande suivante :

```
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. Démarrez le SDK client 5. Pour plus d'informations sur le démarrage, consultez Amorcez le SDK client.
3. Recherchez les fichiers du fournisseur JCE suivants :

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`

Fournir des informations d'identification au fournisseur JCE

Les HSM doivent d'abord authentifier votre application Java avant que l'application ne puisse les utiliser. Les HSM authentifient une session en utilisant la méthode de connexion explicite ou implicite.

Connexion explicite : Cette méthode vous permet de fournir les informations d'identification AWS CloudHSM directement dans l'application. Elle utilise la méthode `AuthProvider`, dans laquelle vous transmettez le nom d'utilisateur et le mot de passe du CU dans le modèle d'accès. Pour plus d'informations, consultez la section Connexion à un exemple de code HSM.

Connexion implicite : Cette méthode vous permet de définir les informations de connexion AWS CloudHSM dans un nouveau fichier de propriétés, dans les propriétés système ou en tant que variables d'environnement.

- Propriétés système : Définissez les informations d'identification par le biais des propriétés système lors de l'exécution de votre application. Les exemples suivants montrent deux manières de le faire :

  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>");
```

- Variables d'environnement : Définissez les informations d'identification en tant que variables d'environnement.

  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>"
  ```

Les informations d'identification peuvent ne pas être disponibles si l'application ne les fournit pas ou si vous essayez une opération avant que le HSM n'authentifie la session. Dans ces cas, la bibliothèque de logiciels CloudHSM pour Java recherche les informations d'identification dans l'ordre suivant :

1. Propriétés système
2. Variables d'environnement

Principes de base de la gestion des clés chez le fournisseur JCE

Les notions de base sur la gestion des clés dans le fournisseur JCE incluent l'importation des clés, l'exportation des clés, le chargement des clés par le handle ou la suppression des clés. Pour plus d'informations sur la gestion des clés, consultez l'exemple de code Gérer les clés.

Vous pouvez également trouver d'autres exemples de code de fournisseur JCE surExemples de code.

Types de clé pris en charge

La bibliothèque AWS CloudHSM logicielle pour Java vous permet de générer les types de clés suivants.

Type de clé	Description
AES	Générez des clés AES de 128, 192 et 256 bits.
Triple DES (3DES, DESede)	Générez une clé triple DES 192 bits [1]
EC	Générez des paires de clés EC : courbes NIST secp224r1 (P-224), secp256r1 (P-256), secp256k1 (Blockchain), secp384r1 (P-384) et secp521r1 (P-521).
GENERIC_SECRET	Générez des secrets génériques de 1 à 800 octets.
HMAC	Support de hachage pour SHA1, SHA224, SHA256, SHA384, SHA512.
RSA	Générez des clés RSA de 2 048 bits à 4 096 bits, par incréments de 256 bits.

[1] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

Mécanismes pris en charge

Pour plus d'informations sur les interfaces JCA (Java Cryptography Architecture) et les classes de moteur prises en charge par AWS CloudHSM, consultez les rubriques suivantes.

Rubriques

- Génération de fonctions de clé et de paire de clés
- Fonctions de chiffrement
- Fonctions de signature et de vérification

- Fonctions de résumé
- Fonctions du code d'authentification de message utilisant hash (HMAC).
- Fonctions du code d'authentification des messages basé sur le chiffrement (CMAC)
- Convertissez les clés en spécifications de clés à l'aide de fabriques de clés
- Annotations du mécanisme

Génération de fonctions de clé et de paire de clés

La bibliothèque AWS CloudHSM logicielle pour Java vous permet d'utiliser les opérations suivantes pour générer des fonctions de clé et de paire de clés.

- `RSA`
- `EC`
- `AES`
- `DESede (Triple DES)`[voir note1]
- `GenericSecret`

Fonctions de chiffrement

La bibliothèque AWS CloudHSM logicielle pour Java prend en charge les combinaisons d'algorithmes, de modes et de remplissage suivantes.

Algorithm	Mode	Remplissage	Remarques

AES	CBC	`AES/CBC/NoPadding`
		
		`AES/CBC/PKCS5Padding`
			Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE` .
			
			Implémente `Cipher.UNWRAP_MODE` for `AES/CBC NoPadding`

Algorithm	Mode	Remplissage	Remarques

AES	ECB	`AES/ECB/PKCS5Padding`
		
		`AES/ECB/NoPadding`
			Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE`.

AES	CTR	`AES/CTR/NoPadding`	Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE`.

Algorithm	Mode	Remplissage	Remarques
AES	GCM	`AES/GCM/NoPadding`	Implémente `Cipher.WRAP_MODE`, `Cipher.UNWRAP_MODE`, `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE`. Lorsque vous effectuez un chiffrement AES-GCM, le HSM ignore le vecteur d'initialisation (IV) dans la demande et utilise un vecteur d'initialisation généré. Lorsque l'opération est terminée, vous devez appeler `Cipher.getIV()` pour obtenir le vecteur d'initialisation.

AESWrap	ECB	`AESWrap/ECB/NoPadding`
		
		`AESWrap/ECB/PKCS5Padding`
		
		`AESWrap/ECB/ZeroPadding`
			Implémente `Cipher.WRAP_MODE` et `Cipher.UNWRAP_MODE`.

Algorithm	Mode	Remplissage	Remarques

DESede (Triple DES)	CBC	`DESede/CBC/PKCS5Padding`
		
		`DESede/CBC/NoPadding`
			Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE`. Voir la note 1 ci-dessous pour un changement à venir.

DESede (Triple DES)	ECB	`DESede/ECB/NoPadding`
		
		`DESede/ECB/PKCS5Padding`
			Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE`. Voir la note 1 ci-dessous pour un changement à venir.

Algorithm	Mode	Remplissage	Remarques

RSA	ECB	`RSA/ECB/PKCS1Padding` Voir note1
		
		`RSA/ECB/OAEPPadding`
		
		`RSA/ECB/OAEPWithSHA-1ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-224ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-256ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-384ANDMGF1Padding`
		
		`RSA/ECB/OAEPWithSHA-512ANDMGF1Padding`
			Implémente `Cipher.WRAP_MODE`, `Cipher.UNWRAP_MODE`, `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE`.

Algorithm	Mode	Remplissage	Remarques
RSA	ECB	`RSA/ECB/NoPadding`	Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE` .

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`
			Implémente `Cipher.WRAP_MODE` et `Cipher.UNWRAP_MODE` .

Fonctions de signature et de vérification

La bibliothèque AWS CloudHSM logicielle pour Java prend en charge les types de signature et de vérification suivants. Avec le SDK client 5 et les algorithmes de signature avec hachage, les données sont hachées localement dans le logiciel avant d'être envoyées au HSM pour signature/vérification. Cela signifie qu'il n'y a aucune limite quant à la taille des données pouvant être hachées par le SDK.

Types de signature RSA

- `NONEwithRSA`
- `RSASSA-PSS`
- `SHA1withRSA`
- `SHA1withRSA/PSS`
- `SHA1withRSAandMGF1`
- `SHA224withRSA`
- `SHA224withRSAandMGF1`
- `SHA224withRSA/PSS`
- `SHA256withRSA`
- `SHA256withRSAandMGF1`
- `SHA256withRSA/PSS`
- `SHA384withRSA`
- `SHA384withRSAandMGF1`
- `SHA384withRSA/PSS`
- `SHA512withRSA`
- `SHA512withRSAandMGF1`
- `SHA512withRSA/PSS`

Types de signature ECDSA

- `NONEwithECDSA`
- `SHA1withECDSA`
- `SHA224withECDSA`
- `SHA256withECDSA`

- `SHA384withECDSA`
- `SHA512withECDSA`

Fonctions de résumé

La bibliothèque AWS CloudHSM logicielle pour Java prend en charge les résumés de messages suivants. Avec le SDK client 5, les données sont hachées localement dans le logiciel. Cela signifie qu'il n'y a aucune limite quant à la taille des données pouvant être hachées par le SDK.

- `SHA-1`
- `SHA-224`
- `SHA-256`
- `SHA-384`
- `SHA-512`

Fonctions du code d’authentification de message utilisant hash (HMAC).

La bibliothèque AWS CloudHSM logicielle pour Java prend en charge les algorithmes HMAC suivants.

- `HmacSHA1`(Taille maximale des données en octets : 16288)
- `HmacSHA224`(Taille maximale des données en octets : 16256)
- `HmacSHA256`(Taille maximale des données en octets : 16288)
- `HmacSHA384`(Taille maximale des données en octets : 16224)
- `HmacSHA512`(Taille maximale des données en octets : 16224)

Fonctions du code d'authentification des messages basé sur le chiffrement (CMAC)

Les CMAC (codes d'authentification de message basés sur le chiffrement) créent des codes d'authentification de message (MAC) à l'aide d'un chiffrement par blocs et d'une clé secrète. Ils diffèrent des HMAC en ce qu'ils utilisent une méthode de clé symétrique par blocs pour les MAC plutôt qu'une méthode de hachage.

La bibliothèque AWS CloudHSM logicielle pour Java prend en charge les algorithmes CMAC suivants.

- `AESCMAC`

Convertissez les clés en spécifications de clés à l'aide de fabriques de clés

Vous pouvez utiliser des usines de clés pour convertir les clés en spécifications clés. AWS CloudHSM possède deux types d'usines clés pour JCE :

SecretKeyFactory: Utilisé pour importer ou dériver des clés symétriques. En utilisant SecretKeyFactory, vous pouvez transmettre une clé prise en charge ou une clé prise en charge KeySpec pour importer ou dériver des clés symétriques. AWS CloudHSM Voici les spécifications prises en charge pour KeyFactory :

- Les KeySpecclasses suivantes SecretKeyFactory de la `generateSecret` méthode de For sont prises en charge :
  - KeyAttributesMappeut être utilisé pour importer des octets de clé avec des attributs supplémentaires en tant que clé CloudHSM. Un exemple peut être trouvé ici.
  - SecretKeySpecpeut être utilisé pour importer une spécification de clé symétrique en tant que clé CloudHSM.
  - AesCmacKdfParameterSpecpeut être utilisé pour dériver des clés symétriques à l'aide d'une autre clé CloudHSM AES.

> **Note**
>
> SecretKeyFactoryprend `translateKey` n'importe quelle clé qui implémente l'interface clé.

KeyFactory: Utilisé pour importer des clés asymétriques. En utilisant KeyFactory, vous pouvez transmettre une clé prise en charge KeySpec ou une clé asymétrique dans AWS CloudHSM laquelle vous pouvez importer une clé asymétrique. Pour plus d'informations, consultez les ressources suivantes :

- Selon KeyFactory la `generatePublic` méthode de For, les KeySpecclasses suivantes sont prises en charge :
- KeyAttributesMap CloudHSM pour RSA et EC, notamment : KeyTypes
  - KeyAttributesMap CloudHSM pour le public RSA et EC. KeyTypes Un exemple peut être trouvé ici

  - X509 EncodedKeySpec pour les clés publiques RSA et EC
  - Clé publique RSA PublicKeySpec pour RSA
  - EC PublicKeySpec pour EC Public Key
- Selon KeyFactory la `generatePrivate` méthode de For, les KeySpecclasses suivantes sont prises en charge :
- KeyAttributesMap CloudHSM pour RSA et EC, notamment : KeyTypes
  - KeyAttributesMap CloudHSM pour le public RSA et EC. KeyTypes Un exemple peut être trouvé ici
  - PKCS8 EncodedKeySpec pour les clés privées EC et RSA
  - Clé privée RSA PrivateCrtKeySpec pour RSA
  - EC PrivateKeySpec pour clé privée EC

KeyFactoryLa `translateKey` méthode de For prend en compte n'importe quelle clé qui implémente l'interface clé.

Annotations du mécanisme

[1] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

Attributs de clé Java pris en charge

Cette rubrique décrit comment utiliser une extension propriétaire pour le fournisseur JCE afin de définir les attributs de clé. Utilisez cette extension pour définir les attributs de clés pris en charge et leurs valeurs au cours des opérations suivantes :

- Génération de clés
- Importation de clés

Pour des exemples d'utilisation des attributs de clé, consultez the section called “Exemples de code”.

Rubriques

- Présentation des attributs
- Attributs pris en charge
- Définition des attributs pour une clé

Présentation des attributs

Les attributs de clé permettent de spécifier les actions autorisées sur les objets de type clé, y compris les clés publiques, privées ou secrètes. Vous définissez les attributs de clés et leurs valeurs lors des opérations de création d'objets de type clé.

Toutefois, l'extension JCE (Java Cryptography Extension) ne spécifie pas comment définir les valeurs des attributs de clé. Dès lors, la plupart des actions étaient autorisées par défaut. En revanche, la norme PKCS #11 définit un ensemble complet d'attributs avec des valeurs par défaut plus restrictives. À partir du fournisseur JCE 3.1, AWS CloudHSM fournit une extension propriétaire qui vous permet de définir des valeurs plus restrictives pour les attributs couramment utilisés.

Attributs pris en charge

Vous pouvez définir des valeurs pour les attributs répertoriés dans le tableau ci-dessous. Il est conseillé de spécifier uniquement des valeurs pour les attributs que vous souhaitez restreindre. Si vous ne spécifiez aucune valeur, AWS CloudHSM utilise la valeur par défaut spécifiée dans le tableau ci-dessous. Une cellule vide dans la colonne Default Value (Valeur par défaut) signale qu'aucune valeur par défaut n'est assignée à l'attribut.

Attribut	Valeur par défaut			Remarques
	Clé symétrique	Clé publique dans la paire de clés	Clé privée dans la paire de clés	
DECRYPT	TRUE		TRUE	True indique que vous pouvez utiliser la clé pour déchiffrer n'importe quelle mémoire tampon. Vous définissez généralement cette valeur sur FALSE pour une clé dont l'attribut

Attribut	Valeur par défaut			Remarques
	Clé symétrique	Clé publique dans la paire de clés	Clé privée dans la paire de clés	
				WRAP est défini sur true.
DERIVE				Permet d'utiliser une clé pour dériver d'autres clés.
ENCRYPT	TRUE	TRUE		True indique que vous pouvez utiliser la clé pour chiffrer n'importe quelle mémoire tampon.
EXTRACTABLE	TRUE		TRUE	True indique que vous pouvez exporter cette clé hors du HSM.
ID				Valeur définie par l'utilisateur utilisée pour identifier la clé.
KEY_TYPE				Utilisé pour identifier le type de clé (AES, DESede, secret générique, EC ou RSA).

Attribut	Valeur par défaut			Remarques
	Clé symétrique	Clé publique dans la paire de clés	Clé privée dans la paire de clés	
LABEL				Chaîne définie par l'utilisateur qui vous permet d'identifier facilement les clés de votre HSM. Conformément aux meilleures pratiques, utilisez une étiquette unique pour chaque clé afin de pouvoir la retrouver plus facilement par la suite.
LOCAL				Indique une clé générée par le HSM.
OBJECT_CLASS				Utilisé pour identifier la classe d'objet d'une clé (SecretKey, PublicKey ou PrivateKey).

Attribut	Valeur par défaut			Remarques
	Clé symétrique	Clé publique dans la paire de clés	Clé privée dans la paire de clés	
PRIVATE	TRUE	TRUE	TRUE	True indique qu'un utilisateur ne peut pas accéder à la clé tant qu'il n'est pas authentifié. Pour plus de clarté, les utilisateurs ne peuvent accéder à aucune clé AWS CloudHSM tant qu'ils ne sont pas authentifiés, même si cet attribut est défini sur FALSE.

Attribut	Valeur par défaut			Remarques
	Clé symétrique	Clé publique dans la paire de clés	Clé privée dans la paire de clés	
SIGN	TRUE		TRUE	True indique que vous pouvez utiliser la clé pour signer le résumé d'un message. Cette valeur est généralement définie sur FALSE pour les clés publiques et les clés privées que vous avez archivées.
SIZE				Attribut qui définit la taille d'une clé. Pour plus de détails sur les tailles de clé prises en charge, reportez-vous à la section Mécanismes pris en charge pour le SDK client 5.

Attribut	Valeur par défaut			Remarques
	Clé symétrique	Clé publique dans la paire de clés	Clé privée dans la paire de clés	
TOKEN	FALSE	FALSE	FALSE	Clé permanente répliquée sur tous les HSM du cluster et incluse dans les sauvegardes. TOKEN = FALSE implique une clé éphémère qui est automatiquement effacée lorsque la connexion au HSM est interrompue.
UNWRAP	TRUE		TRUE	True indique que vous pouvez utiliser la clé pour décencapsuler (importer) une autre clé.

Attribut	Valeur par défaut			Remarques
	Clé symétrique	Clé publique dans la paire de clés	Clé privée dans la paire de clés	
VERIFY	TRUE	TRUE		True indique que vous pouvez utiliser la clé pour valider une signature . Cette valeur est généralement définie sur FALSE pour les clés privées.
WRAP	TRUE	TRUE		True indique que vous pouvez utiliser la clé pour encapsuler une autre clé. Vous définissez généralement ce paramètre sur FALSE pour les clés privées.

Attribut	Valeur par défaut			Remarques
	Clé symétrique	Clé publique dans la paire de clés	Clé privée dans la paire de clés	
WRAP_WITH_TRUSTED	FALSE		FALSE	True indique qu'une clé ne peut être encapsulée et désencapsulée qu'avec des clés pour lesquelles l'attribut TRUSTED est défini sur true. Une fois que l'attribut WRAP_WITH_TRUSTED est défini sur true, cet attribut est en lecture seule et ne peut pas être défini sur false. Pour en savoir plus sur l'encapsulage de confiance, consultez la section Utilisation de clés de confiance pour contrôler le désencapsulage des clés.

> **Note**
>
> Vous bénéficiez d'une prise en charge plus large des attributs dans la bibliothèque PKCS #11. Pour plus d'informations, consultez Attributs PKCS #11 pris en charge.

Définition des attributs pour une clé

`KeyAttributesMap` est un objet de type Java Map que vous pouvez utiliser pour définir des valeurs d'attribut pour les objets de type clé. Les méthodes de `KeyAttributesMap` fonctionne d'une manière similaire à celles utilisées pour la manipulation de Java Map.

Pour définir des valeurs personnalisées pour les attributs, deux options s'offrent à vous :

- Utiliser les méthodes répertoriées dans le tableau suivant
- Utiliser les modèles de générateur illustrés plus loin dans ce document

Les objets de mappage d'attribut prennent en charge les méthodes suivantes pour définir les attributs :

Opération	Valeur renvoyée	Méthode **`KeyAttributesMap`**
Obtenir la valeur d'un attribut pour une clé existante	Objet (contenant la valeur) ou null	get(keyAttribute)
Renseigner la valeur d'un attribut de clé	Valeur précédente associée à l'attribut de clé, ou null en l'absence de mappage pour un attribut de clé	put(keyAttribute, value)
Renseigner des valeurs pour plusieurs attributs de clés	N/A	Tout mettre () keyAttributesMap
Supprimer une paire clé-valeur du mappage d'attributs	Valeur précédente associée à l'attribut de clé, ou null en l'absence de mappage pour un attribut de clé	remove(keyAttribute)

> **Note**
>
> Tous les attributs que vous ne spécifiez pas explicitement utilisent les valeurs par défaut répertoriées dans le tableau précédent, dans the section called "Attributs pris en charge".

Définition des attributs pour une paire de clés

Utilisez la classe Java `KeyPairAttributesMap` pour gérer les attributs d'une paire de clés. `KeyPairAttributesMap` encapsule deux objets `KeyAttributesMap` : un pour une clé publique et un pour une clé privée.

Pour définir des attributs individuels de la clé publique et la clé privée séparément, vous pouvez utiliser la méthode `put()` sur l'objet mappé `KeyAttributes` correspondant à cette clé. Choisissez la méthode `getPublic()` pour récupérer le mappage d'attributs de la clé publique, et utilisez `getPrivate()` pour récupérer le mappage d'attributs de la clé privée. Renseignez la valeur de plusieurs attributs de clés pour les paires de clés publiques et privées en utilisant `putAll()` avec le mappage d'attributs d'une paire de clés comme argument.

Exemples de code pour la bibliothèque de AWS CloudHSM logiciels pour Java

Prérequis

Avant d'exécuter les exemples, vous devez configurer votre environnement :

- Installez et configurez le fournisseur d'extension cryptographique Java (JCE).
- Définissez un nom d'utilisateur et un mot de passe HSM valides. Les autorisations de l'utilisateur de chiffrement (CU) sont suffisantes pour ces tâches. Votre application utilise ces informations d'identification pour se connecter au HSM dans chaque exemple.
- Décidez comment fournir les informations d'identification au fournisseur JCE.

Exemples de code

Les exemples de code suivants vous montrent comment utiliser le fournisseur AWS CloudHSM JCE pour effectuer des tâches de base. D'autres exemples de code sont disponibles sur GitHub.

- Se connecter à un HSM

- Gérer les clés
- Générer des clés symétriques
- Générer des clés asymétriques
- Chiffrer et déchiffrer avec AES-GCM
- Chiffrer et déchiffrer avec AES-CTR
- Chiffrer et déchiffrer avec DESede-ECB[1]
- Signer et vérifier avec des clés RSA
- Signer et vérifier avec des clés EC
- Utiliser les attributs de clé pris en charge
- Utiliser le magasin de clés CloudHSM

[1] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

AWS CloudHSM Javadocs, fournisseur JCE

Utilisez le fournisseur JCE Javadocs pour obtenir des informations d'utilisation sur les types et méthodes Java définis dans le SDK AWS CloudHSM JCE. Pour télécharger la dernière version de Javadocs pour AWS CloudHSM, consultez la Dernière parution section sur la page Téléchargements.

Vous pouvez importer des Javadocs dans un environnement de développement intégré (IDE) ou les afficher dans un navigateur Web.

Utilisation de la classe AWS CloudHSM KeyStore Java

La AWS CloudHSM `KeyStore` classe fournit un magasin de clés PKCS12 à usage spécial. Ce magasin de clés peut stocker des certificats avec vos données clés et les mettre en corrélation avec les données clés stockées sur AWS CloudHSM. La AWS CloudHSM `KeyStore` classe implémente l'interface du fournisseur de `KeyStore` services (SPI) de l'extension de cryptographie Java (JCE). Pour plus d'informations sur l'utilisation`KeyStore`, consultez la section Classe KeyStore.

> **Note**
>
> Les certificats étant des informations publiques, il n'est AWS CloudHSM pas possible de stocker des certificats sur des HSM afin de maximiser la capacité de stockage des clés cryptographiques.

Choisir le magasin de clés approprié

Le fournisseur d'extension cryptographique AWS CloudHSM Java (JCE) propose un AWS CloudHSM à usage spécifique. KeyStore La AWS CloudHSM `KeyStore` classe prend en charge le transfert des opérations clés vers le HSM, le stockage local des certificats et les opérations basées sur les certificats.

Chargez le CloudHSM spécialisé comme suit : KeyStore

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

Initialisation AWS CloudHSM KeyStore

Connectez-vous de AWS CloudHSM KeyStore la même manière que vous vous connectez au fournisseur JCE. Vous pouvez utiliser des variables d'environnement ou le fichier de propriétés du système, et vous devez vous connecter avant de commencer à utiliser CloudHSM KeyStore. Pour obtenir un exemple de connexion à un HSM à l'aide du fournisseur JCE, veuillez consulter Connexion à un HSM.

Si vous le souhaitez, vous pouvez spécifier un mot de passe pour chiffrer le fichier PKCS12 local qui contient les données du magasin de clés. Lorsque vous créez le AWS CloudHSM Keystore, vous définissez le mot de passe et vous le fournissez lorsque vous utilisez les méthodes load, set et get.

Instanciez un nouvel objet KeyStore CloudHSM comme suit :

```
ks.load(null, null);
```

Écrivez les données du keystore dans un fichier à l'aide de la méthode `store`. À partir de ce moment, vous pouvez charger le keystore existant en utilisant la méthode `load` avec le fichier source et le mot de passe comme suit :

```
ks.load(inputStream, password);
```

En utilisant AWS CloudHSM KeyStore

AWS CloudHSM KeyStore est conforme aux KeyStore spécifications de la classe JCE et fournit les fonctions suivantes.

- `load`

Charge le magasin de clés à partir du flux d'entrée donné. Si un mot de passe a été défini lors de l'enregistrement du magasin de clés, ce même mot de passe doit être fourni pour que le chargement réussisse. Définissez les deux paramètres sur null pour initialiser un nouveau magasin de clés vide.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
ks.load(inputStream, password);
```

- `aliases`

  Renvoie une énumération des noms d'alias de toutes les entrées de l'instance de magasin de clés donnée. Les résultats incluent les objets stockés localement dans le fichier PKCS12 et les objets résidant sur le HSM.

  Exemple de code :

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();) {
    String label = entry.nextElement();
    System.out.println(label);
}
```

- `containsalias`

  Renvoie true si le magasin de clés a accès à au moins un objet avec l'alias spécifié. Le magasin de clés vérifie les objets stockés localement dans le fichier PKCS12 et les objets résidant sur le HSM.

- `deleteEntry`

  Supprime une entrée de certificat du fichier PKCS12 local. La suppression de données clés stockées dans un HSM n'est pas prise en charge à l'aide du AWS CloudHSM KeyStore. Vous pouvez supprimer des clés en utilisant la méthode `destroy` de l'interface Destroyable.

```
((Destroyable) key).destroy();
```

- `getCertificate`

  Renvoie le certificat associé à un alias le cas échéant. Si l'alias n'existe pas ou fait référence à un objet qui n'est pas un certificat, la fonction renvoie NULL.

```
KeyStore ks = KeyStore.getInstance("CloudHSM");
```

```
Certificate cert = ks.getCertificate(alias);
```

- `getCertificateAlias`

  Renvoie le nom (alias) de la première entrée de magasin de clés dont les données correspondent au certificat donné.

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  String alias = ks.getCertificateAlias(cert);
  ```

- `getCertificateChain`

  Renvoie la chaîne de certificats associée à l'alias donné. Si l'alias n'existe pas ou fait référence à un objet qui n'est pas un certificat, la fonction renvoie NULL.

- `getCreationDate`

  Renvoie la date de création de l'entrée identifiée par l'alias donné. Si aucune date de création n'est disponible, la fonction renvoie la date à laquelle le certificat est devenu valide.

- `getKey`

  GetKey est transmis au HSM et renvoie un objet clé correspondant à l'étiquette donnée. Comme il interroge `getKey` directement le HSM, il peut être utilisé pour n'importe quelle clé du HSM, qu'elle ait été générée ou non par le. KeyStore

  ```
  Key key = ks.getKey(keyLabel, null);
  ```

- `isCertificateEntry`

  Vérifie si l'entrée avec l'alias donné représente une entrée de certificat.

- `isKeyEntry`

  Vérifie si l'entrée avec l'alias donné représente une entrée de clé. L'action recherche l'alias dans le fichier PKCS12 et le HSM.

- `setCertificateEntry`

  Affecte le certificat donné à l'alias donné. Si l'alias donné est déjà utilisé pour identifier une clé ou un certificat, une `KeyStoreException` est levée. Vous pouvez utiliser le code JCE pour obtenir l'objet clé, puis utiliser la KeyStore `SetKeyEntry` méthode pour associer le certificat à la clé.

- `setKeyEntry` avec la clé `byte[]`

Cette API n'est actuellement pas prise en charge par le SDK client 5.

- `setKeyEntry` avec l'objet `Key`

  Affecte la clé donnée à l'alias donné et la stocke dans le HSM. Si la clé n'existe pas déjà dans le HSM, elle sera importée dans le HSM en tant que clé de session extractible.

  Si l'objet `Key` est de type `PrivateKey`, il doit être accompagné d'une chaîne de certificats correspondante.

  Si l'alias existe déjà, l'appel `SetKeyEntry` lance un `KeyStoreException` et empêche la clé d'être écrasée. Si la clé doit être écrasée, utilisez KMU ou JCE à cet effet.

- `engineSize`

  Renvoie le nombre d'entrées dans le keystore.

- `store`

  Stocke le magasin de clés dans le flux de sortie donné sous la forme d'un fichier PKCS12 et le sécurise avec le mot de passe donné. En outre, il persiste toutes les clés chargées (qui sont définies en utilisant des appels `setKey`).

Migrez votre fournisseur JCE du SDK client 3 vers le SDK client 5

Utilisez cette rubrique pour migrer votre fournisseur JCE du SDK client 3 vers le SDK client 5. Pour connaître les avantages de la migration, voirAvantages du SDK client 5.

Dans AWS CloudHSM, les applications client exécutent des opérations cryptographiques à l'aide du kit de développement logiciel (SDK) AWS CloudHSM client. Le SDK client 5 est le SDK principal auquel de nouvelles fonctionnalités et un support de plateforme continuent d'être ajoutés.

Le fournisseur JCE du SDK client 3 utilise des classes personnalisées et des API qui ne font pas partie de la spécification JCE standard. Le SDK client 5 pour le fournisseur JCE est conforme à la spécification JCE et est rétroincompatible avec le SDK client 3 dans certains domaines. Les applications du client peuvent nécessiter des modifications dans le cadre de la migration vers le SDK client 5. Cette section décrit les modifications requises pour une migration réussie.

Pour consulter les instructions de migration pour tous les fournisseurs, voirMigration du SDK client 3 vers le SDK client 5.

Rubriques

Préparez-vous en faisant face aux changements les plus importants

Passez en revue ces modifications majeures et mettez à jour votre application dans votre environnement de développement en conséquence.

La classe et le nom du fournisseur ont changé

Qu'est-ce qui a changé	Ce que c'était dans le SDK client 3	Qu'est-ce que c'est dans le SDK client 5	Exemple
Classe et nom du fournisseur	La classe de fournisseur JCE du SDK client 3 est appelée `CaviumProvider` et porte le nom du fournisseur. `Cavium`	Dans le SDK client 5, la classe Provider est appelée `CloudHsmProvider` et porte le nom `CloudHSM` du fournisseur.	Un exemple d'initialisation de l'`CloudHsmProvider` objet est disponible dans le référentiel AWS CloudHSM GitHub d'exemples.

La connexion explicite a changé, la connexion implicite n'a pas changé

Qu'est-ce qui a changé	Ce que c'était dans le SDK client 3	Qu'est-ce que c'est dans le SDK client 5	Exemple
Login explicite	Le SDK client 3 utilise la `LoginManager` classe pour une connexion [1] explicite.	Dans le SDK client 5, le `CloudHSM` fournisseur implémente `AuthProvider` une connexion	Pour un exemple d'utilisation de la connexion explicite avec le SDK client 5, consultez l' LoginRunner exemple

Qu'est-ce qui a changé	Ce que c'était dans le SDK client 3	Qu'est-ce que c'est dans le SDK client 5	Exemple
		explicite. `AuthProvider` est une classe Java standard qui suit la méthode idiomatique de Java pour se connecter à un fournisseur. Grâce à la gestion améliorée de l'état de connexion dans le SDK client 5, les applications n'ont plus besoin de surveiller et d'effectuer des connexions lors des [2]reconnexions.	dans le référentiel d'exemples GitHub AWS CloudHSM.
Login implicite	Aucune modification n'est requise pour la connexion implicite. Le même fichier de propriétés et toutes les variables d'environnement continueront de fonctionner pour la connexion implicite lors de la migration du SDK client 3 vers le SDK client 5.		Pour un exemple d'utilisation de la connexion implicite avec le SDK client 5, consultez l'LoginRunner exemple dans le référentiel AWS CloudHSM GitHub d'exemples.

- [1] Extrait de code du SDK client 3 :

```
LoginManager lm = LoginManager.getInstance();

lm.login(partition, user, pass);
```

- [2] Extrait de code du SDK client 5 :

```
// 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);
```

Pour un exemple d'utilisation de la connexion explicite avec le SDK client 5, consultez l'LoginRunner exemple dans le référentiel AWS CloudHSM GitHub d'exemples.

La génération de clés a changé

Qu'est-ce qui a changé	Ce que c'était dans le SDK client 3	Qu'est-ce que c'est dans le SDK client 5	Exemple
Génération de clés	Dans le SDK client 3, `Cavium[Key-type]AlgorithmParameterSpec` est utilisé pour spécifier les paramètres de génération de clés. Pour un extrait de code, voir note de bas de page. 1	Dans le SDK client 5, `KeyAttributesMap` est utilisé pour spécifier les attributs de génération de clés. Pour un extrait de code, voir note de bas de page. 2	Pour un exemple expliquant comment `KeyAttributesMap` générer une clé symétrique, consultez l'exemple dans le référentiel SymmetricKeys d'exemples AWS CloudHSM Github.
Génération de paires de clés	Dans le SDK client 3, `Cavium[Key-type]Algorithmparameterspec` est utilisé pour spécifier les paramètres de génération de paires de clés. Pour un	Dans le SDK client 5, `KeyPairAttributesMap` est utilisé pour spécifier ces paramètres. Pour un extrait de code, voir note de bas de page. 4	Pour un exemple sur la façon de `KeyAttributesMap` générer une clé asymétrique, consultez l'AsymmetricKeys exemple dans le référentiel AWS

Qu'est-ce qui a changé	Ce que c'était dans le SDK client 3	Qu'est-ce que c'est dans le SDK client 5	Exemple
	extrait de code, voir note de bas de page. [3]		CloudHSM GitHub d'échantillons.

- [1] Extrait de code de génération de clé du SDK client 3 :

```
KeyGenerator keyGen = KeyGenerator.getInstance("AES", "Cavium");
CaviumAESKeyGenParameterSpec aesSpec = new CaviumAESKeyGenParameterSpec(
keySizeInBits,
keyLabel,
isExtractable,
isPersistent);
keyGen.init(aesSpec);
SecretKey aesKey = keyGen.generateKey();
```

- [2] Extrait de code de génération de clé du SDK client 5 :

```
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] Extrait de code de génération de paires de clés du SDK client 3 :

```
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] Extrait de code de génération de 5 paires de clés du SDK client :

```
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();
```

La recherche, la suppression et le référencement des clés ont changé

Trouver une clé déjà générée AWS CloudHSM implique d'utiliser le KeyStore. Le SDK client 3 est de deux KeyStore types : `Cavium` et`CloudHSM`. Le SDK client 5 n'a qu'un seul KeyStore type :`CloudHSM`.

Le passage du `Cavium` KeyStore à `CloudHSM` KeyStore nécessite un changement de KeyStore type. En outre, le SDK client 3 utilise des poignées de touches pour référencer les clés, tandis que le SDK client 5 utilise des étiquettes de touches. Les modifications de comportement qui en résultent sont répertoriées ci-dessous.

Qu'est-ce qui a changé	Ce que c'était dans le SDK client 3	Qu'est-ce que c'est dans le SDK client 5	Exemple
Principales références	Avec le SDK client 3, les applications utilisent des libellés ou des descripteurs de touches pour référencer les clés dans le HSM. Ils utilisent des étiquettes KeyStore pour trouver une clé, ou ils utilisent des poignées pour créer des `CaviumKey` objets.	Dans le SDK client 5, les applications peuvent utiliser le Utilisation de la classe AWS CloudHSM KeyStore Java pour rechercher des clés par étiquette. Pour rechercher les clés par poignée, utilisez la touche AWS CloudHSM `KeyStoreWithAttributes` avec AWS CloudHSM `KeyRefereneSpec` .	
Recherche de plusieurs entrées	Lorsque vous recherchez une clé à l'aide de `getEntrygetKey`, ou `getCertificate` dans des scénarios où plusieurs éléments répondant aux mêmes critères existent dans le `Cavium` KeyStore, seule la première	Avec le AWS CloudHSM `KeyStoreetKeyStoreWithAttributes` , ce même scénario entraîner a le lancement d'une exception. Pour résoudre ce problème, il est recommandé de définir des étiquettes	

Qu'est-ce qui a changé	Ce que c'était dans le SDK client 3	Qu'est-ce que c'est dans le SDK client 5	Exemple
	entrée trouvée sera renvoyée.	uniques pour les clés à l'aide de la clé set-attribute commande de la CLI CloudHSM. Ou `KeyStoreWithAttributes#getKeys` utilisez-le pour renvoyer toutes les clés correspondant aux critères.	

Qu'est-ce qui a changé	Ce que c'était dans le SDK client 3	Qu'est-ce que c'est dans le SDK client 5	Exemple
Trouvez toutes les clés	Dans le SDK client 3, il est possible de trouver toutes les clés du HSM à l'aide de. `Util.findAllKeys()`	Le SDK client 5 simplifie et rend la recherche de clés plus efficace en utilisant la `KeyStoreWithAttributes` classe. Dans la mesure du possible, mettez vos clés en cache pour minimiser le temps de latence. Pour plus d'informations, consultez Gérez efficacement les clés de votre application. Lorsque vous devez récupérer toutes les clés du HSM, utilisez-les `KeyStoreWithAttributes#getKeys` avec une clé vide. `KeyAttributesMap`	Un exemple utilisant la `KeyStoreWithAttributes` classe pour trouver une clé est disponible dans le référentiel d'exemples AWS CloudHSM Github et un extrait de code est affiché dans. 1

Qu'est-ce qui a changé	Ce que c'était dans le SDK client 3	Qu'est-ce que c'est dans le SDK client 5	Exemple
Suppression de la clé	Le SDK client 3 est utilisé `Util.deleteKey()` pour supprimer une clé.	L'`Key`objet du SDK client 5 implémente l'`Destroyable` interface qui permet de supprimer des clés à l'aide de la `destroy()` méthode de cette interface.	Un exemple de code illustrant la fonctionnalité de suppression de la clé se trouve dans le référentiel d'exemples de CloudHSM Github. Un exemple d'extrait de code pour chaque SDK est présenté dans. 2

- [1] un extrait est affiché ci-dessous :

```
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] Suppression d'une clé dans le SDK client 3 :

```
Util.deleteKey(key);
```

  Suppression d'une clé dans le SDK client 5 :

```
((Destroyable) key).destroy();
```

Les opérations de décompression du chiffrement ont changé, les autres opérations de chiffrement n'ont pas changé

> **Note**
>
> Aucune modification n'est requise pour les opérations de chiffrement/déchiffrement/encapsulation par chiffrement.

Les opérations de désencapsulage nécessitent le remplacement de la `CaviumUnwrapParameterSpec` classe du SDK client 3 par l'une des classes suivantes, spécifique aux opérations cryptographiques répertoriées.

- `GCMUnwrapKeySpec`pour `AES/GCM/NoPadding` déballer
- `IvUnwrapKeySpec`pour `AESWrap unwrap` et `AES/CBC/NoPadding unwrap`
- `OAEPUnwrapKeySpec` pour `RSA OAEP unwrap`

Exemple d'extrait pour : `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);
```

Les opérations de signature n'ont pas changé

Aucune modification n'est requise pour les opérations de signature.

Migrer vers le SDK client 5

Suivez les instructions de cette section pour migrer du SDK client 3 vers le SDK client 5.

> **Note**
>
> Amazon Linux, Ubuntu 16.04, Ubuntu 18.04, CentOS 6, CentOS 8 et RHEL 6 ne sont actuellement pas pris en charge avec le SDK client 5. Si vous utilisez actuellement l'une de ces plateformes avec le SDK client 3, vous devrez en choisir une autre lors de la migration vers le SDK client 5.

1. Désinstallez le fournisseur JCE pour le SDK client 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. Désinstallez le démon client pour le SDK client 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**
>
> Les configurations personnalisées doivent être réactivées.

3. Installez le fournisseur JCE du SDK client en suivant les étapes décrites dans. Installation et utilisation du fournisseur AWS CloudHSM JCE pour le SDK client 5
4. Le SDK client 5 introduit un nouveau format de fichier de configuration et un outil d'amorçage en ligne de commande. Pour démarrer votre fournisseur Client SDK 5 JCE, suivez les instructions répertoriées dans le guide de l'utilisateur ci-dessous. Amorcez le SDK client
5. Dans votre environnement de développement, testez votre application. Mettez à jour votre code existant pour corriger les modifications importantes avant votre migration finale.

Rubriques en relation

- Les meilleures pratiques pour AWS CloudHSM

Configurations avancées pour JCE

Le fournisseur AWS CloudHSM JCE inclut les configurations avancées suivantes, qui ne font pas partie des configurations générales utilisées par la plupart des clients.

- Connexion à plusieurs clusters

- Extraction de clés à l'aide de JCE

- Réessayer la configuration pour JCE

Connexion à plusieurs clusters avec le fournisseur JCE

Cette configuration permet à une seule instance client de communiquer avec plusieurs clusters. Par rapport au fait qu'une seule instance communique uniquement avec un seul cluster, cette fonctionnalité peut permettre de réaliser des économies dans certains cas d'utilisation. La `CloudHsmProvider` classe est AWS CloudHSM l'implémentation de la classe Provider de Java Security. Chaque instance de cette classe représente une connexion à l'ensemble de votre AWS CloudHSM cluster. Vous instanciez cette classe et vous l'ajoutez à la liste des fournisseurs de Java Security afin de pouvoir interagir avec elle à l'aide des classes JCE standard.

L'exemple suivant instancie cette classe et l'ajoute à la liste des fournisseurs Java Security :

```
if (Security.getProvider(CloudHsmProvider.PROVIDER_NAME) == null) {
    Security.addProvider(new CloudHsmProvider());
}
```

Configuration **`CloudHsmProvider`**

`CloudHsmProvider` peut être configuré de deux façons :

1. Configuration avec un fichier (configuration par défaut)
2. Configuration à l'aide du code

Configuration avec un fichier (configuration par défaut)

Lorsque vous instanciez `CloudHsmProvider` en utilisant le constructeur par défaut, il recherche par défaut le fichier de configuration dans le chemin `/opt/cloudhsm/etc/cloudhsm-jce.cfg` sous Linux. Ce fichier de configuration peut être configuré à l'aide du paramètre `configure-jce`.

Un objet créé à l'aide du constructeur par défaut utilisera le nom du fournisseur CloudHSM par défaut `CloudHSM`. Le nom du fournisseur est utile pour interagir avec JCE afin de lui indiquer quel fournisseur utiliser pour diverses opérations. Voici un exemple d'utilisation du nom du fournisseur CloudHSM pour l'opération Cipher :

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHSM");
```

Configuration à l'aide du code

À partir de la version 5.8.0 du SDK client, vous pouvez également le configurer `CloudHsmProvider` à l'aide de code Java. Pour cela, utilisez un objet de classe `CloudHsmProviderConfig`. Vous pouvez générer cet objet à l'aide de `CloudHsmProviderConfigBuilder`.

`CloudHsmProvider` possède un autre constructeur qui prend l'objet `CloudHsmProviderConfig`, comme le montre l'exemple suivant.

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);
```

Dans cet exemple, le nom du fournisseur JCE est `CloudHsmCluster1`. C'est le nom que l'application peut ensuite utiliser pour interagir avec JCE :

Example

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", "CloudHsmCluster1");
```

Les applications peuvent également utiliser l'objet fournisseur créé ci-dessus pour indiquer à JCE d'utiliser ce fournisseur pour l'opération :

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider);
```

Si aucun identifiant unique n'est spécifié avec la méthode `withClusterUniqueIdentifier`, un nom de fournisseur généré de manière aléatoire est créé pour vous. Pour obtenir cet identifiant généré de manière aléatoire, les applications peuvent appeler `provider.getName()`.

Connexion à plusieurs clusters

Comme indiqué ci-dessus, chaque `CloudHsmProvider` représente une connexion à votre cluster CloudHSM. Si vous souhaitez communiquer avec un autre cluster à partir de la même application, vous pouvez créer un autre objet `CloudHsmProvider` avec des configurations pour votre autre cluster et vous pouvez interagir avec cet autre cluster en utilisant l'objet fournisseur ou en utilisant le nom du fournisseur, comme indiqué dans l'exemple suivant.

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);
}
```

Une fois que vous avez configuré les deux fournisseurs (les deux clusters) ci-dessus, vous pouvez interagir avec eux en utilisant l'objet fournisseur ou en utilisant le nom du fournisseur.

Sur la base de cet exemple qui montre comment parler à`cluster1`, vous pouvez utiliser l'exemple suivant pour une opération NoPadding AES/GCM/ :

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider1);
```

Et dans la même application pour générer une clé « AES » sur le deuxième cluster en utilisant le nom du fournisseur, vous pouvez également utiliser l'exemple suivant :

```
Cipher cipher = Cipher.getInstance("AES/GCM/NoPadding", provider2.getName());
```

Commandes de nouvelle tentative pour JCE

Le SDK client 5.8.0 et versions ultérieures disposent d'une stratégie de relance automatique intégrée qui permet de réessayer les opérations limitées par HSM du côté client. Lorsqu'un HSM limite les opérations parce qu'il est trop occupé à effectuer les opérations précédentes et qu'il ne peut pas traiter plus de demandes, les SDK clients tentent de retenter les opérations limitées jusqu'à 3 fois tout en reculant de manière exponentielle. Cette stratégie de nouvelle tentative automatique peut être réglée sur l'un des deux modes suivants : désactivé et standard.

- désactivé : le SDK client n'exécutera aucune stratégie de nouvelle tentative pour les opérations limitées effectuées par le HSM.
- standard : il s'agit du mode par défaut pour le SDK client 5.8.0 et versions ultérieures. Dans ce mode, les SDK clients réessaieront automatiquement les opérations limitées en reculant de manière exponentielle.

Pour plus d'informations, consultez Limitation du HSM.

Définir des commandes de nouvelle tentative sur le mode désactivé

Linux

Pour définir les commandes de nouvelle tentative sur off pour le SDK client 5 sous Linux

- Vous pouvez utiliser la commande suivante pour définir une nouvelle tentative de configuration sur le mode off :

```
$ sudo /opt/cloudhsm/bin/configure-jce --default-retry-mode off
```

Windows

Pour définir les commandes de nouvelle tentative sur off pour le SDK client 5 sous Windows

- Vous pouvez utiliser la commande suivante pour définir une nouvelle tentative de configuration sur le modeoff :

```
C:\Program Files\Amazon\CloudHSM\bin\ .\configure-jce.exe --default-retry-mode
 off
```

Extraction de clés à l'aide de JCE

L'extension de cryptographie Java (JCE) utilise une architecture qui permet de connecter différentes implémentations de cryptographie. AWS CloudHSM fournit un tel fournisseur JCE qui télécharge les opérations cryptographiques vers le HSM. Pour que la plupart des autres fournisseurs JCE puissent utiliser des clés stockées dans AWS CloudHSM, ils doivent extraire les octets clés de vos HSM en texte clair dans la mémoire de votre machine pour leur utilisation. Les HSM autorisent généralement l'extraction des clés uniquement sous forme d'objets encapsulés, et non sous forme de texte clair. Toutefois, pour prendre en charge les cas d'utilisation liés à l'intégration entre fournisseurs, AWS CloudHSM autorise une option de configuration optionnelle pour permettre l'extraction des octets clés en clair.

> **Important**
>
> JCE décharge les opérations AWS CloudHSM chaque fois que le fournisseur AWS CloudHSM est spécifié ou AWS CloudHSM qu'un objet clé est utilisé. Il n'est pas nécessaire d'extraire les clés en clair si vous vous attendez à ce que votre opération se déroule dans le HSM. L'extraction de clés en texte clair n'est nécessaire que lorsque votre application ne peut pas utiliser de mécanismes sécurisés tels que l'encapsulage et le désencapsulage d'une clé en raison de restrictions imposées par une bibliothèque tierce ou un fournisseur JCE.

Le fournisseur AWS CloudHSM JCE permet l'extraction de clés publiques pour fonctionner avec des fournisseurs JCE externes par défaut. Les méthodes suivantes sont toujours autorisées :

Classe	Méthode	Format (getEncoded)
EcPublicKey	getEncoded()	X.509
	getW()	N/A
RSA PublicKey	getEncoded()	X.509
	getPublicExponent()	N/A
CloudHsmRsaPrivateCrtKey	getPublicExponent()	N/A

Le fournisseur AWS CloudHSM JCE n'autorise pas l'extraction d'octets clés en clair pour les clés privées ou secrètes par défaut. Si votre cas d'utilisation l'exige, vous pouvez activer l'extraction d'octets de clé en clair pour les clés privées ou secrètes dans les conditions suivantes :

1. L'attribut `EXTRACTABLE` des clés privées et secrètes est défini sur true.
   - Par défaut, l'attribut `EXTRACTABLE` des clés privées et secrètes est défini sur true. Les clés `EXTRACTABLE` sont des clés dont l'exportation hors du HSM est autorisée. Pour plus d'informations, consultez Attributs Java pris en charge pour le Client SDK 5.
2. L'attribut `WRAP_WITH_TRUSTED` pour les clés privées et secrètes est défini sur false.
   - `getEncoded`, `getPrivateExponent`, et `getS` ne peuvent pas être utilisés avec des clés privées qui ne peuvent pas être exportées en clair. `WRAP_WITH_TRUSTED` n'autorise pas l'exportation de vos clés privées hors du HSM en clair. Pour plus d'informations, consultez Utilisation de clés fiables pour contrôler le désencapsulage des clés.

Permettre au fournisseur AWS CloudHSM JCE d'extraire les clés privées secrètes de AWS CloudHSM

> **Important**
>
> Cette modification de configuration permet d'extraire à partir de tous les octets de clé `EXTRACTABLE` en clair de votre cluster HSM. Pour une meilleure sécurité, envisagez d'utiliser des méthodes d'encapsulage des clés pour extraire la clé du HSM en toute sécurité. Cela empêche l'extraction involontaire de vos octets de clé du HSM.

1. Utilisez les commandes suivantes pour activer l'extraction de vos clés privées ou secrètes dans JCE :

   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. Une fois que vous avez activé l'extraction de votre clé en clair, les méthodes suivantes sont activées pour extraire les clés privées en mémoire.

Classe	Méthode	Format (getEncoded)
Clé	getEncoded()	RAW
CE PrivateKey	getEncoded()	PKCS #8
	getS()	N/A
RSA PrivateCrtKey	getEncoded()	X.509
	getPrivateExponent()	N/A
	getPrimeP()	N/A
	getPrimeQ()	N/A
	getPrimeExponentP ()	N/A
	getPrimeExponentQ ()	N/A
	getCrtCoefficient()	N/A

Si vous souhaitez rétablir le comportement par défaut et empêcher JCE d'exporter les clés en clair, exécutez la commande suivante :

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
```

API de cryptographie : fournisseurs de stockage de clés (KSP) et de nouvelle génération (CNG) pour Microsoft Windows

Le AWS CloudHSM client pour Windows inclut les fournisseurs CNG et KSP. Actuellement, seul le SDK client 3 prend en charge les fournisseurs CNG et KSP.

Les fournisseurs de stockage de clés (KSP) permettent le stockage et l'extraction de clés. Par exemple, si vous ajoutez le rôle de services de certificats Microsoft Active Directory (AD CS) à votre serveur Windows et choisissez de créer une nouvelle clé privée pour votre autorité de certification (CA), vous pouvez choisir le fournisseur de stockage de clés qui gèrera le stockage. Lorsque vous configurez le rôle AD CS, vous pouvez choisir ce KSP. Pour plus d'informations, consultez Créer une CA Windows Server.

API Cryptography : Next Generation (CNG) est une API de chiffrement spécifique au système d'exploitation Microsoft Windows. CNG permet aux développeurs d'utiliser les techniques de chiffrement pour sécuriser les applications Windows. À un niveau élevé, la AWS CloudHSM mise en œuvre du GNC fournit les fonctionnalités suivantes :

- Primitives de chiffrement - vous permet d'effectuer des opérations de chiffrement fondamentales.
- Importation et exportation de clés - permet d'importer et d'exporter des clés asymétriques.
- API de protection des données (DPAPI) - permet de chiffrer et déchiffrer des données facilement.

- Stockage et extraction de clés - permet de stocker et d'isoler en toute sécurité la clé privée d'une paire de clés asymétriques.

Rubriques

Vérification des fournisseurs KSP et CNG pour Windows

Les fournisseurs KSP et CNG sont installés lorsque vous installez le client Windows AWS CloudHSM . Vous pouvez installer le client en suivant les étapes indiquées dans Installation du client (Windows).

Configuration et exécution du client AWS CloudHSM Windows

Pour démarrer le client Windows CloudHSM, vous devez d'abord respecter les Prérequis. Ensuite, mettez à jour les fichiers de configuration utilisés par les fournisseurs et démarrez le client en suivant les étapes ci-dessous. Vous devez réaliser ces étapes la première fois que vous utilisez les fournisseurs KSP et CNG et après avoir ajouté ou supprimé des HSM dans votre cluster. De cette façon, AWS CloudHSM il est possible de synchroniser les données et de maintenir la cohérence entre tous les HSM du cluster.

Étape 1 : Arrêter le AWS CloudHSM client

Avant de mettre à jour les fichiers de configuration utilisés par les fournisseurs, arrêtez le AWS CloudHSM client. Si le client est déjà été arrêté, l'exécution de la commande stop n'a aucun effet.

- Pour le Client Windows version 1.1.2 et ultérieure :

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Pour les clients Windows version 1.1.1 et antérieure :

  Utilisez Ctrl + C dans la fenêtre de commande dans laquelle vous avez démarré le AWS CloudHSM client.

Étape 2 : mise à jour des fichiers AWS CloudHSM de configuration

Cette étape utilise le paramètre -a de l'outil Configurer pour ajouter l'adresse IP de l'interface réseau Elastic (ENI) de l'un des HSM dans le cluster pour le fichier de configuration.

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

Pour obtenir l'adresse IP ENI d'un HSM de votre cluster, accédez à la AWS CloudHSM console, choisissez les clusters, puis sélectionnez le cluster souhaité. Vous pouvez également utiliser l'DescribeClustersopération, la commande describe-clusters ou l'applet de Get-HSM2Cluster PowerShellcommande. Saisissez une seule adresse IP d'ENI. Peu importe l'adresse IP d'ENI que vous utilisez.

Étape 3 : démarrer le AWS CloudHSM client

Ensuite, démarrez ou redémarrez le AWS CloudHSM client. Lorsque le AWS CloudHSM client démarre, il utilise l'adresse IP ENI dans son fichier de configuration pour interroger le cluster. Ensuite, il ajoute les adresses IP d'ENI de tous les HSM du cluster au fichier d'informations sur le cluster.

- Pour le Client Windows version 1.1.2 et ultérieure :

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Pour les clients Windows version 1.1.1 et antérieure :

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Recherche de fournisseurs KSP et GNC

Vous pouvez utiliser l'une ou l'autre des commandes suivantes afin de déterminer les fournisseurs qui sont installés sur votre système. Les commandes répertorient les fournisseurs KSP et CNG enregistrée. Le client AWS CloudHSM n'a pas besoin d'être en cours d'exécution.

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -enum
```

```
C:\Program Files\Amazon\CloudHSM>cng_config.exe -enum
```

Pour vérifier que les fournisseurs KSP et CNG sont installés sur votre instance Windows Server EC2, vous devriez voir les entrées suivantes dans la liste :

```
Cavium CNG Provider
Cavium Key Storage Provider
```

Si le fournisseur CNG est manquant, exécutez la commande suivante.

```
C:\Program Files\Amazon\CloudHSM>cng_config.exe -register
```

Si le fournisseur KSP est manquant, exécutez la commande suivante.

```
C:\Program Files\Amazon\CloudHSM>ksp_config.exe -register
```

AWS CloudHSM Prérequis pour Windows

Avant de démarrer le AWS CloudHSM client Windows et d'utiliser les fournisseurs KSP et CNG, vous devez définir les informations de connexion du HSM sur votre système. Vous pouvez définir les informations d'identification via le Gestionnaire d'informations d'identification Windows ou la variable d'environnement système. Nous vous recommandons d'utiliser le Gestionnaire d'informations d'identification Windows pour stocker les informations d'identification. Cette option est disponible avec les versions 2.0.4 et ultérieures du AWS CloudHSM client. L'utilisation de la variable d'environnement est plus facile à configurer, mais moins sécurisée que l'utilisation du Gestionnaire d'informations d'identification Windows.

Gestionnaire d'informations d'identification Windows

Vous pouvez utiliser l'utilitaire `set_cloudhsm_credentials` ou l'interface du Gestionnaire d'informations d'identification Windows.

- Utilisation de l'utilitaire **`set_cloudhsm_credentials`** :

  L'utilitaire `set_cloudhsm_credentials` est inclus dans votre programme d'installation Windows. Vous pouvez utiliser cet utilitaire pour transmettre facilement les informations d'identification de connexion HSM au Gestionnaire d'informations d'identification Windows. Si vous souhaitez compiler cet utilitaire à partir de la source, vous pouvez utiliser le code Python inclus dans le programme d'installation.

  1. Accédez au dossier `C:\Program Files\Amazon\CloudHSM\tools\`.

2. Exécutez le fichier `set_cloudhsm_credentials.exe` avec les paramètres de nom d'utilisateur et de mot de passe CU.

```
set_cloudhsm_credentials.exe --username <CU USER> --password <CU PASSWORD>
```

- Utilisation de l'interface du Gestionnaire d'informations d'identification :

  Vous pouvez utiliser l'interface du Gestionnaire d'informations d'identification pour gérer manuellement vos informations d'identification.

  1. Pour ouvrir Credential Manager, tapez `credential manager` dans la zone de recherche de la barre des tâches et sélectionnez Credential Manager (Gestionnaire d'informations d'identification).
  2. Sélectionnez Windows Credentials (Informations d'identification) pour gérer les informations d'identification Windows.
  3. Sélectionnez Add a generic credential (Ajouter une information d'identification générique) et remplissez les détails comme suit :
     - Dans Internet ou Network Address (Adresse Internet ou réseau), entrez le nom de la cible en tant que `cloudhsm_client`.
     - Dans Username (Nom d'utilisateur) et Password (Mot de passe) entrez les informations d'identification CU.
     - Cliquez sur OK.

Variables d'environnement du système

Vous pouvez définir des variables d'environnement système qui identifient un HSM et un utilisateur de chiffrement (CU) pour votre application Windows. Vous pouvez utiliser la commande setx pour définir des variables d'environnement système temporaires ou pour définir des variables d'environnement système permanentes par programmation ou sous l'onglet Avancé du Panneau de configuration Propriétés système de Windows.

> **Warning**
>
> Lorsque vous définissez des informations d'identification via des variables d'environnement système, le mot de passe est disponible en texte brut sur le système d'un utilisateur. Pour résoudre ce problème, utilisez le Gestionnaire d'informations d'identification Windows.

Définissez les variables d'environnement système suivantes :

**`n3fips_password=`*`CU USERNAME`*`:`*`CU PASSWORD`*

Identifie un utilisateur de chiffrement (CU) dans le HSM et fournit toutes les informations de connexion requises. Votre application s'authentifie et s'exécute en tant que ce CU. L'application possède les autorisations de ce CU et peut afficher et gérer uniquement les clés que le CU possède et partage. Pour créer un CU, utilisez createUser. Pour rechercher des CU existants, utilisez listUsers.

Par exemple :

```
setx /m n3fips_password test_user:password123
```

Associer une AWS CloudHSM clé à un certificat

Avant de pouvoir utiliser des AWS CloudHSM clés avec des outils tiers, tels que ceux de Microsoft SignTool, vous devez importer les métadonnées de la clé dans le magasin de certificats local et associer les métadonnées à un certificat. Pour importer les métadonnées de la clé, utilisez l'utilitaire import_key.exe qui est inclus dans CloudHSM version 3.0 et supérieure. Les étapes suivantes fournissent des informations supplémentaires et un exemple de sortie.

Étape 1 : Importer votre certificat

Sous Windows, vous pouvez normalement double-cliquer sur le certificat pour l'importer dans votre magasin de certificats local.

Toutefois, si un double-clic ne fonctionne pas, utilisez l'outil Microsoft Certreq pour importer le certificat dans le gestionnaire de certificats. Par exemple :

```
certreq -accept certificatename
```

Si cette action échoue et que vous recevez l'erreur `Key not found`, passez à l'étape 2. Si le certificat apparaît dans votre magasin de clés, vous avez terminé la tâche et aucune autre action n'est nécessaire.

Étape 2 : Recueillir des renseignements permettant d'identifier les certificats

Si l'étape précédente n'a pas réussi, vous devrez associer votre clé privée à un certificat. Toutefois, avant de pouvoir créer l'association, vous devez d'abord trouver le nom de conteneur unique et

le numéro de série du certificat. Utilisez un utilitaire, tel que certutil, pour afficher les informations de certificat nécessaires. L'exemple de sortie suivant de certutil indique le nom du conteneur et le numéro de série.

```
================ 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
```

Étape 3 : associer la clé AWS CloudHSM privée au certificat

Pour associer la clé au certificat, assurez-vous d'abord de démarrer le daemon AWS CloudHSM client. Ensuite, utilisez import_key.exe (qui est inclus dans CloudHSM version 3.0 et supérieure) pour associer la clé privée au certificat. Lorsque vous spécifiez le certificat, utilisez son nom de conteneur simple. L'exemple suivant montre la commande et la réponse. Cette action copie uniquement les métadonnées de la clé ; la clé reste sur le HSM.

```
$> import_key.exe -RSA CertReq-39c04db0-6aa9-4310-93db-db0d9669f42c

Successfully opened Microsoft Software Key Storage Provider : 0NCryptOpenKey failed :
 80090016
```

Étape 4 : Mettre à jour le magasin de certificats

Assurez-vous que le daemon AWS CloudHSM client est toujours en cours d'exécution. Ensuite, utilisez le verbe certutil, -repairstore, pour mettre à jour le numéro de série du certificat. L'exemple suivant montre la commande et la sortie. Consultez la documentation Microsoft pour plus d'informations sur le -repairstore verb.

```
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.
```

Après avoir mis à jour le numéro de série du certificat, vous pouvez utiliser ce certificat et la clé AWS CloudHSM privée correspondante avec n'importe quel outil de signature tiers sous Windows.

Exemple de code pour le fournisseur CNG

⚠ ** Exemple de code uniquement – Ne pas utiliser en production **
Cet exemple de code est donné uniquement à titre d'illustration. N'exécutez pas ce code dans un environnement de production.

L'exemple suivant montre comment énumérer les fournisseurs de chiffrement enregistrés dans votre système pour trouver le fournisseur CNG installé avec le client CloudHSM pour Windows. L'exemple montre également comment créer une paire de clés asymétrique et comment utiliser la paire de clés pour vous signer les données.

⚠ Important

Avant d'exécuter cet exemple, vous devez configurer les informations d'identification HSM comme expliqué dans les prérequis. Pour plus d'informations, consultez AWS CloudHSM Prérequis pour Windows.

```
// 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;
}
```

SDK client précédent (SDK client 3)

AWS CloudHSM inclut deux versions principales du SDK client :

- SDK client 5 : il s'agit de notre dernier SDK client par défaut. Pour plus d'informations sur ses avantages, consultez Avantages du SDK client 5.
- SDK client 3 : il s'agit de notre ancien SDK client. Il inclut un ensemble complet de composants pour la compatibilité des applications basées sur les plateformes et les langages ainsi que des outils de gestion.

Pour obtenir des instructions sur la migration du SDK client 3 vers le SDK client 5, consultez. Migration du SDK client 3 vers le SDK client 5

La documentation du SDK client 3 est répertoriée dans cette rubrique.

Pour procéder au téléchargement, reportez-vous à Téléchargements.

Vérifiez la version du SDK de votre client

Amazon Linux

Utilisez la commande suivante :

```
rpm -qa | grep ^cloudhsm
```

Amazon Linux 2

Utilisez la commande suivante :

```
rpm -qa | grep ^cloudhsm
```

CentOS 6

Utilisez la commande suivante :

```
rpm -qa | grep ^cloudhsm
```

CentOS 7

Utilisez la commande suivante :

```
rpm -qa | grep ^cloudhsm
```

CentOS 8

Utilisez la commande suivante :

```
rpm -qa | grep ^cloudhsm
```

RHEL 6

Utilisez la commande suivante :

```
rpm -qa | grep ^cloudhsm
```

RHEL 7

Utilisez la commande suivante :

```
rpm -qa | grep ^cloudhsm
```

RHEL 8

Utilisez la commande suivante :

```
rpm -qa | grep ^cloudhsm
```

Ubuntu 16.04 LTS

Utilisez la commande suivante :

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 18.04 LTS

Utilisez la commande suivante :

```
apt list --installed | grep ^cloudhsm
```

Ubuntu 20.04 LTS

Utilisez la commande suivante :

```
apt list --installed | grep ^cloudhsm
```

Windows Server

Utilisez la commande suivante :

```
wmic product get name,version
```

Comparaison des composants du SDK client

Outre les outils de ligne de commande, le SDK client 3 contient des composants qui permettent de décharger les opérations cryptographiques vers le HSM à partir de diverses applications basées sur des plateformes ou des langages. Le SDK client 5 est identique au SDK client 3, sauf qu'il ne prend pas encore en charge les fournisseurs CNG et KSP. Le tableau suivant compare la disponibilité des composants dans le SDK client 3 et le SDK client 5.

Composant	SDK client 5	SDK client 3
Bibliothèque PKCS #11	Oui	Oui
Fournisseur JCE	Oui	Oui
OpenSSL Dynamic Engine	Oui	Oui
Fournisseurs KSP et CNG		Oui
Utilitaire de gestion CloudHSM (CMU) [1]	Oui	Oui

Composant	SDK client 5	SDK client 3
Utilitaire de gestion des clés (KMU) [1]	Oui	Oui
Outil de configuration	Oui	Oui

[1] Les composants CMU et KMU sont inclus dans l'interface de ligne de commande CloudHSM avec le SDK client 5.

Rubriques

- Plateformes prises en charge par le SDK client 3
- Mise à niveau du SDK client 3 sous Linux
- Bibliothèque PKCS #11 pour le SDK client 3
- Installation du SDK client 3 pour le moteur dynamique OpenSSL
- SDK client 3 pour le fournisseur JCE

Plateformes prises en charge par le SDK client 3

Le SDK client 3 nécessite un démon client et propose des outils de ligne de commande, notamment l'Utilitaire de gestion CloudHSM (CMU), l'Utilitaire de gestion d'des clés (KMU) et l'outil de configuration.

Le support de base est différent pour chaque version du SDK AWS CloudHSM client. En général, le support de plateforme pour les composants d'un SDK correspond au support de base, mais pas toujours. Pour déterminer le support de plateforme pour un composant donné, assurez-vous d'abord que la plateforme que vous souhaitez apparaît dans la section de base du SDK, puis recherchez les exclusions ou toute autre information pertinente dans la section du composant.

Le support de la plateforme évolue au fil du temps. Les versions antérieures du SDK du client CloudHSM peuvent ne pas prendre en charge tous les systèmes d'exploitation répertoriés ici. Utilisez les notes de mise à jour pour déterminer le système d'exploitation compatible avec les versions précédentes du SDK du client CloudHSM. Pour plus d'informations, consultez Téléchargements pour AWS CloudHSM le SDK client.

AWS CloudHSM ne prend en charge que les systèmes d'exploitation 64 bits.

Table des matières

Prise en charge de Linux

- 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] Aucune prise en charge pour le moteur dynamique OpenSSL. Pour de plus amples informations, veuillez consulter OpenSSL Dynamic Engine.

[2] Aucune prise en charge pour le SDK client 3.3.0 et versions ultérieures.

[3] Le SDK 3.4 est la dernière version prise en charge sur Ubuntu 16.04.

[4] Le SDK 3.4 est la dernière version prise en charge sur CentOS 8.3+.

Prise en charge de Windows

- Microsoft Windows Server 2012
- Microsoft Windows Server 2012 R2

- Microsoft Windows Server 2016
- Microsoft Windows Server 2019

Composants pris en charge

Bibliothèque PKCS #11

La bibliothèque PKCS #11 est un composant uniquement pour Linux qui correspond au support de base de Linux. Pour plus d'informations, consultez the section called "Prise en charge de Linux".

Fournisseur JCE

Le fournisseur JCE est un composant uniquement pour Linux qui correspond au support de base de Linux. Pour plus d'informations, consultez the section called "Prise en charge de Linux".

- Nécessite OpenJDK 1.8

OpenSSL Dynamic Engine

Le moteur dynamique OpenSSL est un composant uniquement Linux qui ne correspond pas au support de base de Linux. Consultez les exclusions ci-dessous.

- Nécessite OpenSSL 1.0.2[f+]

Plateformes non prises en charge :

- CentOS 8
- Red Hat Enterprise Linux (RHEL) 8
- Ubuntu 18.04 LTS

Ces plateformes sont fournies avec une version d'OpenSSL incompatible avec OpenSSL Dynamic Engine for Client SDK 3. AWS CloudHSM prend en charge ces plateformes avec OpenSSL Dynamic Engine for Client SDK 5.

Fournisseurs KSP et CNG

Les fournisseurs CNG et KSP sont des composants Windows uniquement compatibles avec le support de base de Windows. Pour plus d'informations, voir Prise en charge de Windows.

Mise à niveau du SDK client 3 sous Linux

Avec AWS CloudHSM le SDK client 3.1 et versions ultérieures, la version du démon client et de tous les composants que vous installez doivent correspondre pour effectuer la mise à niveau. Pour tous les systèmes basés sur Linux, vous devez utiliser une seule commande pour mettre à niveau par lots le démon client avec la même version de la bibliothèque PKCS #11, du fournisseur Java Cryptographic Extension (JCE) ou de OpenSSL Dynamic Engine. Cette exigence ne s'applique pas aux systèmes Windows, car les fichiers binaires des fournisseurs KSP et GNC sont déjà inclus dans le package du démon client.

Pour vérifier la version du démon client

- Sur un système Linux basé sur Red Hat (y compris Amazon Linux et CentOS), utilisez la commande suivante :

```
rpm -qa | grep ^cloudhsm
```

- Sur un système Linux basé sur Debian, exécutez la commande suivante :

```
apt list --installed | grep ^cloudhsm
```

- Sur un système Windows, exécutez la commande suivante :

```
wmic product get name,version
```

Rubriques

Prérequis

Téléchargez la dernière version du démon AWS CloudHSM client et choisissez vos composants.

> **Note**
>
> Vous n'avez pas besoin d'installer tous les composants. Pour chaque composant installé, vous devez mettre à niveau ce composant pour qu'il corresponde à la version du démon du client.

Dernier démon client Linux

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
```

Dernière bibliothèque PKCS #11

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
```

Dernier moteur OpenSSL Dynamic Engine

Amazon Linux

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/EL6/cloudhsm-client-dyn-latest.el6.x86_64.rpm
```

Amazon Linux 2

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

CentOS 7

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

RHEL 7

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

Ubuntu 16.04 LTS

```
$ wget https://s3.amazonaws.com/cloudhsmv2-software/CloudHsmClient/Xenial/cloudhsm-client-dyn_latest_amd64.deb
```

Dernier fournisseur JCE

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
```

Étape 1 : arrêter le démon client

Utilisez la commande suivante pour arrêter le démon client.

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
```

Étape 2 : mettre à niveau le SDK client

La commande suivante indique la syntaxe requise pour mettre à niveau le démon client et les composants. Avant d'exécuter la commande, supprimez tous les composants que vous n'avez pas l'intention de mettre à niveau.

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>
```

Étape 3 : démarrer le démon client

Utilisez la commande suivante pour démarrer le démon client.

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

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

Bibliothèque PKCS #11 pour le SDK client 3

PKCS #11 est une norme permettant d'effectuer des opérations cryptographiques sur des modules de sécurité matériels (HSM).

Pour plus d'informations sur l'amorçage, veuillez consulter Connexion au cluster.

Rubriques

Installation du SDK client 3 pour la bibliothèque PKCS #11

Prérequis pour le SDK client 3

La bibliothèque PKCS #11 nécessite le AWS CloudHSM client.

Si vous n'avez pas installé ni configuré le AWS CloudHSM client, faites-le maintenant en suivant les étapes décrites dansInstaller le client (Linux). Une fois que vous avez installé et configuré le client, utilisez la commande suivante pour le démarrer.

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
```

Installation de la bibliothèque PKCS #11 pour le SDK client 3

La commande suivante télécharge et installe la bibliothèque PKCS #11.

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
```

- Si aucun autre composant du SDK client 3 n'est installé sur l'instance EC2 sur laquelle vous avez installé la bibliothèque PKCS #11, vous devez démarrer le SDK client 3. Vous ne devez le faire qu'une seule fois sur chaque instance avec un composant du SDK client 3.
- Vous trouverez les fichiers de bibliothèque PKCS #11 dans les emplacements suivants :

  Binaires, scripts de configuration, certificats et fichiers journaux Linux :

```
/opt/cloudhsm/lib
```

Authentification auprès de la bibliothèque PKCS #11 (SDK client 3)

Lorsque vous utilisez PKCS # 11 avec , votre application s'exécute en tant qu'utilisateur de chiffrement (CU) dans vos HSM. Votre application peut afficher et gérer uniquement les clés que le CU possède et partage. Vous pouvez utiliser un CU existant dans vos HSM ou créer un nouveau CU. Pour plus d'informations sur la gestion des CU, consultez les sections Gestion des utilisateurs HSM à l'aide de la CLI CloudHSM et Gestion des utilisateurs HSM à l'aide de l'Utilitaire de gestion CloudHSM (CMU).

Pour spécifier le CU pour PKCS #11, utilisez le paramètre de code PIN PKCS #11 fonction C_Login. En AWS CloudHSM effet, le paramètre pin a le format suivant :

```
<CU_user_name>:<password>
```

Par exemple, la commande suivante définit le code PIN de bibliothèque PKCS #11 sur le CU avec le nom d'utilisateur `CryptoUser` et le mot de passe `CUPassword123!`.

```
CryptoUser:CUPassword123!
```

Types de clé pris en charge (SDK client 3)

La bibliothèque PKCS #11 prend en charge les Types de clé suivants.

Type de clé	Description
RSA	Générez des clés RSA de 2 048 bits à 4 096 bits, par incréments de 256 bits.
EC	Générez des clés avec les courbes secp224r1 (P-224), secp256r1 (P-256), secp256k1 (Blockchain), secp384r1 (P-384) et secp521r1 (P-521).
AES	Générez des clés AES de 128, 192 et 256 bits.
DES3 (triple DES)	Générez des clés DES3 192 bits. Voir la note 1 ci-dessous pour un changement à venir.
GENERIC_SECRET	Générez des secrets génériques de 1 à 64 octets.

- [1] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

Mécanismes pris en charge (SDK client 3)

La bibliothèque PKCS #11 prend en charge les algorithmes suivants :

- Chiffrement et déchiffrement : AES-CBC, AES-CTR, AES-ECB, AES-GCM, DES3-CBC, DES3-ECB, RSA-OAEP et RSA-PKCS
- Signature et vérification : RSA, HMAC et ECDSA ; avec et sans hachage
- Hachage/résumé : SHA1, SHA224, SHA256, SHA384 et SHA512
- Encapsulage de clés – AES Key Wrap,[4] AES-GCM, RSA-AES et RSA-OAEP
- Dérivation de clés — ECDH, [5]SP800-108 CTR KDF

Le tableau des mécanismes et des fonctions de la bibliothèque PKCS #11

La bibliothèque PKCS #11 est conforme à la version 2.40 de la spécification PKCS #11. Pour appeler une fonction de chiffrement utilisant PKCS #11, appelez une fonction à l'aide d'un mécanisme donné. Le tableau suivant résume les combinaisons de fonctions et de mécanismes prises en charge par AWS CloudHSM.

Interprétation du tableau de mécanismes/fonctions PKCS #11 pris en charge

La marque ✓ indique que le mécanisme de la fonction est pris en AWS CloudHSM charge. Nous ne prenons pas en charge toutes les fonctions possibles répertoriés dans la spécification PKCS #11. La marque ✖ indique qu' AWS CloudHSM elle ne prend pas encore en charge le mécanisme pour la fonction donnée, même si la norme PKCS #11 l'autorise. Les cellules vides indiquent que la norme PKCS #11 ne prend pas en charge le mécanisme pour la fonction donnée.

Mécanismes et fonctions de bibliothèque PKCS # 11 pris en charge

Mécanisme	Fonctions						
	Générer une clé ou une Paire de clés	Connexion et vérification	SR & VR	Digest	Chiffrement et déchiffrement	Dérivation de clé	Envelopper et UnWrap
CKM_RSA_PKCS_KEY_PAIR_GEN	✓						
CKM_RSA_X9_31_KEY_PAIR_GEN	✓[2]						
CKM_RSA_X_509		✓			✓		
CKM_RSA_PKCS voir note 8		✓[1]	✖		✓[1]		✓[1]

Mécanisme	Fonctions						
CKM_RSA_PKCS_OAEP					✓[1]		✓[6]
CKM_SHA1_RSA_PKCS		✓[3.2]					
CKM_SHA224_RSA_PKCS		✓[3.2]					
CKM_SHA256_RSA_PKCS		✓[3.2]					
CKM_SHA384_RSA_PKCS		✓[2,3.2]					
CKM_SHA512_RSA_PKCS		✓[3.2]					
CKM_RSA_PKCS_PSS		✓[1]					
CKM_SHA1_RSA_PKCS_PSS		✓[3.2]					
CKM_SHA224_RSA_PKCS_PSS		✓[3.2]					
CKM_SHA256_RSA_PKCS_PSS		✓[3.2]					

Mécanisme	Fonctions						
CKM_SHA384_RSA_PKCS_PSS		✓[2,3.2]					
CKM_SHA512_RSA_PKCS_PSS		✓[3.2]					
CKM_EC_KEY_PAIR_GEN	✓						
CKM_ECDSA		✓[1]					
CKM_ECDSA_SHA1		✓[3.2]					
CKM_ECDSA_SHA224		✓[3.2]					
CKM_ECDSA_SHA256		✓[3.2]					
CKM_ECDSA_SHA384		✓[3.2]					
CKM_ECDSA_SHA512		✓[3.2]					
CKM_ECDH1_DERIVE						✓[5]	
CKM_SP800_108_COUNTER_KDF						✓	

Mécanisme	Fonctions						
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 voir note 8	✓						
CKM_DES3_CBC voir note 8					✓[3.3]		✖
CKM_DES3_CBC_PAD voir note 8					✓		✖
CKM_DES3_ECB voir note 8					✓		✖

Mécanisme	Fonctions						
CKM_AES_GCM					✓[3.3, 4]		✓[7.1]
CKM_CLOUDHSM_AES_GCM					✓[7.1]		✓[7.1]
CKM_SHA_1				✓[3.1]			
CKM_SHA_1_HMAC		✓[3.3]					
CKM_SHA224				✓[3.1]			
CKM_SHA224_HMAC		✓[3.3]					
CKM_SHA256				✓[3.1]			
CKM_SHA256_HMAC		✓[3.3]					
CKM_SHA384				✓[3.1]			
CKM_SHA384_HMAC		✓[3.3]					
CKM_SHA512				✓[3.1]			
CKM_SHA512_HMAC		✓[3.3]					

Mécanisme	Fonctions						
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]

Annotations du mécanisme

- [1] Opérations à une seule partie uniquement.
- [2] Le mécanisme est fonctionnellement identique au mécanisme CKM_RSA_PKCS_KEY_PAIR_GEN, mais offre de meilleures garanties pour la génération de p et q.
- [3.1] AWS CloudHSM aborde le hachage différemment en fonction du SDK client. Pour le SDK client 3, l'endroit où nous effectuons le hachage dépend de la taille des données et du fait que vous utilisez des opérations en une ou plusieurs parties.

Opérations en une seule partie dans le SDK client 3

Le tableau 3.1 répertorie la taille maximale de l'ensemble de données pour chaque mécanisme du SDK client 3. Le hachage complet est calculé dans le HSM. Aucune prise en charge pour les tailles de données supérieures à 16 Ko.

Tableau 3.1, Taille maximale de l'ensemble de données pour les opérations en une seule partie

Mécanisme	Taille maximale des données
CKM_SHA_1	16296
CKM_SHA224	16264
CKM_SHA256	16296
CKM_SHA384	16232
CKM_SHA512	16232

SDK client 3 pour les opérations en plusieurs parties

Prise en charge pour les tailles de données supérieures à 16 Ko, mais la taille des données détermine l'endroit où le hachage a lieu. Les tampons de données de moins de 16 Ko sont hachés dans le HSM. Les tampons compris entre 16 Ko et la taille de données maximale de votre système sont hachés localement dans le logiciel. N'oubliez pas : les fonctions de hachage ne nécessitent pas de secrets cryptographiques, vous pouvez donc les calculer en toute sécurité en dehors du HSM.

- [3.2] AWS CloudHSM aborde le hachage différemment en fonction du SDK client. Pour le SDK client 3, l'endroit où nous effectuons le hachage dépend de la taille des données et du fait que vous utilisez des opérations en une ou plusieurs parties.

  SDK client 3 pour les opérations en une seule partie

  Le tableau 3.2 répertorie la taille maximale de l'ensemble de données pour chaque mécanisme du SDK client 3. Aucune prise en charge pour les tailles de données supérieures à 16 Ko.

Tableau 3.2, Taille maximale de l'ensemble de données pour les opérations en une seule partie

Mécanisme	Taille maximale des données
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

SDK client 3 pour les opérations en plusieurs parties

Prise en charge pour les tailles de données supérieures à 16 Ko, mais la taille des données détermine l'endroit où le hachage a lieu. Les tampons de données de moins de 16 Ko sont hachés dans le HSM. Les tampons compris entre 16 Ko et la taille de données maximale de votre système sont hachés localement dans le logiciel. N'oubliez pas : les fonctions de hachage ne nécessitent

pas de secrets cryptographiques, vous pouvez donc les calculer en toute sécurité en dehors du HSM.

- [3.3] Lors de l'utilisation de données avec l'un des mécanismes suivants, si la mémoire tampon des données dépasse la taille maximale des données, l'opération génère une erreur. Pour ces mécanismes, tout le traitement des données doit avoir lieu à l'intérieur du HSM. Le tableau suivant répertorie la taille maximale des données définie pour chaque mécanisme :

  Tableau 3.3, Taille maximale de l'ensemble de données

Mécanisme	Taille maximale des données
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] Lorsque vous procédez au chiffrement AES GCM, le HSM n'accepte pas de données du vecteur d'initialisation (VI) de l'application. Vous devez utiliser un vecteur d'initialisation qu'il génère. Le vecteur d'initialisation 12 octets fourni par le HSM est écrit dans la référence en mémoire vers lequel pointe l'élément pIV des paramètres CK_GCM_PARAMS de la structure que vous fournissez. Pour éviter toute confusion de l'utilisateur, le kit SDK PKCS#11 version 1.1.1 et ultérieure s'assure que cet élément pIV pointe vers une mémoire tampon mise à zéro lorsque le chiffrement AES-GCM est initialisé.
- [5] SDK client 3 uniquement. Ce mécanisme est mis en œuvre pour prendre en charge des cas de transfert de charge SSL/TLS et est exécuté uniquement partiellement dans le HSM. Avant d'utiliser ce mécanisme, consultez « Problème : La dérivation de clés ECDH est exécutée

uniquement partiellement dans le HSM » dans Problèmes connus pour la bibliothèque PKCS#11. CKM_ECDH1_DERIVE ne prend pas en charge la courbe secp521r1 (P-521).

- [6] Les CK_MECHANISM_TYPE et CK_RSA_PKCS_MGF_TYPE suivants sont pris en charge en tant que CK_RSA_PKCS_OAEP_PARAMS pour CKM_RSA_PKCS_OAEP :
  - CKM_SHA_1 utilisant CKG_MGF1_SHA1
  - CKM_SHA224 utilisant CKG_MGF1_SHA224
  - CKM_SHA256 utilisant CKG_MGF1_SHA256
  - CKM_SHA384 utilisant CKM_MGF1_SHA384
  - CKM_SHA512 utilisant CKM_MGF1_SHA512
- [7.1] Mécanisme défini par le fournisseur. Afin d'utiliser les mécanismes définis par le fournisseur CloudHSM, les applications PKCS #11 doivent inclure /opt/cloudhsm/include/pkcs11t.h lors de la compilation.

  CKM_CLOUDHSM_AES_GCM : Ce mécanisme propriétaire est une alternative plus sûre par programme à la norme CKM_AES_GCM. Il ajoute le IV généré par le HSM au chiffrement au lieu de l'écrire dans la structure CK_GCM_PARAMS fournie lors de l'initialisation du chiffrement. Vous pouvez utiliser ce mécanisme avec les fonctions C_Encrypt, C_WrapKey, C_Decrypt et C_UnwrapKey. Lors de l'utilisation de ce mécanisme, la variable PiV dans la structure CK_GCM_PARAMS doit être définie sur NULL. Lors de l'utilisation de ce mécanisme avec C_Decrypt et C_UnwrapKey, le IV doit être ajouté au texte chiffré qui est en cours de désencapsulage.

  CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD : Encapsulage des clés AES avec remplissage PKCS #5

  CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD : Encapsulage des clés AES avec remplissage à l'aide de zéros

  Pour plus d'informations sur l'encapsulage de clé AES, voir Encapsulage de clé AES.
- [8] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

Opérations d'API prises en charge (SDK client 3)

La bibliothèque PKCS #11 prend en charge les opérations d'API PKCS #11 suivantes.

- 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` (prise en charge du SDK client 3 uniquement)
- `C_SignRecoverInit` (prise en charge du SDK client 3 uniquement)
- `C_SignUpdate`
- `C_UnWrapKey`
- `C_Verify`
- `C_VerifyFinal`
- `C_VerifyInit`
- `C_VerifyRecover` (prise en charge du SDK client 3 uniquement)
- `C_VerifyRecoverInit` (prise en charge du SDK client 3 uniquement)
- `C_VerifyUpdate`
- `C_WrapKey`

Attributs de clés pris en charge (SDK client 3)

Un objet de clé peut être une clé publique, privée ou secrète. Les actions autorisées sur un objet de clé sont spécifiées via des attributs. Les attributs sont définis lorsque l'objet de clé est créé. Lorsque vous utilisez la bibliothèque SDK PKCS #11, des valeurs par défaut sont attribuées, comme indiqué par la norme PKCS #11.

AWS CloudHSM ne prend pas en charge tous les attributs répertoriés dans la spécification PKCS #11. Nous nous conformons à la spécification pour tous les attributs que nous prenons en charge. Ces attributs sont indiqués dans les tableaux respectifs.

Les fonctions cryptographiques telles que `C_CreateObject`, `C_GenerateKey`, `C_GenerateKeyPair`, `C_UnwrapKey` et `C_DeriveKey` qui créent, modifient ou copient des objets utilisent un modèle d'attribut en tant que paramètre. Pour plus d'informations sur la transmission d'un modèle d'attributs lors de la création d'un objet, consultez l'échantillon Generate keys through PKCS #11 library.

Interprétation du tableau d'attributs de la bibliothèque PKCS #11

Le tableau de la bibliothèque PKCS #11 contient une liste d'attributs qui diffèrent en fonction des types de clé. Il indique si un attribut donné est pris en charge pour un type de clé particulier lors de l'utilisation d'une fonction cryptographique spécifique avec AWS CloudHSM.

Légende :

- ✓ indique que CloudHSM prend en charge l'attribut pour le type de clé spécifique.
- ✖ indique que CloudHSM ne prend pas en charge l'attribut pour le type de clé spécifique.
- R indique que la valeur de l'attribut est définie en lecture seule pour le type de clé spécifique.
- S indique que l'attribut ne peut pas être lu par `GetAttributeValue` car sensible.
- Une cellule vide dans la colonne Default Value (Valeur par défaut) indique qu'il n'y a aucune valeur par défaut attribuée à l'attribut.

GenerateKeyPair

Attribut	Type de clé				Valeur par défaut
	EC privée	EC publique	RSA privée	RSA publique	
`CKA_CLASS`	✓	✓	✓	✓	
`CKA_KEY_TYPE`	✓	✓	✓	✓	
`CKA_LABEL`	✓	✓	✓	✓	

Attribut	Type de clé				Valeur par défaut
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
CKA_SIGN	✓	✖	✓	✖	False
CKA_SIGN_RECOVER	✖	✖	✓[3]	✖	
CKA_VERIFY	✖	✓	✖	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✓[4]	

Attribut	Type de clé				Valeur par défaut
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	✖	
CKA_EXTRACTABLE	✓	✖	✓	✖	True
CKA_NEVER_EXTRACTABLE	R	✖	R	✖	
CKA_MODULUS	✖	✖	✖	✖	

Attribut	Type de clé				Valeur par défaut
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]	
CKA_EC_PARAMS	✖	✓[2]	✖	✖	
CKA_EC_POINT	✖	✖	✖	✖	
CKA_VALUE	✖	✖	✖	✖	

Attribut	Type de clé				Valeur par défaut
CKA_VALUE_LEN	✖	✖	✖	✖	
CKA_CHECK_VALUE	R	R	R	R	

GenerateKey

Attribut	Type de clé			Valeur par défaut
	AES	DES3	Secrète générique	
CKA_CLASS	✓	✓	✓	
CKA_KEY_TYPE	✓	✓	✓	
CKA_LABEL	✓	✓	✓	
CKA_ID	✓	✓	✓	
CKA_LOCAL	R	R	R	True
CKA_TOKEN	✓	✓	✓	False
CKA_PRIVATE	✓[1]	✓[1]	✓[1]	True
CKA_ENCRYPT	✓	✓	✖	False
CKA_DECRYPT	✓	✓	✖	False

Attribut	Type de clé			Valeur par défaut
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
CKA_UNWRAP	✓	✓	✖	False

Attribut	Type de clé			Valeur par défaut
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	✖	✖	✖	
CKA_EXPONENT_1	✖	✖	✖	

Attribut	Type de clé							Valeur par défaut
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

Attribut	Type de clé							Valeur par défaut
		EC privée	EC publique	RSA privée	RSA publique	AES	DES3	Secrète générique	

Attribut	Type de clé							Valeur par défaut
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

Attribut	Type de clé							Valeur par défaut
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	

Attribut	Type de clé							Valeur par défaut
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]	✖	✖	✖	

Attribut	Type de clé							Valeur par défaut
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

Attribut	Type de clé						Valeur par défaut
	EC privée	RSA privée	AES	DES3	Secrète générique		
CKA_CLASS	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]		
CKA_KEY_TYPE	✓[2]	✓[2]	✓[2]	✓[2]	✓[2]		
CKA_LABEL	✓	✓	✓	✓	✓		
CKA_ID	✓	✓	✓	✓	✓		

Attribut	Type de clé					Valeur par défaut
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	✖	✓[3]	✖	✖	✖	False
CKA_VERIFY	✖	✖	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	✖	✖	

Attribut	Type de clé					Valeur par défaut
CKA_WRAP	✖	✖	✓	✓	✖	False
CKA_UNWRAP	✖	✓	✓	✓	✖	False
CKA_SENSITIVE	✓	✓	✓	✓	✓	True
CKA_EXTRACTABLE	✓	✓	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	R	R	
CKA_MODULUS	✖	✖	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	✖	✖	
CKA_PRIME_1	✖	✖	✖	✖	✖	
CKA_PRIME_2	✖	✖	✖	✖	✖	
CKA_COEFFICIENT	✖	✖	✖	✖	✖	

Attribut	Type de clé					Valeur par défaut
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

Attribut	Type de clé			Valeur par défaut
	AES	DES3	Secrète générique	
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

Attribut	Type de clé			Valeur par défaut
CKA_SIGN_RECOVER	✖	✖	✖	
CKA_VERIFY	✓	✓	✓	False
CKA_VERIFY_RECOVER	✖	✖	✖	
CKA_WRAP	✓	✓	✖	False
CKA_UNWRAP	✓	✓	✖	False
CKA_SENSITIVE	✓	✓	✓	True
CKA_EXTRACTABLE	✓	✓	✓	True
CKA_NEVER_EXTRACTABLE	R	R	R	
CKA_ALWAYS_SENSITIVE	R	R	R	
CKA_MODULUS	✖	✖	✖	
CKA_MODULUS_BITS	✖	✖	✖	

Attribut	Type de clé			Valeur par défaut
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]	
CKA_CHECK_VALUE	R	R	R	

GetAttributeValue

Attribut	Type de clé						
	EC privée	EC publique	RSA privée	RSA publique	AES	DES3	Secrète générique
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	✓	✓	✓	✓	✓	✓	✓

Attribut	Type de clé						
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	✓	✖	✓	✖	✓	✓	✓

Attribut	Type de clé						
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	✖	✖	✖	✖

Attribut	Type de clé						
CKA_PUBLIC_EXPONENT	✖	✖	✓	✓	✖	✖	✖
CKA_EC_PARAMS	✓	✓	✖	✖	✖	✖	✖
CKA_EC_POINT	✖	✓	✖	✖	✖	✖	✖
CKA_VALUE	S	✖	✖	✖	✓[2]	✓[2]	✓[2]
CKA_VALUE_LEN	✖	✖	✖	✖	✓	✖	✓
CKA_CHECK_VALUE	✓	✓	✓	✓	✓	✓	✖

Annotations d'attributs

- [1] Cet attribut est partiellement pris en charge par le micrologiciel et doit être explicitement défini sur la valeur par défaut.
- [2] Attribut obligatoire.
- [3] SDK client 3 uniquement. L'attribut `CKA_SIGN_RECOVER` est dérivé de l'attribut `CKA_SIGN`. S'il est défini, il doit uniquement l'être avec la même valeur que celle définie pour `CKA_SIGN`. Dans le cas contraire, il prend la valeur par défaut de `CKA_SIGN`. Étant donné que CloudHSM prend uniquement en charge les mécanismes de signature récupérable basés sur RSA, cet attribut est actuellement applicable uniquement aux clés publiques RSA.
- [4] SDK client 3 uniquement. L'attribut `CKA_VERIFY_RECOVER` est dérivé de l'attribut `CKA_VERIFY`. S'il est défini, il doit uniquement l'être avec la même valeur que celle définie pour `CKA_VERIFY`. Dans le cas contraire, il prend la valeur par défaut de `CKA_VERIFY`. Étant donné que CloudHSM prend uniquement en charge les mécanismes de signature récupérable basés sur RSA, cet attribut est actuellement applicable uniquement aux clés publiques RSA.

Modification d'attributs

Certains attributs d'un objet peuvent être modifiés une fois que l'objet a été créé, tandis que d'autres ne le peuvent pas. Pour modifier des attributs, utilisez la commande setAttribute à partir de cloudhsm_mgmt_util. Vous pouvez également obtenir une liste des attributs et des constantes qui les représentent en utilisant la commande listAttribute à partir de cloudhsm_mgmt_util.

La liste suivante contient les attributs que vous pouvez modifier après la création de l'objet :

- `CKA_LABEL`
- `CKA_TOKEN`

> **Note**
>
> La modification est autorisée uniquement pour changer une clé de session en une clé de jeton. Utilisez la commande setAttribute à partir de key_mgmt_util pour modifier la valeur de l'attribut.

- `CKA_ENCRYPT`
- `CKA_DECRYPT`
- `CKA_SIGN`
- `CKA_VERIFY`
- `CKA_WRAP`
- `CKA_UNWRAP`
- `CKA_LABEL`
- `CKA_SENSITIVE`
- `CKA_DERIVE`

> **Note**
>
> Cet attribut prend en charge la dérivation de clé. Il doit être `False` pour toutes les clés publiques et ne peut pas être défini sur `True`. Pour les clés privées EC et secrètes, il peut être défini sur `True` ou `False`.

- `CKA_TRUSTED`

> **Note**
>
> Cet attribut peut être défini sur `True` ou `False` par le responsable du chiffrement uniquement.

- `CKA_WRAP_WITH_TRUSTED`

> **Note**
>
> Appliquez cet attribut à une clé de données exportable pour indiquer que vous ne pouvez encapsuler cette clé qu'avec des clés marquées comme `CKA_TRUSTED`. Une fois `CKA_WRAP_WITH_TRUSTED` défini sur true, l'attribut passe en lecture seule et vous ne pouvez ni le modifier ni le supprimer.

Interprétation des codes d'erreur

Si vous spécifiez dans le modèle un attribut qui n'est pas pris en charge par une clé spécifique, une erreur se produit. Le tableau suivant contient des codes d'erreur qui sont générés lorsque vous violez des spécifications :

Code d'erreur	Description
`CKR_TEMPLATE_INCONSISTENT`	Vous recevez cette erreur lorsque vous spécifiez, dans le modèle d'attributs, un attribut conforme à la spécification PKCS #11 mais non pris en charge par CloudHSM.
`CKR_ATTRIBUTE_TYPE_INVALID`	Vous recevez cette erreur lorsque vous récupérez la valeur d'un attribut conforme à la spécification PKCS #11 mais non pris en charge par CloudHSM.
`CKR_ATTRIBUTE_INCOMPLETE`	Vous recevez cette erreur lorsque vous ne spécifiez pas l'attribut obligatoire dans le modèle d'attributs.

Code d'erreur	Description
`CKR_ATTRIBUTE_READ_ONLY`	Vous recevez cette erreur lorsque vous spécifiez un attribut en lecture seule dans le modèle d'attributs.

Exemples de code pour la bibliothèque PKCS #11 (SDK client 3)

Les exemples de code ci-dessous vous GitHub montrent comment accomplir des tâches de base à l'aide de la bibliothèque PKCS #11.

Prérequis d'exemples de code

Avant d'exécuter les exemples, effectuez les étapes suivantes pour configurer votre environnement :

- Installez et configurez la bibliothèque PKCS #11 pour le SDK client 3.
- Configurez un utilisateur de chiffrement(CU). Votre application utilise ce compte HSM pour exécuter les exemples de code sur le HSM.

Exemples de code

Des exemples de code pour la bibliothèque AWS CloudHSM logicielle de PKCS #11 sont disponibles sur GitHub. Ce référentiel contient des exemples sur la façon d'effectuer des opérations courantes à l'aide de PKCS #11, y compris le chiffrement, le déchiffrement, la signature et la vérification.

- Générer des clés (AES, RSA, EC)
- Afficher les attributs des clés
- Chiffrer et déchiffrer les données avec AES GCM
- Chiffrer et déchiffrer les données avec AES_CTR
- Chiffrer et déchiffrer les données avec 3DES
- Signer et vérifier les données avec RSA
- Dériver des clés à l'aide de HMAC KDF
- Encapsuler et désencapsuler les clés avec AES en utilisant le remplissage PKCS #5
- Encapsuler et désencapsuler les clés avec AES sans remplissage
- Encapsuler et désencapsuler les clés avec AES à l'aide du remplissage avec des zéros

- Encapsuler et désencapsuler les clés avec AES-GCM
- Encapsuler et désencapsuler les clés avec RSA

Installation du SDK client 3 pour le moteur dynamique OpenSSL

Le SDK client 3 nécessite un démon client pour se connecter au cluster. Il prend en charge :

- Génération de clés RSA de 2 048, 3 072 et 4 096 bits.
- Signature vérification RSA.
- Chiffrement/Déchiffrement RSA.
- Génération de nombres aléatoires sécurisée par voie cryptographique et conforme à la spécification FIPS.

Rubriques

- Prérequis pour le moteur dynamique OpenSSL avec le SDK client 3
- Installation du moteur dynamique OpenSSL pour le SDK client 3
- Utiliser le moteur dynamique OpenSSL pour le SDK client 3

Prérequis pour le moteur dynamique OpenSSL avec le SDK client 3

Pour plus d'informations sur les plateformes prises en charge, veuillez consulter Plateformes prises en charge par le SDK client 3.

Avant de pouvoir utiliser le moteur AWS CloudHSM dynamique pour OpenSSL, vous avez besoin du client. AWS CloudHSM

Le client est un démon qui établit une communication end-to-end cryptée avec les HSM de votre cluster, et le moteur OpenSSL communique localement avec le client. Pour installer et configurer le AWS CloudHSM client, consultezInstaller le client (Linux). Utilisez la commande suivante pour le démarrer.

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
```

Installation du moteur dynamique OpenSSL pour le SDK client 3

Les étapes suivantes décrivent comment installer et configurer le moteur AWS CloudHSM dynamique pour OpenSSL. Pour plus d'informations sur la mise à niveau, veuillez consulter Mise à niveau du SDK client 3.

Pour installer et configurer le moteur OpenSSL

1. Utilisez les commandes suivantes pour télécharger et installer le moteur OpenSSL.

   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
```

Le moteur OpenSSL est installé sur `/opt/cloudhsm/lib/libcloudhsm_openssl.so`.

2. Utilisez la commande suivante pour définir la variable d'environnement `n3fips_password` qui contient les informations d'identification d'un utilisateur de chiffrement (CU).

```
$ export n3fips_password=<HSM user name>:<password>
```

Utiliser le moteur dynamique OpenSSL pour le SDK client 3

Pour utiliser le moteur AWS CloudHSM dynamique pour OpenSSL à partir d'une application intégrée à OpenSSL, assurez-vous que votre application utilise le moteur dynamique OpenSSL nommé. `cloudhsm` La bibliothèque partagée du moteur dynamique est située dans `/opt/cloudhsm/lib/libcloudhsm_openssl.so`.

Pour utiliser le moteur AWS CloudHSM dynamique pour OpenSSL à partir de la ligne de commande OpenSSL, utilisez l'`-engine`option permettant de spécifier le moteur dynamique OpenSSL nommé. `cloudhsm` Par exemple :

```
$ openssl s_server -cert server.crt -key server.key -engine cloudhsm
```

SDK client 3 pour le fournisseur JCE

Le fournisseur AWS CloudHSM JCE est une implémentation de fournisseur créée à partir du framework de fournisseur Java Cryptographic Extension (JCE). Le JCE vous permet d'effectuer des opérations cryptographiques à l'aide du kit de développement Java (JDK). Dans ce guide, le fournisseur AWS CloudHSM JCE est parfois appelé fournisseur JCE. Utilisez le fournisseur JCE et le JDK pour décharger les opérations cryptographiques vers le HSM.

Rubriques

Installation et utilisation du fournisseur AWS CloudHSM JCE pour le SDK client 3

Avant de pouvoir utiliser le fournisseur JCE, vous avez besoin du AWS CloudHSM client.

Le client est un démon qui établit une communication end-to-end cryptée avec les HSM de votre cluster. Le fournisseur JCE communique localement avec le client. Si vous n'avez pas installé ni configuré le package AWS CloudHSM client, faites-le maintenant en suivant les étapes décrites dansInstaller le client (Linux). Une fois que vous avez installé et configuré le client, utilisez la commande suivante pour le démarrer.

Le fournisseur JCE est uniquement pris en charge sur les systèmes d'exploitation Linux et compatibles.

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
```

Rubriques

Installation du fournisseur JCE

Utilisez les commandes suivantes pour télécharger et installer le fournisseur JCE. Cette bibliothèque est prise en charge uniquement sur les systèmes d'exploitation Linux et compatibles.

> **Note**
>
> Pour la mise à niveau, consultez Mise à niveau du SDK client 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
```

Une fois que vous avez exécuté les commandes précédentes, vous pouvez trouver les fichiers du fournisseur JCE suivants :

- `/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`

Validation de l'installation

Effectuez des opérations de base sur le HSM pour valider l'installation.

Pour valider l'installation du fournisseur JCE

1. (Facultatif) Si besoin, utilisez la commande suivante pour installer Java dans votre environnement.

   Linux (and compatible libraries)

   ```
   $ sudo yum install java-1.8.0-openjdk
   ```

   Ubuntu

   ```
   $ sudo apt-get install openjdk-8-jre
   ```

2. Utilisez les commandes suivantes pour définir les variables d'environnement nécessaires. Remplacez *<HSM user name>* et *<password>* par les informations d'identification d'un utilisateur de chiffrement (CU).

   ```
   $ export LD_LIBRARY_PATH=/opt/cloudhsm/lib
   ```

   ```
   $ export HSM_PARTITION=PARTITION_1
   ```

   ```
   $ export HSM_USER=<HSM user name>
   ```

   ```
   $ export HSM_PASSWORD=<password>
   ```

3. Utilisez la commande suivante pour lancer le test des fonctionnalités de base. Si elle aboutit, la sortie de commande devrait ressembler à la sortie ci-dessous.

   ```
   $ 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)
```

Fournir des informations d'identification au fournisseur JCE

Les HSM doivent authentifier votre application Java avant que l'application ne puisse les utiliser. Chaque application peut utiliser une session. Les HSM authentifient une session en utilisant la méthode de connexion explicite ou implicite.

Connexion explicite : Cette méthode vous permet de fournir les informations d'identification CloudHSM directement dans l'application. Elle utilise la méthode `LoginManager.login()`, dans laquelle vous transmettez le nom d'utilisateur et le mot de passe du CU, ainsi que l'ID de partition du HSM. Pour plus d'informations sur l'utilisation de la méthode de connexion explicite, consultez l'exemple de code Connexion à un HSM.

Connexion implicite : Cette méthode vous permet de définir les informations de connexion CloudHSM dans un nouveau fichier de propriétés, dans les propriétés système ou en tant que variables d'environnement.

- Nouveau fichier de propriétés : Créez un nouveau fichier intitulé `HsmCredentials.properties` et ajoutez-le dans le `CLASSPATH` de votre application. Le fichier doit contenir ce qui suit :

```
HSM_PARTITION = PARTITION_1
HSM_USER = <HSM user name>
HSM_PASSWORD = <password>
```

- Propriétés système : Définissez les informations d'identification par le biais des propriétés système lors de l'exécution de votre application. Les exemples suivants montrent deux manières de le faire :

```
$ 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>");
```

- Variables d'environnement : Définissez les informations d'identification en tant que variables d'environnement.

```
$ export HSM_PARTITION=PARTITION_1
$ export HSM_USER=<HSM user name>
$ export HSM_PASSWORD=<password>
```

Les informations d'identification peuvent ne pas être disponibles si l'application ne les fournit pas ou si vous essayez une opération avant que le HSM n'authentifie la session. Dans ces cas, la bibliothèque de logiciels CloudHSM pour Java recherche les informations d'identification dans l'ordre suivant :

1. `HsmCredentials.properties`
2. Propriétés système
3. Variables d'environnement

Gestion des erreurs

La gestion des erreurs est plus facile avec la méthode de connexion explicite qu'avec la méthode de connexion implicite. Lorsque vous utilisez la classe `LoginManager`, vous maîtrisez mieux la manière dont votre application gère les pannes. La méthode de connexion implicite rend la gestion des erreurs difficile à comprendre lorsque les informations d'identification sont invalides ou lorsque les HSM ont du mal à authentifier la session.

Principes de base de la gestion des clés chez le fournisseur JCE

Les notions de base sur la gestion des clés dans le fournisseur JCE incluent l'importation des clés, l'exportation des clés, le chargement des clés par le handle ou la suppression des clés. Pour plus d'informations sur la gestion des clés, consultez l'exemple de code Gérer les clés.

Vous pouvez également trouver d'autres exemples de code de fournisseur JCE surExemples de code.

Mécanismes pris en charge pour le SDK client 3

Pour plus d'informations sur les interfaces JCA (Java Cryptography Architecture) et les classes de moteur prises en charge par AWS CloudHSM, consultez les rubriques suivantes.

Rubriques

- Clés prises en charge
- Chiffrements pris en charge
- Synthèses prises en charge
- Algorithmes HMAC (Hash-Based Message Authentication Code) pris en charge
- Mécanismes de signature/vérification pris en charge
- Annotations du mécanisme

Clés prises en charge

La bibliothèque AWS CloudHSM logicielle pour Java vous permet de générer les types de clés suivants.

- AES : Clés AES de 128, 192 et 256 bits.
- DESede : clé 3DES 92 bits. Voir la note 1 ci-dessous pour un changement à venir.
- Paires de clés ECC pour courbes NIST secp256r1 (P-256), secp384r1 (P-384) et secp256k1 (Blockchain).
- RSA : Clés RSA de 2 048 bits à 4 096 bits, par incréments de 256 bits.

Outre les paramètres standard, nous prenons en charge les paramètres suivants pour chaque clé générée.

- Label : étiquette de clé que vous pouvez utiliser pour rechercher des clés.
- isExtractable : indique si la clé peut être exportée depuis le HSM.
- isPersistent : indique si la clé reste sur le HSM lorsque la session en cours se termine.

> **Note**
>
> La version 3.1 de la bibliothèque Java permet de spécifier les paramètres plus en détail. Pour plus d'informations, consultez Attributs Java pris en charge.

Chiffrements pris en charge

La bibliothèque AWS CloudHSM logicielle pour Java prend en charge les combinaisons d'algorithmes, de modes et de remplissage suivantes.

Algorithm	Mode	Remplissage	Remarques
AES	CBC	`AES/CBC/NoPadding` `AES/CBC/PKCS5Padding`	Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE` .
AES	ECB	`AES/ECB/NoPadding` `AES/ECB/PKCS5Padding`	Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE` . Utilisez Transformation AES.
AES	CTR	`AES/CTR/NoPadding`	Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE` .

Algorithm	Mode	Remplissage	Remarques
AES	GCM	`AES/GCM/NoPadding`	Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE`, `Cipher.WRAP_MODE` et `Cipher.UNWRAP_MODE`. Lorsque vous effectuez un chiffrement AES-GCM, le HSM ignore le vecteur d'initialisation (IV) dans la demande et utilise un vecteur d'initialisation généré. Lorsque l'opération est terminée, vous devez appeler `Cipher.getIV()` pour obtenir le vecteur d'initialisation.

AESWrap	ECB	`AESWrap/ECB/ZeroPadding`
		
		`AESWrap/ECB/NoPadding`
		
		`AESWrap/ECB/PKCS5Padding`
			Implémente `Cipher.WRAP_MODE` et `Cipher.UNWRAP_MODE`. Utilisez Transformation AES.

Algorithm	Mode	Remplissage	Remarques

DESede (Triple DES)	CBC	`DESede/CBC/NoPadding`
		
		`DESede/CBC/PKCS5Padding`
			Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE` .
			
			Les routines de génération de clé acceptent une taille de 168 ou 192 bits. Toutefois, en interne, tous les clés DESede ont une taille de 192 bits.
			
			Voir la note 1 ci-dessous pour un changement à venir.

Algorithm	Mode	Remplissage	Remarques

DESede (Triple DES)	ECB	`DESede/ECB/NoPadding`
		
		`DESede/ECB/PKCS5Padding`
			Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE` .
			
			Les routines de génération de clé acceptent une taille de 168 ou 192 bits. Toutefois, en interne, tous les clés DESede ont une taille de 192 bits.
			
			Voir la note 1 ci-dessous pour un changement à venir.

RSA	ECB	`RSA/ECB/NoPadding`
		
		`RSA/ECB/PKCS1Padding`
			Implémente `Cipher.ENCRYPT_MODE` et `Cipher.DECRYPT_MODE` .
			
			Voir la note 1 ci-dessous pour un changement à venir.

Algorithm	Mode	Remplissage	Remarques

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`
			Implémente `Cipher.ENCRYPT_MODE`, `Cipher.DECRYPT_MODE`, `Cipher.WRAP_MODE` et `Cipher.UNWRAP_MODE`.
			
			`OAEPPadding` est OAEP avec le type de remplissage `SHA-1`.

RSAAESWrap	ECB	`OAEPPADDING`	Implémente `Cipher.WRAP_Mode` et `Cipher.UNWRAP_MODE`.

Synthèses prises en charge

La bibliothèque AWS CloudHSM logicielle pour Java prend en charge les résumés de messages suivants.

- `SHA-1`
- `SHA-224`
- `SHA-256`
- `SHA-384`
- `SHA-512`

> **Note**
>
> Les données inférieures à 16 Ko en longueur sont hachées sur le HSM, tandis que les données plus grandes le sont localement dans le logiciel.

Algorithmes HMAC (Hash-Based Message Authentication Code) pris en charge

La bibliothèque AWS CloudHSM logicielle pour Java prend en charge les algorithmes HMAC suivants.

- `HmacSHA1`
- `HmacSHA224`
- `HmacSHA256`
- `HmacSHA384`
- `HmacSHA512`

Mécanismes de signature/vérification pris en charge

La bibliothèque AWS CloudHSM logicielle pour Java prend en charge les types de signature et de vérification suivants.

Types de signature RSA

- `NONEwithRSA`

- SHA1withRSA
- SHA224withRSA
- SHA256withRSA
- SHA384withRSA
- SHA512withRSA
- SHA1withRSA/PSS
- SHA224withRSA/PSS
- SHA256withRSA/PSS
- SHA384withRSA/PSS
- SHA512withRSA/PSS

Types de signature ECDSA

- NONEwithECDSA
- SHA1withECDSA
- SHA224withECDSA
- SHA256withECDSA
- SHA384withECDSA
- SHA512withECDSA

Annotations du mécanisme

[1] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

Attributs de clé Java pris en charge pour le SDK client 3

Cette rubrique décrit comment utiliser une extension propriétaire pour la bibliothèque Java version 3.1 afin de définir les attributs des clés. Utilisez cette extension pour définir les attributs de clés pris en charge et leurs valeurs au cours des opérations suivantes :

- Génération de clés
- Importation de clés

- Désencapsulage de clés

> **Note**
>
> L'extension permettant de définir des attributs de clés personnalisés est une fonctionnalité facultative. Si vous avez déjà du code qui fonctionne dans la bibliothèque Java version 3.0, vous n'avez pas besoin de modifier ce code. Les clés que vous créez continueront de contenir les mêmes attributs qu'auparavant.

Rubriques

Présentation des attributs

Les attributs de clés permettent de spécifier les actions autorisées sur les objets de type clé, y compris les clés publiques, privées ou secrètes. Vous définissez les attributs de clés et leurs valeurs lors des opérations de création d'objets de type clé.

Toutefois, l'extension JCE (Java Cryptography Extension) ne spécifie pas comment définir les valeurs des attributs de clés. Dès lors, la plupart des actions étaient autorisées par défaut. En revanche, la norme PKCS #11 définit un ensemble complet d'attributs avec des valeurs par défaut plus restrictives. À partir de la version 3.1 de la bibliothèque Java, CloudHSM fournit une extension propriétaire qui vous permet de définir des valeurs plus restrictives pour les attributs couramment utilisés.

Attributs pris en charge

Vous pouvez définir des valeurs pour les attributs répertoriés dans le tableau ci-dessous. Il est conseillé de spécifier uniquement des valeurs pour les attributs que vous souhaitez restreindre. Si vous n'indiquez pas de valeur, CloudHSM utilise la valeur par défaut spécifiée dans le tableau ci-dessous. Une cellule vide dans la colonne Default Value (Valeur par défaut) signale qu'aucune valeur par défaut n'est assignée à l'attribut.

Attribut	Valeur par défaut			Remarques
	Clé symétrique	Clé publique dans la paire de clés	Clé privée dans la paire de clés	
CKA_TOKEN	FALSE	FALSE	FALSE	Clé permanente répliquée sur tous les HSM du cluster et incluse dans les sauvegardes. CKA_TOKEN = FALSE implique une clé de session, qui n'est chargée que sur un seul HSM et automatiquement effacée lorsque la connexion au HSM est interrompue.
CKA_LABEL				Chaîne définie par l'utilisateur. Permet d'identifier facilement les clés de votre HSM.
CKA_EXTRACTABLE	TRUE		TRUE	True indique que vous pouvez exporter cette clé hors du HSM.

Attribut	Valeur par défaut			Remarques
CKA_ENCRYPT	TRUE	TRUE		True indique que vous pouvez utiliser la clé pour chiffrer n'importe quelle mémoire tampon.
CKA_DECRYPT	TRUE		TRUE	True indique que vous pouvez utiliser la clé pour déchiffrer n'importe quelle mémoire tampon. Vous définissez généralement cette valeur sur FALSE pour une clé dont l'attribut CKA_WRAP est défini sur true.
CKA_WRAP	TRUE	TRUE		True indique que vous pouvez utiliser la clé pour encapsuler une autre clé. Vous définissez généralement ce paramètre sur FALSE pour les clés privées.

Attribut	Valeur par défaut			Remarques
CKA_UNWRAP	TRUE		TRUE	True indique que vous pouvez utiliser la clé pour décencapsuler (importer) une autre clé.
CKA_SIGN	TRUE		TRUE	True indique que vous pouvez utiliser la clé pour signer le résumé d'un message. Cette valeur est généralement définie sur FALSE pour les clés publiques et les clés privées que vous avez archivées.
CKA_VERIFY	TRUE	TRUE		True indique que vous pouvez utiliser la clé pour valider une signature. Cette valeur est généralement définie sur FALSE pour les clés privées.

Attribut	Valeur par défaut			Remarques
CKA_PRIVATE	TRUE	TRUE	TRUE	True indique qu'un utilisateur ne peut pas accéder à la clé tant qu'il n'est pas authentifié. Pour plus de clarté, les utilisateurs ne peuvent accéder à aucune clé sur CloudHSM tant qu'ils n'ont pas été authentifiés, même si cet attribut est défini sur FALSE.

> **Note**
>
> Vous bénéficiez d'une prise en charge plus large des attributs dans la bibliothèque PKCS #11. Pour plus d'informations, consultez Attributs PKCS #11 pris en charge.

Définition des attributs pour une clé

`CloudHsmKeyAttributesMap` est un objet de type Java Map que vous pouvez utiliser pour définir des valeurs d'attribut pour les objets de type clé. Les méthodes de `CloudHsmKeyAttributesMap` fonctionne d'une manière similaire à celles utilisées pour la manipulation de Java Map.

Pour définir des valeurs personnalisées pour les attributs, deux options s'offrent à vous :

- Utiliser les méthodes répertoriées dans le tableau suivant
- Utiliser les modèles de générateur illustrés plus loin dans ce document

Les objets de mappage d'attribut prennent en charge les méthodes suivantes pour définir les attributs :

Opération	Valeur renvoyée	Méthode **CloudHSMKeyAttributesMap**
Obtenir la valeur d'un attribut pour une clé existante	Objet (contenant la valeur) ou null	get(keyAttribute)
Renseigner la valeur d'un attribut de clé	Valeur précédente associée à l'attribut de clé, ou null en l'absence de mappage pour un attribut de clé	put(keyAttribute, value)
Renseigner des valeurs pour plusieurs attributs de clés	N/A	Tout mettre () keyAttributesMap
Supprimer une paire clé-valeur du mappage d'attributs	Valeur précédente associée à l'attribut de clé, ou null en l'absence de mappage pour un attribut de clé	remove(keyAttribute)

> **Note**
>
> Tous les attributs que vous ne spécifiez pas explicitement utilisent les valeurs par défaut répertoriées dans le tableau précédent, dans the section called "Attributs pris en charge".

Exemple de modèle de générateur

Les développeurs trouvent généralement plus pratique d'utiliser les classes via le modèle de générateur. À titre d'exemple :

```
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();
```

Les développeurs peuvent également utiliser des ensembles d'attributs prédéfinis comme un moyen pratique d'imposer le respect des bonnes pratiques dans les modèles de clés. À titre d'exemple :

```
//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
```

Définition des attributs pour une paire de clés

Utilisez la classe Java `CloudHsmKeyPairAttributesMap` pour gérer les attributs d'une paire de clés. `CloudHsmKeyPairAttributesMap` encapsule deux objets `CloudHsmKeyAttributesMap` : un pour une clé publique et un pour une clé privée.

Pour définir des attributs individuels de la clé publique et la clé privée séparément, vous pouvez utiliser la méthode `put()` sur l'objet mappé `CloudHsmKeyAttributes` correspondant à cette clé. Choisissez la méthode `getPublic()` pour récupérer le mappage d'attributs de la clé publique, et utilisez `getPrivate()` pour récupérer le mappage d'attributs de la clé privée. Renseignez la valeur de plusieurs attributs de clés pour les paires de clés publiques et privées en utilisant `putAll()` avec le mappage d'attributs d'une paire de clés comme argument.

Exemple de modèle de générateur

Les développeurs trouvent généralement plus pratique de définir des attributs de clés via le modèle de générateur. Par exemple :

```
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**
>
> Pour plus d'informations sur cette extension propriétaire, consultez l'archive Javadoc et l'exemple sur. GitHub Pour explorer Javadoc, téléchargez et développez l'archive.

Tout mettre en place

Pour spécifier des attributs de clés avec vos opérations de clés, procédez comme suit :

1. Instancier `CloudHsmKeyAttributesMap` pour les clés symétriques ou `CloudHsmKeyPairAttributesMap` pour les paires de clés.
2. Définissez l'objet Attributs de l'étape 1 avec les attributs et valeurs de clés requis.
3. Instanciez une classe `Cavium*ParameterSpec`, correspondant à votre type de clé spécifique, et transmettez cet objet d'attributs configuré à son constructeur.
4. Transmettez cet objet `Cavium*ParameterSpec` à une classe ou une méthode de chiffrement correspondante.

Pour référence, le tableau suivant contient les classes et méthodes `Cavium*ParameterSpec` qui prennent en charge les attributs de clé personnalisés.

Type de clé	Classe de spécification de paramètre	Exemples de constructeurs
Classe de base	`CaviumKeyGenAlgorithmParameterSpec`	`CaviumKeyGenAlgorithmParameterSpec(CloudHsmKeyAttributesMap keyAttributesMap)`
DES	`CaviumDESKeyGenParameterSpec`	`CaviumDESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap keyAttributesMap)`

Type de clé	Classe de spécification de paramètre	Exemples de constructeurs
RSA	`CaviumRSAKeyGenParameterSpec`	`CaviumRSAKeyGenParameterSpec(int keysize, BigInteger publicExponent, CloudHsmKeyPairAttributesMap keyPairAttributesMap)`
Secret	`CaviumGenericSecretKeyGenParameterSpec`	`CaviumGenericSecretKeyGenParameterSpec(int size, CloudHsmKeyAttributesMap keyAttributesMap)`
AES	`CaviumAESKeyGenParameterSpec`	`CaviumAESKeyGenParameterSpec(int keySize, byte[] iv, CloudHsmKeyAttributesMap keyAttributesMap)`
EC	`CaviumECGenParameterSpec`	`CaviumECGenParameterSpec(String stdName, CloudHsmKeyPairAttributesMap keyPairAttributesMap)`

Exemple de code : générer et encapsuler une clé

Ces brefs exemples de code illustrent les étapes de deux opérations différentes : la génération d'une clé et son encapsulage :

```
// 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);
```

Exemples de code pour la bibliothèque de AWS CloudHSM logiciels pour Java for Client SDK 3

Prérequis

Avant d'exécuter les exemples, vous devez configurer votre environnement :

- Installez et configurez le fournisseur d'extension cryptographique Java (JCE) et le package client AWS CloudHSM.
- Définissez un nom d'utilisateur et un mot de passe HSM valides. Les autorisations de l'utilisateur de chiffrement (CU) sont suffisantes pour ces tâches. Votre application utilise ces informations d'identification pour se connecter au HSM dans chaque exemple.
- Décidez comment fournir les informations d'identification au fournisseur JCE.

Exemples de code

Les exemples de code suivants vous montrent comment utiliser le fournisseur AWS CloudHSM JCE pour effectuer des tâches de base. D'autres exemples de code sont disponibles sur GitHub.

- Se connecter à un HSM
- Gérer les clés
- Générer une clé AES
- Chiffrer et déchiffrer avec AES-GCM
- Chiffrer et déchiffrer avec AES-CTR
- Chiffrer et déchiffrer avec D3DES-ECB[voir note 1]
- Encapsuler et désencapsuler les clés avec AES-GCM
- Encapsuler et désencapsuler les clés avec AES
- Encapsuler et désencapsuler les clés avec RSA
- Utiliser les attributs de clé pris en charge
- Énumérer les clés dans le magasin de clés
- Utiliser le magasin de clés CloudHSM
- Signer les messages d'un échantillon multi-thread
- Signer et vérifier à l'aide des clés EC

[1] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

Utilisation de la classe AWS CloudHSM KeyStore Java pour le SDK client 3

La AWS CloudHSM **`KeyStore`** classe fournit un magasin de clés PKCS12 spécial qui permet d'accéder aux AWS CloudHSM clés via des applications telles que keytool et jarsigner. Ce magasin de clés peut stocker des certificats avec vos données clés et les mettre en corrélation avec les données clés stockées sur AWS CloudHSM.

> **Note**
>
> Les certificats étant des informations publiques, il n'est AWS CloudHSM pas possible de stocker des certificats sur des HSM afin d'optimiser la capacité de stockage des clés cryptographiques.

La AWS CloudHSM `KeyStore` classe implémente l'interface du fournisseur de `KeyStore` services (SPI) de l'extension de cryptographie Java (JCE). Pour plus d'informations sur l'utilisation`KeyStore`, voir Classe KeyStore.

Choisir le magasin de clés approprié

Le fournisseur d'extension cryptographique AWS CloudHSM Java (JCE) est fourni avec un magasin de clés par défaut en lecture seule qui transmet toutes les transactions au HSM. Ce magasin de clés par défaut est distinct du magasin spécialisé. AWS CloudHSM KeyStore Dans la plupart des cas, vous obtiendrez de meilleures performances d'exécution et débit en utilisant la valeur par défaut. Vous ne devez l'utiliser que AWS CloudHSM KeyStore pour les applications pour lesquelles vous avez besoin de support pour les certificats et les opérations basées sur des certificats, en plus du transfert des opérations clés vers le HSM.

Bien que les deux magasins de clés utilisent le fournisseur JCE pour leurs opérations, ce sont des entités indépendantes qui n'échangent pas d'informations entre elles.

Chargez le magasin de clés par défaut pour votre application Java comme suit :

```
KeyStore ks = KeyStore.getInstance("Cavium");
```

Chargez le CloudHSM spécialisé comme suit : KeyStore

```
KeyStore ks = KeyStore.getInstance("CloudHSM")
```

Initialisation AWS CloudHSM KeyStore

Connectez-vous de AWS CloudHSM KeyStore la même manière que vous vous connectez au fournisseur JCE. Vous pouvez utiliser des variables d'environnement ou le fichier de propriétés du système, et vous devez vous connecter avant de commencer à utiliser CloudHSM KeyStore. Pour obtenir un exemple de connexion à un HSM à l'aide du fournisseur JCE, veuillez consulter Connexion à un HSM.

Si vous le souhaitez, vous pouvez spécifier un mot de passe pour chiffrer le fichier PKCS12 local qui contient les données du magasin de clés. Lorsque vous créez le AWS CloudHSM Keystore, vous définissez le mot de passe et vous le fournissez lorsque vous utilisez les méthodes load, set et get.

Instanciez un nouvel objet KeyStore CloudHSM comme suit :

```
ks.load(null, null);
```

Écrivez les données du keystore dans un fichier à l'aide de la méthode `store`. À partir de ce moment, vous pouvez charger le keystore existant en utilisant la méthode `load` avec le fichier source et le mot de passe comme suit :

```
ks.load(inputStream, password);
```

En utilisant AWS CloudHSM KeyStore

Un objet KeyStore CloudHSM est généralement utilisé par le biais d'une application tierce telle que jarsigner ou keytool. Vous pouvez également accéder directement à l'objet avec le code.

AWS CloudHSM KeyStore est conforme aux KeyStore spécifications de la classe JCE et fournit les fonctions suivantes.

- `load`

  Charge le magasin de clés à partir du flux d'entrée donné. Si un mot de passe a été défini lors de l'enregistrement du magasin de clés, ce même mot de passe doit être fourni pour que le chargement réussisse. Définissez les deux paramètres sur null pour initialiser un nouveau magasin de clés vide.

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  ks.load(inputStream, password);
  ```

- `aliases`

  Renvoie une énumération des noms d'alias de toutes les entrées de l'instance de magasin de clés donnée. Les résultats incluent les objets stockés localement dans le fichier PKCS12 et les objets résidant sur le HSM.

  Exemple de code :

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  for(Enumeration<String> entry = ks.aliases(); entry.hasMoreElements();)
   {
    String label = entry.nextElement();
    System.out.println(label);
   }
  ```

- `ContainsAlias`

Renvoie true si le magasin de clés a accès à au moins un objet avec l'alias spécifié. Le magasin de clés vérifie les objets stockés localement dans le fichier PKCS12 et les objets résidant sur le HSM.

- `DeleteEntry`

  Supprime une entrée de certificat du fichier PKCS12 local. La suppression de données clés stockées dans un HSM n'est pas prise en charge à l'aide du AWS CloudHSM KeyStore. Vous pouvez supprimer des clés avec l'outil key_mgmt_util de CloudHSM.

- `GetCertificate`

  Renvoie le certificat associé à un alias le cas échéant. Si l'alias n'existe pas ou fait référence à un objet qui n'est pas un certificat, la fonction renvoie NULL.

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  Certificate cert = ks.getCertificate(alias)
  ```

- `GetCertificateAlias`

  Renvoie le nom (alias) de la première entrée de magasin de clés dont les données correspondent au certificat donné.

  ```
  KeyStore ks = KeyStore.getInstance("CloudHSM");
  String alias = ks.getCertificateAlias(cert)
  ```

- `GetCertificateChain`

  Renvoie la chaîne de certificats associée à l'alias donné. Si l'alias n'existe pas ou fait référence à un objet qui n'est pas un certificat, la fonction renvoie NULL.

- `GetCreationDate`

  Renvoie la date de création de l'entrée identifiée par l'alias donné. Si aucune date de création n'est disponible, la fonction renvoie la date à laquelle le certificat est devenu valide.

- `GetKey`

  GetKey est transmis au HSM et renvoie un objet clé correspondant à l'étiquette donnée. Comme il interroge `getKey` directement le HSM, il peut être utilisé pour n'importe quelle clé du HSM, qu'elle ait été générée ou non par le. KeyStore

  ```
  Key key = ks.getKey(keyLabel, null);
  ```

- `IsCertificateEntry`

  Vérifie si l'entrée avec l'alias donné représente une entrée de certificat.

- `IsKeyEntry`

  Vérifie si l'entrée avec l'alias donné représente une entrée de clé. L'action recherche l'alias dans le fichier PKCS12 et le HSM.

- `SetCertificateEntry`

  Affecte le certificat donné à l'alias donné. Si l'alias donné est déjà utilisé pour identifier une clé ou un certificat, une `KeyStoreException` est levée. Vous pouvez utiliser le code JCE pour obtenir l'objet clé, puis utiliser la KeyStore `SetKeyEntry` méthode pour associer le certificat à la clé.

- `SetKeyEntry` avec la clé `byte[]`

  Cette API n'est actuellement pas prise en charge par le SDK client 3.

- `SetKeyEntry` avec l'objet `Key`

  Affecte la clé donnée à l'alias donné et la stocke dans le HSM. Si l'objet `Key` n'est pas de type `CaviumKey`, la clé est importée dans le HSM en tant que clé de session extractible.

  Si l'objet `Key` est de type `PrivateKey`, il doit être accompagné d'une chaîne de certificats correspondante.

  Si l'alias existe déjà, l'appel `SetKeyEntry` lance un `KeyStoreException` et empêche la clé d'être écrasée. Si la clé doit être écrasée, utilisez KMU ou JCE à cet effet.

- `EngineSize`

  Renvoie le nombre d'entrées dans le keystore.

- `Store`

  Stocke le magasin de clés dans le flux de sortie donné sous la forme d'un fichier PKCS12 et le sécurise avec le mot de passe donné. En outre, il persiste toutes les clés chargées (qui sont définies en utilisant des appels `setKey`).

Intégration des applications tierces à AWS CloudHSM

Certains des cas d'utilisation AWS CloudHSM impliquent l'intégration d'applications logicielles tierces au HSM de votre AWS CloudHSM cluster. En intégrant des logiciels tiers AWS CloudHSM, vous pouvez atteindre divers objectifs liés à la sécurité. Les rubriques suivantes décrivent comment atteindre certains de ces objectifs.

Rubriques

- Améliorez la sécurité de votre serveur Web grâce au déchargement SSL/TLS dans AWS CloudHSM
- Configuration de Windows Server en tant qu'autorité de certification (CA) avec AWS CloudHSM
- Chiffrement transparent des données (TDE) des bases de données Oracle avec AWS CloudHSM
- Utiliser Microsoft SignTool with AWS CloudHSM pour signer des fichiers
- Java Keytool et Jarsigner
- Autres intégrations de fournisseur tiers

Améliorez la sécurité de votre serveur Web grâce au déchargement SSL/TLS dans AWS CloudHSM

Les serveurs Web et leurs clients (navigateurs Web) peuvent utiliser les protocoles SSL (Secure Sockets Layer) ou TLS (Transport Layer Security) pour confirmer l'identité du serveur Web et établir une connexion sécurisée qui envoie et reçoit des pages Web ou d'autres données sur Internet. C'est ce que l'on appelle communément HTTPS. Le serveur web utilise une paire clé privée-clé publique et un certificat de clé publique SSL/TLS pour établir une session HTTPS avec chaque client. Ce processus implique de nombreux calculs pour les serveurs Web, mais vous pouvez en décharger une partie sur votre AWS CloudHSM cluster, ce que l'on appelle l'accélération SSL. Le déchargement réduit la charge de calcul qui s'exerce sur vos serveurs web et offre une sécurité supplémentaire en stockant les clés privées des serveurs dans des HSM.

Les rubriques suivantes fournissent une vue d'ensemble du fonctionnement du déchargement SSL/TLS et des AWS CloudHSM didacticiels pour configurer le déchargement SSL/TLS sur les plateformes suivantes. AWS CloudHSM

Pour Linux, utilisez le moteur dynamique OpenSSL sur le logiciel de serveur Web NGINX ou Apache HTTP Server

Pour Windows, utilisez le logiciel de serveur web Internet Information Services (IIS) for Windows Server

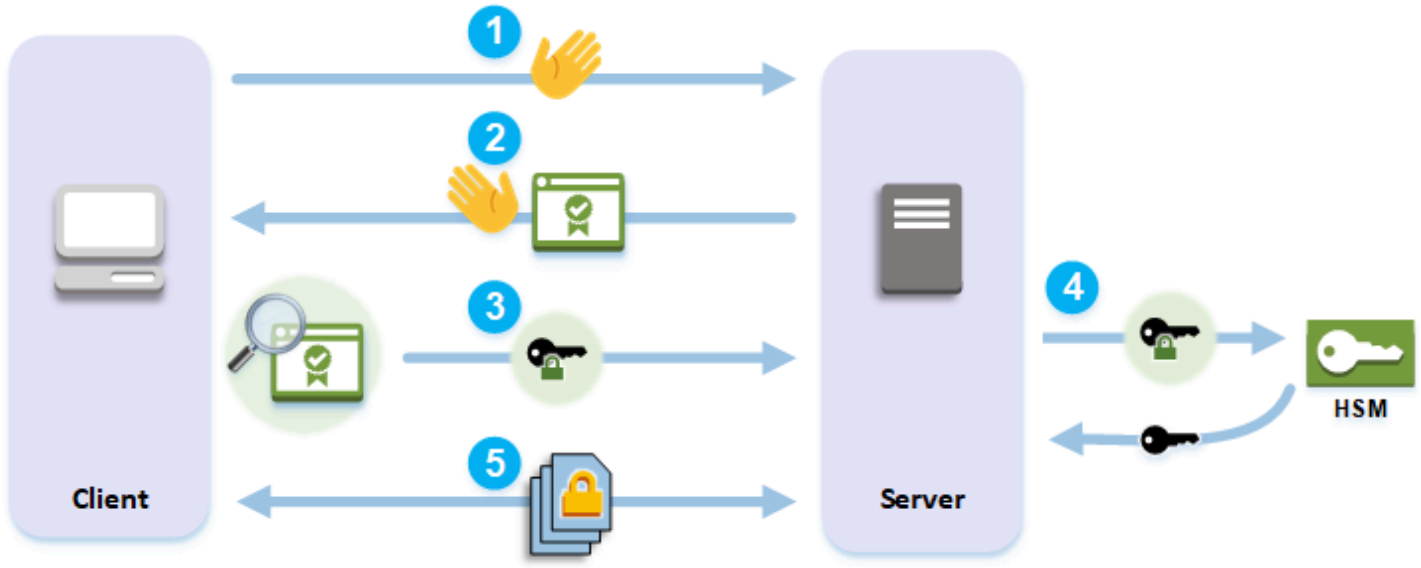
Rubriques

- Comment fonctionne le déchargement SSL/TLS AWS CloudHSM
- Déchargement SSL/TLS sur Linux
- Utilisation d'IIS avec CNG pour le déchargement SSL/TLS sous Windows
- Ajouter un équilibreur de charge avec Elastic Load Balancing (facultatif)

Comment fonctionne le déchargement SSL/TLS AWS CloudHSM

Pour établir une connexion HTTPS, votre serveur web effectue un processus de négociation avec les clients. Dans le cadre de ce processus, le serveur décharge une partie du traitement cryptographique sur les HSM, comme indiqué dans la figure suivante. Chaque étape du processus est expliquée sous la figure.

> **Note**
>
> L'image et le processus suivants supposent que RSA est utilisé pour la vérification du serveur et l'échange de clés. Le processus est légèrement différent lorsque Diffie-Hellman est utilisé à la place de RSA.



1. Le client envoie un message Hello au serveur.
2. Le serveur répond par un message Hello et envoie son certificat de serveur.
3. Le client effectue les actions suivantes :
   a. Il vérifie que le certificat du serveur SSL/TLS est signé par un certificat racine auquel il fait confiance.
   b. Il extrait la clé publique du certificat du serveur.
   c. Il génère un prémaster secret et le chiffre avec la clé publique du serveur.
   d. Il envoie le prémaster secret chiffré au serveur.
4. Pour déchiffrer le prémaster secret du client, le serveur l'envoie au HSM. Le HSM utilise la clé privée dans le HSM pour déchiffrer le prémaster secret, puis il envoie le prémaster secret au serveur. Indépendamment, le client et le serveur utilisent tous les deux le prémaster secret et certaines informations issues des messages Hello pour calculer un secret principal.
5. Le processus de négociation se termine. Dans le reste de la session, tous les messages envoyés entre le client et le serveur sont chiffrés avec des dérivés du secret principal.

Pour savoir comment configurer le déchargement SSL/TLS avec AWS CloudHSM, consultez l'une des rubriques suivantes :

- Déchargement SSL/TLS sur Linux
- Utilisation d'IIS avec CNG pour le déchargement SSL/TLS sous Windows

Déchargement SSL/TLS sur Linux

Avec AWS CloudHSM, vous pouvez effectuer un déchargement SSL/TLS sous Linux avec NGINX, Apache et Tomcat. For more information, see the related topics below.

Rubriques

- Utiliser NGINX ou Apache avec OpenSSL pour le déchargement SSL/TLS sous Linux
- Utilisation de Tomcat avec JSSE pour le déchargement SSL/TLS sous Linux

Utiliser NGINX ou Apache avec OpenSSL pour le déchargement SSL/TLS sous Linux

Cette rubrique fournit des step-by-step instructions pour configurer le déchargement SSL/TLS AWS CloudHSM sur un serveur Web Linux.

Rubriques

Présentation

Sur Linux, les applications serveur web NGINX et Apache HTTP Server s'intègrent à OpenSSL pour prendre en charge le protocole HTTPS. Le moteur dynamique AWS CloudHSM pour OpenSSL fournit une interface qui permet à l'application serveur web d'utiliser les HSM de votre cluster pour le déchargement cryptographique et le stockage de clé. Le moteur OpenSSL est le pont qui relie le serveur web à votre cluster AWS CloudHSM .

Pour suivre ce didacticiel, vous devez d'abord choisir d'utiliser l'application serveur web NGINX ou Apache sur Linux. Ce didacticiel vous montre ensuite comment effectuer les opérations suivantes :

- Installez le logiciel du serveur web sur une instance Amazon EC2.
- Configurez le logiciel du serveur Web pour qu'il prenne en charge le protocole HTTPS avec une clé privée stockée dans votre cluster AWS CloudHSM .
- (Facultatif) Utilisez Amazon EC2 pour créer une deuxième instance de serveur Web et Elastic Load Balancing pour créer un équilibreur de charge. L'utilisation d'un équilibreur de charge peut accroître les performances en répartissant la charge sur plusieurs serveurs. Elle peut également assurer la redondance et une meilleure disponibilité en cas de défaillance d'un ou plusieurs serveurs.

Lorsque vous êtes prêt à commencer, consultez Étape 1 : Configurer les prérequis.

Étape 1 : Configurer les prérequis

Les différentes plateformes ont des prérequis différents. Utilisez la section des prérequis ci-dessous qui correspond à votre plateforme.

Rubriques

Prérequis pour le SDK client 5

Pour configurer le déchargement SSL/TLS du serveur web avec le SDK client 5, vous avez besoin des prérequis suivants :

- Un AWS CloudHSM cluster actif avec au moins deux modules de sécurité matériels (HSM)

> **Note**
>
> Vous pouvez utiliser un seul cluster HSM, mais vous devez d'abord désactiver la durabilité des clés client. Pour plus d'informations, voir Gérer les paramètres de durabilité des clés client et Outil de configuration du SDK client 5.

- Une instance Amazon EC2 exécutant un système d'exploitation Linux avec les logiciels suivants installés :
  - Un serveur Web (NGINX ou Apache)
  - Le moteur dynamique OpenSSL pour le SDK client 5
- Un utilisateur de chiffrement (CU) propriétaire et gestionnaire de la clé privée du serveur web sur le HSM.

Pour configurer une instance de serveur web Linux et créer un utilisateur de chiffrement sur le HSM

1. Installez et configurez le moteur dynamique OpenSSL pour. AWS CloudHSM Pour plus d'informations sur l'installation du moteur dynamique OpenSSL, consultez OpenSSL Dynamic Engine for Client SDK 5.
2. Sur une instance Linux EC2 ayant accès à votre cluster, installez le serveur Web NGINX ou Apache :

   Amazon Linux

   - NGINX

     ```
     $ sudo yum install nginx
     ```

   - Apache

     ```
     $ sudo yum install httpd24 mod24_ssl
     ```

Amazon Linux 2

- Pour plus d'informations sur le téléchargement de la dernière version de NGINX sur Amazon Linux 2, consultez le site Web de NGINX.

  La dernière version de NGINX disponible pour Amazon Linux 2 utilise une version d'OpenSSL plus récente que la version système d'OpenSSL. Après avoir installé NGINX, vous devez créer un lien symbolique depuis la bibliothèque AWS CloudHSM OpenSSL Dynamic Engine vers l'emplacement attendu par cette version d'OpenSSL

  ```
  $ 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

- Pour plus d'informations sur le téléchargement de la dernière version de NGINX sur CentOS 7, consultez le site Web de NGINX.

  La dernière version de NGINX disponible pour CentOS 7 utilise une version d'OpenSSL plus récente que la version système d'OpenSSL. Après avoir installé NGINX, vous devez créer un lien symbolique depuis la bibliothèque AWS CloudHSM OpenSSL Dynamic Engine vers l'emplacement attendu par cette version d'OpenSSL

  ```
  $ 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

- Pour plus d'informations sur le téléchargement de la dernière version de NGINX sur Red Hat 7, consultez le site Web de NGINX.

  La dernière version de NGINX disponible pour Red Hat 7 utilise une version d'OpenSSL plus récente que la version système d'OpenSSL. Après avoir installé NGINX, vous devez créer un lien symbolique depuis la bibliothèque AWS CloudHSM OpenSSL Dynamic Engine vers l'emplacement attendu par cette version d'OpenSSL

  ```
  $ 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

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

3. Utilisez la CLI CloudHSM pour créer un CU. Pour plus d'informations sur la gestion des utilisateurs HSM, consultez la section Gestion des utilisateurs HSM avec la CLI CloudHSM.

> **Tip**
>
> Conservez le nom d'utilisateur et le mot de passe du CU. Vous en aurez besoin plus tard pour générer ou importer la clé privée HTTPS et le certificat de votre serveur web.

Une fois que vous avez terminé ces étapes, consultez Étape 2 : importer ou générer une clé privée et un certificat SSL/TLS.

Remarques

- Pour utiliser Security-Enhanced Linux (SELinux) et les serveurs Web, vous devez autoriser les connexions TCP sortantes sur le port 2223, qui est le port utilisé par le SDK client 5 pour communiquer avec le HSM.
- Pour créer et activer un cluster et donner à une instance EC2 l'accès au cluster, suivez les étapes décrites dans Démarrer avec AWS CloudHSM. La section Getting Started fournit des step-by-step instructions pour créer un cluster actif avec un HSM et une instance client Amazon EC2. Vous pouvez utiliser cette instance client comme serveur web.
- Pour éviter de désactiver la durabilité des clés client, ajoutez plusieurs HSM à votre cluster. Pour plus d'informations, consultez Ajout d'un HSM.
- Vous pouvez utiliser un SSH ou PuTTY pour vous connecter à votre instance client. Pour plus d'informations, consultez Connexion à votre instance Linux à l'aide de SSH ou Connexion à votre instance Linux à partir de Windows à l'aide de PuTTY dans la documentation Amazon EC2.

Prérequis pour le SDK client 3

Pour configurer le déchargement SSL/TLS du serveur web avec le SDK client 3, vous avez besoin des prérequis suivants :

- Un AWS CloudHSM cluster actif avec au moins un HSM.
- Une instance Amazon EC2 exécutant un système d'exploitation Linux avec les logiciels suivants installés :
  - Le AWS CloudHSM client et les outils de ligne de commande.
  - Application de serveur web NGINX ou Apache.
  - Le moteur AWS CloudHSM dynamique d'OpenSSL.
- Un utilisateur de chiffrement (CU) propriétaire et gestionnaire de la clé privée du serveur web sur le HSM.

Pour configurer une instance de serveur web Linux et créer un utilisateur de chiffrement sur le HSM

1. Suivez les étapes de Premiers pas. Vous disposerez ensuite d'un cluster actif avec un HSM et une instance client Amazon EC2. Votre instance EC2 sera configurée avec les outils de ligne de commande. Utilisez cette instance client comme serveur web.

2. Connectez-vous à votre instance client . Pour plus d'informations, consultez Connexion à votre instance Linux à l'aide de SSH ou Connexion à votre instance Linux à partir de Windows à l'aide de PuTTY dans la documentation Amazon EC2.
3. Sur une instance Linux EC2 ayant accès à votre cluster, installez le serveur Web NGINX ou Apache :

   Amazon Linux

   - NGINX

     ```
     $ sudo yum install nginx
     ```

   - Apache

     ```
     $ sudo yum install httpd24 mod24_ssl
     ```

   Amazon Linux 2

   - La version 1.19 de NGINX est la dernière version de NGINX compatible avec le moteur Client SDK 3 sur Amazon Linux 2.

     Pour plus d'informations et pour télécharger la version 1.19 de NGINX, consultez le site Web de NGINX.

   - Apache

     ```
     $ sudo yum install httpd mod_ssl
     ```

   CentOS 7

   - La version 1.19 de NGINX est la dernière version de NGINX compatible avec le moteur Client SDK 3 sur CentOS 7.

     Pour plus d'informations et pour télécharger la version 1.19 de NGINX, consultez le site Web de NGINX.

   - Apache

     ```
     $ sudo yum install httpd mod_ssl
     ```

Red Hat 7

- La version 1.19 de NGINX est la dernière version de NGINX compatible avec le moteur Client SDK 3 sur Red Hat 7.

  Pour plus d'informations et pour télécharger la version 1.19 de NGINX, consultez le site Web de NGINX.

- 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. (Facultatif) Ajoutez d'autres HSM à votre cluster. Pour plus d'informations, consultez Ajout d'un HSM.
5. Utilisez cloudhsm_mgmt_util pour créer un CU. Pour plus d'informations, consultez Gestion des utilisateurs HSM. Conservez le nom d'utilisateur et le mot de passe du CU. Vous en aurez besoin plus tard pour générer ou importer la clé privée HTTPS et le certificat de votre serveur web.

Une fois que vous avez terminé ces étapes, consultez Étape 2 : importer ou générer une clé privée et un certificat SSL/TLS.

Étape 2 : importer ou générer une clé privée et un certificat SSL/TLS

Pour activer le protocole HTTPS, votre application de serveur web (NGINX ou Apache) nécessite une clé privée et un certificat SSL/TLS correspondant. Pour utiliser le déchargement SSL/TLS du serveur Web avec AWS CloudHSM, vous devez stocker la clé privée dans un HSM de votre cluster. AWS CloudHSM Vous pouvez effectuer cette opération de différentes manières :

- Si vous ne disposez pas encore d'une clé privée et d'un certificat correspondant, générez une clé privée dans un HSM. Vous utilisez la clé privée pour créer une demande de signature de certificat (CSR), que vous utilisez pour créer le certificat SSL/TLS.
- Si vous disposez déjà d'une clé privée et du certificat correspondant, importez la clé privée dans un HSM.

Quelle que soit la méthode choisie ci-dessus, vous exportez une fausse clé privée PEM depuis le HSM, qui est un fichier de clé privée au format PEM contenant une référence à la clé privée stockée sur le HSM (il ne s'agit pas de la véritable clé privée). Votre serveur Web utilise le faux fichier de clé privée PEM pour identifier la clé privée sur le HSM lors du déchargement SSL/TLS.

Effectuez l'une des actions suivantes :

- Générer une clé privée et un certificat
- Importer une clé privée et un certificat existants

Générer une clé privée et un certificat

Générer une clé privée

Cette section explique comment générer une paire de clés à l'aide de l'Utilitaire de gestion des clés (KMU) à partir du SDK client 3. Une fois qu'une paire de clés est générée dans le HSM, vous pouvez l'exporter sous forme de faux fichier PEM et générer le certificat correspondant.

Les clés privées générées à l'aide de l'Utilitaire de gestion des clés (KMU) peuvent être utilisées à la fois avec le SDK client 3 et le SDK client 5.

Installation et configuration de l'Utilitaire de gestion des clés (KMU)

1. Connectez-vous à votre instance client .

2. Installez et configurez le SDK client 3.
3. Exécutez la commande suivante pour démarrer le AWS CloudHSM client.

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

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

4. Exécutez la commande suivante pour démarrer l'outil de ligne de commande key_mgmt_util.

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. Exécutez la commande suivante pour vous connecter au HSM. Remplacez *<user name>* et *<password>* par le nom d'utilisateur et le mot de passe de l'utilisateur de chiffrement (CU).

```
Command: loginHSM -u CU -s <user name> -p <password>>
```

Générer une clé privée

En fonction de votre cas d'utilisation, vous pouvez générer une paire de clés RSA ou EC. Effectuez l'une des actions suivantes :

- Pour générer une clé privée RSA sur un HSM

  Utilisez la commande `genRSAKeyPair` pour générer une paire de clés RSA. Cet exemple génère une paire de clés RSA avec un module de 2048, un exposant public de 65537 et une étiquette *`tls_rsa_keypair`*.

```
Command: genRSAKeyPair -m 2048 -e 65537 -l tls_rsa_keypair
```

  Si la commande s'est correctement déroulée, vous devriez voir le résultat suivant indiquant que vous avez correctement généré une paire de clés RSA.

```
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
```

- Pour générer une clé privée EC sur un HSM

  Utilisez la commande `genECCKeyPair` pour générer une paire de clés EC. Cet exemple génère une paire de clés EC avec un ID de courbe de 2 (correspondant à la courbe `NID_X9_62_prime256v1`) et une étiquette *`tls_ec_keypair`*.

  ```
  Command: genECCKeyPair -i 2 -l tls_ec_keypair
  ```

  Si la commande s'est correctement déroulée, vous devriez voir le résultat suivant indiquant que vous avez correctement généré une paire de clés EC.

  ```
  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
  ```

Exporter un faux fichier de clé privée PEM

Une fois que vous avez une clé privée sur le HSM, vous devez exporter un faux fichier de clé privée PEM. Ce fichier ne contient pas les données clés réelles, mais il permet au moteur dynamique OpenSSL d'identifier la clé privée sur le HSM. Vous pouvez ensuite utiliser la clé privée pour créer une demande de signature de certificat (CSR) et signer la CSR pour créer le certificat.

> **Note**
>
> Les faux fichiers PEM générés à l'aide de l'Utilitaire de gestion des clés (KMU) peuvent être utilisés à la fois avec le SDK client 3 et le SDK client 5.

Identifiez le handle de clé correspondant à la clé que vous souhaitez exporter en tant que faux PEM, puis exécutez la commande suivante pour exporter la clé privée au faux format PEM et l'enregistrer dans un fichier. Remplacez les valeurs suivantes par vos propres valeurs :

- *`<private_key_handle>`* – Handle de la clé privée générée. Ce handle a été généré par l'une des commandes de génération de clé de l'étape précédente. Dans l'exemple précédent, le handle de la clé privée est 8.

- *<web_server_fake_PEM.key>* – Nom du fichier dans lequel votre fausse clé PEM sera écrite.

```
Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
```

Quitter

Exécutez la commande suivante pour arrêter le key_mgmt_util.

```
Command: exit
```

Vous devriez maintenant avoir un nouveau fichier sur votre système, situé sur le chemin indiqué par *<web_server_fake_PEM.key>* dans la commande précédente. Ce fichier est le faux fichier de clé privée PEM.

Générer un certificat signé automatiquement.

Une fois que vous avez généré une fausse clé privée PEM, vous pouvez utiliser ce fichier pour générer une demande de signature de certificat (CSR) et un certificat.

Dans un environnement de production, vous utilisez généralement une autorité de certification (CA) pour créer un certificat émis par une demande de signature de certificat (CSR). L'autorité de certification n'est pas nécessaire pour un environnement de test. Si vous utilisez une autorité de certification, envoyez-lui le fichier CSR et utilisez le certificat SSL/TLS signé qu'elle vous fournit sur votre serveur Web pour HTTPS.

Au lieu d'utiliser une autorité de certification, vous pouvez utiliser le moteur dynamique AWS CloudHSM OpenSSL pour créer un certificat auto-signé. Les certificats auto-signés ne sont pas approuvées par les navigateurs et ne doivent pas être utilisés dans les environnements de production. Ils peuvent cependant être utilisés dans les environnements de test.

> **Warning**
>
> Les certificats auto-signés doivent uniquement être utilisés dans un environnement de test. Pour un environnement de production, utilisez une méthode plus sécurisée telle qu'une autorité de certification pour créer un certificat.

Installer et configurer le moteur dynamique OpenSSL

1. Connectez-vous à votre instance client .
2. Pour installer et configurer, réalisez l'un des éléments suivants :
   - the section called "Installation du moteur dynamique OpenSSL"
   - the section called "OpenSSL Dynamic Engine"

Générez un certificat

1. Obtenez une copie de votre faux fichier PEM généré lors d'une étape précédente.
2. Création d'une CSR

   Exécutez la commande suivante pour utiliser le moteur dynamique AWS CloudHSM OpenSSL afin de créer une demande de signature de certificat (CSR). Remplacez *<web_server_fake_PEM.key>* par le nom du fichier qui contient votre fausse clé privée PEM. Remplacez *<web_server.csr>* par le nom du fichier qui contient votre demande CSR.

   La commande `req` est interactive. Renseignez chaque champ. Les informations du champ sont copiées dans votre certificat SSL/TLS.

   ```
   $ openssl req -engine cloudhsm -new -key <web_server_fake_PEM.key> -
   out <web_server.csr>
   ```
3. Créer un certificat auto-signé

   Exécutez la commande suivante pour utiliser le moteur dynamique AWS CloudHSM OpenSSL afin de signer votre CSR avec votre clé privée sur votre HSM. Cette opération crée un certificat auto-signé. Remplacez les valeurs suivantes par vos propres valeurs dans la commande :

   - *<web_server.csr>* – Nom du fichier qui contient la CSR.
   - *<web_server_fake_PEM.key>* – Nom du fichier qui contient la fausse clé privée PEM.
   - *<web_server.crt>* – Nom du fichier qui contiendra votre certificat de serveur web.

   ```
   $ openssl x509 -engine cloudhsm -req -days 365 -in <web_server.csr> -
   signkey <web_server_fake_PEM.key> -out <web_server.crt>
   ```

Une fois que vous avez terminé ces étapes, consultez Étape 3 : Configurer le serveur web.

Importer une clé privée et un certificat existants

Il est possible que vous disposiez déjà d'une clé privée et d'un certificat SSL/TLS correspondant que vous utilisez pour les connexions HTTPS sur votre serveur web. Si tel est le cas, vous pouvez importer cette clé dans un HSM en suivant les étapes de cette section.

> **Note**
>
> Quelques remarques sur les importations de clés privées et la compatibilité avec le SDK client :
>
> - L'importation d'une clé privée existante nécessite le SDK client 3.
> - Vous pouvez utiliser les clés privées du SDK client 3 avec le SDK client 5.
> - OpenSSL Dynamic Engine for Client SDK 3 ne prend pas en charge les dernières plateformes Linux, mais l'implémentation d'OpenSSL Dynamic Engine for Client SDK 5 le fait. Vous pouvez importer une clé privée existante à l'aide de l'Utilitaire de gestion des clés (KMU) fourni avec le SDK client 3, puis utiliser cette clé privée et l'implémentation du moteur dynamique OpenSSL avec le SDK client 5 pour prendre en charge le déchargement SSL/TLS sur les dernières plateformes Linux.

Pour importer une clé privée existante dans un HSM avec le SDK client 3

1. Connectez-vous à votre instance client EC2 Amazon. Si nécessaire, copiez votre clé privée et le certificat sur l'instance.
2. Installez et configurez le SDK client 3
3. Exécutez la commande suivante pour démarrer le AWS CloudHSM client.

   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

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

4. Exécutez la commande suivante pour démarrer l'outil de ligne de commande key_mgmt_util.

```
$ /opt/cloudhsm/bin/key_mgmt_util
```

5. Exécutez la commande suivante pour vous connecter au HSM. Remplacez *<user name>* et *<password>* par le nom d'utilisateur et le mot de passe de l'utilisateur de chiffrement (CU).

```
Command: loginHSM -u CU -s <user name> -p <password>
```

6. Exécutez les commandes suivantes pour importer votre clé privée dans un HSM.

   a. Exécutez la commande suivante pour créer une clé d'encapsulage symétrique valable uniquement pour la session en cours. La commande et la sortie sont affichées.

```
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. Exécutez la commande suivante pour importer votre clé privée existante dans un HSM. La commande et la sortie sont affichées. Remplacez les valeurs suivantes par vos propres valeurs :

   - *<web_server_existing.key>* – Nom du fichier qui contient votre clé privée.
   - *<web_server_imported_key>* – Étiquette de votre clé privée importée.
   - *<wrapping_key_handle>* – Handle de la clé d'encapsulage générée par la commande précédente.d Dans l'exemple précédent, le handle de clé d'encapsulage est 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. Exécutez la commande suivante pour exporter la fausse clé privée au format PEM et l'enregistrer dans un fichier. Remplacez les valeurs suivantes par vos propres valeurs :

- *<private_key_handle>* – Handle de la clé privée importée. Ce handle a été généré par la deuxième commande de l'étape précédente. Dans l'exemple précédent, le handle de la clé privée est 8.
- *<web_server_fake_PEM.key>* – Nom du fichier qui contient votre fausse clé privée PEM exportée.

```
Command: getCaviumPrivKey -k <private_key_handle> -out <web_server_fake_PEM.key>
```

8. Exécutez la commande suivante pour arrêter key_mgmt_util.

```
Command: exit
```

Une fois que vous avez terminé ces étapes, consultez Étape 3 : Configurer le serveur web.

Étape 3 : Configurer le serveur web

Mettez à jour votre configuration de logiciel serveur web pour utiliser le certificat HTTPS et la fausse clé privée PEM correspondante que vous avez créée à l'étape précédente. N'oubliez pas de sauvegarder vos certificats et clés existants avant de commencer. Cela permettra de terminer la configuration de votre logiciel serveur web Linux pour le déchargement SSL/TLS avec AWS CloudHSM.

Complétez les étapes de l'une des sections suivantes.

Rubriques

- Configuration du serveur Web NGINX
- Configurer le serveur web Apache

Configuration du serveur Web NGINX

Utilisez cette section pour configurer NGINX sur les plateformes prises en charge.

Pour mettre à jour la configuration de serveur web pour NGINX

1. Connectez-vous à votre instance client .
2. Exécutez la commande suivante pour créer les répertoires requis pour le certificat de serveur web et la fausse clé privée PEM.

```
$ sudo mkdir -p /etc/pki/nginx/private
```

3. Exécutez la commande suivante pour copier votre certificat de serveur web à l'emplacement requis. Remplacez *<web_server.crt>* par le nom de votre certificat de serveur web.

```
$ sudo cp <web_server.crt> /etc/pki/nginx/server.crt
```

4. Exécutez la commande suivante pour copier votre fausse clé privée PEM à l'emplacement requis. Remplacez *<web_server_fake_PEM.key>* par le nom du fichier qui contient votre fausse clé privée PEM.

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/nginx/private/server.key
```

5. Exécutez la commande suivante pour modifier la propriété des fichiers afin que l'utilisateur nommé nginx puisse les lire.

```
$ sudo chown nginx /etc/pki/nginx/server.crt /etc/pki/nginx/private/server.key
```

6. Exécutez la commande suivante pour sauvegarder le fichier `/etc/nginx/nginx.conf`.

```
$ sudo cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
```

7. Mise à jour de la configuration pour NGINX.

> **Note**
>
> Chaque cluster peut prendre en charge un maximum de 1 000 processus de travail NGINX sur tous les serveurs Web NGINX.

Amazon Linux

Utilisez un éditeur de texte pour modifier le fichier `/etc/nginx/nginx.conf`. Ceci nécessite des autorisations racine Linux. En haut du fichier, ajoutez les lignes suivantes :

- Si vous utilisez le SDK client 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Si vous utilisez le SDK client 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Ensuite, ajoutez ce qui suit à la section TLS du fichier :

```
# 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

Utilisez un éditeur de texte pour modifier le fichier `/etc/nginx/nginx.conf`. Ceci nécessite des autorisations racine Linux. En haut du fichier, ajoutez les lignes suivantes :

- Si vous utilisez le SDK client 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Si vous utilisez le SDK client 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Ensuite, ajoutez ce qui suit à la section TLS du fichier :

```
# 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

Utilisez un éditeur de texte pour modifier le fichier `/etc/nginx/nginx.conf`. Ceci nécessite des autorisations racine Linux. En haut du fichier, ajoutez les lignes suivantes :

- Si vous utilisez le SDK client 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Si vous utilisez le SDK client 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Ensuite, ajoutez ce qui suit à la section TLS du fichier :

```
# 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

Utilisez un éditeur de texte pour modifier le fichier `/etc/nginx/nginx.conf`. Ceci nécessite des autorisations racine Linux. En haut du fichier, ajoutez les lignes suivantes :

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Ensuite, ajoutez ce qui suit à la section TLS du fichier :

```
# 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

Utilisez un éditeur de texte pour modifier le fichier `/etc/nginx/nginx.conf`. Ceci nécessite des autorisations racine Linux. En haut du fichier, ajoutez les lignes suivantes :

- Si vous utilisez le SDK client 3

```
ssl_engine cloudhsm;
env n3fips_password;
```

- Si vous utilisez le SDK client 5

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Ensuite, ajoutez ce qui suit à la section TLS du fichier :

```
# 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

Utilisez un éditeur de texte pour modifier le fichier `/etc/nginx/nginx.conf`. Ceci nécessite des autorisations racine Linux. En haut du fichier, ajoutez les lignes suivantes :

```
ssl_engine cloudhsm;
env CLOUDHSM_PIN;
```

Ensuite, ajoutez ce qui suit à la section TLS du fichier :

```
# 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

Utilisez un éditeur de texte pour modifier le fichier `/etc/nginx/nginx.conf`. Ceci nécessite des autorisations racine Linux. En haut du fichier, ajoutez les lignes suivantes :

```
ssl_engine cloudhsm;
    env n3fips_password;
```

Ensuite, ajoutez ce qui suit à la section TLS du fichier :

```
# 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

Utilisez un éditeur de texte pour modifier le fichier `/etc/nginx/nginx.conf`. Ceci nécessite des autorisations racine Linux. En haut du fichier, ajoutez les lignes suivantes :

```
ssl_engine cloudhsm;
    env CLOUDHSM_PIN;
```

Ensuite, ajoutez ce qui suit à la section TLS du fichier :

```
# 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

Utilisez un éditeur de texte pour modifier le fichier `/etc/nginx/nginx.conf`. Ceci nécessite des autorisations racine Linux. En haut du fichier, ajoutez les lignes suivantes :

```
ssl_engine cloudhsm;
    env CLOUDHSM_PIN;
```

Ensuite, ajoutez ce qui suit à la section TLS du fichier :

```
# 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

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

Enregistrez le fichier.

8. Sauvegardez le fichier de configuration `systemd`, puis définissez le chemin d'accès à `EnvironmentFile`.

   Amazon Linux

   Aucune action requise.

   Amazon Linux 2

   1. Sauvegardez le fichier `nginx.service`.

      ```
      $ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
      nginx.service.backup
      ```

   2. Ouvrez le fichier `/lib/systemd/system/nginx.service` dans un éditeur de texte, puis sous la section [Service], ajoutez le chemin d'accès suivant :

      ```
      EnvironmentFile=/etc/sysconfig/nginx
      ```

   CentOS 7

   Aucune action requise.

   CentOS 8

   1. Sauvegardez le fichier `nginx.service`.

      ```
      $ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
      nginx.service.backup
      ```

   2. Ouvrez le fichier `/lib/systemd/system/nginx.service` dans un éditeur de texte, puis sous la section [Service], ajoutez le chemin d'accès suivant :

```
EnvironmentFile=/etc/sysconfig/nginx
```

Red Hat 7

Aucune action requise.

Red Hat 8

1. Sauvegardez le fichier `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. Ouvrez le fichier `/lib/systemd/system/nginx.service` dans un éditeur de texte, puis sous la section [Service], ajoutez le chemin d'accès suivant :

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 16.04

1. Sauvegardez le fichier `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. Ouvrez le fichier `/lib/systemd/system/nginx.service` dans un éditeur de texte, puis sous la section [Service], ajoutez le chemin d'accès suivant :

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 18.04

1. Sauvegardez le fichier `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. Ouvrez le fichier `/lib/systemd/system/nginx.service` dans un éditeur de texte, puis sous la section [Service], ajoutez le chemin d'accès suivant :

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 20.04 LTS

1. Sauvegardez le fichier `nginx.service`.

```
$ sudo cp /lib/systemd/system/nginx.service /lib/systemd/system/
nginx.service.backup
```

2. Ouvrez le fichier `/lib/systemd/system/nginx.service` dans un éditeur de texte, puis sous la section [Service], ajoutez le chemin d'accès suivant :

```
EnvironmentFile=/etc/sysconfig/nginx
```

Ubuntu 22.04 LTS

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

9. Vérifiez si le fichier `/etc/sysconfig/nginx` existe, puis effectuez l'une des actions suivantes :

   - Si le fichier existe, sauvegardez le fichier en exécutant la commande suivante :

```
$ sudo cp /etc/sysconfig/nginx /etc/sysconfig/nginx.backup
```

   - Si le fichier n'existe pas, ouvrez un éditeur de texte et créez un fichier nommé `nginx` dans le dossier `/etc/sysconfig/`.

10. Configurez l'environnement NGINX.

> **Note**
>
> Le SDK client 5 introduit la variable d'environnement `CLOUDHSM_PIN` permettant de stocker les informations d'identification du CU.

Amazon Linux

Ouvrez le fichier `/etc/sysconfig/nginx` dans un éditeur de texte. Ceci nécessite des autorisations racine Linux. Ajoutez les informations d'identification de l'utilisateur de chiffrement (CU) :

- Si vous utilisez le SDK client 3

```
n3fips_password=<CU user name>:<password>
```

- Si vous utilisez le SDK client 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Enregistrez le fichier.

Amazon Linux 2

Ouvrez le fichier `/etc/sysconfig/nginx` dans un éditeur de texte. Ceci nécessite des autorisations racine Linux. Ajoutez les informations d'identification de l'utilisateur de chiffrement (CU) :

- Si vous utilisez le SDK client 3

```
n3fips_password=<CU user name>:<password>
```

- Si vous utilisez le SDK client 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Enregistrez le fichier.

CentOS 7

Ouvrez le fichier `/etc/sysconfig/nginx` dans un éditeur de texte. Ceci nécessite des autorisations racine Linux. Ajoutez les informations d'identification de l'utilisateur de chiffrement (CU) :

- Si vous utilisez le SDK client 3

```
n3fips_password=<CU user name>:<password>
```

- Si vous utilisez le SDK client 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Enregistrez le fichier.

CentOS 8

Ouvrez le fichier `/etc/sysconfig/nginx` dans un éditeur de texte. Ceci nécessite des autorisations racine Linux. Ajoutez les informations d'identification de l'utilisateur de chiffrement (CU) :

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Enregistrez le fichier.

Red Hat 7

Ouvrez le fichier `/etc/sysconfig/nginx` dans un éditeur de texte. Ceci nécessite des autorisations racine Linux. Ajoutez les informations d'identification de l'utilisateur de chiffrement (CU) :

- Si vous utilisez le SDK client 3

```
n3fips_password=<CU user name>:<password>
```

- Si vous utilisez le SDK client 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Enregistrez le fichier.

Red Hat 8

Ouvrez le fichier `/etc/sysconfig/nginx` dans un éditeur de texte. Ceci nécessite des autorisations racine Linux. Ajoutez les informations d'identification de l'utilisateur de chiffrement (CU) :

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Enregistrez le fichier.

Ubuntu 16.04 LTS

Ouvrez le fichier `/etc/sysconfig/nginx` dans un éditeur de texte. Ceci nécessite des autorisations racine Linux. Ajoutez les informations d'identification de l'utilisateur de chiffrement (CU) :

```
n3fips_password=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Enregistrez le fichier.

Ubuntu 18.04 LTS

Ouvrez le fichier `/etc/sysconfig/nginx` dans un éditeur de texte. Ceci nécessite des autorisations racine Linux. Ajoutez les informations d'identification de l'utilisateur de chiffrement (CU) :

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Enregistrez le fichier.

Ubuntu 20.04 LTS

Ouvrez le fichier `/etc/sysconfig/nginx` dans un éditeur de texte. Ceci nécessite des autorisations racine Linux. Ajoutez les informations d'identification de l'utilisateur de chiffrement (CU) :

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Enregistrez le fichier.

Ubuntu 22.04 LTS

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

11. Démarrez le serveur web NGINX.

Amazon Linux

Ouvrez le fichier `/etc/sysconfig/nginx` dans un éditeur de texte. Ceci nécessite des autorisations racine Linux. Ajoutez les informations d'identification de l'utilisateur de chiffrement (CU) :

```
$ sudo service nginx start
```

Amazon Linux 2

Arrêtez tout processus NGINX en cours d'exécution

```
$ sudo systemctl stop nginx
```

Rechargez la configuration `systemd` pour récupérer les dernières modifications

```
$ sudo systemctl daemon-reload
```

Démarrez le processus NGINX

```
$ sudo systemctl start nginx
```

CentOS 7

Arrêtez tout processus NGINX en cours d'exécution

```
$ sudo systemctl stop nginx
```

Rechargez la configuration `systemd` pour récupérer les dernières modifications

```
$ sudo systemctl daemon-reload
```

Démarrez le processus NGINX

```
$ sudo systemctl start nginx
```

CentOS 8

Arrêtez tout processus NGINX en cours d'exécution

```
$ sudo systemctl stop nginx
```

Rechargez la configuration `systemd` pour récupérer les dernières modifications

```
$ sudo systemctl daemon-reload
```

Démarrez le processus NGINX

```
$ sudo systemctl start nginx
```

Red Hat 7

Arrêtez tout processus NGINX en cours d'exécution

```
$ sudo systemctl stop nginx
```

Rechargez la configuration `systemd` pour récupérer les dernières modifications

```
$ sudo systemctl daemon-reload
```

Démarrez le processus NGINX

```
$ sudo systemctl start nginx
```

Red Hat 8

Arrêtez tout processus NGINX en cours d'exécution

```
$ sudo systemctl stop nginx
```

Rechargez la configuration `systemd` pour récupérer les dernières modifications

```
$ sudo systemctl daemon-reload
```

Démarrez le processus NGINX

```
$ sudo systemctl start nginx
```

Ubuntu 16.04 LTS

Arrêtez tout processus NGINX en cours d'exécution

```
$ sudo systemctl stop nginx
```

Rechargez la configuration `systemd` pour récupérer les dernières modifications

```
$ sudo systemctl daemon-reload
```

Démarrez le processus NGINX

```
$ sudo systemctl start nginx
```

Ubuntu 18.04 LTS

Arrêtez tout processus NGINX en cours d'exécution

```
$ sudo systemctl stop nginx
```

Rechargez la configuration `systemd` pour récupérer les dernières modifications

```
$ sudo systemctl daemon-reload
```

Démarrez le processus NGINX

```
$ sudo systemctl start nginx
```

Ubuntu 20.04 LTS

Arrêtez tout processus NGINX en cours d'exécution

```
$ sudo systemctl stop nginx
```

Rechargez la configuration `systemd` pour récupérer les dernières modifications

```
$ sudo systemctl daemon-reload
```

Démarrez le processus NGINX

```
$ sudo systemctl start nginx
```

Ubuntu 22.04 LTS

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

12. (Facultatif) Configurez votre plateforme pour démarrer NGINX au démarrage.

Amazon Linux

```
$ sudo chkconfig nginx on
```

Amazon Linux 2

```
$ sudo systemctl enable nginx
```

CentOS 7

Aucune action requise.

CentOS 8

```
$ sudo systemctl enable nginx
```

Red Hat 7

Aucune action requise.

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

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

Après que vous avez mis à jour votre configuration de serveur web, accédez à Étape 4 : Activer le trafic HTTPS et vérifier le certificat.

Configurer le serveur web Apache

Utilisez cette section pour configurer Apache sur les plateformes prises en charge.

Pour mettre à jour la configuration de serveur web pour Apache

1. Connectez-vous à votre instance client EC2 Amazon.
2. Définissez les emplacements par défaut pour les certificats et les clés privées de votre plateforme.

   Amazon Linux

   Dans le fichier `/etc/httpd/conf.d/ssl.conf`, assurez-vous que les valeurs suivantes existent :

   ```
   SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
   SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
   ```

   Amazon Linux 2

   Dans le fichier `/etc/httpd/conf.d/ssl.conf`, assurez-vous que les valeurs suivantes existent :

   ```
   SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
   SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
   ```

CentOS 7

Dans le fichier `/etc/httpd/conf.d/ssl.conf`, assurez-vous que les valeurs suivantes existent :

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

CentOS 8

Dans le fichier `/etc/httpd/conf.d/ssl.conf`, assurez-vous que les valeurs suivantes existent :

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 7

Dans le fichier `/etc/httpd/conf.d/ssl.conf`, assurez-vous que les valeurs suivantes existent :

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Red Hat 8

Dans le fichier `/etc/httpd/conf.d/ssl.conf`, assurez-vous que les valeurs suivantes existent :

```
SSLCertificateFile      /etc/pki/tls/certs/localhost.crt
SSLCertificateKeyFile   /etc/pki/tls/private/localhost.key
```

Ubuntu 16.04 LTS

Dans le fichier `/etc/apache2/sites-available/default-ssl.conf`, assurez-vous que les valeurs suivantes existent :

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 18.04 LTS

Dans le fichier `/etc/apache2/sites-available/default-ssl.conf`, assurez-vous que les valeurs suivantes existent :

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 20.04 LTS

Dans le fichier `/etc/apache2/sites-available/default-ssl.conf`, assurez-vous que les valeurs suivantes existent :

```
SSLCertificateFile      /etc/ssl/certs/localhost.crt
SSLCertificateKeyFile   /etc/ssl/private/localhost.key
```

Ubuntu 22.04 LTS

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

3. Copiez le certificat de votre serveur Web à l'emplacement requis pour votre plateforme.

Amazon Linux

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Remplacez *<web_server.crt>* par le nom de votre certificat de serveur web.

Amazon Linux 2

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Remplacez *<web_server.crt>* par le nom de votre certificat de serveur web.

CentOS 7

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Remplacez *<web_server.crt>* par le nom de votre certificat de serveur web.

CentOS 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Remplacez *<web_server.crt>* par le nom de votre certificat de serveur web.

Red Hat 7

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Remplacez *<web_server.crt>* par le nom de votre certificat de serveur web.

Red Hat 8

```
$ sudo cp <web_server.crt> /etc/pki/tls/certs/localhost.crt
```

Remplacez *<web_server.crt>* par le nom de votre certificat de serveur web.

Ubuntu 16.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

Remplacez *<web_server.crt>* par le nom de votre certificat de serveur web.

Ubuntu 18.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

Remplacez *<web_server.crt>* par le nom de votre certificat de serveur web.

Ubuntu 20.04 LTS

```
$ sudo cp <web_server.crt> /etc/ssl/certs/localhost.crt
```

Remplacez *<web_server.crt>* par le nom de votre certificat de serveur web.

Ubuntu 22.04 LTS

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

4. Copiez votre fausse clé privée PEM à l'emplacement requis pour votre plateforme.

Amazon Linux

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Remplacez *<web_server_fake_PEM.key>* par le nom du fichier qui contient votre fausse clé privée PEM.

Amazon Linux 2

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Remplacez *<web_server_fake_PEM.key>* par le nom du fichier qui contient votre fausse clé privée PEM.

CentOS 7

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Remplacez *<web_server_fake_PEM.key>* par le nom du fichier qui contient votre fausse clé privée PEM.

CentOS 8

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Remplacez *<web_server_fake_PEM.key>* par le nom du fichier qui contient votre fausse clé privée PEM.

Red Hat 7

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Remplacez *<web_server_fake_PEM.key>* par le nom du fichier qui contient votre fausse clé privée PEM.

Red Hat 8

```
$ sudo cp <web_server_fake_PEM.key> /etc/pki/tls/private/localhost.key
```

Remplacez *<web_server_fake_PEM.key>* par le nom du fichier qui contient votre fausse clé privée PEM.

Ubuntu 16.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

Remplacez *<web_server_fake_PEM.key>* par le nom du fichier qui contient votre fausse clé privée PEM.

Ubuntu 18.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

Remplacez *<web_server_fake_PEM.key>* par le nom du fichier qui contient votre fausse clé privée PEM.

Ubuntu 20.04 LTS

```
$ sudo cp <web_server_fake_PEM.key> /etc/ssl/private/localhost.key
```

Remplacez *<web_server_fake_PEM.key>* par le nom du fichier qui contient votre fausse clé privée PEM.

Ubuntu 22.04 LTS

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

5. Changez le propriétaire de ces fichiers si votre plateforme l'exige.

Amazon Linux

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Fournit une autorisation de lecture à l'utilisateur nommé Apache.

Amazon Linux 2

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Fournit une autorisation de lecture à l'utilisateur nommé Apache.

CentOS 7

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Fournit une autorisation de lecture à l'utilisateur nommé Apache.

CentOS 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Fournit une autorisation de lecture à l'utilisateur nommé Apache.

Red Hat 7

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Fournit une autorisation de lecture à l'utilisateur nommé Apache.

Red Hat 8

```
$ sudo chown apache /etc/pki/tls/certs/localhost.crt /etc/pki/tls/private/
localhost.key
```

Fournit une autorisation de lecture à l'utilisateur nommé Apache.

Ubuntu 16.04 LTS

Aucune action requise.

Ubuntu 18.04 LTS

Aucune action requise.

Ubuntu 20.04 LTS

Aucune action requise.

Ubuntu 22.04 LTS

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

6. Configurez les directives Apache pour votre plateforme.

Amazon Linux

Localisez le fichier SSL pour cette plateforme :

```
/etc/httpd/conf.d/ssl.conf
```

Ce fichier contient les directives Apache qui définissent le mode de fonctionnement de votre serveur. Les directives apparaissent sur la gauche, suivies d'une valeur. Utilisez un éditeur de texte pour modifier ce fichier. Ceci nécessite des autorisations racine Linux.

Mettez à jour ou entrez les directives suivantes avec ces valeurs :

```
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
```

Enregistrez le fichier.

Amazon Linux 2

Localisez le fichier SSL pour cette plateforme :

```
/etc/httpd/conf.d/ssl.conf
```

Ce fichier contient les directives Apache qui définissent le mode de fonctionnement de votre serveur. Les directives apparaissent sur la gauche, suivies d'une valeur. Utilisez un éditeur de texte pour modifier ce fichier. Ceci nécessite des autorisations racine Linux.

Mettez à jour ou entrez les directives suivantes avec ces valeurs :

```
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
```

Enregistrez le fichier.

CentOS 7

Localisez le fichier SSL pour cette plateforme :

```
/etc/httpd/conf.d/ssl.conf
```

Ce fichier contient les directives Apache qui définissent le mode de fonctionnement de votre serveur. Les directives apparaissent sur la gauche, suivies d'une valeur. Utilisez un éditeur de texte pour modifier ce fichier. Ceci nécessite des autorisations racine Linux.

Mettez à jour ou entrez les directives suivantes avec ces valeurs :

```
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
```

Enregistrez le fichier.

CentOS 8

Localisez le fichier SSL pour cette plateforme :

```
/etc/httpd/conf.d/ssl.conf
```

Ce fichier contient les directives Apache qui définissent le mode de fonctionnement de votre serveur. Les directives apparaissent sur la gauche, suivies d'une valeur. Utilisez un éditeur de texte pour modifier ce fichier. Ceci nécessite des autorisations racine Linux.

Mettez à jour ou entrez les directives suivantes avec ces valeurs :

```
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
```

Enregistrez le fichier.

Red Hat 7

Localisez le fichier SSL pour cette plateforme :

```
/etc/httpd/conf.d/ssl.conf
```

Ce fichier contient les directives Apache qui définissent le mode de fonctionnement de votre serveur. Les directives apparaissent sur la gauche, suivies d'une valeur. Utilisez un éditeur de texte pour modifier ce fichier. Ceci nécessite des autorisations racine Linux.

Mettez à jour ou entrez les directives suivantes avec ces valeurs :

```
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
```

Enregistrez le fichier.

Red Hat 8

Localisez le fichier SSL pour cette plateforme :

```
/etc/httpd/conf.d/ssl.conf
```

Ce fichier contient les directives Apache qui définissent le mode de fonctionnement de votre serveur. Les directives apparaissent sur la gauche, suivies d'une valeur. Utilisez un éditeur de texte pour modifier ce fichier. Ceci nécessite des autorisations racine Linux.

Mettez à jour ou entrez les directives suivantes avec ces valeurs :

```
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
```

Enregistrez le fichier.

Ubuntu 16.04 LTS

Localisez le fichier SSL pour cette plateforme :

```
/etc/apache2/mods-available/ssl.conf
```

Ce fichier contient les directives Apache qui définissent le mode de fonctionnement de votre serveur. Les directives apparaissent sur la gauche, suivies d'une valeur. Utilisez un éditeur de texte pour modifier ce fichier. Ceci nécessite des autorisations racine Linux.

Mettez à jour ou entrez les directives suivantes avec ces valeurs :

```
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
```

Enregistrez le fichier.

Activez le module SSL et la configuration du site SSL par défaut :

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 18.04 LTS

Localisez le fichier SSL pour cette plateforme :

```
/etc/apache2/mods-available/ssl.conf
```

Ce fichier contient les directives Apache qui définissent le mode de fonctionnement de votre serveur. Les directives apparaissent sur la gauche, suivies d'une valeur. Utilisez un éditeur de texte pour modifier ce fichier. Ceci nécessite des autorisations racine Linux.

Mettez à jour ou entrez les directives suivantes avec ces valeurs :

```
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
```

Enregistrez le fichier.

Activez le module SSL et la configuration du site SSL par défaut :

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 20.04 LTS

Localisez le fichier SSL pour cette plateforme :

```
/etc/apache2/mods-available/ssl.conf
```

Ce fichier contient les directives Apache qui définissent le mode de fonctionnement de votre serveur. Les directives apparaissent sur la gauche, suivies d'une valeur. Utilisez un éditeur de texte pour modifier ce fichier. Ceci nécessite des autorisations racine Linux.

Mettez à jour ou entrez les directives suivantes avec ces valeurs :

```
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
```

Enregistrez le fichier.

Activez le module SSL et la configuration du site SSL par défaut :

```
$ sudo a2enmod ssl
$ sudo a2ensite default-ssl
```

Ubuntu 22.04 LTS

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

7. Configurez un fichier de valeurs d'environnement pour votre plateforme.

Amazon Linux

Aucune action requise. Les valeurs environnementales vont dans `/etc/sysconfig/httpd`

Amazon Linux 2

Ouvrez le fichier de service httpd :

```
/lib/systemd/system/httpd.service
```

Ajoutez le code suivant dans la section `[Service]` :

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 7

Ouvrez le fichier de service httpd :

```
/lib/systemd/system/httpd.service
```

Ajoutez le code suivant dans la section `[Service]` :

```
EnvironmentFile=/etc/sysconfig/httpd
```

CentOS 8

Ouvrez le fichier de service httpd :

```
/lib/systemd/system/httpd.service
```

Ajoutez le code suivant dans la section `[Service]` :

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 7

Ouvrez le fichier de service httpd :

```
/lib/systemd/system/httpd.service
```

Ajoutez le code suivant dans la section `[Service]` :

```
EnvironmentFile=/etc/sysconfig/httpd
```

Red Hat 8

Ouvrez le fichier de service httpd :

```
/lib/systemd/system/httpd.service
```

Ajoutez le code suivant dans la section `[Service]` :

```
EnvironmentFile=/etc/sysconfig/httpd
```

Ubuntu 16.04 LTS

Aucune action requise. Les valeurs environnementales vont dans `/etc/sysconfig/httpd`

Ubuntu 18.04 LTS

Aucune action requise. Les valeurs environnementales vont dans `/etc/sysconfig/httpd`

Ubuntu 20.04 LTS

Aucune action requise. Les valeurs environnementales vont dans `/etc/sysconfig/httpd`

Ubuntu 22.04 LTS

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

8. Dans le fichier qui stocke les variables d'environnement de votre plateforme, définissez une variable d'environnement contenant les informations d'identification de l'utilisateur de chiffrement (CU) :

   Amazon Linux

   Utilisez un éditeur de texte pour modifier le `/etc/sysconfig/httpd`.

   - Si vous utilisez le SDK client 3

     ```
     n3fips_password=<CU user name>:<password>
     ```

   - Si vous utilisez le SDK client 5

     ```
     CLOUDHSM_PIN=<CU user name>:<password>
     ```

   Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

   Amazon Linux 2

   Utilisez un éditeur de texte pour modifier le `/etc/sysconfig/httpd`.

   - Si vous utilisez le SDK client 3

```
n3fips_password=<CU user name>:<password>
```

- Si vous utilisez le SDK client 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

CentOS 7

Utilisez un éditeur de texte pour modifier le `/etc/sysconfig/httpd`.

- Si vous utilisez le SDK client 3

```
n3fips_password=<CU user name>:<password>
```

- Si vous utilisez le SDK client 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

CentOS 8

Utilisez un éditeur de texte pour modifier le `/etc/sysconfig/httpd`.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Red Hat 7

Utilisez un éditeur de texte pour modifier le `/etc/sysconfig/httpd`.

- Si vous utilisez le SDK client 3

```
n3fips_password=<CU user name>:<password>
```

- Si vous utilisez le SDK client 5

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Red Hat 8

Utilisez un éditeur de texte pour modifier le `/etc/sysconfig/httpd`.

```
CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

> **Note**
>
> Le SDK client 5 introduit la variable d'environnement `CLOUDHSM_PIN` permettant de stocker les informations d'identification du CU.

Ubuntu 16.04 LTS

Utilisez un éditeur de texte pour modifier le `/etc/apache2/envvars`.

```
export n3fips_password=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

Ubuntu 18.04 LTS

Utilisez un éditeur de texte pour modifier le `/etc/apache2/envvars`.

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

> **Note**
>
> Le SDK client 5 introduit la variable d'environnement `CLOUDHSM_PIN` permettant de stocker les informations d'identification du CU. Dans le SDK client 3, vous avez stocké les informations d'identification CU dans la variable d'environnement `n3fips_password`. Le SDK client 5 prend en charge les deux variables d'environnement, mais nous vous recommandons d'utiliser `CLOUDHSM_PIN`.

Ubuntu 20.04 LTS

Utilisez un éditeur de texte pour modifier le `/etc/apache2/envvars`.

```
export CLOUDHSM_PIN=<CU user name>:<password>
```

Remplacez *<CU user name>* et *<password>* par les informations d'identification de l'utilisateur de chiffrement.

> **Note**
>
> Le SDK client 5 introduit la variable d'environnement `CLOUDHSM_PIN` permettant de stocker les informations d'identification du CU. Dans le SDK client 3, vous avez stocké les informations d'identification CU dans la variable d'environnement `n3fips_password`. Le SDK client 5 prend en charge les deux variables d'environnement, mais nous vous recommandons d'utiliser `CLOUDHSM_PIN`.

Ubuntu 22.04 LTS

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

9. Démarrez le serveur web Apache.

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

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

10. (Facultatif) Configurez votre plateforme pour démarrer Apache au démarrage.

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

La prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible.

Après que vous avez mis à jour votre configuration de serveur web, accédez à Étape 4 : Activer le trafic HTTPS et vérifier le certificat.

Étape 4 : Activer le trafic HTTPS et vérifier le certificat

Après avoir configuré votre serveur Web pour le déchargement SSL/TLS avec AWS CloudHSM, ajoutez votre instance de serveur Web à un groupe de sécurité qui autorise le trafic HTTPS entrant. Cela permet aux clients, tels que les navigateurs Web, d'établir une connexion HTTPS avec votre serveur Web. Établissez ensuite une connexion HTTPS avec votre serveur Web et vérifiez qu'il utilise le certificat que vous avez configuré pour le déchargement SSL/TLS. AWS CloudHSM

Rubriques

- Activation des connexions HTTPS entrantes
- Vérification que le protocole HTTPS utilise le certificat que vous avez configuré

Activation des connexions HTTPS entrantes

Pour vous connecter à votre serveur web à partir d'un client (par exemple, un navigateur web), créez un groupe de sécurité qui autorise les connexions HTTPS entrantes. En particulier, il doit autoriser les connexions TCP entrantes sur le port 443. Affectez ce groupe de sécurité à votre serveur web.

Pour créer un groupe de sécurité pour le protocole HTTPS et l'affecter à votre serveur web

1. Ouvrez la console Amazon EC2 à l'adresse https://console.aws.amazon.com/ec2/.
2. Choisissez Groupes de sécurité dans le panneau de navigation.

3. Sélectionnez Create security group (Créer un groupe de sécurité).
4. Pour Créer un groupe de sécurité, procédez comme suit :

    a. Pour Nom du groupe de sécurité, tapez un nom pour le groupe de sécurité que vous créez.
    b. (Facultatif) Tapez une description du groupe de sécurité que vous créez.
    c. Pour VPC, choisissez le VPC qui contient votre instance de serveur web Amazon EC2.
    d. Sélectionnez Ajouter une règle.
    e. Pour Type, sélectionnez HTTPS dans la fenêtre déroulante.
    f. Pour Source, entrez l'emplacement de la source.
    g. Sélectionnez Create security group (Créer un groupe de sécurité).
5. Dans le panneau de navigation, sélectionnez Instances.
6. Cochez la case située en regard de votre instance de serveur web.
7. Choisissez le menu déroulant Actions en haut de la page. Sélectionnez Sécurité, puis Modifier les groupes de sécurité.
8. Pour Groupes de sécurité associés, veuillez consulter la zone de recherche, puis choisissez le groupe de sécurité que vous avez créé pour HTTPS. Ensuite, choisissez Ajouter des groupes de sécurité.
9. Sélectionnez Save.

Vérification que le protocole HTTPS utilise le certificat que vous avez configuré

Après avoir ajouté le serveur web à un groupe de sécurité, vous pouvez vérifier que le déchargement SSL/TLS fonctionne avec votre certificat signé par vous-même. Vous pouvez faire cela à l'aide d'un navigateur web ou avec un outil tel qu'OpenSSL s_client.

Pour vérifier le déchargement SSL/TLS à l'aide d'un navigateur web

1. Utilisez un navigateur web pour vous connecter à votre serveur web à l'aide du nom DNS public ou de l'adresse IP du serveur. Assurez-vous que l'URL dans la barre d'adresse commence par https://. Par exemple, **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

    > **Tip**
    >
    > Vous pouvez utiliser un service DNS tel que Amazon Route 53 pour router le nom de domaine de votre site web (par exemple, https://www.exemple.com/) vers votre serveur

web. Pour plus d'informations, consultez Routage du trafic vers une instance Amazon EC2 dans le Guide du Développeur Amazon Route 53 ou dans la documentation de votre service DNS.

2. Utilisez votre navigateur web pour afficher le certificat de serveur web. Pour plus d'informations, consultez les ressources suivantes :

   - Pour Mozilla Firefox, consultez View a Certificate sur le site web de support Mozilla.
   - Pour Google Chrome, consultez Understand Security Issues sur les Outils Google pour site web des développeurs Google.

   D'autres navigateurs web peuvent avoir des fonctions similaires que vous pouvez utiliser pour afficher le certificat de serveur web.

3. Assurez-vous que le certificat SSL/TLS est celui que vous avez configuré votre serveur web à utiliser.

Pour vérifier le déchargement SSL/TLS avec OpenSSL s_client

1. Exécutez la commande OpenSSL suivante pour vous connecter à votre serveur web en utilisant le protocole HTTPS. Remplacez *<server name>* par le nom DNS public ou l'adresse IP de votre serveur web.

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> Vous pouvez utiliser un service DNS tel que Amazon Route 53 pour router le nom de domaine de votre site web (par exemple, https://www.exemple.com/) vers votre serveur web. Pour plus d'informations, consultez Routage du trafic vers une instance Amazon EC2 dans le Guide du Développeur Amazon Route 53 ou dans la documentation de votre service DNS.

2. Assurez-vous que le certificat SSL/TLS est celui que vous avez configuré votre serveur web à utiliser.

Vous disposez maintenant d'un site web sécurisé avec HTTPS. La clé privée du serveur Web est stockée dans un HSM de votre AWS CloudHSM cluster.

Pour ajouter un équilibreur de charge, veuillez consulter Ajouter un équilibreur de charge avec Elastic Load Balancing (facultatif).

Utilisation de Tomcat avec JSSE pour le déchargement SSL/TLS sous Linux

Cette rubrique fournit des step-by-step instructions pour configurer le déchargement SSL/TLS à l'aide de Java Secure Socket Extension (JSSE) avec le SDK JCE. AWS CloudHSM

Rubriques

Présentation

Dans AWS CloudHSM, les serveurs Web Tomcat fonctionnent sous Linux pour prendre en charge le protocole HTTPS. Le SDK AWS CloudHSM JCE fournit une interface qui peut être utilisée avec JSSE (Java Secure Socket Extension) pour permettre l'utilisation de HSM pour de tels serveurs Web. AWS CloudHSM JCE est le pont qui connecte JSSE à votre cluster AWS CloudHSM. JSSE est une API Java pour les protocoles SSL (Secure Sockets Layer) et TLS (Transport Layer Security).

Étape 1 : Configurer les prérequis

Suivez ces prérequis pour utiliser un serveur Web Tomcat AWS CloudHSM pour le déchargement SSL/TLS sous Linux. Ces prérequis doivent être remplis pour configurer le déchargement SSL/TLS du serveur Web avec le SDK client 5 et un serveur Web Tomcat.

> **Note**
>
> Les différentes plateformes ont des prérequis différents. Suivez toujours les étapes d'installation adaptées à votre plateforme.

Prérequis

- Une instance Amazon EC2 exécutant un système d'exploitation Linux avec un serveur Web Tomcat installé.
- Un utilisateur de chiffrement (CU) propriétaire et gestionnaire de la clé privée du serveur web sur le HSM.
- AWS CloudHSM Cluster actif avec au moins deux modules de sécurité matériels (HSM) sur lesquels JCE for Client SDK 5 est installé et configuré.

> **Note**
>
> Vous pouvez utiliser un seul cluster HSM, mais vous devez d'abord désactiver la durabilité des clés client. Pour plus d'informations, voir Gérer les paramètres de durabilité des clés client et Outil de configuration du SDK client 5.

Comment répondre aux prérequis

1. Installez et configurez le JCE AWS CloudHSM sur un AWS CloudHSM cluster actif comportant au moins deux modules de sécurité matériels (HSM). Pour plus d'informations sur l'installation, consultez JCE for Client SDK 5.
2. Sur une instance Linux EC2 ayant accès à votre AWS CloudHSM cluster, suivez les instructions d'Apache Tomcat pour télécharger et installer le serveur Web Tomcat.
3. Utilisez la CLI CloudHSM pour créer un utilisateur de chiffrement (CU). Pour plus d'informations sur la gestion des utilisateurs HSM, consultez la section Gestion des utilisateurs HSM avec la CLI CloudHSM.

   > **Tip**
   >
   > Conservez le nom d'utilisateur et le mot de passe du CU. Vous en aurez besoin plus tard pour générer ou importer la clé privée HTTPS et le certificat de votre serveur web.

4. Pour configurer JCE avec Java Keytool, suivez les instructions de Utilisation du SDK client 5 pour intégrer Java Keytool et Jarsigner.

Une fois que vous avez terminé ces étapes, consultez Étape 2 : importer ou générer une clé privée et un certificat SSL/TLS.

Remarques

- Pour utiliser Security-Enhanced Linux (SELinux) et les serveurs Web, vous devez autoriser les connexions TCP sortantes sur le port 2223, qui est le port utilisé par le SDK client 5 pour communiquer avec le HSM.
- Pour créer et activer un cluster et donner à une instance EC2 l'accès au cluster, suivez les étapes décrites dans Démarrer avec AWS CloudHSM. Cette section fournit des step-by-step instructions pour créer un cluster actif avec un HSM et une instance client Amazon EC2. Vous pouvez utiliser cette instance client comme serveur web.
- Pour éviter de désactiver la durabilité des clés client, ajoutez plusieurs HSM à votre cluster. Pour plus d'informations, consultez Ajout d'un HSM.
- Vous pouvez utiliser un SSH ou PuTTY pour vous connecter à votre instance client. Pour plus d'informations, consultez Connexion à votre instance Linux à l'aide de SSH ou Connexion à votre instance Linux à partir de Windows à l'aide de PuTTY dans la documentation Amazon EC2.

Étape 2 : importer ou générer une clé privée et un certificat SSL/TLS

Pour activer le protocole HTTPS, votre application de serveur web Tomcat nécessite une clé privée et un certificat SSL/TLS correspondant. Pour utiliser le déchargement SSL/TLS du serveur Web avec AWS CloudHSM, vous devez stocker la clé privée dans un HSM de votre cluster. AWS CloudHSM

> **Note**
>
> Si vous ne disposez pas encore d'une clé privée et d'un certificat correspondant, générez une clé privée dans un HSM. Vous utilisez la clé privée pour créer une demande de signature de certificat (CSR), que vous utilisez pour créer le certificat SSL/TLS.

Vous créez un AWS CloudHSM KeyStore fichier local qui contient une référence à votre clé privée sur le HSM et le certificat associé. Votre serveur Web utilise le AWS CloudHSM KeyStore fichier pour identifier la clé privée sur le HSM lors du déchargement SSL/TLS.

Rubriques

- Générer une clé privée
- Générer un certificat signé automatiquement.

Générer une clé privée

Cette section explique comment générer une paire de clés à l'aide du JDK à KeyTool partir du JDK. Une fois qu'une paire de clés est générée dans le HSM, vous pouvez l'exporter sous forme de KeyStore fichier et générer le certificat correspondant.

En fonction de votre cas d'utilisation, vous pouvez générer une paire de clés RSA ou EC. Les étapes suivantes montrent comment générer une paire de clés RSA.

Utilisez la **`genkeypair`** commande in KeyTool pour générer une paire de clés RSA

1. Après avoir remplacé les *<VARIABLES>* ci-dessous par vos données spécifiques, utilisez la commande suivante pour générer un fichier keystore nommé `jsse_keystore.keystore`, qui contiendra une référence à votre clé privée sur le HSM.

```
$ 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>*** : chemin vers lequel vous souhaitez générer votre fichier keystore.
   - ***<UNIQUE ALIAS FOR KEYS>*** : Ceci est utilisé pour identifier de manière unique votre clé sur le HSM. Cet alias sera défini comme attribut LABEL pour la clé.
   - ***<KEY PASSWORD>*** : Nous stockons la référence à votre clé dans le fichier keystore local, et ce mot de passe protège cette référence locale.
   - ***<KEYSTORE PASSWORD>*** : Il s'agit du mot de passe de votre fichier keystore local.
   - ***<JSSE KEYSTORE NAME>*** : nom du fichier Keystore.
   - ***<CERT DOMAIN NAME>*** : Nom distinctif X.500.
   - ***<KEY ALGORITHM>*** : algorithme clé pour générer une paire de clés (par exemple, RSA et EC).
   - ***<KEY SIZE>*** : taille de clé pour générer une paire de clés (par exemple, 2048, 3072 et 4096).
   - ***<SIGN ALGORITHM>*** : taille de clé pour générer une paire de clés (par exemple, SHA1WithRSA, SHA224withRSA, SHA256withRSA, SHA384withRSA et SHA512withRSA).

2. Pour confirmer le succès de la commande, entrez la commande suivante et vérifiez que vous avez correctement généré une paire de clés RSA.

```
$ ls <PATH>/<JSSE KEYSTORE NAME>.keystore
```

Générer un certificat signé automatiquement.

Une fois que vous avez généré une clé privée avec le fichier keystore, vous pouvez utiliser ce fichier pour générer une demande de signature de certificat (CSR) et un certificat.

Dans un environnement de production, vous utilisez généralement une autorité de certification (CA) pour créer un certificat émis par une demande de signature de certificat (CSR). L'autorité de certification n'est pas nécessaire pour un environnement de test. Si vous utilisez une autorité de certification, envoyez-lui le fichier CSR et utilisez le certificat SSL/TLS signé qu'elle vous fournit sur votre serveur Web pour HTTPS.

Au lieu d'utiliser une autorité de certification, vous pouvez utiliser le KeyTool pour créer un certificat auto-signé. Les certificats auto-signés ne sont pas approuvées par les navigateurs et ne doivent pas être utilisés dans les environnements de production. Ils peuvent cependant être utilisés dans les environnements de test.

> **Warning**
>
> Les certificats auto-signés doivent uniquement être utilisés dans un environnement de test. Pour un environnement de production, utilisez une méthode plus sécurisée telle qu'une autorité de certification pour créer un certificat.

Générez un certificat

1. Obtenez une copie de votre fichier keystore généré lors d'une étape précédente.
2. Exécutez la commande suivante pour KeyTool créer une demande de signature de certificat (CSR).

```
$ keytool -certreq -keyalg RSA -alias unique_alias_for_key -file certreq.csr \
        -keystore <JSSE KEYSTORE NAME>.keystore -storetype CLOUDHSM \
        -J-classpath '-J$JAVA_LIB/*:/opt/cloudhsm/java/*:./*' \
        -keypass <KEY PASSWORD> -storepass <KEYSTORE PASSWORD>
```

> **Note**
>
> Le fichier de sortie de la demande de signature de certificat est `certreq.csr`.

Signer un certificat

- Après avoir remplacé les *<VARIABLES>* ci-dessous par vos données spécifiques, exécutez la commande suivante pour signer votre CSR avec votre clé privée sur votre HSM. Cette opération crée un certificat auto-signé.

```
$ 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` est le certificat signé qui utilise la clé privée de l'alias.

Importer un certificat dans Keystore

- Après avoir remplacé les *<VARIABLES>* ci-dessous par vos données spécifiques, exécutez la commande suivante pour importer un certificat signé en tant que certificat fiable. Cette étape stockera le certificat dans l'entrée du keystore identifiée par un alias.

```
$ 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>
```

Convertir un certificat en PEM

- Exécutez la commande suivante pour convertir le fichier de certificat signé (.crt) en PEM. Le fichier PEM sera utilisé pour envoyer la demande depuis le client HTTP.

```
$ openssl x509 -inform der -in certificate.crt -out certificate.pem
```

Une fois ces étapes terminées, passez à l'étape 3 : Configuration du serveur Web.

Étape 3 : Configurer le serveur web Tomcat

Mettez à jour votre configuration de logiciel serveur web pour utiliser le certificat HTTPS et le fichier PEM correspondant que vous avez créé à l'étape précédente. N'oubliez pas de sauvegarder vos certificats et clés existants avant de commencer. Cela permettra de terminer la configuration de votre logiciel serveur web Linux pour le déchargement SSL/TLS avec AWS CloudHSM. Pour plus d'informations, consultez Référence de configuration Apache Tomcat 9.

Arrêtez le server.

- Après avoir remplacé les *<VARIABLES>* ci-dessous par vos données spécifiques, exécutez la commande suivante pour arrêter le serveur Tomcat avant de mettre à jour la configuration

  ```
  $ /<TOMCAT DIRECTORY>/bin/shutdown.sh
  ```

  - ***<TOMCAT DIRECTORY>*** : votre répertoire d'installation de Tomcat.

Mettre à jour le chemin de classe de Tomcat

1. Connectez-vous à votre instance client .
2. Localisez le dossier d'installation de Tomcat.
3. Après avoir remplacé les *<VARIABLES>* ci-dessous par vos données spécifiques, utilisez la commande suivante pour ajouter la bibliothèque Java et le chemin Java Cloudhsm dans le chemin de classe Tomcat, situé dans le fichier Tomcat/bin/catalina.sh.

   ```
   $ 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>*** : emplacement de la bibliothèque Java JRE.
   - ***<TOMCAT PATH>*** : dossier d'installation de Tomcat.

Ajoutez un connecteur HTTPS dans la configuration du serveur.

1. Accédez au dossier d'installation de Tomcat.
2. Après avoir remplacé les *<VARIABLES>* ci-dessous par vos données spécifiques, utilisez la commande suivante pour ajouter un connecteur HTTPS afin d'utiliser les certificats générés dans les prérequis :

```
$ 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>*** : répertoire dans lequel se trouve le fichier keystore.
   - ***<JSSE KEYSTORE NAME>*** : nom du fichier Keystore.
   - ***<KEYSTORE PASSWORD>*** : Il s'agit du mot de passe de votre fichier keystore local.
   - ***<KEY PASSWORD>*** : Nous stockons la référence à votre clé dans le fichier keystore local, et ce mot de passe protège cette référence locale.
   - ***<UNIQUE ALIAS FOR KEYS>*** : Ceci est utilisé pour identifier de manière unique votre clé sur le HSM. Cet alias sera défini comme attribut LABEL pour la clé.
   - ***<TOMCAT PATH>*** : chemin d'accès à votre dossier Tomcat.

Démarrez le serveur

- Après avoir remplacé les *<VARIABLES>* ci-dessous par vos données spécifiques, utilisez la commande suivante pour démarrer le serveur Tomcat :

```
$ /<TOMCAT DIRECTORY>/bin/startup.sh
```

> **Note**
>
> ***<TOMCAT DIRECTORY>*** est le nom de votre répertoire d'installation Tomcat.

Après que vous avez mis à jour votre configuration de serveur web, accédez à Étape 4 : Activer le trafic HTTPS et vérifier le certificat.

Étape 4 : Activer le trafic HTTPS et vérifier le certificat

Après avoir configuré votre serveur Web pour le déchargement SSL/TLS avec AWS CloudHSM, ajoutez votre instance de serveur Web à un groupe de sécurité qui autorise le trafic HTTPS entrant. Cela permet aux clients, tels que les navigateurs Web, d'établir une connexion HTTPS avec votre serveur Web. Établissez ensuite une connexion HTTPS avec votre serveur Web et vérifiez qu'il utilise le certificat que vous avez configuré pour le déchargement SSL/TLS. AWS CloudHSM

Rubriques

- Activation des connexions HTTPS entrantes
- Vérification que le protocole HTTPS utilise le certificat que vous avez configuré

Activation des connexions HTTPS entrantes

Pour vous connecter à votre serveur web à partir d'un client (par exemple, un navigateur web), créez un groupe de sécurité qui autorise les connexions HTTPS entrantes. En particulier, il doit autoriser les connexions TCP entrantes sur le port 443. Affectez ce groupe de sécurité à votre serveur web.

Pour créer un groupe de sécurité pour le protocole HTTPS et l'affecter à votre serveur web

1. Ouvrez la console Amazon EC2 à l'adresse https://console.aws.amazon.com/ec2/.
2. Choisissez Groupes de sécurité dans le panneau de navigation.
3. Sélectionnez Create security group (Créer un groupe de sécurité).
4. Pour Créer un groupe de sécurité, procédez comme suit :
    a. Pour Nom du groupe de sécurité, tapez un nom pour le groupe de sécurité que vous créez.
    b. (Facultatif) Tapez une description du groupe de sécurité que vous créez.
    c. Pour VPC, choisissez le VPC qui contient votre instance de serveur web Amazon EC2.
    d. Sélectionnez Ajouter une règle.
    e. Pour Type, sélectionnez HTTPS dans la fenêtre déroulante.
    f. Pour Source, entrez l'emplacement de la source.
    g. Sélectionnez Create security group (Créer un groupe de sécurité).
5. Dans le panneau de navigation, sélectionnez Instances.

6. Cochez la case située en regard de votre instance de serveur web.
7. Choisissez le menu déroulant Actions en haut de la page. Sélectionnez Sécurité, puis Modifier les groupes de sécurité.
8. Pour Groupes de sécurité associés, veuillez consulter la zone de recherche, puis choisissez le groupe de sécurité que vous avez créé pour HTTPS. Ensuite, choisissez Ajouter des groupes de sécurité.
9. Sélectionnez Save.

Vérification que le protocole HTTPS utilise le certificat que vous avez configuré

Après avoir ajouté le serveur web à un groupe de sécurité, vous pouvez vérifier que le déchargement SSL/TLS fonctionne avec votre certificat signé par vous-même. Vous pouvez faire cela à l'aide d'un navigateur web ou avec un outil tel qu'OpenSSL s_client.

Pour vérifier le déchargement SSL/TLS à l'aide d'un navigateur web

1. Utilisez un navigateur web pour vous connecter à votre serveur web à l'aide du nom DNS public ou de l'adresse IP du serveur. Assurez-vous que l'URL dans la barre d'adresse commence par https://. Par exemple, **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

   > **Tip**
   >
   > Vous pouvez utiliser un service DNS tel que Amazon Route 53 pour router le nom de domaine de votre site web (par exemple, https://www.exemple.com/) vers votre serveur web. Pour plus d'informations, consultez Routage du trafic vers une instance Amazon EC2 dans le Guide du Développeur Amazon Route 53 ou dans la documentation de votre service DNS.

2. Utilisez votre navigateur web pour afficher le certificat de serveur web. Pour plus d'informations, consultez les ressources suivantes :

   - Pour Mozilla Firefox, consultez View a Certificate sur le site web de support Mozilla.
   - Pour Google Chrome, consultez Understand Security Issues sur les Outils Google pour site web des développeurs Google.

D'autres navigateurs web peuvent avoir des fonctions similaires que vous pouvez utiliser pour afficher le certificat de serveur web.

3. Assurez-vous que le certificat SSL/TLS est celui que vous avez configuré votre serveur web à utiliser.

Pour vérifier le déchargement SSL/TLS avec OpenSSL s_client

1. Exécutez la commande OpenSSL suivante pour vous connecter à votre serveur web en utilisant le protocole HTTPS. Remplacez *<server name>* par le nom DNS public ou l'adresse IP de votre serveur web.

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> Vous pouvez utiliser un service DNS tel que Amazon Route 53 pour router le nom de domaine de votre site web (par exemple, https://www.exemple.com/) vers votre serveur web. Pour plus d'informations, consultez Routage du trafic vers une instance Amazon EC2 dans le Guide du Développeur Amazon Route 53 ou dans la documentation de votre service DNS.

2. Assurez-vous que le certificat SSL/TLS est celui que vous avez configuré votre serveur web à utiliser.

Vous disposez maintenant d'un site web sécurisé avec HTTPS. La clé privée du serveur Web est stockée dans un HSM de votre AWS CloudHSM cluster.

Pour ajouter un équilibreur de charge, veuillez consulter Ajouter un équilibreur de charge avec Elastic Load Balancing (facultatif).

Utilisation d'IIS avec CNG pour le déchargement SSL/TLS sous Windows

Ce didacticiel fournit des step-by-step instructions pour configurer le déchargement SSL/TLS AWS CloudHSM sur un serveur Web Windows.

Rubriques

Présentation

Sous Windows, l'application serveur web Internet Information Services (IIS) pour Windows Server prend en charge HTTPS en mode natif. Le fournisseur de stockage de clés (KSP, Key Storage Provider)AWS CloudHSM pour l'API Cryptography: Next Generation (CNG) de Microsoft fournit l'interface qui autorise IIS à utiliser les HSM de votre cluster pour le déchargement cryptographique et le stockage de clés. Le AWS CloudHSM KSP est le pont qui connecte IIS à votre AWS CloudHSM cluster.

Ce didacticiel vous montre comment effectuer les opérations suivantes :

- Installez le logiciel du serveur web sur une instance Amazon EC2.
- Configurez le logiciel du serveur Web pour qu'il prenne en charge le protocole HTTPS avec une clé privée stockée dans votre cluster AWS CloudHSM .
- (Facultatif) Utilisez Amazon EC2 pour créer une deuxième instance de serveur Web et Elastic Load Balancing pour créer un équilibreur de charge. L'utilisation d'un équilibreur de charge peut accroître les performances en répartissant la charge sur plusieurs serveurs. Elle peut également assurer la redondance et une meilleure disponibilité en cas de défaillance d'un ou plusieurs serveurs.

Lorsque vous êtes prêt à commencer, consultez Étape 1 : Configurer les prérequis.

Étape 1 : Configurer les prérequis

Pour configurer le déchargement SSL/TLS du serveur Web avec AWS CloudHSM, vous avez besoin des éléments suivants :

- Un AWS CloudHSM cluster actif avec au moins un HSM.
- Une instance Amazon EC2 exécutant un système d'exploitation Windows avec les logiciels suivants installés :
  - Le logiciel AWS CloudHSM client pour Windows.

  - Internet Information Services (IIS) pour Windows Server.
- Un utilisateur de chiffrement (CU) propriétaire et gestionnaire de la clé privée du serveur web sur le HSM.

> **Note**
>
> Ce didacticiel utilise Microsoft Windows Server 2016. Microsoft Windows Server 2012 est également pris en charge, mais Microsoft Windows Server 2012 R2 ne l'est pas.

Pour configurer une instance Windows Server et créer un utilisateur de chiffrement sur le HSM

1. Suivez les étapes de Premiers pas. Lorsque vous lancez le client Amazon EC2, choisissez une AMI Windows Server 2016 ou Windows Server 2012. Une fois que vous avez terminé ces étapes, vous disposez d'un cluster actif avec au moins un HSM. Vous disposez également d'une instance client Amazon EC2 exécutant Windows Server sur laquelle le logiciel AWS CloudHSM client pour Windows est installé.
2. (Facultatif) Ajoutez d'autres HSM à votre cluster. Pour plus d'informations, consultez Ajout d'un HSM.
3. Connectez-vous à votre serveur Windows. Pour plus d'informations, consultez Connexion à votre instance dans le Guide de l'utilisateur Amazon EC2 pour les instances Windows.
4. Utilisez la CLI CloudHSM pour créer un utilisateur de chiffrement (CU). Conservez le nom d'utilisateur et le mot de passe du CU. Vous en aurez besoin pour terminer l'étape suivante.

   > **Note**
   >
   > Pour plus d'informations sur la création d'un utilisateur, consultez Gestion des utilisateurs HSM avec la CLI CloudHSM.

5. Définissez les informations d'identification de connexion pour le HSM, à l'aide du nom d'utilisateur et du mot de passe CU que vous avez créés à l'étape précédente.
6. À l'étape 5, si vous avez utilisé le Gestionnaire d'informations d'identification Windows pour définir les informations d'identification HSM, téléchargez `psexec.exe`depuis SysInternals pour exécuter la commande suivante sous le nom NT Authority \ SYSTEM :

```
psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
 --username <USERNAME> --password <PASSWORD>
```

Remplacez *<USERNAME>* et *<PASSWORD>* par les informations d'identification HSM.

Pour installer IIS sur Windows Server

1. Si ce n'est pas déjà fait, connectez-vous à votre serveur Windows Server. Pour plus d'informations, consultez Connexion à votre instance dans le Guide de l'utilisateur Amazon EC2 pour les instances Windows.
2. Sur votre serveur Windows Server, démarrez le Server Manager (Gestionnaire de serveurs).
3. Dans le tableau de bord du Server Manager (Gestionnaire de serveurs), choisissez Add roles and features (Ajouter des rôles et des fonctions).
4. Prenez connaissances des informations contenues dans le fichier Before you begin (Avant de commencer), puis choisissez Next (Suivant).
5. Pour Installation Type (Type d'installation), choisissez Role-based or feature-based installation (Installation basée sur un rôle ou une fonction). Ensuite, sélectionnez Suivant.
6. Pour Server Selection (Sélection de serveur), choisissez Select a server from the server pool (Sélectionner un serveur du pool de serveurs). Ensuite, sélectionnez Suivant.
7. Pour Server Roles (Rôles de serveur), procédez comme suit :
   a. Sélectionnez Web Server (IIS).
   b. Pour Add features that are required for Web Server (IIS) (Ajouter des fonctions qui sont requises pour le serveur Web (IIS)), choisissez Add Features (Ajouter des fonctions).
   c. Choisissez Suivant pour finaliser la sélection de rôles de serveur.
8. Pour Features (Fonctions), acceptez les valeurs par défaut. Ensuite, sélectionnez Suivant.
9. Lisez les informations Web Server Role (IIS) (Rôle du serveur Web (IIS)). Ensuite, sélectionnez Suivant.
10. Pour Select role services (Sélectionner les services de rôle), acceptez les valeurs par défaut ou modifiez les paramètres comme souhaité. Ensuite, sélectionnez Suivant.
11. Pour Confirmation, lisez les informations de confirmation. Choisissez ensuite Install (Installer).
12. Une fois l'installation terminée, choisissez Close (Fermer).

Une fois que vous avez terminé ces étapes, consultez Étape 2 : Créer une demande de signature de certificat (CSR) et un certificat.

Étape 2 : Créer une demande de signature de certificat (CSR) et un certificat

Pour activer le protocole HTTPS, votre serveur nécessite un certificat SSL/TLS et une clé privée correspondante. Pour utiliser le déchargement SSL/TLS avec AWS CloudHSM, vous devez stocker la clé privée dans le HSM de votre cluster. AWS CloudHSM Pour ce faire, vous utilisez le fournisseur de stockage de clés (KSP)AWS CloudHSM pour l'API Cryptography : Next Generation (CNG) de Microsoft afin de créer une demande de signature de certificat (CSR). Ensuite, vous accordez la CSR à une autorité de certification (CA), qui signe la CSR pour produire un certificat.

Rubriques

- Création d'une CSR
- Obtention et importation d'un certificat signé

Création d'une CSR

Utilisez le AWS CloudHSM KSP sur votre serveur Windows pour créer un CSR.

Pour créer un CSR

1. Si ce n'est pas déjà fait, connectez-vous à votre serveur Windows Server. Pour plus d'informations, consultez Connexion à votre instance dans le Guide de l'utilisateur Amazon EC2 pour les instances Windows.
2. Utilisez la commande suivante pour démarrer le daemon AWS CloudHSM client.

   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

- Pour le Client Windows version 1.1.2 et ultérieure :

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Pour les clients Windows version 1.1.1 et antérieure :

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

3. Sur votre Windows Server, utilisez un éditeur de texte pour créer un fichier de demande de certificat nommé `IISCertRequest.inf`. L'exemple suivant montre le contenu d'un exemple de fichier `IISCertRequest.inf`. Pour plus d'informations sur les sections, les clés et les valeurs que vous pouvez spécifier dans le fichier, consultez la documentation Microsoft. Ne modifiez pas la valeur `ProviderName`.

```
[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. Utilisez la commande Windows certreq pour créer une CSR à partir du fichier `IISCertRequest.inf` que vous avez créé à l'étape précédente. L'exemple suivant enregistre la CSR dans un fichier nommé `IISCertRequest.csr`. Si vous avez utilisé un nom de fichier différent pour votre fichier de demande de certificat, remplacez *`IIS CertRequest .inf`* par le nom de fichier approprié. Vous pouvez éventuellement remplacer le fichier *`IIS CertRequest .csr`* par un nom de fichier différent pour votre fichier CSR.

```
C:\>certreq -new IISCertRequest.inf IISCertRequest.csr
        SDK Version: 2.03

CertReq: Request Created
```

   Le fichier `IISCertRequest.csr` contient votre CSR. Vous avez besoin de cette CSR pour obtenir un certificat signé.

Obtention et importation d'un certificat signé

Dans un environnement de production, vous utilisez généralement une autorité de certification (CA) pour créer un certificat émis par une demande de signature de certificat (CSR). L'autorité de certification n'est pas nécessaire pour un environnement de test. Si vous utilisez une autorité de certification, envoyez-lui le fichier CSR (`IISCertRequest.csr`) et utilisez l'autorité de certification pour créer un certificat SSL/TLS signé.

Au lieu d'utiliser une autorité de certification, vous pouvez utiliser un outil comme OpenSSL pour créer un certificat auto-signé.

> **Warning**
>
> Les certificats auto-signés ne sont pas approuvées par les navigateurs et ne doivent pas être utilisés dans les environnements de production. Ils peuvent cependant être utilisés dans les environnements de test.

Les procédures suivantes montrent comment créer un certificat auto-signé et l'utiliser pour signer la CSR de votre serveur web.

Pour créer un certificat auto-signé

1. Utilisez la commande OpenSSL suivante pour créer une clé privée. Vous pouvez éventuellement remplacer *SelfSignedCA.key* par le nom du fichier contenant votre clé privée.

```
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. Utilisez la commande OpenSSL suivante pour créer un certificat d'émission auto-signé avec la clé privée que vous avez créée à l'étape précédente. Il s'agit d'une commande interactive. Lisez les instructions à l'écran et suivez les invites. Remplacez *SelfSignedCA.key* par le nom du fichier contenant votre clé privée (s'il est différent). Vous pouvez éventuellement remplacer *SelfSignedCA.crt* par le nom du fichier contenant votre certificat auto-signé.

```
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 []:
```

Pour utiliser votre certificat auto-signé pour signer la CSR du serveur web

- Utilisez la commande OpenSSL suivante pour utiliser votre clé privée et le certificat auto-signé pour signer la CSR. Remplacez les éléments suivants par les noms des fichiers qui contiennent les données correspondantes (en cas de différence).
  - *IIS CertRequest .csr* : nom du fichier contenant le CSR de votre serveur Web
  - *SelfSignedCA.crt* — Le nom du fichier contenant votre certificat auto-signé
  - *SelfSignedCA.key* — Le nom du fichier contenant votre clé privée
  - *IISCert.crt* – Nom du fichier qui contient le certificat auto-signé de votre serveur web

```
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:
```

Une fois que vous avez terminé l'étape précédente, vous avez un certificat signé pour votre serveur web (`IISCert.crt`) et un certificat auto-signé (`SelfSignedCA.crt`). Lorsque vous avez ces fichiers, accédez à Étape 3 : Configurer le serveur web.

Étape 3 : Configurer le serveur web

Mettez à jour la configuration de votre site web IIS pour utiliser le certificat HTTPS que vous avez créé à la fin de l'étape précédente. Cela permettra de terminer la configuration de votre logiciel serveur web Windows (IIS) pour le déchargement SSL/TLS avec AWS CloudHSM.

Si vous avez utilisé un certificat auto-signé pour signer votre CSR, vous devez d'abord importer le certificat auto-signé dans les autorités de certification racine approuvées Windows.

Pour importer votre certificat auto-signé dans les autorités de certification racine approuvées Windows

1. Si ce n'est pas déjà fait, connectez-vous à votre serveur Windows Server. Pour plus d'informations, consultez Connexion à votre instance dans le Guide de l'utilisateur Amazon EC2 pour les instances Windows.
2. Copiez votre certificat auto-signé sur votre serveur Windows.
3. Sur votre serveur Windows, ouvrez le Panneau de configuration.
4. Pour Search Control Panel (Recherche dans le panneau de configuration), tapez **`certificates`**. Ensuite, choisissez Manage computer certificates (Gérer les certificats de l'ordinateur).
5. Dans la fenêtre Certificates - Local Computer (Certificats - Ordinateur local), double-cliquez sur Trusted Root Certification Authorities (Autorités de certification racine approuvées).
6. Cliquez avec le bouton droit sur Certificates (Certificats), puis choisissez All Tasks (Toutes les tâches), Import (Importer).
7. Dans l'assistant Certificate Import (Importation de certificat), choisissez Next (Suivant).
8. Choisissez Browse (Parcourir), puis recherchez et sélectionnez votre certificat auto-signé. Si vous avez créé votre certificat auto-signé en suivant les instructions de l'étape précédente de ce didacticiel, votre certificat auto-signé se nomme `SelfSignedCA.crt`. Choisissez Ouvrir.
9. Choisissez Suivant.
10. Pour Certificate Store (Magasin de certificats), choisissez Place all certificates in the following store (Placer tous les certificats dans les éléments suivants). Ensuite, vérifiez que Trusted Root Certification Authorities (Autorités de certification racine approuvées) est sélectionné pour Certificate store (Magasin de certificats).
11. Cliquez sur Suivant, puis sur Terminer.

Pour mettre à jour la configuration du site web IIS

1. Si ce n'est pas déjà fait, connectez-vous à votre serveur Windows Server. Pour plus d'informations, consultez Connexion à votre instance dans le Guide de l'utilisateur Amazon EC2 pour les instances Windows.
2. Démarrez le daemon AWS CloudHSM client.
3. Copiez le certificat signé de votre serveur web, celui que vous avez créé à la fin de l'étape précédente de ce didacticiel, sur votre serveur Windows.

4. Sur votre serveur Windows, utilisez la commande certreq Windows pour accepter le certificat signé, comme dans l'exemple suivant. Remplacez *IISCert.crt* par le nom du fichier qui contient le certificat signé de votre serveur web.

```
C:\>certreq -accept IISCert.crt
        SDK Version: 2.03
```

5. Sur votre serveur Windows Server, démarrez le Server Manager (Gestionnaire de serveurs).
6. Dans le tableau de bord Server Manager (Gestionnaire de serveurs), dans le coin supérieur droit, choisissez Tools (Outils), Internet Information Services (IIS) Manager.
7. Dans la fenêtre Internet Information Services (IIS) Manager, double-cliquez sur le nom de votre serveur. Ensuite, double-cliquez sur Sites. Sélectionnez votre site web.
8. Sélectionnez SSL Settings (Paramètres SSL). Puis, sur la droite de la fenêtre, choisissez Bindings (Liaisons).
9. Dans la fenêtre Site Bindings (Liaisons de site), choisissez Add (Ajouter).
10. Pour Type, choisissez https. Pour SSL certificate (Certificat SSL), choisissez le certificat HTTPS que vous avez créé à la fin de l'étape précédente du didacticiel.

> **Note**
>
> Si vous rencontrez une erreur au cours de cette liaison de certificat, redémarrez votre serveur et réessayez cette étape.

11. Choisissez OK.

Après que vous avez mis à jour la configuration de votre site web, accédez à Étape 4 : Activer le trafic HTTPS et vérifier le certificat.

Étape 4 : Activer le trafic HTTPS et vérifier le certificat

Après avoir configuré votre serveur Web pour le déchargement SSL/TLS avec AWS CloudHSM, ajoutez votre instance de serveur Web à un groupe de sécurité qui autorise le trafic HTTPS entrant. Cela permet aux clients, tels que les navigateurs Web, d'établir une connexion HTTPS avec votre serveur Web. Établissez ensuite une connexion HTTPS avec votre serveur Web et vérifiez qu'il utilise le certificat que vous avez configuré pour le déchargement SSL/TLS. AWS CloudHSM

Rubriques

Activation des connexions HTTPS entrantes

Pour vous connecter à votre serveur web à partir d'un client (par exemple, un navigateur web), créez un groupe de sécurité qui autorise les connexions HTTPS entrantes. En particulier, il doit autoriser les connexions TCP entrantes sur le port 443. Affectez ce groupe de sécurité à votre serveur web.

Pour créer un groupe de sécurité pour le protocole HTTPS et l'affecter à votre serveur web

1. Ouvrez la console Amazon EC2 à l'adresse https://console.aws.amazon.com/ec2/.
2. Choisissez Groupes de sécurité dans le panneau de navigation.
3. Sélectionnez Create security group (Créer un groupe de sécurité).
4. Pour Créer un groupe de sécurité, procédez comme suit :
    a. Pour Nom du groupe de sécurité, tapez un nom pour le groupe de sécurité que vous créez.
    b. (Facultatif) Tapez une description du groupe de sécurité que vous créez.
    c. Pour VPC, choisissez le VPC qui contient votre instance de serveur web Amazon EC2.
    d. Sélectionnez Ajouter une règle.
    e. Pour Type, sélectionnez HTTPS dans la fenêtre déroulante.
    f. Pour Source, entrez l'emplacement de la source.
    g. Sélectionnez Create security group (Créer un groupe de sécurité).
5. Dans le panneau de navigation, sélectionnez Instances.
6. Cochez la case située en regard de votre instance de serveur web.
7. Choisissez le menu déroulant Actions en haut de la page. Sélectionnez Sécurité, puis Modifier les groupes de sécurité.
8. Pour Groupes de sécurité associés, veuillez consulter la zone de recherche, puis choisissez le groupe de sécurité que vous avez créé pour HTTPS. Ensuite, choisissez Ajouter des groupes de sécurité.
9. Sélectionnez Save.

Vérification que le protocole HTTPS utilise le certificat que vous avez configuré

Après avoir ajouté le serveur web à un groupe de sécurité, vous pouvez vérifier que le déchargement SSL/TLS fonctionne avec votre certificat signé par vous-même. Vous pouvez faire cela à l'aide d'un navigateur web ou avec un outil tel qu'OpenSSL s_client.

Pour vérifier le déchargement SSL/TLS à l'aide d'un navigateur web

1. Utilisez un navigateur web pour vous connecter à votre serveur web à l'aide du nom DNS public ou de l'adresse IP du serveur. Assurez-vous que l'URL dans la barre d'adresse commence par https://. Par exemple, **`https://ec2-52-14-212-67.us-east-2.compute.amazonaws.com/`**.

   > **Tip**
   >
   > Vous pouvez utiliser un service DNS tel que Amazon Route 53 pour router le nom de domaine de votre site web (par exemple, https://www.exemple.com/) vers votre serveur web. Pour plus d'informations, consultez Routage du trafic vers une instance Amazon EC2 dans le Guide du Développeur Amazon Route 53 ou dans la documentation de votre service DNS.

2. Utilisez votre navigateur web pour afficher le certificat de serveur web. Pour plus d'informations, consultez les ressources suivantes :

   - Pour Mozilla Firefox, consultez View a Certificate sur le site web de support Mozilla.
   - Pour Google Chrome, consultez Understand Security Issues sur les Outils Google pour site web des développeurs Google.

   D'autres navigateurs web peuvent avoir des fonctions similaires que vous pouvez utiliser pour afficher le certificat de serveur web.

3. Assurez-vous que le certificat SSL/TLS est celui que vous avez configuré votre serveur web à utiliser.

Pour vérifier le déchargement SSL/TLS avec OpenSSL s_client

1. Exécutez la commande OpenSSL suivante pour vous connecter à votre serveur web en utilisant le protocole HTTPS. Remplacez *`<server name>`* par le nom DNS public ou l'adresse IP de votre serveur web.

```
openssl s_client -connect <server name>:443
```

> **Tip**
>
> Vous pouvez utiliser un service DNS tel que Amazon Route 53 pour router le nom de domaine de votre site web (par exemple, https://www.exemple.com/) vers votre serveur web. Pour plus d'informations, consultez Routage du trafic vers une instance Amazon EC2 dans le Guide du Développeur Amazon Route 53 ou dans la documentation de votre service DNS.

2. Assurez-vous que le certificat SSL/TLS est celui que vous avez configuré votre serveur web à utiliser.

Vous disposez maintenant d'un site web sécurisé avec HTTPS. La clé privée du serveur Web est stockée dans un HSM de votre AWS CloudHSM cluster.

Pour ajouter un équilibreur de charge, veuillez consulter Ajouter un équilibreur de charge avec Elastic Load Balancing (facultatif).

Ajouter un équilibreur de charge avec Elastic Load Balancing (facultatif)

Une fois que vous avez configuré le déchargement SSL/TLS avec un serveur web, vous pouvez créer d'autres serveurs web et un équilibreur de charge Elastic Load Balancing qui achemine le trafic HTTPS vers les serveurs web. Un équilibreur de charge peut réduire la charge sur vos serveurs web individuels en équilibrant le trafic entre deux serveurs ou plus. Il peut également augmenter la disponibilité de votre site web, car l'équilibreur de charge surveille l'état de vos serveurs web et achemine uniquement le trafic vers les serveurs sains. Si un serveur web échoue, l'équilibreur de charge arrête automatiquement l'acheminement du trafic vers le serveur.

Rubriques

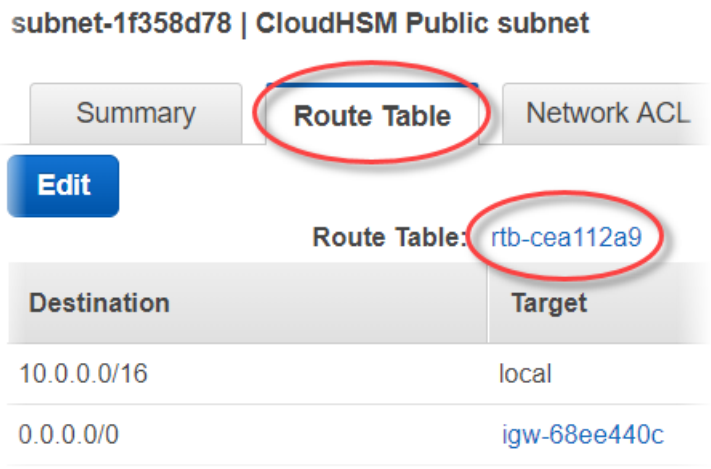
- Créer un sous-réseau pour le deuxième serveur web
- Création du deuxième serveur web
- Créer l'équilibreur de charge

Créer un sous-réseau pour le deuxième serveur web

Avant de créer un autre serveur Web, vous devez créer un nouveau sous-réseau dans le même VPC qui contient votre serveur AWS CloudHSM Web et votre cluster existants.

Pour créer un sous-réseau

1. Ouvrez la section Sous-réseaux de la console Amazon VPC.
2. Choisissez Create Subnet.
3. Dans la boîte de dialogue Créer le sous-réseau (subnet), procédez comme suit :
   a. Pour Balise Nom, saisissez un nom pour votre sous-réseau.
   b. Pour le VPC, choisissez le AWS CloudHSM VPC qui contient votre serveur Web et votre cluster existants. AWS CloudHSM
   c. Pour Zone de disponibilité, choisissez une zone de disponibilité différente de celle qui contient votre serveur web existant.
   d. Pour Bloc d'adresse CIDR IPv4, tapez le bloc d'adresse CIDR à utiliser pour le sous-réseau. Par exemple, saisissez **`10.0.10.0/24`**.
   e. Choisissez Yes, Create.
4. Cochez la case située en regard du sous-réseau public qui contient votre serveur web existant. Ce sous-réseau est différent du sous-réseau public que vous avez créé à l'étape précédente.
5. Dans le volet de contenu, sélectionnez l'onglet Table de routage. Ensuite, cliquez sur le lien de la table de routage.



6. Cochez la case en regard de la table de routage.
7. Sélectionnez l'onglet Associations de sous-réseau. Puis, choisissez Modifier.

8. Cochez la case située en regard du sous-réseau public que vous avez créé précédemment dans cette procédure. Ensuite, choisissez Save (Enregistrer).

Création du deuxième serveur web

Effectuez les étapes suivantes pour créer un deuxième serveur web avec la même configuration que votre serveur web existant.

Pour créer un deuxième serveur web

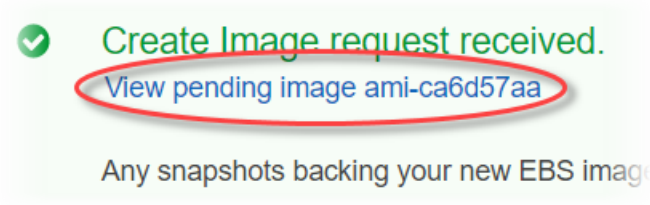
1. Ouvrez la section Instances de la console Amazon EC2.
2. Cochez la case située en regard de votre instance existante de serveur web.
3. Sélectionnez Actions, Image, puis Créer une image.
4. Dans la boîte de dialogue Créer une image, procédez comme suit :

   a. Dans Nom de l'image, tapez un nom pour l'image.

   b. Pour Description de l'image, tapez une description pour l'image.

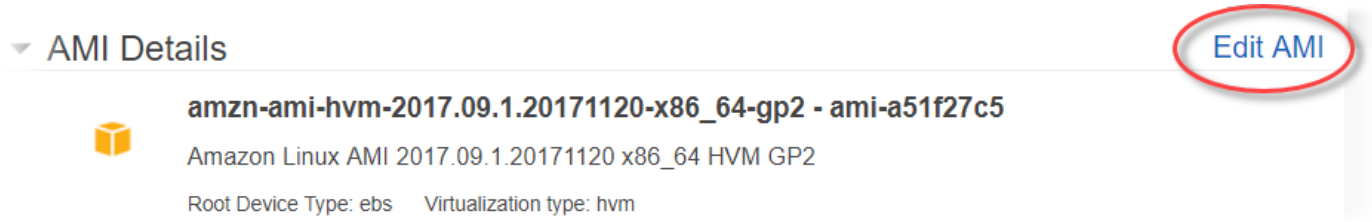
   c. Choisissez Créer une image. Cette action redémarre votre serveur web existant.

   d. Sélectionnez le lien Afficher l'ami-*ID AMI>* de l'image en attente.



   Dans la colonne Statut, notez le statut de votre image. Lorsque votre image a pour statut available (cela peut prendre plusieurs minutes), passez à l'étape suivante.

5. Dans le panneau de navigation, sélectionnez Instances.
6. Cochez la case située en regard de votre serveur web existant.
7. Choisissez Actions, puis En lancer plus comme ceci.
8. Choisissez Modifier l'AMI.



9. Dans le volet de navigation gauche, choisissez Mes AMI. Ensuite, effacez le texte dans la zone de recherche.
10. À côté de l'image de votre serveur web, choisissez Sélectionner.
11. Choisissez Oui, je veux continuer avec cette AMI (***<image name>*** - ami-***<AMI ID>***).
12. Choisissez Suivant.
13. Sélectionnez un type d'instance, puis choisissez Suivant : Configurer les détails de l'instance.
14. Pour Etape 3 : Configurer les détails de l'instance, procédez comme suit :

    a. Pour Réseau, choisissez le VPC qui contient votre serveur web existant.

    b. Pour Sous-réseau, choisissez le sous-réseau public que vous avez créé pour le deuxième serveur web.

    c. Pour Attribuer automatiquement l'adresse IP publique, choisissez Activer.

    d. Modifiez les autres options des détails de l'instance comme vous le souhaitez. Puis choisissez Suivant : Ajouter le stockage.
15. Modifiez les paramètres de stockage comme souhaité. Choisissez ensuite Suivant : Ajouter des balises.
16. Ajoutez ou modifiez les balises comme souhaité. Choisissez ensuite Suivant : Configurer le groupe de sécurité.
17. Pour Etape 6 : Configurer le groupe de sécurité, procédez comme suit :

    a. Pour Attribuer un groupe de sécurité, choisissez Select an existing security group (Sélectionner un groupe de sécurité existant).

    b. Cochez la case à côté du groupe de sécurité nommé cloudhsm-***<cluster ID>***-sg. AWS CloudHSM a créé ce groupe de sécurité en votre nom lorsque vous avez créé le cluster. Vous devez choisir ce groupe de sécurité pour autoriser l'instance de serveur web à se connecter aux modules HSM du cluster.

    c. Cochez la case en regard du groupe de sécurité qui autorise le trafic HTTPS entrant. Vous avez créé ce groupe de sécurité précédemment.

    d. (Facultatif) Activez la case à cocher en regard d'un groupe de sécurité qui autorise le trafic SSH entrant (pour Linux) ou RDP (pour Windows) depuis votre réseau. En d'autres termes, le groupe de sécurité doit autoriser le trafic TCP entrant sur le port 22 (pour SSH sous Linux) ou le port 3389 (pour RDP sous Windows). Sinon, vous ne pouvez pas vous connecter à votre instance client. Si vous n'avez pas de groupe de sécurité de ce type, vous devez en créer un, puis l'attribuer ultérieurement à votre instance client.

Choisissez Review and Launch.

18. Vérifiez les détails de votre instance, puis choisissez Lancement.
19. Choisissez si vous souhaitez démarrer votre instance avec une paire de clés existante, créer une paire de clés ou lancer votre instance sans paire de clés.

    - Pour utiliser une paire de clés existante, procédez comme suit :
      1. Choisissez Choisir une paire de clés existante.
      2. Pour Sélectionner une paire de clés, choisissez la paire de clés à utiliser.
      3. Activez la case à cocher en regard de Je reconnais que j'ai accès au fichier de clé privée sélectionné (pem ***<private key file name>***), et que, sans ce fichier, je ne suis pas capable de me connecter à mon instance.
    - Pour créer une paire de clés, procédez comme suit :
      1. Choisissez Créer une nouvelle paire de clés.
      2. Pour Nom de la paire de clés, saisissez un nom de paire de clés.
      3. Choisissez Télécharger une paire de clés et enregistrer le fichier de clé privée dans un endroit sûr et accessible.

      > ⚠ Warning
      >
      > Vous ne pouvez pas télécharger une nouvelle fois le fichier de clé privée après ce stade. Si vous ne téléchargez pas le fichier de clé privée maintenant, vous ne pourrez pas accéder à l'instance client.

    - Pour démarrer votre instance sans une paire de clés, effectuez les opérations suivantes :
      1. Choisissez Continuer sans paire de clés.
      2. Cochez la case en regard de Je reconnais ne pas être en mesure de me connecter à cette instance si je ne connais pas déjà le mot de passe intégré à cette AMI.

    Choisissez Launch Instances (Démarrer les instances).

Créer l'équilibreur de charge

Effectuez les étapes suivantes pour créer un équilibreur de charge Elastic Load Balancing qui achemine le trafic HTTPS vers vos serveurs web.

Pour créer un équilibreur de charge

1. Ouvrez la page des équilibreurs de charge de la console Amazon EC2.
2. Sélectionnez Create Load Balancer (Créer un équilibreur de charge).
3. Dans la section Équilibreur de charge du réseau, choisissez Créer.
4. Pour Étape 1 : Configurer l'équilibreur de charge, procédez comme suit :
    a. Pour Nom, entrez un nom pour l'équilibreur de charge que vous créez.
    b. Dans la section Écouteurs, pour Port de l'équilibreur de charge, modifiez la valeur en **443**.
    c. Dans la section Zones de disponibilité, pour VPC, choisissez le VPC qui contient vos serveurs web.
    d. Dans la section Zones de disponibilité, choisissez les sous-réseaux qui contiennent vos serveurs web.
    e. Choisissez Next: Configure Routing (Suivant :Configurer le routage).
5. Pour Étape 2 : Configurer le routage, procédez comme suit :
    a. Pour Nom, entrez le nom du groupe cible que vous créez.
    b. Pour Port, modifiez la valeur en **443**.
    c. Choisissez Next: Register Targets (Suivant : Enregistrer des cibles).
6. Pour Étape 3 : Enregistrer les cibles, procédez comme suit :
    a. Dans la section Instances, cochez les cases en regard de vos instances de serveur web. Ensuite, choisissez Ajouter au membre.
    b. Choisissez Suivant : vérification.
7. Passez en revue les détails de votre équilibreur de charge, puis cliquez sur Créer.
8. Lorsque l'équilibreur de charge a été créé avec succès, cliquez sur Fermer.

Une fois que vous avez terminé les étapes précédentes, la console Amazon EC2 affiche votre équilibreur de charge Elastic Load Balancing.

Lorsque l'état de votre équilibreur de charge est actif, vous pouvez vérifier que l'équilibreur de charge fonctionne. Autrement dit, vous pouvez vérifier qu'il envoie le trafic HTTPS à vos serveurs Web avec déchargement SSL/TLS avec AWS CloudHSM. Vous pouvez pour cela utiliser un navigateur web ou un outil tel qu'OpenSSL s_client.

Pour vérifier que votre équilibreur de charge fonctionne avec un navigateur web

1. Dans la console Amazon EC2, trouvez le nom DNS de l'équilibreur de charge que vous venez de créer. Puis, sélectionnez le nom DNS et copiez-le.
2. Utilisez un navigateur web tel que Mozilla Firefox ou Google Chrome pour vous connecter à votre équilibreur de charge en utilisant son nom DNS. Assurez-vous que l'URL dans la barre d'adresse commence par https://.

   > **Tip**
   >
   > Vous pouvez utiliser un service DNS tel que Amazon Route 53 pour router le nom de domaine de votre site web (par exemple, https://www.exemple.com/) vers votre serveur web. Pour plus d'informations, consultez Routage du trafic vers une instance Amazon EC2 dans le Guide du Développeur Amazon Route 53 ou dans la documentation de votre service DNS.

3. Utilisez votre navigateur web pour afficher le certificat de serveur web. Pour plus d'informations, consultez les ressources suivantes :

   - Pour Mozilla Firefox, consultez View a Certificate sur le site web de support Mozilla.
   - Pour Google Chrome, consultez Understand Security Issues sur les Outils Google pour site web des développeurs Google.

   D'autres navigateurs web peuvent avoir des fonctions similaires que vous pouvez utiliser pour afficher le certificat de serveur web.
4. Assurez-vous que le certificat est celui que vous avez configuré le serveur web à utiliser.

Pour vérifier que votre équilibreur de charge fonctionne avec OpenSSL s_client

1. Utilisez la commande OpenSSL suivante pour vous connecter à votre équilibreur de charge en utilisant le protocole HTTPS. Remplacez *DNS name>* par le nom DNS de votre équilibreur de charge.

```
openssl s_client -connect <DNS name>:443
```

> **Tip**
>
> Vous pouvez utiliser un service DNS tel que Amazon Route 53 pour router le nom de domaine de votre site web (par exemple, https://www.exemple.com/) vers votre serveur web. Pour plus d'informations, consultez Routage du trafic vers une instance Amazon EC2 dans le Guide du Développeur Amazon Route 53 ou dans la documentation de votre service DNS.

2. Assurez-vous que le certificat est celui que vous avez configuré le serveur web à utiliser.

Vous disposez désormais d'un site Web sécurisé par HTTPS, la clé privée du serveur Web étant stockée dans un HSM de votre AWS CloudHSM cluster. Votre site web dispose de deux serveurs web et d'un équilibreur de charge afin d'améliorer l'efficacité et la disponibilité.

Configuration de Windows Server en tant qu'autorité de certification (CA) avec AWS CloudHSM

Dans une infrastructure à clés publiques (PKI), une autorité de certification est une entité de confiance qui émet des certificats numériques. Ces certificats numériques lient une clé publique à une identité (une personne ou une organisation) à l'aide du chiffrement de la clé publique et des signatures numériques. Pour exploiter une autorité de certification, vous devez garantir la confiance en protégeant les clés privées qui signent les certificats délivrés par votre autorité de certification. Vous pouvez stocker ces clés privées dans le HSM de votre cluster AWS CloudHSM , et utiliser celui-ci pour effectuer les opérations de signature par chiffrement.

Dans ce didacticiel, vous allez utiliser Windows Server et AWS CloudHSM configurer une autorité de certification. Vous installez le logiciel client AWS CloudHSM pour Windows sur votre serveur Windows Server, puis vous ajoutez le rôle de services de certificats Active Directory (AD CS) à votre serveur Windows Server. Lorsque vous configurez ce rôle, vous utilisez un fournisseur de stockage de AWS CloudHSM clés (KSP) pour créer et stocker la clé privée de l'autorité de certification sur votre AWS CloudHSM cluster. Le KSP est le pont qui connecte votre serveur Windows à votre AWS CloudHSM cluster. Pour finir, vous signez une demande de signature de certificat (CSR) avec votre CA Windows Server.

Pour plus d'informations, consultez les rubriques suivantes :

Rubriques

- CA Windows Server étape 1: Définir les prérequis
- CA Windows Server étape 2 : Créer une CA Windows Server avec AWS CloudHSM
- Étape 3 de l'autorité de certification Windows Server : signez une demande de signature de certificat (CSR) auprès de votre autorité de certification Windows Server avec AWS CloudHSM

CA Windows Server étape 1: Définir les prérequis

Pour configurer Windows Server en tant qu'autorité de certification (CA) auprès de AWS CloudHSM, vous avez besoin des éléments suivants :

- AWS CloudHSM Cluster actif avec au moins un HSM.
- Une instance Amazon EC2 exécutant un système d'exploitation Windows Server avec le logiciel AWS CloudHSM client pour Windows installé. Ce didacticiel utilise Microsoft Windows Server 2016.
- Un utilisateur de chiffrement (CU) propriétaire et gestionnaire de la clé privée de la CA sur le HSM.

Pour configurer les conditions requises pour une autorité de certification Windows Server avec AWS CloudHSM

1. Suivez les étapes de Premiers pas. Lorsque vous lancez le client Amazon EC2, choisissez une AMI Windows Server. Ce didacticiel utilise Microsoft Windows Server 2016. Une fois que vous avez terminé ces étapes, vous disposez d'un cluster actif avec au moins un HSM. Vous disposez également d'une instance client Amazon EC2 exécutant Windows Server sur laquelle le logiciel AWS CloudHSM client pour Windows est installé.
2. (Facultatif) Ajoutez d'autres HSM à votre cluster. Pour plus d'informations, consultez Ajout d'un HSM.
3. Connectez-vous à votre instance client . Pour plus d'informations, consultez Connexion à votre instance dans le Guide de l'utilisateur Amazon EC2 pour les instances Windows.
4. Créez un utilisateur de chiffrement (CU) à l'aide de la gestion des utilisateurs HSM avec la CLI CloudHSM ou de la gestion des utilisateurs HSM à l'aide de l'Utilitaire de gestion CloudHSM (CMU). Conservez le nom d'utilisateur et le mot de passe du CU. Vous en aurez besoin pour terminer l'étape suivante.

5. Définissez les informations d'identification de connexion pour le HSM, à l'aide du nom d'utilisateur et du mot de passe CU que vous avez créés à l'étape précédente.
6. À l'étape 5, si vous avez utilisé le Gestionnaire d'informations d'identification Windows pour définir les informations d'identification HSM, téléchargez `psexec.exe`depuis SysInternals pour exécuter la commande suivante sous le nom NT Authority \ SYSTEM :

   ```
   psexec.exe -s "C:\Program Files\Amazon\CloudHsm\tools\set_cloudhsm_credentials.exe"
    --username <USERNAME> --password <PASSWORD>
   ```

   Remplacez *<USERNAME>* et *<PASSWORD>* par les informations d'identification HSM.

Pour créer une autorité de certification Windows Server avec AWS CloudHSM, rendez-vous surCréer une CA Windows Server.

CA Windows Server étape 2 : Créer une CA Windows Server avec AWS CloudHSM

Pour créer une CA Windows Server, vous ajoutez le rôle de services de certificat Active Directory (AD CS) à votre serveur Windows Server. Lorsque vous ajoutez ce rôle, vous utilisez un fournisseur de stockage de AWS CloudHSM clés (KSP) pour créer et stocker la clé privée de l'autorité de certification sur votre AWS CloudHSM cluster.

> **Note**
>
> Lorsque vous créez votre CA Windows Server, vous pouvez choisir de créer une CA racine ou CA subordonnée. En général, vous prenez cette décision en fonction de la conception de votre infrastructure à clés publiques et des stratégies de sécurité de votre organisation. Ce didacticiel explique comment créer une CA racine pour plus de simplicité.

Pour ajouter le rôle AD CS à votre serveur Windows Server et créer la clé privée de la CA

1. Si ce n'est pas déjà fait, connectez-vous à votre serveur Windows Server. Pour plus d'informations, consultez Connexion à votre instance dans le Guide de l'utilisateur Amazon EC2 pour les instances Windows.
2. Sur votre serveur Windows Server, démarrez le Server Manager (Gestionnaire de serveurs).

3. Dans le tableau de bord du Server Manager (Gestionnaire de serveurs), choisissez Add roles and features (Ajouter des rôles et des fonctions).
4. Prenez connaissances des informations contenues dans le fichier Before you begin (Avant de commencer), puis choisissez Next (Suivant).
5. Pour Installation Type (Type d'installation), choisissez Role-based or feature-based installation (Installation basée sur un rôle ou une fonction). Ensuite, sélectionnez Suivant.
6. Pour Server Selection (Sélection de serveur), choisissez Select a server from the server pool (Sélectionner un serveur du pool de serveurs). Ensuite, sélectionnez Suivant.
7. Pour Server Roles (Rôles de serveur), procédez comme suit :
    a. Sélectionnez Active Directory Certificate Services (Services de certificat Active Directory).
    b. Pour Add features that are required for Active Directory Certificate Services (Ajouter des fonctions qui sont requises pour les services de certificats Active Directory), choisissez Add Features (Ajouter des fonctions).
    c. Choisissez Suivant pour finaliser la sélection de rôles de serveur.
8. Pour Fonctions, acceptez les valeurs par défaut, puis choisissez Next (Suivant).
9. Pour AD CS, procédez comme suit :
    a. Choisissez Suivant.
    b. Sélectionnez Certification Authority (Autorité de certification), puis choisissez Next (Suivant).
10. Pour Confirmation, lisez les informations de confirmation, puis choisissez Installer. Ne fermez pas la fenêtre.
11. Cliquez sur le lien en surbrillance Configurer les services de certificats Active Directory sur le serveur de destination.
12. Pour Informations d'identification, vérifiez ou modifiez les informations d'identification affichées. Ensuite, sélectionnez Suivant.
13. Pour Role Services (Services de rôle), sélectionnez Certification Authority (Autorité de certification). Ensuite, sélectionnez Suivant.
14. Pour Setup Type (Type de configuration), sélectionnez Standalone CA (CA autonome). Ensuite, sélectionnez Suivant.
15. Pour CA Type (Type de CA), sélectionnez Root CA (CA racine). Ensuite, sélectionnez Suivant.

> **Note**
>
> Vous pouvez choisir de créer une CA racine ou une CA subordonnée en fonction de la conception de votre infrastructure à clés publiques et des stratégies de sécurité de votre organisation. Ce didacticiel explique comment créer une CA racine pour plus de simplicité.

16. Pour Private Key (Clé privée), sélectionnez Create a new private key (Créer une nouvelle clé privée). Ensuite, sélectionnez Suivant.
17. Dans Cryptography (Chiffrement), procédez comme suit :

    a. Pour Select a cryptographic provider (Sélectionner un fournisseur de services de chiffrement), choisissez l'une des options Cavium Key Storage Provider (Fournisseur de stockage de clés Cavium) du menu. Il s'agit des principaux fournisseurs de stockage de clés AWS CloudHSM . Par exemple, vous pouvez choisir RSA # Cavium Key Storage Provider (Fournisseur de stockage de clés Cavium RSA #).

    b. Pour Key length (Longueur de la clé), choisissez l'une des options de longueur de clé.

    c. Pour Select the hash algorithm for signing certificates issued by this CA (Sélectionner l'algorithme de hachage pour signer les certificats émis par cette CA), choisissez l'une des options de l'algorithme de hachage.

    Choisissez Suivant.
18. Dans CA Name (Nom de la CA), procédez comme suit :

    a. (Facultatif) Modifiez le nom commun.

    b. (Facultatif) Tapez un suffixe de nom unique.

    Choisissez Suivant.
19. Pour Période de validité, spécifiez une période de plusieurs années, mois, semaines ou jours. Ensuite, sélectionnez Suivant.
20. Pour Certificate Database (Base de données de certificats), vous pouvez accepter les valeurs par défaut ou, le cas échéant, modifier l'emplacement pour la base de données et le journal de base de données. Ensuite, sélectionnez Suivant.
21. Pour Confirmation, vérifiez les informations sur la CA, puis choisissez Configurer.

22. Choisissez Close (Fermer), puis à nouveau Close (Fermer).

Vous disposez désormais d'une autorité de certification Windows Server avec AWS CloudHSM. Pour savoir comment signer une demande de signature de certificat (CSR) avec votre CA, consultez Signer une CSR.

Étape 3 de l'autorité de certification Windows Server : signez une demande de signature de certificat (CSR) auprès de votre autorité de certification Windows Server avec AWS CloudHSM

Vous pouvez utiliser votre autorité de certification Windows Server AWS CloudHSM pour signer une demande de signature de certificat (CSR). Pour effectuer ces étapes, vous avez besoin d'une CSR valide. Vous pouvez créer une CSR de différentes manières, notamment :

- À l'aide d'OpenSSL
- À l'aide du gestionnaire d'Internet Information Services (IIS) de Windows Server
- À l'aide du composant logiciel enfichable des certificats dans la console de gestion Microsoft
- À l'aide de l'utilitaire de ligne de commande certreq sous Windows

La procédure à suivre pour créer une CSR ne sont pas pris en compte dans le cadre de ce didacticiel. Une fois que vous disposez d'une CSR, vous pouvez la signer avec votre CA Windows Server.

Pour signer une CSR avec votre CA Windows Server

1. Si ce n'est pas déjà fait, connectez-vous à votre serveur Windows Server. Pour plus d'informations, consultez Connexion à votre instance dans le Guide de l'utilisateur Amazon EC2 pour les instances Windows.
2. Sur votre serveur Windows Server, démarrez le Server Manager (Gestionnaire de serveurs).
3. Dans le tableau de bord Server Manager (Gestionnaire de serveurs), dans le coin supérieur droit, choisissez Outils, Certification Authority (Autorité de certification).
4. Dans la fenêtre Certification Authority (Autorité de certification), choisissez le nom de votre ordinateur.
5. Dans le menu Action, choisissez All Tasks (Toutes les tâches), Submit new request (Envoyer une nouvelle demande).

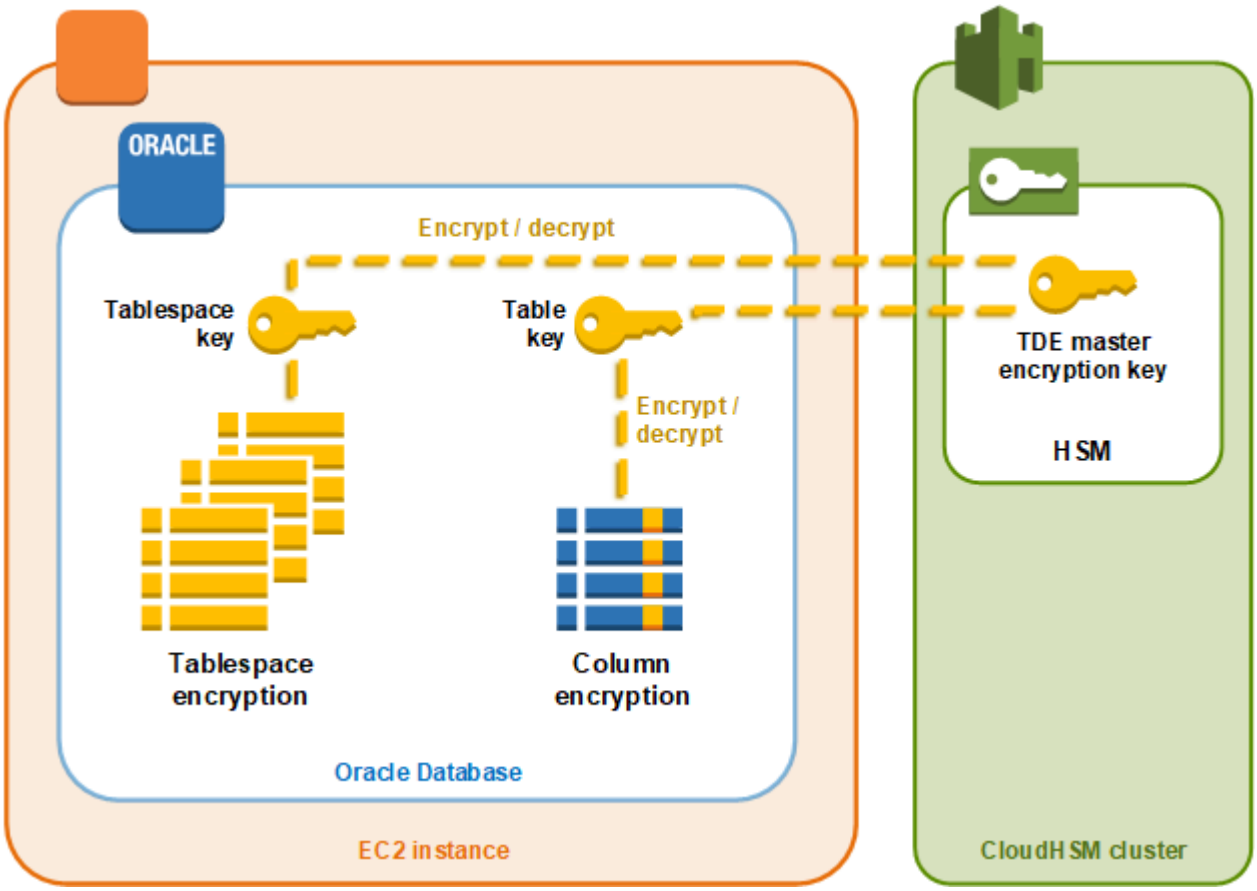
6. Sélectionnez votre fichier de CSR, puis choisissez Ouvrir.
7. Dans la fenêtre Certification Authority (Autorité de certification), double-cliquez sur Pending Requests (Demandes en attente).
8. Sélectionnez la demande en attente. Puis, dans le menu Action, choisissez All Tasks (Toutes les tâches), Issue (Émettre).
9. Dans la fenêtre Certification Authority (Autorité de certification), double-cliquez sur Issued Requests (Demandes émises) pour afficher le certificat signé.
10. (Facultatif) Pour exporter le certificat signé vers un fichier, procédez comme suit :

    a. Dans la fenêtre Certification Authority (Autorité de certification), double-cliquez sur le certificat.
    b. Choisissez l'onglet Détails, puis Copy to File (Copier dans un fichier).
    c. Suivez les instructions fournies dans Certificate Export Wizard (Assistant d'exportation de certificat).

Vous disposez désormais d'une autorité de certification Windows Server avec AWS CloudHSM et d'un certificat valide signé par l'autorité de certification Windows Server.

Chiffrement transparent des données (TDE) des bases de données Oracle avec AWS CloudHSM

Le chiffrement TDE (Transparent Data Encryption) est utilisé pour chiffrer les fichiers de base de données. Avec le TDE, le logiciel de base de données chiffre les données avant de les stocker sur le disque. Les données contenues dans les colonnes et les espaces de table de la base de données sont chiffrées avec une clé de table ou une clé d'espace de table. Certaines versions du logiciel de base de données d'Oracle proposent le TDE. Dans Oracle TDE, ces clés sont chiffrées avec une clé de chiffrement principale du TDE. Vous pouvez améliorer la sécurité en stockant la clé de chiffrement principale TDE dans les HSM de votre AWS CloudHSM cluster.



Dans cette solution, vous utilisez le système Oracle Database installé sur une instance Amazon EC2. Oracle Database s'intègre à la bibliothèque de logiciels AWS CloudHSM pour PKCS #11 pour stocker la clé principale TDE dans les HSM de votre cluster.

> **Important**
>
> - Nous recommandons d'installer Oracle Database sur une instance Amazon EC2.

Exécutez les étapes suivantes pour réaliser l'intégration d'Oracle TDE avec AWS CloudHSM.

Pour configurer l'intégration d'Oracle TDE avec AWS CloudHSM

1. Suivez les étapes de Configuration des prérequis pour préparer votre environnement.
2. Suivez les étapes décrites Configurer la base de données pour configurer Oracle Database afin de l'intégrer à votre AWS CloudHSM cluster.

Oracle TDE avec AWS CloudHSM : Configuration des prérequis

Pour réaliser l'intégration avec Oracle TDE AWS CloudHSM, vous avez besoin des éléments suivants :

- AWS CloudHSM Cluster actif avec au moins un HSM.
- Une instance Amazon EC2 exécutant un système d'exploitation Linux avec les logiciels suivants installés :
  - Le AWS CloudHSM client et les outils de ligne de commande.
  - La bibliothèque AWS CloudHSM logicielle pour PKCS #11.
  - Base de données Oracle. AWS CloudHSM prend en charge l'intégration d'Oracle TDE. Le SDK client 5.6 et versions ultérieures prennent en charge Oracle TDE for Oracle Database 19c. Le SDK client 3 prend en charge Oracle TDE pour les versions 11g et 12c d'Oracle Database.
- Un utilisateur de chiffrement (CU) propriétaire et gestionnaire de la clé de chiffrement principale TDE sur les HSM de votre cluster.

Exécutez les étapes suivantes pour installer tous les prérequis.

Pour configurer les conditions préalables à l'intégration d'Oracle TDE avec AWS CloudHSM

1. Suivez les étapes de Premiers pas. Une fois que vous aurez terminé, vous disposerez d'un cluster actif avec un HSM. Vous aurez aussi une instance Amazon EC2 exécutant le système d'exploitation Amazon Linux. Le AWS CloudHSM client et les outils de ligne de commande seront également installés et configurés.
2. (Facultatif) Ajoutez d'autres HSM à votre cluster. Pour plus d'informations, consultez Ajout d'un HSM.
3. Connectez-vous à votre instance client Amazon EC2 et procédez comme suit :
   a. Installez la bibliothèque AWS CloudHSM logicielle pour PKCS #11.
   b. Installez Oracle Database. Pour plus d'informations, consultez la documentation Oracle Database. Le SDK client 5.6 et versions ultérieures prennent en charge Oracle TDE for Oracle Database 19c. Le SDK client 3 prend en charge Oracle TDE pour les versions 11g et 12c d'Oracle Database.
   c. Utilisez l'outil de ligne de commande cloudhsm_mgmt_util pour créer un utilisateur de chiffrement (CU) sur votre cluster. Pour plus d'informations sur la création d'un CU, consultez Comment gérer les utilisateurs HSM avec CMU et Gestion des utilisateurs HSM.

Une fois que vous avez terminé ces étapes, vous pouvez Configurer la base de données.

Oracle TDE avec AWS CloudHSM : Configuration de la base de données et génération de la clé de chiffrement principale

Pour intégrer Oracle TDE à votre AWS CloudHSM cluster, consultez les rubriques suivantes :

1. Mise à jour de la configuration d'Oracle Database pour utiliser les HSM de votre cluster en tant que module de sécurité externe. Pour plus d'informations sur les modules de sécurité externes, consultez Introduction to Transparent Data Encryption dans le manuel Oracle Database Advanced Security Guide.
2. Génération de la clé de chiffrement principale Oracle TDE sur les HSM de votre cluster.

Mise à jour de la configuration d'Oracle Database

Pour mettre à jour la configuration d'Oracle Database afin d'utiliser un HSM dans votre cluster comme module de sécurité externe, exécutez les étapes suivantes. Pour plus d'informations sur les modules de sécurité externes, consultez Introduction to Transparent Data Encryption dans le manuel Oracle Database Advanced Security Guide.

Pour mettre à jour la configuration Oracle

1. Connectez-vous à votre instance client EC2 Amazon. Il s'agit de l'instance sur laquelle vous avez installé Oracle Database.
2. Effectuez une copie de sauvegarde du fichier `sqlnet.ora`. Pour l'emplacement du fichier, reportez-vous à la documentation Oracle.
3. Utilisez un éditeur de texte pour modifier le fichier de nommé `sqlnet.ora`.
   Ajoutez la ligne suivante. Si une ligne existante dans le fichier commence par `encryption_wallet_location`, remplacez la ligne existante par la suivante.

   ```
   encryption_wallet_location=(source=(method=hsm))
   ```

   Enregistrez le fichier.
4. Exécutez la commande suivante pour créer le répertoire dans lequel Oracle Database prévoit de trouver le fichier de bibliothèque de la bibliothèque logicielle AWS CloudHSM PKCS #11.

   ```
   sudo mkdir -p /opt/oracle/extapi/64/hsm
   ```

5. Exécutez la commande suivante pour copier la bibliothèque AWS CloudHSM logicielle du fichier PKCS #11 dans le répertoire que vous avez créé à l'étape précédente.

```
sudo cp /opt/cloudhsm/lib/libcloudhsm_pkcs11.so /opt/oracle/extapi/64/hsm/
```

> **Note**
>
> Le répertoire `/opt/oracle/extapi/64/hsm` doit contenir un seul fichier de bibliothèque. Supprimez tous les autres fichiers qui existent dans ce répertoire.

6. Exécutez la commande suivante pour modifier le propriétaire du répertoire `/opt/oracle` et tout ce qui y figure à l'intérieur.

```
sudo chown -R oracle:dba /opt/oracle
```

7. Démarrez Oracle Database.

Génération de la clé de chiffrement principale Oracle TDE

Pour générer la clé principale Oracle TDE sur les HSM de votre cluster, procédez comme indiqué dans la procédure suivante.

Pour générer la clé principale

1. Utilisez la commande suivante pour ouvrir Oracle SQL*Plus. Lorsque vous y êtes invité, saisissez le mot de passe système que vous avez défini lors de l'installation d'Oracle Database.

```
sqlplus / as sysdba
```

> **Note**
>
> Pour le SDK client 3, vous devez définir la variable d'environnement `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE` chaque fois que vous générez une clé principale. Cette variable n'est nécessaire que pour la génération de la clé principale. Pour plus d'informations, consultez « Problème : Oracle définit l'attribut PCKS #11 `CKA_MODIFIABLE` lors de la génération de la clé principale, mais le HSM ne le prend pas en charge » dans Problèmes connus liés à l'intégration d'applications tierces.

2. Exécutez l'instruction SQL qui crée la clé principale de chiffrement, comme indiqué dans les exemples suivants. Utilisez l'instruction qui correspond à votre version d'Oracle Database. Remplacez *<CU user name>* par le nom d'utilisateur de l'utilisateur de chiffrement (CU). Remplacez *<password>* par le mot de passe de l'utilisateur de chiffrement (CU).

   > **Important**
   >
   > Exécutez la commande suivante une seule fois. Chaque fois que la commande est exécutée, elle crée une nouvelle clé de chiffrement principale.

   - Pour Oracle Database version 11, exécutez l'instruction SQL suivante.

     ```
     SQL> alter system set encryption key identified by "<CU user name>:<password>";
     ```

   - Pour Oracle Database version 12 et version 19c, exécutez l'instruction SQL suivante.

     ```
     SQL> administer key management set key identified by "<CU user
      name>:<password>";
     ```

   Si la réponse est `System altered` ou `keystore altered`, vous avez généré et défini avec succès la clé principale pour Oracle TDE.

3. (Facultatif) Exécutez la commande suivante pour vérifier le statut du portefeuille Oracle.

   ```
   SQL> select * from v$encryption_wallet;
   ```

   Si le portefeuille n'est pas ouvert, utilisez l'une des commandes suivantes pour l'ouvrir. Remplacez *<CU user name>* par le nom de l'utilisateur de chiffrement (CU). Remplacez *<password>* par le mot de passe de l'utilisateur de chiffrement (CU).

   - Pour Oracle 11, exécutez la commande suivante pour ouvrir le portefeuille.

     ```
     SQL> alter system set encryption wallet open identified by "<CU user
      name>:<password>";
     ```

     Pour fermer manuellement le portefeuille, exécutez la commande suivante.

```
SQL> alter system set encryption wallet close identified by "<CU user
 name>:<password>";
```

- Pour Oracle 12 et Oracle 19c, exécutez la commande suivante pour ouvrir le portefeuille.

```
SQL> administer key management set keystore open identified by "<CU user
 name>:<password>";
```

Pour fermer manuellement le portefeuille, exécutez la commande suivante.

```
SQL> administer key management set keystore close identified by "<CU user
 name>:<password>";
```

Utiliser Microsoft SignTool with AWS CloudHSM pour signer des fichiers

Dans la cryptographie et l'infrastructure à clés publiques (PKI), les signatures numériques sont utilisées pour confirmer que les données ont été envoyés par une entité de confiance. Les signatures indiquent également que les données n'ont pas été altérées pendant le transit. Une signature est un hachage chiffré qui est généré avec la clé privée de l'expéditeur. Le destinataire peut vérifier l'intégrité des données en déchiffrant la signature de hachage avec la clé publique de l'expéditeur. En revanche, il est de la responsabilité de l'expéditeur de conserver un certificat numérique. Le certificat numérique démontre la propriété de la clé privée par l'expéditeur et fournit au destinataire la clé publique qui est nécessaire pour le déchiffrement. Tant que la clé privée appartient à l'expéditeur, la signature est fiable. AWS CloudHSM fournit du matériel sécurisé validé FIPS 140-2 de niveau 3 pour vous permettre de sécuriser ces clés avec un accès exclusif à locataire unique.

De nombreuses entreprises utilisent Microsoft SignTool, un outil de ligne de commande qui signe, vérifie et horodate les fichiers afin de simplifier le processus de signature de code. Vous pouvez les utiliser AWS CloudHSM pour stocker en toute sécurité vos paires de clés jusqu'à ce qu'elles soient nécessaires SignTool, créant ainsi un flux de travail facilement automatisable pour la signature des données.

Les rubriques suivantes fournissent une vue d'ensemble de la façon de l'utiliser SignTool avec AWS CloudHSM :

Rubriques

Microsoft SignTool avec AWS CloudHSM étape 1 : configurer les prérequis

Pour utiliser Microsoft SignTool avec AWS CloudHSM, vous devez disposer des éléments suivants :

- Une instance de client Amazon EC2 exécutant un système d'exploitation Windows.
- Une autorité de certification (CA), soit en auto-gestion ou établie par un fournisseur tiers.
- Un AWS CloudHSM cluster actif dans le même cloud public virtuel (VPC) que votre instance EC2. Le cluster doit contenir au moins un HSM.
- Un utilisateur cryptographique (CU) pour posséder et gérer les clés du AWS CloudHSM cluster.
- Un fichier non signé ou un fichier exécutable.
- Le kit de développement logiciel (SDK) Microsoft Windows.

Pour configurer les conditions requises pour une utilisation AWS CloudHSM avec Windows SignTool

1. Suivez les instructions fournies dans la section Premiers pas de ce guide pour démarrer une instance Windows EC2 et un cluster AWS CloudHSM .
2. Si vous souhaitez héberger votre propre autorité de certification Windows Server, suivez les étapes 1 et 2 de la section Configuration de Windows Server en tant qu'autorité de certification avec AWS CloudHSM. Dans le cas contraire, continuez à utiliser votre autorité de certification tierce approuvée publiquement.
3. Téléchargez et installez l'une des versions suivantes du kit SDK Microsoft Windows sur votre instance Windows EC2 :

   - Kit SDK Microsoft Windows 10
   - Kit SDK Microsoft Windows 8.1
   - Kit SDK Microsoft Windows 7

   Le fichier exécutable `SignTool` fait partie de la fonction d'installation Windows SDK Signing Tools for Desktop Apps. Vous pouvez omettre l'installation des autres fonctionnalités si vous n'en avez pas besoin. L'emplacement d'installation par défaut est :

```
C:\Program Files (x86)\Windows Kits\<SDK version>\bin\<version number>\<CPU
 architecture>\signtool.exe
```

Vous pouvez désormais utiliser le SDK Microsoft Windows, votre AWS CloudHSM cluster et votre autorité de certification pour créer un certificat de signature.

Microsoft SignTool avec AWS CloudHSM étape 2 : créer un certificat de signature

Maintenant que vous avez téléchargé le kit SDK Windows sur votre instance EC2, vous pouvez l'utiliser pour générer une demande de signature de certificat (CSR). La CSR est un certificat non signé qui est, à terme, transmis à votre CA pour signature. Dans cet exemple, nous utilisons le fichier exécutable `certreq` qui est inclus dans le kit SDK Windows pour générer la CSR.

Pour générer une CSR en utilisant le fichier exécutable **`certreq`**

1. Si ce n'est pas déjà fait, connectez-vous à votre instance EC2 Windows. Pour plus d'informations, consultez Connexion à votre instance dans le Guide de l'utilisateur Amazon EC2 pour les instances Windows.
2. Créez un fichier nommé `request.inf` qui contient les lignes ci-dessous. Remplacez les informations `Subject` par celles de votre entreprise. Pour une explication de chaque paramètre, consultez la documentation de Microsoft.

```
[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. Exécutez `certreq.exe`. Pour cet exemple, nous enregistrons la CSR en tant que `request.csr`.

```
certreq.exe -new request.inf request.csr
```

   En interne, une nouvelle paire de clés est générée sur votre AWS CloudHSM cluster, et la clé privée de la paire est utilisée pour créer le CSR.

4. Envoyez la CSR à votre autorité de certification. Si vous utilisez une CA Windows Server, procédez comme suit :

   a. Entrez la commande suivante pour ouvrir l'outil CA

```
certsrv.msc
```

   b. Dans la nouvelle fenêtre, cliquez avec le bouton droit de la souris sur le nom du serveur CA. Choisissez All Tasks (Toutes les tâches), puis choisissez Submit new request (Envoyer une nouvelle demande).

   c. Accédez à l'emplacement de `request.csr` et choisissez Ouvrir.

   d. Accédez au dossier Pending Requests (Demandes en attente) en développant le menu Server CA. Cliquez avec le bouton droit de la souris sur la demande que vous venez de créer puis, dans All Tasks (Toutes les tâches), choisissez Issue.

   e. Maintenant, accédez au dossier Issued Certificates (Certificats émis) (au-dessus du dossier Demandes en attente).

   f. Choisissez Ouvrir pour afficher le certificat, puis cliquez sur l'onglet Détails.

   g. Choisissez Copy to File (Copier dans un fichier) pour démarrer le Certificate Export Wizard (Assistant d'exportation de certificat). Enregistrez le fichier X.509 codé DER dans un emplacement sûr en tant que `signedCertificate.cer`.

   h. Quittez l'outil CA et utilisez la commande suivante, qui permet de déplacer le fichier de certificat dans le magasin de certificats personnel dans Windows. Il peut ensuite être utilisé par d'autres applications.

```
certreq.exe -accept signedCertificate.cer
```

Vous pouvez désormais utiliser votre certificat importé pour signer un fichier.

Microsoft SignTool avec AWS CloudHSM étape 3 : signer un fichier

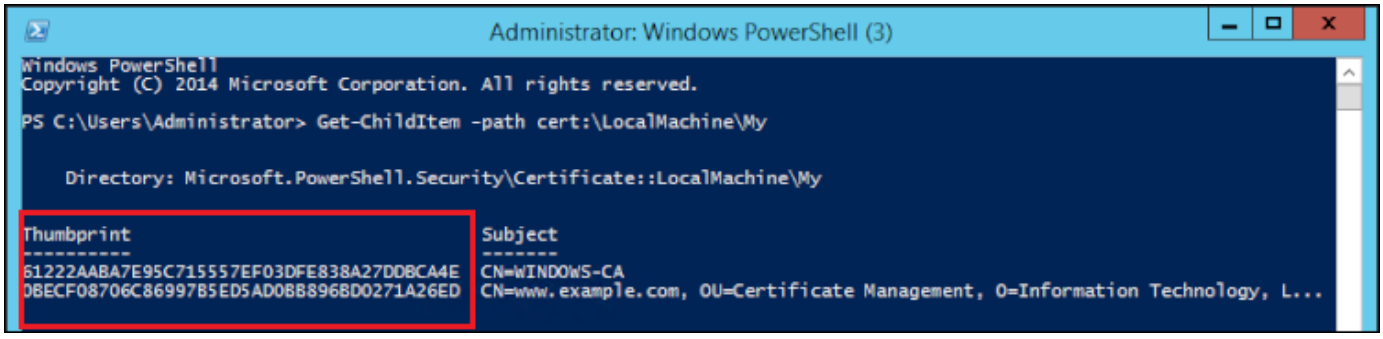
Vous êtes maintenant prêt à utiliser SignTool votre certificat importé pour signer votre fichier d'exemple. Pour ce faire, vous devez connaître le hachage SHA-1 ou l'empreinte du certificat. L'empreinte numérique est utilisée pour garantir que SignTool seuls les certificats vérifiés par. AWS CloudHSM Dans cet exemple, nous utilisons PowerShell pour obtenir le hachage du certificat. Vous pouvez également utiliser l'interface graphique utilisateur de CA ou le fichier exécutable `certutil` du kit SDK Windows.

Pour obtenir l'empreinte d'un certificat et l'utiliser pour signer un fichier

1. Ouvrez PowerShell en tant qu'administrateur et exécutez la commande suivante :

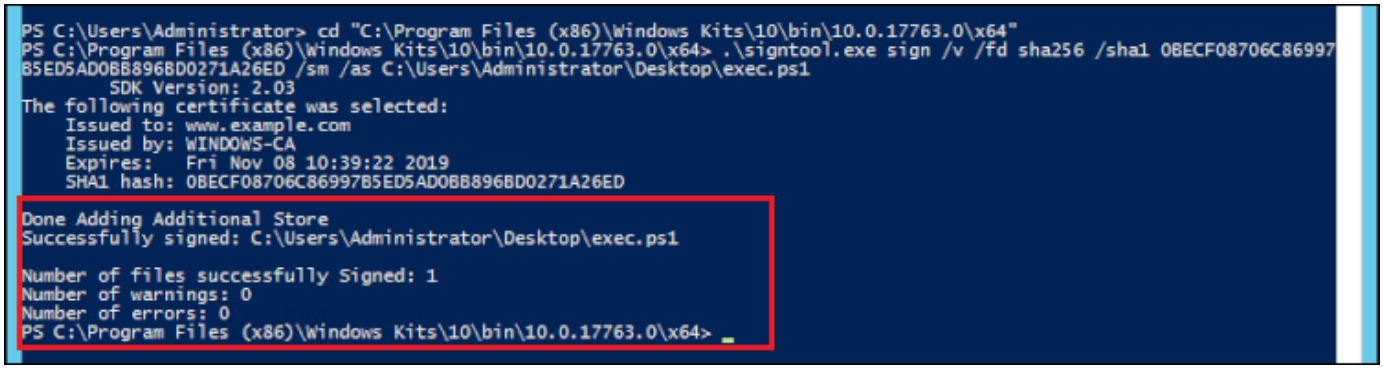
   ```
   Get-ChildItem -path cert:\LocalMachine\My
   ```

   Copiez la valeur `Thumbprint` qui est renvoyée.

2. Accédez au répertoire PowerShell qui contient`SignTool.exe`. L'emplacement par défaut est `C:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64`.
3. Enfin, signez votre fichier en exécutant la commande suivante. Si la commande aboutit, PowerShell renvoie un message de réussite.

   ```
   signtool.exe sign /v /fd sha256 /sha1 <thumbprint> /sm C:\Users\Administrator
   \Desktop\<test>.ps1
   ```



4. (Facultatif) Pour vérifier la signature sur le fichier, utilisez la commande suivante :

```
signtool.exe verify /v /pa C:\Users\Administrator\Desktop\<test>.ps1
```

Java Keytool et Jarsigner

AWS CloudHSM offre une intégration avec les utilitaires Java Keytool et Jarsigner via le SDK client 3 et le SDK client 5. Les étapes d'utilisation de ces outils varient en fonction de la version du SDK client dans laquelle vous avez actuellement téléchargé :

- Utilisation du SDK client 5 pour intégrer Java Keytool et Jarsigner
- Utilisation du SDK client 3 pour intégrer Java Keytool et Jarsigner

Utilisation du SDK client 5 pour intégrer Java Keytool et Jarsigner

AWS CloudHSM Le magasin de clés est un magasin de clés JCE spécial qui utilise les certificats associés aux clés de votre HSM via des outils tiers tels que et. `keytool` `jarsigner` AWS CloudHSM ne stocke pas les certificats sur le HSM, car les certificats sont des données publiques non confidentielles. Le magasin de AWS CloudHSM clés stocke les certificats dans un fichier local et les mappe aux clés correspondantes sur votre HSM.

Lorsque vous utilisez le magasin de AWS CloudHSM clés pour générer de nouvelles clés, aucune entrée n'est générée dans le fichier de magasin de clés local ; les clés sont créées sur le HSM. De même, lorsque vous utilisez le magasin de clés AWS CloudHSM pour rechercher des clés, la recherche est transmise au HSM. Lorsque vous stockez des certificats dans le magasin de AWS CloudHSM clés, le fournisseur vérifie qu'une paire de clés portant l'alias correspondant existe sur le HSM, puis associe le certificat fourni à la paire de clés correspondante.

Rubriques

Prérequis

Pour utiliser le magasin de AWS CloudHSM clés, vous devez d'abord initialiser et configurer le SDK AWS CloudHSM JCE.

Étape 1 : Installation du JCE

Pour installer le JCE, y compris les prérequis du AWS CloudHSM client, suivez les étapes d'installation de la bibliothèque Java.

Étape 2 : Ajouter des informations d'identification de connexion HSM aux variables d'environnement

Configurez les variables d'environnement pour qu'elles contiennent vos informations d'identification de connexion HSM.

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**
>
> Le AWS CloudHSM JCE propose différentes options de connexion. Pour utiliser le magasin de AWS CloudHSM clés avec des applications tierces, vous devez utiliser une connexion

implicite avec des variables d'environnement. Si vous souhaitez utiliser une connexion explicite via le code de l'application, vous devez créer votre propre application à l'aide du magasin de AWS CloudHSM clés. Pour plus d'informations, consultez l'article sur l'utilisation de AWS CloudHSM Key Store.

Étape 3 : Enregistrement du fournisseur JCE

Pour enregistrer le fournisseur JCE dans la CloudProvider configuration Java, procédez comme suit :

1. Ouvrez le fichier de configuration `java.security` dans votre installation Java, pour modification.
2. Dans le fichier de configuration `java.security`, ajoutez `com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider` comme dernier fournisseur. Par exemple, s'il y a neuf fournisseurs dans le fichier `java.security`, ajoutez le fournisseur suivant comme dernier fournisseur dans la section :

   ```
   security.provider.10=com.amazonaws.cloudhsm.jce.provider.CloudHsmProvider
   ```

> **Note**
>
> L'ajout du AWS CloudHSM fournisseur à une priorité plus élevée peut avoir un impact négatif sur les performances de votre système, car le AWS CloudHSM fournisseur sera priorisé pour les opérations qui peuvent être déchargées en toute sécurité vers le logiciel. Il est recommandé de toujours spécifier le fournisseur que vous souhaitez utiliser pour une opération, qu'il s'agisse d'un fournisseur logiciel AWS CloudHSM ou d'un fournisseur basé sur un logiciel.

> **Note**
>
> La spécification des options de ligne de commande `-providerName`, `-providerclass` et `-providerpath` lors de la génération de clés à l'aide de keytool avec le magasin de clés AWS CloudHSM peut provoquer des erreurs.

Utilisation du magasin de AWS CloudHSM clés avec keytool

keytool est un utilitaire de ligne de commande populaire pour les tâches courantes de clés et de certificats. La documentation AWS CloudHSM n'inclut pas de didacticiel complet sur keytool. Cet article explique les paramètres spécifiques que vous devez utiliser avec les différentes fonctions de keytool lorsque vous les utilisez AWS CloudHSM comme racine de confiance via le magasin de AWS CloudHSM clés.

Lorsque vous utilisez keytool avec le magasin de AWS CloudHSM clés, spécifiez les arguments suivants pour chaque commande keytool :

Linux

```
-storetype CLOUDHSM -J-classpath< '-J/opt/cloudhsm/java/*'>
```

Windows

```
-storetype CLOUDHSM -J-classpath<'-J"C:\Program Files\Amazon\CloudHSM\java\*"'>
```

Si vous souhaitez créer un nouveau fichier de stockage de clés à l'aide du magasin de AWS CloudHSM clés, consultezEn utilisant AWS CloudHSM KeyStore. Pour utiliser un magasin de clés existant, spécifiez son nom (y compris le chemin) à l'aide de l'argument —keystore à keytool. Si vous spécifiez un fichier de stockage de clés inexistant dans une commande keytool, le magasin de AWS CloudHSM clés crée un nouveau fichier de stockage de clés.

Créer de nouvelles clés avec keytool

Vous pouvez utiliser keytool pour générer un type de clé RSA, AES et DESede pris en charge par le SDK JCE d' AWS CloudHSM.

> **Important**
>
> Une clé générée par keytool est générée dans le logiciel, puis importée AWS CloudHSM sous forme de clé persistante extractible.

Nous vous recommandons fortement de générer des clés non exportables en dehors de keytool, puis d'importer les certificats correspondants dans le magasin de clés. Si vous utilisez des clés

RSA ou EC extractibles via keytool et Jarsigner, les fournisseurs exportent les clés depuis le, AWS CloudHSM puis les utilisent localement pour les opérations de signature.

Si plusieurs instances clientes sont connectées à votre AWS CloudHSM cluster, sachez que l'importation d'un certificat dans le magasin de clés d'une instance client ne rendra pas automatiquement les certificats disponibles sur d'autres instances clientes. Pour enregistrer la clé et les certificats associés sur chaque instance client, vous devez exécuter une application Java comme décrit dans the section called "Générer une CSR à l'aide de Keytool". Vous pouvez également apporter les modifications nécessaires sur un client et copier le fichier de stockage de clés résultant sur chaque autre instance cliente.

Exemple 1 : Pour générer une clé AES-256 symétrique et l'enregistrer dans un fichier de magasin de clés nommé « my_keystore.store », dans le répertoire de travail. Remplacez *<secret label>* par une étiquette unique.

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\*"'
```

Exemple 2 : Pour générer une paire de clés RSA 2048 et l'enregistrer dans un fichier de magasin de clés nommé « my_keystore.store » dans le répertoire de travail. Remplacez *<RSA key pair label>* par une étiquette unique.

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\*"'
```

Vous trouverez une liste des algorithmes de signature pris en charge dans la bibliothèque Java.

Supprimer une clé à l'aide de Keytool

Le magasin de AWS CloudHSM clés ne prend pas en charge la suppression de clés. Vous pouvez supprimer des clés à l'aide de la méthode de destruction de l'interface Destroyable.

```
((Destroyable) key).destroy();
```

Générer une CSR à l'aide de Keytool

Vous bénéficiez de la plus grande flexibilité dans la génération d'une demande de signature de certificat (CSR) si vous utilisez le OpenSSL Dynamic Engine. La commande suivante utilise keytool pour générer un CSR pour une paire de clés avec l'alias, `my-key-pair`.

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**
>
> Pour utiliser une paire de clés de keytool, cette paire de clés doit avoir une entrée dans le fichier de stockage de clés spécifié. Si vous souhaitez utiliser une paire de clés générée en dehors de keytool, vous devez importer les métadonnées de clé et de certificat dans le magasin de clés. Pour obtenir des instructions sur l'importation des données du keystore, consultez the section called "Utilisation de keytool pour importer des certificats intermédiaires et racines dans le magasin de AWS CloudHSM clés ".

Utilisation de keytool pour importer des certificats intermédiaires et racines dans le magasin de AWS CloudHSM clés

Pour importer un certificat d'autorité de certification, vous devez activer la vérification d'une chaîne de certificats complète sur un certificat nouvellement importé. Voici un exemple de commande.

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\*"'
```

Si vous connectez plusieurs instances clientes à votre AWS CloudHSM cluster, l'importation d'un certificat dans le magasin de clés d'une instance client ne le rendra pas automatiquement disponible sur les autres instances clientes. Vous devez importer le certificat sur chaque instance client.

Utilisation de keytool pour supprimer des certificats du magasin de AWS CloudHSM clés

La commande suivante montre un exemple de suppression d'un certificat d'un magasin de clés Java keytool.

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\*"'
```

Si vous connectez plusieurs instances clientes à votre AWS CloudHSM cluster, la suppression d'un certificat dans le magasin de clés d'une instance client ne supprimera pas automatiquement le certificat des autres instances clientes. Vous devez supprimer le certificat sur chaque instance cliente.

Importation d'un certificat fonctionnel dans le magasin de AWS CloudHSM clés à l'aide de keytool

Une fois qu'une demande de signature de certificat (CSR) est signée, vous pouvez l'importer dans le magasin de clés AWS CloudHSM et l'associer à la paire de clés appropriée. La commande suivante fournit un exemple.

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\*"'
```

L'alias doit être une paire de clés avec un certificat associé dans le magasin de clés. Si la clé est générée en dehors de keytool, ou si elle est générée sur une autre instance cliente, vous devez d'abord importer la clé et les métadonnées de certificat dans le magasin de clés.

La chaîne de certificats doit être vérifiable. Si vous ne parvenez pas à vérifier le certificat, vous devrez peut-être importer le certificat de signature (autorité de certification) dans le magasin de clés afin que la chaîne puisse être vérifiée.

Exportation d'un certificat à l'aide de Keytool

L'exemple suivant génère un certificat au format binaire X.509. Pour exporter un certificat lisible par l'homme, ajoutez `-rfc` à la commande `-exportcert`.

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\*"'
```

Utiliser le magasin de AWS CloudHSM clés avec Jarsigner

Jarsigner est un utilitaire de ligne de commande populaire permettant de signer des fichiers JAR à l'aide d'une clé stockée de manière sécurisée sur un HSM. Un didacticiel complet sur Jarsigner n'entre pas dans le cadre de la documentation AWS CloudHSM . Cette section explique les paramètres Jarsigner que vous devez utiliser pour signer et vérifier les signatures en utilisant le référentiel de AWS CloudHSM clés AWS CloudHSM comme source de confiance.

Configuration des clés et des certificats

Avant de pouvoir signer des fichiers JAR avec Jarsigner, assurez-vous d'avoir configuré ou effectué les étapes suivantes :

1. Suivez les instructions fournies dans les conditions préalables du magasin de clés AWS CloudHSM.
2. Configurez vos clés de signature ainsi que les certificats et la chaîne de certificats associés, qui doivent être stockés dans le magasin de AWS CloudHSM clés de l'instance actuelle du serveur ou du client. Créez les clés sur le, AWS CloudHSM puis importez les métadonnées associées dans votre magasin de AWS CloudHSM clés. Si vous souhaitez utiliser keytool pour configurer les clés et les certificats, reportez-vous à la section the section called "Créer de nouvelles clés avec keytool". Si vous utilisez plusieurs instances clientes pour signer vos fichiers JAR, créez la clé et importez la chaîne de certificats. Copiez ensuite le fichier de stockage de clés obtenu sur chaque instance cliente. Si vous générez fréquemment de nouvelles clés, il peut être plus facile d'importer individuellement des certificats dans chaque instance cliente.
3. Toute la chaîne de certificats doit être vérifiable. Pour que la chaîne de certificats soit vérifiable, vous devrez peut-être ajouter le certificat CA et les certificats intermédiaires au magasin de AWS CloudHSM clés. Veuillez consulter l'extrait de code dans the section called "Signer un fichier JAR en utilisant AWS CloudHSM et Jarsigner" pour apprendre à utiliser le code Java afin de vérifier la chaîne de certificats. Si vous préférez, vous pouvez utiliser keytool pour importer des certificats. Pour des instructions sur l'utilisation de keytool, consultez the section called "Utilisation de keytool pour importer des certificats intermédiaires et racines dans le magasin de AWS CloudHSM clés ".

Signer un fichier JAR en utilisant AWS CloudHSM et Jarsigner

Utilisez la commande suivante pour signer un fichier JAR :

Linux;

Pour 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>
```

Pour OpenJDK 11, OpenJDK 17 et 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

Pour 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>
```

Pour OpenJDK 11, OpenJDK 17 et 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>
```

Utilisez la commande suivante pour vérifier un JAR signé :

Linux

Pour 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>
```

Pour OpenJDK 11, OpenJDK 17 et 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

Pour 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>
```

Pour OpenJDK 11, OpenJDK 17 et 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>
```

Problèmes connus

1. Nous ne prenons pas en charge les clés EC avec Keytool et Jarsigner.

Utilisation du SDK client 3 pour intégrer Java Keytool et Jarsigner

AWS CloudHSM Le magasin de clés est un magasin de clés JCE spécial qui utilise les certificats associés aux clés de votre HSM via des outils tiers tels que et. `keytool` `jarsigner` AWS CloudHSM ne stocke pas les certificats sur le HSM, car les certificats sont des données publiques non confidentielles. Le magasin de AWS CloudHSM clés stocke les certificats dans un fichier local et les mappe aux clés correspondantes sur votre HSM.

Lorsque vous utilisez le magasin de AWS CloudHSM clés pour générer de nouvelles clés, aucune entrée n'est générée dans le fichier de magasin de clés local ; les clés sont créées sur le HSM. De même, lorsque vous utilisez le magasin de clés AWS CloudHSM pour rechercher des clés, la recherche est transmise au HSM. Lorsque vous stockez des certificats dans le magasin de AWS CloudHSM clés, le fournisseur vérifie qu'une paire de clés portant l'alias correspondant existe sur le HSM, puis associe le certificat fourni à la paire de clés correspondante.

Rubriques

Prérequis

Pour utiliser le magasin de AWS CloudHSM clés, vous devez d'abord initialiser et configurer le SDK AWS CloudHSM JCE.

Étape 1 : Installation du JCE

Pour installer le JCE, y compris les prérequis du AWS CloudHSM client, suivez les étapes d'installation de la bibliothèque Java.

Étape 2 : Ajouter des informations d'identification de connexion HSM aux variables d'environnement

Configurez les variables d'environnement pour qu'elles contiennent vos informations d'identification de connexion HSM.

```
export HSM_PARTITION=PARTITION_1
export HSM_USER=<HSM user name>
export HSM_PASSWORD=<HSM password>
```

> **Note**
>
> Le CloudHSM JCE offre diverses options de connexion. Pour utiliser le magasin de AWS CloudHSM clés avec des applications tierces, vous devez utiliser une connexion implicite avec des variables d'environnement. Si vous souhaitez utiliser une connexion explicite via le code de l'application, vous devez créer votre propre application à l'aide du magasin de AWS CloudHSM clés. Pour plus d'informations, consultez l'article sur l'utilisation de AWS CloudHSM Key Store.

Étape 3 : Enregistrement du fournisseur JCE

Pour enregistrer le fournisseur JCE, dans la CloudProvider configuration Java.

1. Ouvrez le fichier de configuration java.security dans votre installation Java, pour modification.
2. Dans le fichier de configuration java.security, ajoutez `com.cavium.provider.CaviumProvider` comme dernier fournisseur. Par exemple, s'il y a neuf fournisseurs dans le fichier java.security, ajoutez le fournisseur suivant comme dernier fournisseur dans la section. L'ajout du fournisseur de Cavium à une priorité plus élevée peut avoir un impact négatif sur les performances de votre système.

   ```
   security.provider.10=com.cavium.provider.CaviumProvider
   ```

   > **Note**
   >
   > Les utilisateurs de puissance peuvent être habitués à spécifier les options de lignes de commande `-providerName`, `-providerclass` et `-providerpath` lors de l'utilisation de keytool, au lieu de mettre à jour le fichier de configuration de sécurité. Si vous essayez

de spécifier des options de ligne de commande lors de la génération de AWS CloudHSM clés avec le magasin de clés, cela provoquera des erreurs.

Utilisation du magasin de AWS CloudHSM clés avec keytool

Keytool est un utilitaire de ligne de commande populaire pour les tâches courantes de clés et de certificats sur les systèmes Linux. La documentation AWS CloudHSM n'inclut pas de didacticiel complet sur keytool. Cet article explique les paramètres spécifiques que vous devez utiliser avec les différentes fonctions de keytool lorsque vous les utilisez AWS CloudHSM comme racine de confiance via le magasin de AWS CloudHSM clés.

Lorsque vous utilisez keytool avec le magasin de AWS CloudHSM clés, spécifiez les arguments suivants pour chaque commande keytool :

```
-storetype CLOUDHSM \
  -J-classpath '-J/opt/cloudhsm/java/*' \
  -J-Djava.library.path=/opt/cloudhsm/lib
```

Si vous souhaitez créer un nouveau fichier de stockage de clés à l'aide du magasin de AWS CloudHSM clés, consultezEn utilisant AWS CloudHSM KeyStore. Pour utiliser un magasin de clés existant, spécifiez son nom (y compris le chemin) à l'aide de l'argument —keystore à keytool. Si vous spécifiez un fichier de stockage de clés inexistant dans une commande keytool, le magasin de AWS CloudHSM clés crée un nouveau fichier de stockage de clés.

Créer de nouvelles clés avec keytool

Vous pouvez utiliser keytool pour générer n'importe quel type de clé pris en charge par AWS CloudHSM le SDK JCE. Consultez la liste complète des clés et longueurs dans l'article Clés prises en charge dans la bibliothèque Java.

> **Important**
>
> Une clé générée par keytool est générée dans le logiciel, puis importée AWS CloudHSM sous forme de clé persistante extractible.

Les instructions pour créer des clés non extractibles directement sur le HSM, puis les utiliser avec keytool ou Jarsigner, sont présentées dans l'exemple de code de la section Enregistrement de clés

préexistantes avec le magasin de clés. AWS CloudHSM Nous vous recommandons fortement de générer des clés non exportables en dehors de keytool, puis d'importer les certificats correspondants dans le magasin de clés. Si vous utilisez des clés RSA ou EC extractibles via keytool et jarsigner, les fournisseurs exportent les clés depuis le, AWS CloudHSM puis les utilisent localement pour les opérations de signature.

Si plusieurs instances clientes sont connectées à votre cluster CloudHSM, sachez que l'importation d'un certificat sur le magasin de clés d'une instance cliente ne rend pas automatiquement les certificats disponibles sur d'autres instances clientes. Pour enregistrer la clé et les certificats associés sur chaque instance client, vous devez exécuter une application Java comme décrit dans Générer un CSR à l'aide de Keytool. Vous pouvez également apporter les modifications nécessaires sur un client et copier le fichier de stockage de clés résultant sur chaque autre instance cliente.

Exemple 1 : Pour générer une clé AES-256 symétrique et l'enregistrer dans un fichier de magasin de clés nommé « my_keystore.store », dans le répertoire de travail. Remplacez *<secret label>* par une étiquette unique.

```
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/
```

Exemple 2 : Pour générer une paire de clés RSA 2048 et l'enregistrer dans un fichier de magasin de clés nommé « my_keystore.store » dans le répertoire de travail. Remplacez *<RSA key pair label>* par une étiquette unique.

```
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/
```

Exemple 3 : Pour générer une clé p256 ED et l'enregistrer dans un fichier de banque de clés nommé « my_keystore.store » dans le répertoire de travail. Remplacez *<ec key pair label>* par une étiquette unique.

```
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/
```

Vous trouverez une liste des algorithmes de signature pris en charge dans la bibliothèque Java.

Supprimer une clé à l'aide de Keytool

Le magasin de AWS CloudHSM clés ne prend pas en charge la suppression de clés. Pour supprimer la clé, vous devez utiliser la `deleteKey` fonction de l'outil AWS CloudHSM de ligne de commande,deleteKey.

Générer une CSR à l'aide de Keytool

Vous bénéficiez de la plus grande flexibilité dans la génération d'une demande de signature de certificat (CSR) si vous utilisez le OpenSSL Dynamic Engine. La commande suivante utilise keytool pour générer un CSR pour une paire de clés avec l'alias, `my-key-pair`.

```
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**
>
> Pour utiliser une paire de clés de keytool, cette paire de clés doit avoir une entrée dans le fichier de stockage de clés spécifié. Si vous souhaitez utiliser une paire de clés générée en dehors de keytool, vous devez importer les métadonnées de clé et de certificat dans le magasin de clés. Pour obtenir des instructions sur l'importation des données du magasin de clés, voir Importation de certificats intermédiaires et racines dans le magasin de AWS CloudHSM clés à l'aide de Keytool.

Utilisation de keytool pour importer des certificats intermédiaires et racines dans le magasin de AWS CloudHSM clés

Pour importer un certificat d'autorité de certification, vous devez activer la vérification d'une chaîne de certificats complète sur un certificat nouvellement importé. Voici un exemple de commande.

```
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/
```

Si vous connectez plusieurs instances clientes à votre AWS CloudHSM cluster, l'importation d'un certificat dans le magasin de clés d'une instance client ne le rendra pas automatiquement disponible sur les autres instances clientes. Vous devez importer le certificat sur chaque instance client.

Utilisation de keytool pour supprimer des certificats du magasin de AWS CloudHSM clés

La commande suivante montre un exemple de suppression d'un certificat d'un magasin de clés Java keytool.

```
keytool -delete -alias mydomain -keystore \
        -keystore my_keystore.store \
        -storetype CLOUDHSM \
        -J-classpath '-J/opt/cloudhsm/java/*' \
        -J-Djava.library.path=/opt/cloudhsm/lib/
```

Si vous connectez plusieurs instances clientes à votre AWS CloudHSM cluster, la suppression d'un certificat dans le magasin de clés d'une instance client ne supprimera pas automatiquement le certificat des autres instances clientes. Vous devez supprimer le certificat sur chaque instance cliente.

Importation d'un certificat fonctionnel dans le magasin de AWS CloudHSM clés à l'aide de keytool

Une fois qu'une demande de signature de certificat (CSR) est signée, vous pouvez l'importer dans le magasin de clés AWS CloudHSM et l'associer à la paire de clés appropriée. La commande suivante fournit un exemple.

```
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/
```

L'alias doit être une paire de clés avec un certificat associé dans le magasin de clés. Si la clé est générée en dehors de keytool, ou si elle est générée sur une autre instance cliente, vous devez d'abord importer la clé et les métadonnées de certificat dans le magasin de clés. Pour obtenir des instructions sur l'importation des métadonnées du certificat, consultez l'exemple de code dans Enregistrement de clés préexistantes auprès du magasin de AWS CloudHSM clés.

La chaîne de certificats doit être vérifiable. Si vous ne parvenez pas à vérifier le certificat, vous devrez peut-être importer le certificat de signature (autorité de certification) dans le magasin de clés afin que la chaîne puisse être vérifiée.

Exportation d'un certificat à l'aide de Keytool

L'exemple suivant génère un certificat au format binaire X.509. Pour exporter un certificat lisible par l'homme, ajoutez `-rfc` à la commande `-exportcert`.

```
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/
```

Utiliser le magasin de AWS CloudHSM clés avec JarSigner

Jarsigner est un utilitaire de ligne de commande populaire permettant de signer des fichiers JAR à l'aide d'une clé stockée de manière sécurisée sur un HSM. Un didacticiel complet sur Jarsigner n'entre pas dans le cadre de la documentation AWS CloudHSM . Cette section explique les paramètres Jarsigner que vous devez utiliser pour signer et vérifier les signatures en utilisant le référentiel de AWS CloudHSM clés AWS CloudHSM comme source de confiance.

Configuration des clés et des certificats

Avant de pouvoir signer des fichiers JAR avec Jarsigner, assurez-vous d'avoir configuré ou effectué les étapes suivantes :

1. Suivez les instructions fournies dans les conditions préalables du magasin de clés AWS CloudHSM.

2. Configurez vos clés de signature ainsi que les certificats et la chaîne de certificats associés, qui doivent être stockés dans le magasin de AWS CloudHSM clés de l'instance actuelle du serveur ou du client. Créez les clés sur le, AWS CloudHSM puis importez les métadonnées associées dans votre magasin de AWS CloudHSM clés. Utilisez l'exemple de code de la section Enregistrement de clés préexistantes auprès du magasin de AWS CloudHSM clés pour importer des métadonnées dans le magasin de clés. Si vous souhaitez utiliser keytool pour configurer les clés et les certificats, reportez-vous à la section Créer de nouvelles clés avec keytool. Si vous utilisez plusieurs instances clientes pour signer vos fichiers JAR, créez la clé et importez la chaîne de certificats. Copiez ensuite le fichier de stockage de clés obtenu sur chaque instance cliente. Si vous générez fréquemment de nouvelles clés, il peut être plus facile d'importer individuellement des certificats dans chaque instance cliente.
3. Toute la chaîne de certificats doit être vérifiable. Pour que la chaîne de certificats soit vérifiable, vous devrez peut-être ajouter le certificat CA et les certificats intermédiaires au magasin de AWS CloudHSM clés. Consultez l'extrait de code dans Signer un fichier JAR à l'aide de Jarsigner AWS CloudHSM et Jarsigner pour obtenir des instructions sur l'utilisation du code Java pour vérifier la chaîne de certificats. Si vous préférez, vous pouvez utiliser keytool pour importer des certificats. Pour obtenir des instructions sur l'utilisation de keytool, consultez la section Utilisation de Keytool pour importer des certificats intermédiaires et racines dans AWS CloudHSM Key Store.

Signer un fichier JAR en utilisant AWS CloudHSM et Jarsigner

Utilisez la commande suivante pour signer un fichier JAR :

```
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>
```

Utilisez la commande suivante pour vérifier un JAR signé :

```
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>
```

Problèmes connus

La liste suivante contient les problèmes connus à ce jour.

- Lorsque vous générez des clés à l'aide de keytool, le premier fournisseur dans la configuration du fournisseur ne peut pas l'être CaviumProvider.
- Lors de la génération de clés à l'aide de keytool, le premier fournisseur (pris en charge) dans le fichier de configuration de sécurité est utilisé pour générer la clé. Il s'agit généralement d'un fournisseur de logiciels. La clé générée reçoit ensuite un alias et est importée dans le AWS CloudHSM HSM en tant que clé persistante (jeton) pendant le processus d'ajout de clé.
- Lorsque vous utilisez keytool avec le magasin de AWS CloudHSM clés, ne spécifiez pas `-providerName-providerclass`, ni `-providerpath` options sur la ligne de commande. Spécifiez ces options dans le fichier du fournisseur de sécurité, comme décrit dans les conditions préalables du magasin de clés.
- Lors de l'utilisation de clés EC non extractibles via keytool et Jarsigner, le fournisseur SunEC doit être supprimé/désactivé de la liste des fournisseurs dans le fichier java.security. Si vous utilisez des clés EC extractibles via keytool et Jarsigner, les fournisseurs exportent les bits clés du AWS CloudHSM HSM et utilisent la clé localement pour les opérations de signature. Nous ne vous recommandons pas d'utiliser des clés exportables avec keytool ou Jarsigner.

Enregistrement de clés préexistantes avec le magasin de AWS CloudHSM clés

Pour une sécurité et une flexibilité maximales dans les attributs et l'étiquetage, nous vous recommandons de générer vos clés de signature à l'aide de key_mgmt_util. Vous pouvez également utiliser une application Java pour générer la clé dans AWS CloudHSM.

La section suivante fournit un exemple de code qui montre comment générer une nouvelle paire de clés sur le HSM et l'enregistrer à l'aide de clés existantes importées dans le magasin de AWS CloudHSM clés. Les clés importées peuvent être utilisées avec des outils tiers tels que keytool et Jarsigner.

Pour utiliser une clé préexistante, modifiez l'exemple de code pour rechercher une clé par étiquette au lieu de générer une nouvelle clé. Un exemple de code permettant de rechercher une clé par étiquette est disponible dans l'exemple KeyUtilitiesRunner .java sur GitHub.

> **Important**
>
> L'enregistrement d'une clé enregistrée dans un magasin de clés local n'exporte pas la clé. AWS CloudHSM Lorsque la clé est enregistrée, le magasin de clés enregistre l'alias (ou l'étiquette) de la clé et met en corrélation localement les objets de certificat de magasin avec une paire de clés sur le AWS CloudHSM. Tant que la paire de clés est créée comme non exportable, les bits de clé ne quitteront pas le 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());
        }
    }

}
```

Autres intégrations de fournisseur tiers

Le support AWS CloudHSM de plusieurs fournisseurs tiers est une source de confiance. Cela signifie que vous pouvez utiliser une solution logicielle de votre choix lorsque vous créez et stockez la clé sous-jacente dans votre cluster CloudHSM. Par conséquent, votre charge de travail AWS peut

compter sur les avantages de latence, de disponibilité, de fiabilité et d'élasticité de CloudHSM. La liste suivante inclut les fournisseurs tiers qui prennent en charge CloudHSM.

> **Note**
>
> AWS n'approuve ni ne garantit aucun fournisseur tiers.

- Hashicorp Vault est un outil de gestion des secrets conçu pour favoriser la collaboration et la gouvernance entre les organisations. Elle soutient AWS Key Management Service et AWS CloudHSM sert de base à la confiance pour une protection supplémentaire.
- Thycotic Secrets Server aide les clients à gérer les informations d'identification sensibles entre des comptes privilégiés. Il soutient en AWS CloudHSM tant que source de confiance.
- L'adaptateur KMIP du P6R vous permet d'utiliser vos AWS CloudHSM instances via une interface KMIP standard.
- PrimeKey EJBCA est une solution open source populaire pour le PKI. Il vous permet de créer et de stocker des paires de clés en toute sécurité avec AWS CloudHSM.
- Box KeySafe fournit la gestion des clés de chiffrement pour le contenu cloud à de nombreuses entreprises soumises à des exigences strictes en matière de sécurité, de confidentialité et de conformité réglementaire. Les clients peuvent également sécuriser leurs KeySafe clés directement AWS Key Management Service ou indirectement AWS CloudHSM via AWS KMS Custom Key Store.
- Insyde Software prend en charge la signature du microprogramme en AWS CloudHSM tant que source de confiance.
- Le support F5 BIG-IP LTM est une source de AWS CloudHSM confiance.
- Cloudera Navigator Key HSM permet d'utiliser votre cluster CloudHSM afin de créer et stocker des clé pour Cloudera Navigator Key Trustee Server.
- Venafi Trust Protection Platform fournit une gestion complète de l'identité des machines pour le TLS, le SSH et la signature de code grâce à la génération et à la protection de clés AWS CloudHSM.

Surveillance AWS CloudHSM

Outre les fonctionnalités de journalisation intégrées au SDK client, vous pouvez également utiliser AWS CloudTrail Amazon CloudWatch Logs et Amazon CloudWatch à des fins de surveillance AWS CloudHSM.

Journaux du SDK client

Utilisez la journalisation du SDK client pour surveiller les informations de diagnostic et de dépannage provenant des applications que vous créez.

CloudTrail

CloudTrail Utilisez-le pour surveiller tous les appels d'API de votre AWS compte, y compris les appels que vous effectuez pour créer et supprimer des clusters, des modules de sécurité matériels (HSM) et des balises de ressources.

CloudWatch Journaux

Utilisez CloudWatch les journaux pour surveiller les journaux de vos instances HSM, qui incluent les événements relatifs à la création et à la suppression d'utilisateurs HSM, à la modification des mots de passe des utilisateurs, à la création et à la suppression de clés, etc.

CloudWatch

CloudWatch Utilisez-le pour surveiller l'état de santé de votre cluster en temps réel.

Rubriques

Utilisation des journaux du SDK client

Vous pouvez récupérer les journaux générés par le SDK client. AWS CloudHSM propose une implémentation de la journalisation avec le SDK client 3 et le SDK client 5.

Rubriques

Journalisation du SDK client 5

Les journaux du SDK client 5 contiennent des informations pour chaque composant dans un fichier portant le nom du composant. Vous pouvez utiliser l'outil de configuration du SDK client 5 pour configurer la journalisation pour chaque composant.

Si vous ne spécifiez pas d'emplacement pour le fichier, le système écrit les journaux à l'emplacement par défaut :

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
```

Pour plus d'informations sur la configuration de la journalisation pour le SDK client 5, veuillez consulter l'outil de configuration du SDK client 5

Journalisation du SDK client 3

Les journaux du SDK client 3 contiennent des informations détaillées provenant du démon AWS CloudHSM client. L'emplacement des journaux dépend du système d'exploitation de l'instance client Amazon EC2 dans laquelle vous exécutez le démon client.

Amazon Linux

Dans Amazon Linux, les journaux des AWS CloudHSM clients sont écrits dans le fichier nommé `/opt/cloudhsm/run/cloudhsm_client.log`. Vous pouvez utiliser logrotate ou un outil similaire pour faire pivoter et gérer ces journaux.

Amazon Linux 2

Dans Amazon Linux 2, les journaux des AWS CloudHSM clients sont collectés et stockés dans le journal. Vous pouvez utiliser journalctl pour afficher et gérer ces journaux. Par exemple, utilisez la commande suivante pour afficher les journaux du AWS CloudHSM client.

```
journalctl -f -u cloudhsm-client
```

CentOS 7

Dans CentOS 7, les journaux du AWS CloudHSM client sont collectés et stockés dans le journal. Vous pouvez utiliser journalctl pour afficher et gérer ces journaux. Par exemple, utilisez la commande suivante pour afficher les journaux du AWS CloudHSM client.

```
journalctl -f -u cloudhsm-client
```

CentOS 8

Dans CentOS 8, les journaux du AWS CloudHSM client sont collectés et stockés dans le journal. Vous pouvez utiliser journalctl pour afficher et gérer ces journaux. Par exemple, utilisez la commande suivante pour afficher les journaux du AWS CloudHSM client.

```
journalctl -f -u cloudhsm-client
```

RHEL 7

Dans Red Hat Enterprise Linux 7, les journaux des AWS CloudHSM clients sont collectés et stockés dans le journal. Vous pouvez utiliser journalctl pour afficher et gérer ces journaux. Par exemple, utilisez la commande suivante pour afficher les journaux du AWS CloudHSM client.

```
journalctl -f -u cloudhsm-client
```

RHEL 8

Dans Red Hat Enterprise Linux 8, les journaux des AWS CloudHSM clients sont collectés et stockés dans le journal. Vous pouvez utiliser journalctl pour afficher et gérer ces journaux. Par exemple, utilisez la commande suivante pour afficher les journaux du AWS CloudHSM client.

```
journalctl -f -u cloudhsm-client
```

Ubuntu 16.04

Dans Ubuntu 16.04, les journaux du AWS CloudHSM client sont collectés et stockés dans le journal. Vous pouvez utiliser journalctl pour afficher et gérer ces journaux. Par exemple, utilisez la commande suivante pour afficher les journaux du AWS CloudHSM client.

```
journalctl -f -u cloudhsm-client
```

Ubuntu 18.04

Dans Ubuntu 18.04, les journaux du AWS CloudHSM client sont collectés et stockés dans le journal. Vous pouvez utiliser journalctl pour afficher et gérer ces journaux. Par exemple, utilisez la commande suivante pour afficher les journaux du AWS CloudHSM client.

```
journalctl -f -u cloudhsm-client
```

Windows

- Pour le Client Windows version 1.1.2 et ultérieure :

  AWS CloudHSM les journaux des clients sont écrits dans un `cloudhsm.log` fichier du dossier des fichiers AWS CloudHSM du programme (`C:\Program Files\Amazon\CloudHSM\`).

Chaque nom de fichier journal est suffixé par un horodatage indiquant la date de démarrage du AWS CloudHSM client.

- Pour le client Windows version 1.1.1 et antérieure :

  Les journaux client ne sont pas écrits dans un fichier. Les journaux sont affichés à l'invite de commande ou dans la PowerShell fenêtre dans laquelle vous avez démarré le AWS CloudHSM client.

Travailler avec AWS CloudTrail et AWS CloudHSM

AWS CloudHSM est intégré à AWS CloudTrail un service qui fournit un enregistrement des actions entreprises par un utilisateur, un rôle ou un AWS service dans AWS CloudHSM. CloudTrail capture tous les appels d'API AWS CloudHSM sous forme d'événements. Les appels capturés incluent des appels provenant de la AWS CloudHSM console et des appels de code vers les opérations de l' AWS CloudHSM API. Si vous créez un suivi, vous pouvez activer la diffusion continue d' CloudTrail événements vers un compartiment Amazon S3, y compris les événements pour AWS CloudHSM. Si vous ne configurez pas de suivi, vous pouvez toujours consulter les événements les plus récents dans la CloudTrail console dans Historique des événements. À l'aide des informations collectées par CloudTrail, vous pouvez déterminer la demande qui a été faite AWS CloudHSM, l'adresse IP à partir de laquelle la demande a été faite, qui a fait la demande, quand elle a été faite et des détails supplémentaires.

Pour en savoir plus CloudTrail, consultez le guide de AWS CloudTrail l'utilisateur. Pour obtenir la liste complète des opérations d' AWS CloudHSM API, consultez la section Actions de la référence AWS CloudHSM d'API.

AWS CloudHSM informations dans CloudTrail

CloudTrail est activé sur votre AWS compte lorsque vous le créez. Lorsqu'une activité se produit dans AWS CloudHSM, cette activité est enregistrée dans un CloudTrail événement avec d'autres événements de AWS service dans l'historique des événements. Vous pouvez afficher, rechercher et télécharger les événements récents dans votre compte AWS . Pour plus d'informations, consultez la section Affichage des événements avec l'historique des CloudTrail événements.

Pour un enregistrement continu des événements de votre AWS compte, y compris des événements pour AWS CloudHSM, créez un parcours. Un suivi permet CloudTrail de fournir des fichiers journaux à un compartiment Amazon S3. Par défaut, lorsque vous créez un journal d'activité dans la console,

il s'applique à toutes les régions AWS. Le journal enregistre les événements de toutes les régions de la AWS partition et transmet les fichiers journaux au compartiment Amazon S3 que vous spécifiez. En outre, vous pouvez configurer d'autres AWS services pour analyser plus en détail les données d'événements collectées dans les CloudTrail journaux et agir en conséquence. Pour plus d'informations, consultez les ressources suivantes :

- Vue d'ensemble de la création d'un journal d'activité
- CloudTrail Services et intégrations pris en charge
- Configuration des notifications Amazon SNS pour CloudTrail
- Réception de fichiers CloudTrail journaux de plusieurs régions et réception de fichiers CloudTrail journaux de plusieurs comptes

CloudTrail enregistre toutes les AWS CloudHSM opérations, y compris les opérations en lecture seule, telles que `DescribeClusters` et`ListTags`, et les opérations de gestion, telles que `InitializeClusterCreatHsm`, et. `DeleteBackup`

Chaque événement ou entrée de journal contient des informations sur la personne ayant initié la demande. Les informations relatives à l'identité permettent de déterminer les éléments suivants :

- Si la demande a été faite avec les informations d'identification de l'utilisateur root ou AWS Identity and Access Management (IAM).
- Si la demande a été effectuée avec les informations d'identification de sécurité temporaires d'un rôle ou d'un utilisateur fédéré.
- Si la demande a été faite par un autre AWS service.

Pour plus d'informations, consultez l'élément CloudTrail UserIdentity.

Comprendre les entrées du fichier AWS CloudHSM journal

Un suivi est une configuration qui permet de transmettre des événements sous forme de fichiers journaux à un compartiment Amazon S3 que vous spécifiez. CloudTrail les fichiers journaux contiennent une ou plusieurs entrées de journal. Un événement représente une demande unique provenant de n'importe quelle source et inclut des informations sur l'action demandée, la date et l'heure de l'action, les paramètres de la demande, etc. CloudTrail les fichiers journaux ne constituent pas une trace ordonnée des appels d'API publics, ils n'apparaissent donc pas dans un ordre spécifique.

L'exemple suivant montre une entrée de CloudTrail journal illustrant l' AWS CloudHSM `CreateHsm`action.


```
{
    "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"
}
```


Utilisation d'Amazon CloudWatch Logs et AWS CloudHSM d'Audit Logs

Lorsqu'un HSM de votre compte reçoit une commande des outils de ligne de AWS CloudHSM commande ou des bibliothèques logicielles, il enregistre son exécution de la commande sous forme de journal d'audit. Les journaux d'audit d'un HSM incluent toutes les commandes de gestion initiées par le client, y compris celles qui créent et suppriment le HSM, se connectent ou se déconnectent du HSM, et gèrent les utilisateurs et les clés. Ces journaux fournissent un enregistrement fiable des actions qui ont changé l'état du HSM.

AWS CloudHSM collecte vos journaux d'audit HSM et les envoie à Amazon CloudWatch Logs en votre nom. Vous pouvez utiliser les fonctionnalités de CloudWatch Logs pour gérer vos journaux AWS CloudHSM d'audit, notamment en recherchant et en filtrant les journaux et en exportant les données des journaux vers Amazon S3. Vous pouvez utiliser vos journaux d'audit HSM dans la CloudWatch console Amazon ou utiliser les commandes CloudWatch Logs des CLI et CloudWatch des SDK Logs.

Rubriques

- Fonctionnement de la journalisation d'audit HSM
- Afficher les journaux d'audit HSM dans CloudWatch les journaux
- Interprétation des journaux d'audit HSM
- Référence des journaux d'audit HSM

Fonctionnement de la journalisation d'audit HSM

La journalisation des audits est automatiquement activée dans tous les AWS CloudHSM clusters. Il ne peut pas être désactivé ou désactivé, et aucun paramètre ne peut AWS CloudHSM empêcher

l'exportation des CloudWatch journaux vers Logs. Chaque événement de journal possède un horodatage et un numéro de séquence qui indiquent l'ordre des événements et vous aident à détecter les falsifications de journal.

Chaque instance HSM génère son propre journal. Les journaux d'audit des différents HSM, même dans le même cluster, sont susceptibles de différer. Par exemple, seul le premier HSM de chaque cluster enregistre l'initialisation du HSM. Les événements d'initialisation n'apparaissent pas dans les journaux des HSM qui sont clonés à partir de sauvegardes. De même, lorsque vous créez une clé, le HSM qui génère la clé enregistre un événement de génération de clé. Les autres HSM du cluster enregistrent un événement lorsqu'ils reçoivent la clé via la synchronisation.

AWS CloudHSM collecte les journaux et les publie dans les CloudWatch journaux de votre compte. Pour communiquer avec le service CloudWatch Logs en votre nom, AWS CloudHSM utilise un rôle lié au service. La politique IAM associée au rôle permet d' AWS CloudHSM effectuer uniquement les tâches requises pour envoyer les journaux d'audit à CloudWatch Logs.

> **Important**
>
> Si vous avez créé un cluster avant le 20 janvier 2018, et que vous n'avez pas encore créé un rôle lié à un service attaché, vous devez en créer un manuellement. Cela est nécessaire pour CloudWatch recevoir les journaux d'audit de votre AWS CloudHSM cluster. Pour plus d'informations sur la création d'un rôle lié à un service, consultez Présentation des rôles liés à un service, ainsi que Création d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Afficher les journaux d'audit HSM dans CloudWatch les journaux

Amazon CloudWatch Logs organise les journaux d'audit en groupes de journaux et, au sein d'un groupe de journaux, en flux de journaux. Chaque entrée du journal est un événement. AWS CloudHSM crée un groupe de journaux pour chaque cluster et un flux de journaux pour chaque HSM du cluster. Il n'est pas nécessaire de créer des composants CloudWatch Logs ni de modifier les paramètres.

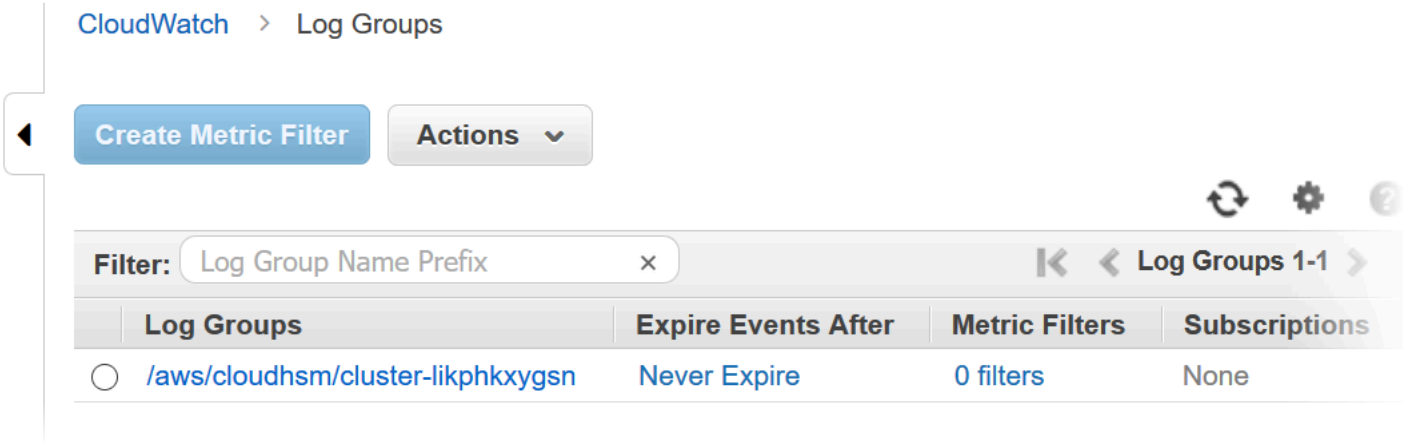
- Le nom du groupe de journaux est `/aws/cloudhsm/<cluster ID>`, par exemple `/aws/cloudhsm/cluster-likphkxygsn`. Lorsque vous utilisez le nom du groupe de journaux dans une CLI ou une PowerShell commande, veillez à le placer entre guillemets doubles.
- Le nom de flux de journaux est l'ID du HSM, par exemple `hsm-nwbbiqbj4jk`.

En général, il existe un flux de journaux pour chaque HSM. Toutefois, toute action qui modifie l'ID d'un HSM, par exemple lorsqu'un HSM est défaillant et remplacé, crée un nouveau flux de journaux.

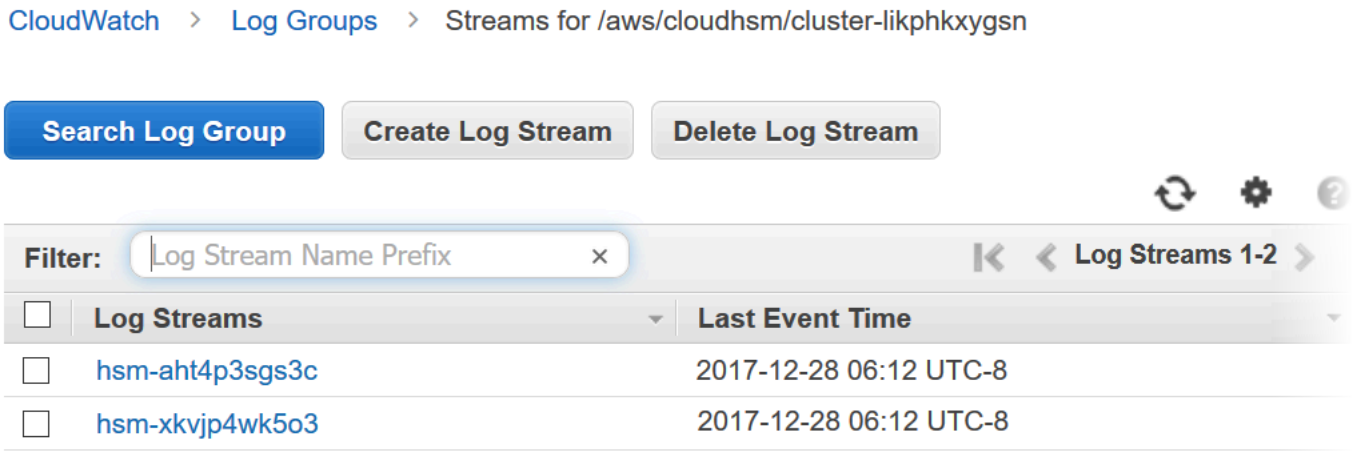
Pour plus d'informations sur CloudWatch les concepts relatifs aux journaux, consultez la section Concepts du guide de l'utilisateur Amazon CloudWatch Logs.

Vous pouvez consulter les journaux d'audit d'un HSM à partir de la page CloudWatch Logs du AWS Management Console, des commandes CloudWatch Logs de la CLI, des PowerShellapplets de commande CloudWatch Logs ou des CloudWatch SDK Logs. Pour obtenir des instructions, consultez la section Afficher les données des CloudWatch journaux dans le guide de l'utilisateur Amazon Logs.

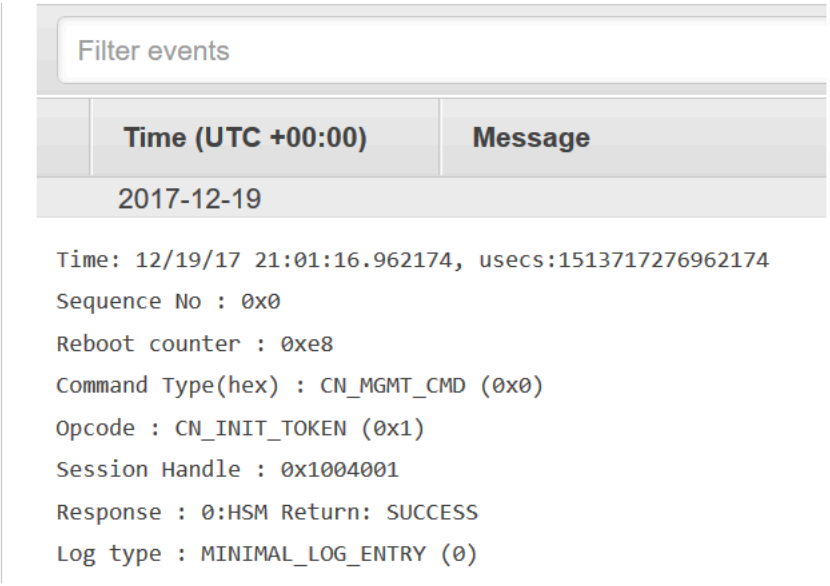
Par exemple, l'image suivante affiche le groupe de journaux pour le cluster `cluster-likphkxygsn` dans le AWS Management Console.



Lorsque vous choisissez le nom de groupe de journaux du cluster, vous pouvez consulter le flux de journaux pour chacun des HSM du cluster. L'image suivante affiche les flux de journaux pour les HSM du cluster `cluster-likphkxygsn`.

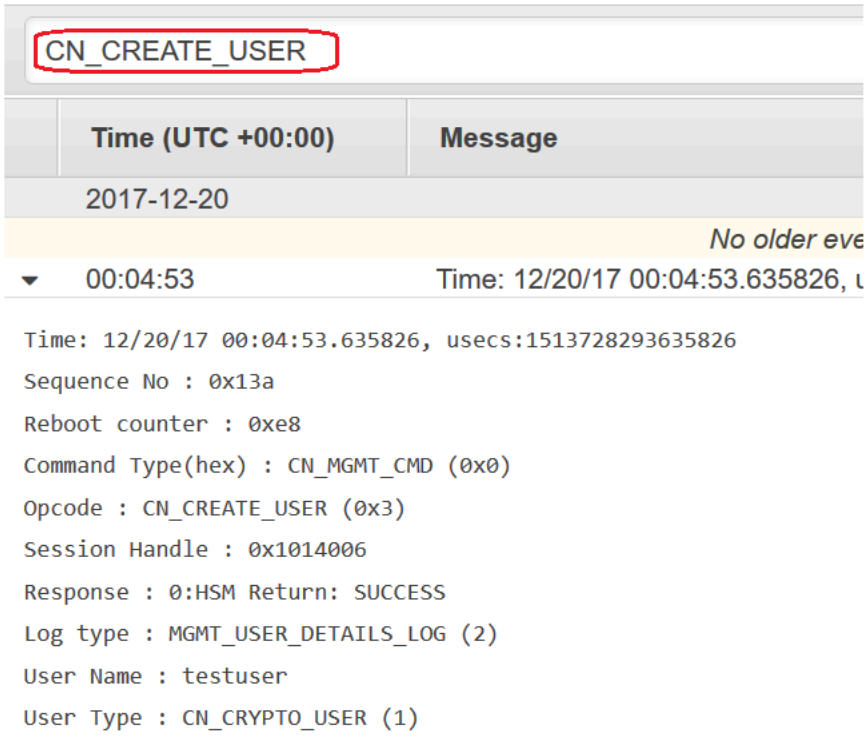


Lorsque vous choisissez un nom de flux de journaux de HSM, vous pouvez afficher les événements dans le journal d'audit. Par exemple, cet événement, qui a le numéro de séquence 0x0 et un `Opcode` de `CN_INIT_TOKEN` est généralement le premier événement pour le premier HSM de chaque cluster. Il enregistre l'initialisation du HSM dans le cluster.



Vous pouvez utiliser les nombreuses fonctionnalités de CloudWatch Logs pour gérer vos journaux d'audit. Par exemple, vous pouvez utiliser la fonction Filtrer les événements pour trouver un texte spécifique dans un événement, comme le `CN_CREATE_USER Opcode`.

Pour rechercher tous les événements qui n'incluent pas le texte spécifié, ajoutez un signe moins (-) avant le texte. Par exemple, pour trouver les événements qui n'incluent pas `CN_CREATE_USER`, saisissez -CN_CREATE_USER.



Interprétation des journaux d'audit HSM

Les événements des journaux d'audit de HSM ont des champs standard. Certains types d'événement ont des champs supplémentaires qui capturent des informations utiles sur l'événement. Par exemple, les événements de connexion utilisateur et de gestion des utilisateurs incluent le nom de l'utilisateur et le type d'utilisateur de l'utilisateur. Les commandes de gestion de clé incluent le handle de la clé.

Plusieurs des champs fournissent des informations particulièrement importantes. Le champ `Opcode` identifie la commande de gestion qui est en cours d'enregistrement. La `Sequence No` identifie un événement dans le flux de journaux et indique l'ordre dans lequel il a été enregistré.

Par exemple, l'exemple d'événement suivant est le deuxième événement (`Sequence No: 0x1`) dans le flux de journal pour un HSM. Il indique le HSM qui génère un mot de passe clé de chiffrement, ce qui fait partie de sa routine de démarrage.

```
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)
```

Les champs suivants sont communs à tous les AWS CloudHSM événements du journal d'audit.

Heure

Heure à la laquelle l'événement s'est produit dans le fuseau horaire UTC. L'heure est affichée dans un format compréhensible par les utilisateurs et au format d'heure Unix en microsecondes.

Reboot counter (Compteur de redémarrages)

Compteur ordinal 32 bits persistant, incrémenté quand le matériel HSM est redémarré.

Tous les événements d'un flux de journaux ont la même valeur de compteur de redémarrages. Toutefois, le compteur de redémarrages peut ne pas être unique pour un flux de journaux, car il peut varier entre les différentes instances HSM dans le même cluster.

Sequence No (N° de séquence)

Compteur ordinal 64 bits incrémenté pour chaque événement de journal. Le premier événement de chaque flux de journaux a le numéro de séquence 0x0. Il ne doit y avoir aucun écart dans les valeurs `Sequence No`. Le numéro de séquence est unique au sein d'un flux de journaux uniquement.

Command type (Type de commande)

Valeur hexadécimale qui représente la catégorie de la commande. Les commandes dans les flux de journaux AWS CloudHSM ont un type de commande `CN_MGMT_CMD` (0x0) ou `CN_CERT_AUTH_CMD` (0x9).

Opcode (Code d'opération)

Identifie la commande de gestion qui a été exécutée. Pour obtenir la liste des `Opcode` valeurs figurant dans les journaux AWS CloudHSM d'audit, consultezRéférence des journaux d'audit HSM.

Session handle (Handle de session)

Identifie la session dans laquelle la commande a été exécutée et l'événement a été consigné.

Réponse

Enregistre la réponse à la commande de gestion. Vous pouvez rechercher les valeurs `SUCCESS` et `ERROR` dans le champ `Response`.

Type de journal

Indique le type de AWS CloudHSM journal du journal qui a enregistré la commande.

- `MINIMAL_LOG_ENTRY (0)`
- `MGMT_KEY_DETAILS_LOG (1)`
- `MGMT_USER_DETAILS_LOG (2)`
- `GENERIC_LOG`

Exemples d'événements de journaux d'audit

Les événements d'un flux de journaux enregistrent l'historique du HSM de sa création à sa suppression. Vous pouvez utiliser le journal pour vérifier le cycle de vie de vos HSM et mieux comprendre leur fonctionnement. Lorsque vous interprétez les événements, notez le `Opcode`, qui indique la commande de gestion ou l'action et le `Sequence No`, qui indique l'ordre des événements.

Rubriques

Exemple : Initialiser le premier HSM dans un cluster

Le flux de journaux d'audit pour le premier HSM de chaque cluster diffère de manière significative des flux de journaux des autres HSM du cluster. Le journal d'audit du premier HSM de chaque cluster enregistre la création et l'initialisation de celui-ci. Les journaux des autres HSM du cluster, qui sont générées à partir de sauvegardes, commencent par un événement de connexion.

> **Important**
>
> Les entrées d'initialisation suivantes n'apparaîtront pas dans les CloudWatch journaux des clusters initialisés avant le lancement de la fonctionnalité de journalisation d'audit CloudHSM (30 août 2018). Pour plus d'informations, consultez Historique du document.

Les exemples d'événements suivants apparaissent dans le flux de journaux pour le premier HSM d'un cluster. Le premier événement dans le journal, celui contenant `Sequence No 0x0`, représente la commande permettant d'initialiser le HSM (`CN_INIT_TOKEN`). La réponse indique que la commande a réussi (`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)
```

Le deuxième événement de cet exemple de flux de journaux (`Sequence No 0x1`) enregistre la commande pour créer la clé de chiffrement de mot de passe que le HSM utilise (`CN_GEN_PSWD_ENC_KEY`).

Il s'agit d'une séquence de démarrage classique pour le premier HSM de chaque cluster. Comme les HSM suivants du même cluster sont des clones du premier HSM, ils utilisent la même clé de chiffrement de mot de passe.

```
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)
```

Le troisième événement de cet exemple de flux de journaux (`Sequence No 0x2`) est la création de l'utilisateur de l'appareil (AU), qui est le service AWS CloudHSM . Les événements qui impliquent

des utilisateurs de HSM incluent des champs supplémentaires pour le nom d'utilisateur et le type d'utilisateur.

```
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)
```

Le quatrième événement de cet exemple de flux de journaux (`Sequence No 0x3`) enregistre l'événement `CN_INIT_DONE`, qui termine l'initialisation du HSM.

```
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)
```

Vous pouvez suivre les autres événements dans la séquence de démarrage. Ces événements peuvent inclure plusieurs événements de connexion et de déconnexion, et la génération de la clé de chiffrement de clé (KEK). L'événement suivant enregistre la commande qui modifie le mot de passe du responsable du pré-chiffrement (PRECO). Cette commande active le cluster.

```
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)
```

Événements de connexion et de déconnexion

Lors de l'interprétation de votre journal d'audit, notez les événements qui enregistrent la connexion au HSM et la déconnexion de celui-ci. Ces événements pour vous aident à déterminer quel utilisateur est responsable des commandes de gestion qui apparaissent dans la séquence entre les commandes de connexion et de déconnexion.

Par exemple, cette entrée de journal enregistre une connexion par un responsable de chiffrement nommé `admin`. Le numéro de séquence, `0x0`, indique qu'il s'agit du premier événement de ce flux de journaux.

Lorsqu'un utilisateur se connecte à un HSM, les autres HSM du cluster enregistrent également un événement de connexion pour l'utilisateur. Vous pouvez trouver les événements de connexion correspondants dans les flux de journaux des autres HSM du cluster peu de temps après l'événement de connexion initiale.

```
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)
```

L'exemple d'événement suivant enregistre la déconnexion du responsable de chiffrement `admin`. Le numéro de séquence, `0x2`, indique qu'il s'agit du troisième événement du flux de journaux.

Si l'utilisateur connecté ferme la session sans se déconnecter, le flux de journaux inclut un `CN_APP_FINALIZE` ou un événement de fermeture de session (`CN_SESSION_CLOSE`), au lieu d'un événement `CN_LOGOUT`. Contrairement à l'événement de connexion, cet événement de déconnexion est enregistré généralement uniquement par le HSM qui exécute la commande.

```
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)
```

Si une tentative de connexion échoue parce que le nom d'utilisateur n'est pas valide, le HSM enregistre un événement `CN_LOGIN` avec le nom et le type d'utilisateur fournis dans la commande de connexion. La réponse affiche un message d'erreur 157, qui explique que le nom d'utilisateur n'existe pas.

```
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)
```

Si une tentative de connexion échoue parce que le mot de passe n'est pas valide, le HSM enregistre un événement `CN_LOGIN` avec le nom et le type d'utilisateur fournis dans la commande de connexion. La réponse affiche le message d'erreur avec le code d'erreur `RET_USER_LOGIN_FAILURE`.

```
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)
```

Exemple : Créer et supprimer des utilisateurs

Cet exemple montre les événements de journal qui sont enregistrés lorsqu'un responsable de chiffrement (CO) crée et supprime des utilisateurs.

Le premier événement enregistre un CO, `admin`, qui se connecte au HSM. Le numéro de séquence `0x0` indique qu'il s'agit du premier événement du flux de journaux. Le nom et le type de l'utilisateur qui s'est connecté sont inclus dans l'événement.

```
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)
```

L'événement suivant dans le flux de journaux (séquence `0x1`) enregistre que le responsable de chiffrement (CO) crée un utilisateur de chiffrement (CU). Le nom et le type du nouvel utilisateur sont inclus dans l'événement.

```
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)
```

Ensuite, le CO crée un autre responsable de chiffrement, `alice`. Le numéro de séquence indique que cette action suivait l'action précédente sans action intermédiaire.

```
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)
```

Plus tard, le CO nommé `admin` se connecte et supprime le responsable de chiffrement nommé `alice`. Le HSM enregistre un événement `CN_DELETE_USER`. Le nom et le type de l'utilisateur supprimé sont inclus dans l'événement.

```
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)
```

Exemple : Créer et supprimer une paire de clés

Cet exemple montre les événements qui sont enregistrés dans le journal d'audit d'un HSM lorsque vous créez et supprimez une paire de clés.

L'événement suivant enregistre l'utilisateur de chiffrement (CU) nommé `crypto_user` qui se connecte au 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)
```

Ensuite, le CU génère une paire de clés (`CN_GENERATE_KEY_PAIR`). La clé privée dispose du handle de clé `131079`. La clé publique dispose du handle de clé `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
```

Le CU supprime immédiatement la paire de clés. Un événement CN_DESTROY_OBJECT enregistre la suppression de la clé publique (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
```

Ensuite, un deuxième événement CN_DESTROY_OBJECT enregistre la suppression de la clé privée (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
```

Enfin, le CU se déconnecte.

```
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)
```

Exemple : Générer et Synchroniser une clé

Cet exemple montre l'effet de la création d'une clé dans un cluster avec plusieurs HSM. La clé est générée sur un HSM, extraite du HSM en tant qu'objet masqué, et insérée dans les autres HSM en tant qu'objet masqué.

> **Note**
>
> Les outils du client peuvent échouer à synchroniser la clé. Ou la commande peut inclure le paramètre min_srv qui synchronise la clé uniquement pour le nombre spécifié de HSM. Dans les deux cas, le AWS CloudHSM service synchronise la clé avec les autres HSM du cluster. Comme les HSM enregistrent uniquement les commandes de gestion côté client dans leurs journaux, la synchronisation côté serveur n'est pas consignée dans le journal du HSM.

Examinez d'abord le flux de journaux du HSM qui reçoit et exécute les commandes. Le flux de journaux est nommé pour le HSM ID `hsm-abcde123456`, mais l'ID du HSM n'apparaît pas dans les événements du journal.

Tout d'abord, l'utilisateur de chiffrement (CU) `testuser` se connecte au HSM `hsm-abcde123456`.

```
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)
```

La CU exécute une exSymKeycommande pour générer une clé symétrique. Le HSM `hsm-abcde123456` génère une clé symétrique avec un handle de clé `262152`. Le HSM enregistre un événement `CN_GENERATE_KEY` dans son journal.

```
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
```

L'événement suivant dans le flux de journaux pour `hsm-abcde123456` enregistre la première étape du processus de synchronisation de clé. La nouvelle clé (handle de clé `262152`) est extraite du HSM en tant qu'objet masqué.

```
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
```

Examinez maintenant le flux de journaux pour le HSM `hsm-zyxwv987654`, un autre HSM du même cluster. Ce flux de journaux inclut également un événement de connexion pour l'utilisateur de chiffrement (CU) `testuser`. La valeur temporelle montre que cela se produit peu de temps après que l'utilisateur se connecte au HSM `hsm-abcde123456`.

```
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)
```

Ce flux de journaux pour ce HSM n'a pas d'événement `CN_GENERATE_KEY`. Mais il y a un événement qui enregistre la synchronisation de la clé pour ce HSM. L'événement `CN_INSERT_MASKED_OBJECT_USER` enregistre la réception de la clé `262152` en tant qu'objet masqué. Maintenant, la clé `262152` existe sur les deux HSM du cluster.

```
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
```

Lorsque l'utilisateur de chiffrement (CU) se déconnecte, cet événement `CN_LOGOUT` s'affiche uniquement dans le flux de journaux du HSM qui reçoit les commandes.

Exemple : Exporter une clé

Cet exemple montre les événements de journal d'audit qui sont enregistrés lorsqu'un utilisateur de chiffrement (CU) exporte des clés à partir d'un cluster avec plusieurs HSM.

L'événement suivant enregistre la journalisation CU (`testuser`) dans key_mgmt_util.

```
Time: 01/24/18 19:42:22.695884, usecs:1516822942695884
Sequence No : 0x26
Reboot counter : 0x107
Command Type(hex) : CN_MGMT_CMD (0x0)
Opcode : CN_LOGIN (0xd)
Session Handle : 0x7004004
Response : 0:HSM Return: SUCCESS
Log type : MGMT_USER_DETAILS_LOG (2)
User Name : testuser
User Type : CN_CRYPTO_USER (1)
```

La CU exécute une exSymKeycommande pour exporter la clé7, une clé AES 256 bits. La commande utilise la clé 6, une clé AES 256 bits sur le HSM, comme clé d'encapsulage.

Le HSM qui reçoit la commande enregistre un événement `CN_WRAP_KEY` pour la clé 7, la clé qui est exportée.

```
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
```

Ensuite, le HSM enregistre un événement `CN_NIST_AES_WRAP` pour la clé d'encapsulage, la clé 6. La clé est encapsulée, puis immédiatement désencapsulée, mais le HSM n'enregistre qu'un seul événement.

```
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
```

La commande exSymKey écrit les clés exportées dans un fichier, mais ne modifie pas la clé dans le HSM. Par conséquent, il n'y a pas d'événements correspondants dans les journaux des autres HSM du cluster.

Exemple : Importer une clé

Cet exemple montre les événements de journal d'audit qui sont enregistrés lorsque vous importez des clés dans les HSM d'un cluster. Dans cet exemple, l'utilisateur cryptographique (CU) utilise la

imSymKeycommande pour importer une clé AES dans les HSM. La commande utilise la clé 6 en tant que clé d'encapsulage.

Le HSM qui reçoit les commandes enregistre un événement CN_NIST_AES_WRAP pour la clé 6, la clé d'encapsulage.

```
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
```

Ensuite, le HSM enregistre un événement CN_UNWRAP_KEY qui représente l'opération d'importation. Le handle de clé 11 est attribué à la clé importée.

```
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
```

Lorsqu'une nouvelle clé est générée ou importée, les outils client tentent automatiquement de synchroniser la nouvelle clé pour les autres HSM du cluster. Dans ce cas, le HSM enregistre un événement CN_EXTRACT_MASKED_OBJECT_USER lorsque la clé 11 est extraite du HSM en tant qu'objet masqué.

```
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
```

Les flux de journaux des autres HSM du cluster reflètent l'arrivée de la nouvelle clé importée.

Par exemple, cet événement a été enregistré dans le flux de journaux d'un autre HSM du même cluster. Cet événement `CN_INSERT_MASKED_OBJECT_USER` enregistre l'arrivée d'un objet masqué qui représente la clé 11.

```
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
```

Exemple : Partager une clé et annuler le partage d'une clé

Cet exemple illustre l'événement de journal d'audit qui est enregistré lorsqu'un utilisateur de chiffrement (CU) partage ou annule le partage d'une clé privée ECC avec d'autres utilisateurs de chiffrement. Le CU utilise la commande shareKey et fournit le handle de clé, l'ID utilisateur et la valeur 1 pour partager ou la valeur `0` pour annuler le partage de la clé.

Dans l'exemple suivant, le HSM qui reçoit la commande, enregistre un événement `CM_SHARE_OBJECT` qui représente l'opération de partage.

```
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)
```

Référence des journaux d'audit HSM

AWS CloudHSM enregistre les commandes de gestion HSM dans les événements du journal d'audit. Chaque événement comporte une valeur de code d'opération (`0pcode`) qui identifie l'action qui s'est produite et sa réponse. Vous pouvez utiliser les valeurs de `0pcode` pour rechercher, trier et filtrer les journaux.

Le tableau suivant définit les `0pcode` valeurs d'un journal AWS CloudHSM d'audit.

Code d'opération (Opcode)	Description
User Login : Ces événements incluent le nom d'utilisateur et le type d'utilisateur.	
`CN_LOGIN (0xd)`	Login utilisateur
`CN_LOGOUT (0xe)`	Déconnexion de l'utilisateur
`CN_APP_FINALIZE`	La connexion avec le HSM a été fermée. Toutes les clés de session ou jetons de quorum de cette connexion ont été supprimés.
`CN_CLOSE_SESSION`	La session avec le HSM a été fermée. Toutes les clés de session ou jetons de quorum de cette session ont été supprimés.
User Management : Ces événements incluent le nom d'utilisateur et le type d'utilisateur.	
`CN_CREATE_USER (0x3)`	Créez un utilisateur de chiffrement (CU).
`CN_CREATE_CO`	Créer un responsable de chiffrement (CO)
`CN_DELETE_USER`	Suppression d'un utilisateur
`CN_CHANGE_PSWD`	Modifier un mot de passe utilisateur
`CN_SET_M_VALUE`	Définir l'authentification par quorum (M ou N) pour une action utilisateur
`CN_APPROVE_TOKEN`	Approuver un jeton d'authentification du quorum pour une action utilisateur

Code d'opération (Opcode)	Description
`CN_DELETE_TOKEN`	Supprimer un ou plusieurs jetons de quorum
`CN_GET_TOKEN`	Demander un jeton de signature pour lancer une opération de quorum
Key Management : Ces événements incluent le handle de la clé.	
`CN_GENERATE_KEY`	Générer une clé symétrique
`CN_GENERATE_KEY_PAIR (0x19)`	Génération d'une paire de clés asymétrique
`CN_CREATE_OBJECT`	Importer une clé publique (sans encapsulage)
`CN_MODIFY_OBJECT`	Définissez un attribut clé
`CN_DESTROY_OBJECT (0x11)`	Suppression d'une clé de session
`CN_TOMBSTONE_OBJECT`	Suppression d'une clé de jeton
`CN_SHARE_OBJECT`	Partager ou annuler le partage d'une clé
`CN_WRAP_KEY`	Exporter une copie chiffrée d'une clé (wrapKey)
`CN_UNWRAP_KEY`	Importer une copie chiffrée d'une clé (unwrapKey)
`CN_DERIVE_KEY`	Dériver une clé symétrique à partir d'une clé existante
`CN_NIST_AES_WRAP`	Chiffrer ou déchiffrer une clé avec une clé AES
`CN_INSERT_MASKED_OBJECT_USER`	Insérez une clé chiffrée avec les attributs d'un autre HSM du cluster.
`CN_EXTRACT_MASKED_OBJECT_USER`	Enveloppe ou chiffre une clé avec les attributs du HSM à envoyer à un autre HSM du cluster.
Back up HSMs	
`CN_BACKUP_BEGIN`	Commencer le processus de sauvegarde

Code d'opération (Opcode)	Description
`CN_BACKUP_END`	Le processus de sauvegarde est terminé
`CN_RESTORE_BEGIN`	Commencer la restauration à partir d'une sauvegarde
`CN_RESTORE_END`	Vous avez terminé le processus de restauration à partir d'une sauvegarde
Certificate-Based Authentication	
`CN_CERT_AUTH_STORE_CERT`	Stocke le certificat du cluster
HSM Instance Commands	
`CN_INIT_TOKEN (0x1)`	Démarrez le processus d'initialisation du HSM
`CN_INIT_DONE`	Le processus d'initialisation du HSM est terminé
`CN_GEN_KEY_ENC_KEY`	Générer une clé de chiffrement de clé (KEK)
`CN_GEN_PSWD_ENC_KEY (0x1d)`	Générer une clé de chiffrement de mot de passe (PEK)
HSM crypto commands	
`CN_FIPS_RAND`	Génération d'un nombre aléatoire conforme à la norme FIPS

Obtenir CloudWatch des métriques pour AWS CloudHSM

CloudWatch Utilisez-le pour surveiller votre AWS CloudHSM cluster en temps réel. Les métriques peuvent être regroupées par région, par ID de cluster ou par ID de cluster et ID HSM.

L'espace de noms `AWS/CloudHSM` inclut les métriques suivantes :

Métrique	Description
HsmUnhealthy	L'instance HSM ne fonctionne pas correctement. AWS CloudHSM remplace automatiquement les instances défectueuses pour vous. Vous pouvez choisir d'étendre de façon proactive la taille des clusters pour réduire l'impact sur les performances pendant que nous remplaçons le HSM.
HsmTemperature	La température de jonction du processeur matériel. Le système s'arrête si la température atteint 110 degrés centigrades.
HsmKeysSessionOccupied	Le nombre de clés de session utilisées par l'instance HSM.
HsmKeysTokenOccupied	Le nombre de clés de jeton utilisées par l'instance HSM et le cluster.
HsmSslCtxsOccupied	Le nombre de canaux end-to-end chiffrés actuellement établis pour l'instance HSM. Le nombre maximum de canaux autorisé est de 2 048.
HsmSessionCount	Le nombre de connexions ouvertes avec l'instance HSM. Le nombre maximum de connexions autorisé est de 2 048. Par défaut, le daemon client est configuré pour ouvrir deux sessions avec chaque instance HSM sous un canal end-to-end crypté. AWS CloudHSM peut également avoir jusqu'à 2 connexions ouvertes avec le HSM pour surveiller l'état des HSM.
HsmUsersAvailable	Le nombre d'utilisateurs supplémentaires pouvant être créés. Cela correspond au nombre maximum d'utilisateurs (listés dans HsmUsersMax) moins les utilisateurs créés à ce jour.
HsmUsersMax	Le nombre maximum d'utilisateurs pouvant être créés sur l'instance HSM. Ce nombre est actuellement de 1 024.
InterfaceEth2OctetsInput	La somme cumulée du trafic entrant vers le HSM à ce jour.
InterfaceEth2OctetsOutput	La somme cumulée du trafic sortant vers le HSM à ce jour.

AWS CloudHSM Rendement

Pour les clusters de production, vous devriez disposer d'au moins deux instances HSM réparties sur deux zones de disponibilité dans une région. Nous vous recommandons de tester la charge de votre cluster pour déterminer le pic de charge auquel vous devez vous attendre, puis d'y ajouter un ou plusieurs HSM supplémentaires pour garantir une haute disponibilité. Pour les applications nécessitant une durabilité des clés nouvellement générés, nous recommandons au moins trois instances HSM réparties sur les différentes zones de disponibilité d'une région.

Données de performance

Les performances des AWS CloudHSM clusters varient en fonction de la charge de travail spécifique. Le contenu qui suit montre les performances approximatives des algorithmes cryptographiques courants exécutés sur l'instance EC2. Pour améliorer les performances, vous pouvez ajouter d'autres instances HSM à vos clusters. Les performances peuvent varier en fonction de la configuration, de la taille des données et de la charge d'application supplémentaire sur vos instances EC2. Nous vous encourageons à tester la charge de votre application afin de déterminer les besoins de mise à l'échelle.

Opération	Cluster à deux HSM [1]	Cluster à trois HSM [2]	Cluster à six HSM [3]
Signe RSA 2048 bits	2 000 ops/s	3 000 ops/s	5 000 ops/s
Signe EC P256	500 ops/s	750 ops/s	1 500 ops/s

- [1] Un cluster à deux HSM avec l'application multithread Java exécutée sur une instance EC2 c4.large avec un HSM situé dans la même zone AZ que l'instance EC2.
- [2] Un cluster à trois HSM avec l'application multithread Java exécutée sur une instance EC2 c4.large avec un HSM situé dans la même zone AZ que l'instance EC2.
- [3] Un cluster à six HSM avec l'application multithread Java exécutée sur une instance EC2 c4.large avec deux HSM situés dans la même zone AZ que l'instance EC2.

Limitation du HSM

Lorsque votre charge de travail dépasse la capacité du HSM de votre cluster, vous recevez des messages d'erreur indiquant que les HSM sont occupés ou limités. Pour plus de détails sur la marche à suivre dans ce cas de figure, consultez la section Limitation du HSM

Sécurité dans AWS CloudHSM

La sécurité du cloud AWS est la priorité absolue. En tant que AWS client, vous bénéficiez d'un centre de données et d'une architecture réseau conçus pour répondre aux exigences des entreprises les plus sensibles en matière de sécurité.

La sécurité est une responsabilité partagée entre vous AWS et vous. Le modèle de responsabilité partagée décrit cela comme la sécurité du cloud et la sécurité dans le cloud :

- Sécurité du cloud : AWS est chargée de protéger l'infrastructure qui exécute les AWS services dans le AWS cloud. AWS vous fournit également des services que vous pouvez utiliser en toute sécurité. Des auditeurs tiers testent et vérifient régulièrement l'efficacité de notre sécurité dans le cadre des programmes de AWS conformité Programmes de de conformité. Pour en savoir plus sur les programmes de conformité applicables à AWS CloudHSM, consultez la section Services AWS concernés par programme de conformité .
- Sécurité dans le cloud — Votre responsabilité est déterminée par le AWS service que vous utilisez. Vous êtes également responsable d'autres facteurs, y compris de la sensibilité de vos données, des exigences de votre entreprise, ainsi que de la législation et de la réglementation applicables.

Cette documentation vous aide à comprendre comment appliquer le modèle de responsabilité partagée lors de son utilisation AWS CloudHSM. Les rubriques suivantes expliquent comment procéder à la configuration AWS CloudHSM pour atteindre vos objectifs de sécurité et de conformité. Vous apprendrez également à utiliser d'autres services AWS qui vous aident à surveiller et à sécuriser vos AWS CloudHSM ressources.

Table des matières

Protection des données dans AWS CloudHSM

Le modèle de responsabilité AWS partagée de s'applique à la protection des données dans AWS CloudHSM. Comme décrit dans ce modèle, AWS est chargé de protéger l'infrastructure mondiale qui gère tous les AWS Cloud. La gestion du contrôle de votre contenu hébergé sur cette infrastructure relève de votre responsabilité. Vous êtes également responsable des tâches de configuration et de gestion de la sécurité des Services AWS que vous utilisez. Pour plus d'informations sur la confidentialité des données, consultez Questions fréquentes (FAQ) sur la confidentialité des données. Pour en savoir plus sur la protection des données en Europe, consultez le billet de blog Modèle de responsabilité partagée AWS et RGPD (Règlement général sur la protection des données) sur le Blog de sécuritéAWS .

À des fins de protection des données, nous vous recommandons de protéger les Compte AWS informations d'identification et de configurer les utilisateurs individuels avec AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Ainsi, chaque utilisateur se voit attribuer uniquement les autorisations nécessaires pour exécuter ses tâches. Nous vous recommandons également de sécuriser vos données comme indiqué ci-dessous :

- Utilisez l'authentification multifactorielle (MFA) avec chaque compte.
- Utilisez le protocole SSL/TLS pour communiquer avec les ressources. AWS Nous exigeons TLS 1.2 et recommandons TLS 1.3.
- Configurez l'API et la journalisation de l'activité des utilisateurs avec AWS CloudTrail.
- Utilisez des solutions de AWS chiffrement, ainsi que tous les contrôles de sécurité par défaut qu'ils contiennent Services AWS.
- Utilisez des services de sécurité gérés avancés tels qu'Amazon Macie, qui contribuent à la découverte et à la sécurisation des données sensibles stockées dans Amazon S3.
- Si vous avez besoin de modules cryptographiques validés par la norme FIPS 140-2 pour accéder AWS via une interface de ligne de commande ou une API, utilisez un point de terminaison FIPS. Pour plus d'informations sur les points de terminaison FIPS (Federal Information Processing Standard) disponibles, consultez Federal Information Processing Standard (FIPS) 140-2 (Normes de traitement de l'information fédérale).

Nous vous recommandons fortement de ne jamais placer d'informations confidentielles ou sensibles, telles que les adresses e-mail de vos clients, dans des balises ou des champs de texte libre tels que le champ Name (Nom). Cela inclut lorsque vous travaillez avec AWS CloudHSM ou d'autres utilisateurs de la Services AWS console, de l'API, de la CLI ou AWS des SDK. Toutes les données

que vous entrez dans des balises ou des champs de texte de forme libre utilisés pour les noms peuvent être utilisées à des fins de facturation ou dans les journaux de diagnostic. Si vous fournissez une adresse URL à un serveur externe, nous vous recommandons fortement de ne pas inclure d'informations d'identification dans l'adresse URL permettant de valider votre demande adressée à ce serveur.

Chiffrement au repos

Lorsqu'il AWS CloudHSM effectue une sauvegarde à partir d'un HSM, celui-ci chiffre ses données avant de les envoyer à. AWS CloudHSM Les données sont chiffrées à l'aide d'une clé de chiffrement unique et éphémère. Pour plus d'informations, consultez AWS CloudHSM sauvegardes en cluster.

Chiffrement en transit

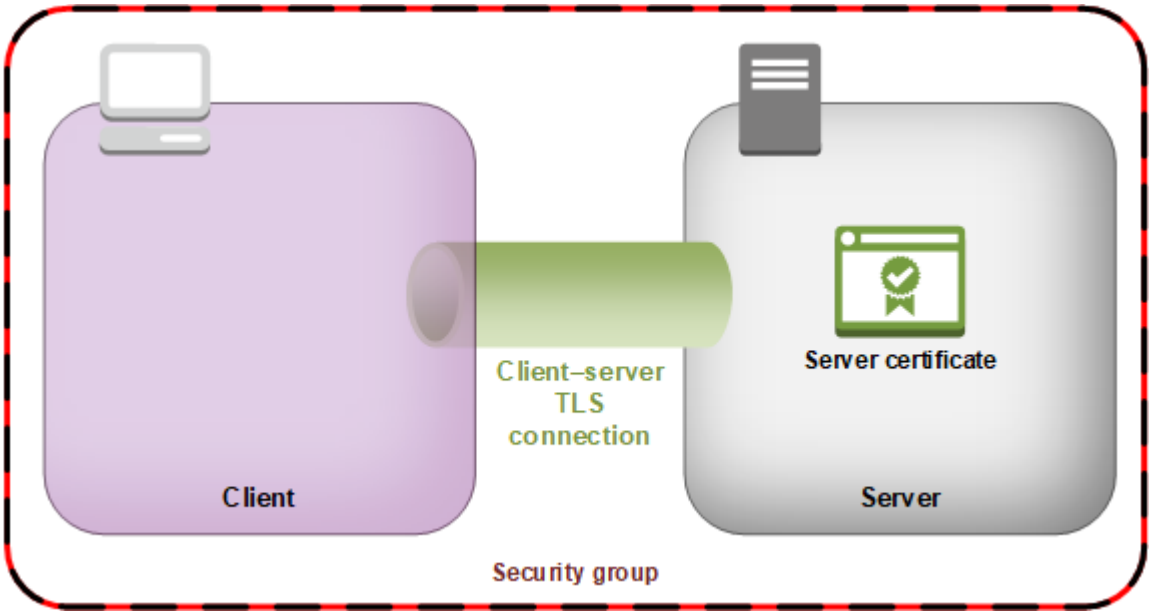
Les communications entre le AWS CloudHSM client et le HSM de votre cluster sont chiffrées de bout en bout. Cette communication ne peut être déchiffrée que par votre client et vos HSM. Pour plus d'informations, consultez nd-to-end Chiffrement électronique.

AWS CloudHSM end-to-end chiffrement du client

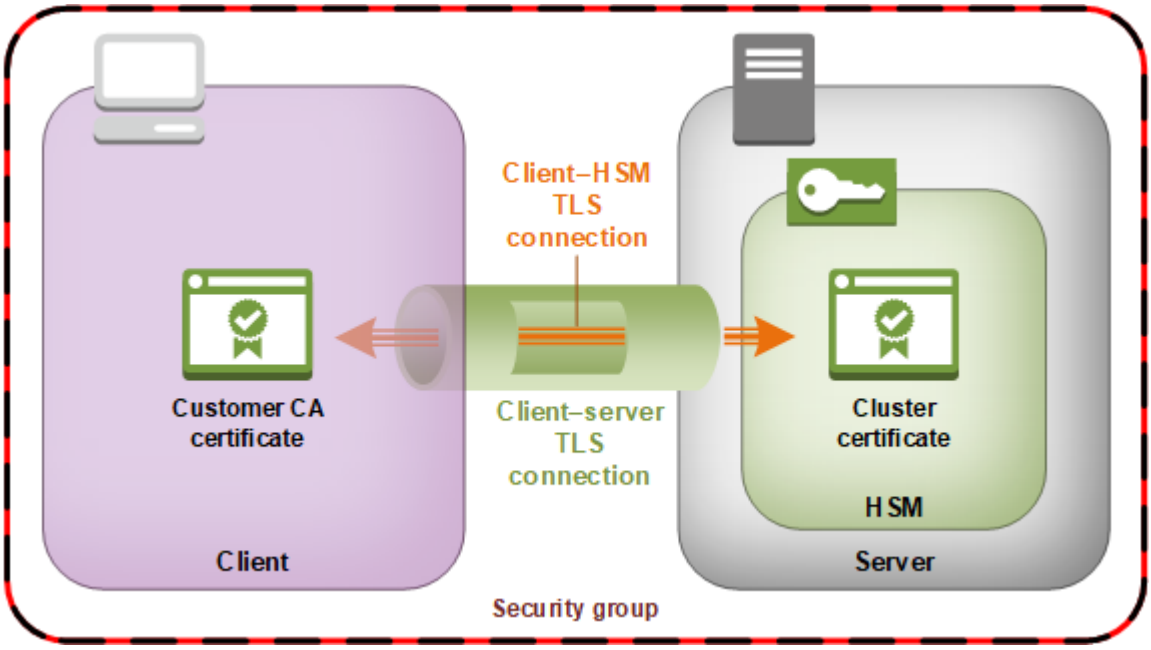
La communication entre l'instance du client et le HSM de votre cluster est chiffrée de bout en bout. Seuls votre client et vos HSM peuvent déchiffrer la communication.

Le processus suivant explique comment le client établit une communication end-to-end cryptée avec un HSM.

1. Votre client met en place une connexion TLS (Transport Layer Security) avec le serveur qui héberge votre matériel HSM. Le groupe de sécurité de votre cluster autorise le trafic entrant vers le serveur uniquement à partir d'instances client du groupe de sécurité. Le client vérifie également le certificat du serveur afin de garantir qu'il s'agit d'un serveur approuvé.



2. Le client établit ensuite une connexion chiffrée avec le matériel HSM. Le HSM comporte le certificat de cluster que vous avez signé avec votre propre autorité de certification (CA) et le client comporte le certificat racine de l'autorité de certification. Avant que la connexion chiffrée entre le client et HSM soit établie, le client vérifie le certificat de cluster du HSM par rapport à son certificat racine. La connexion est établie uniquement lorsque la vérification par le client confirme que le HSM est approuvé.



Sécurité des sauvegardes de clusters

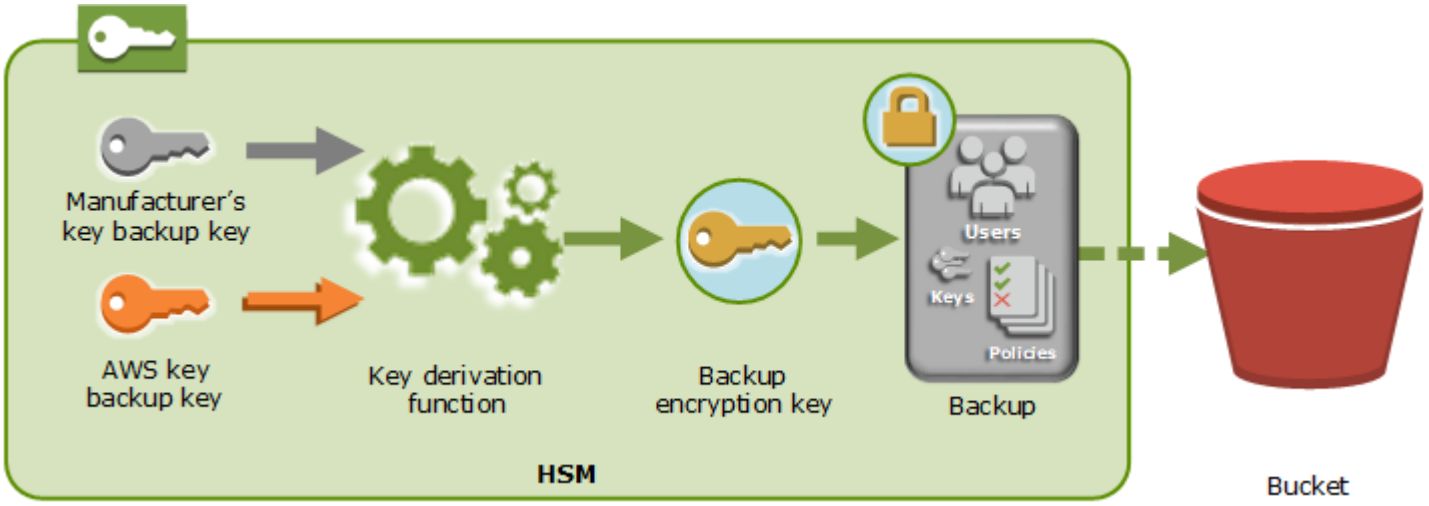
Lorsqu'il AWS CloudHSM effectue une sauvegarde à partir du HSM, celui-ci chiffre toutes ses données avant de les envoyer à. AWS CloudHSM Les données ne quittent jamais le HSM dans le formulaire en texte brut. De plus, les sauvegardes ne peuvent pas être déchiffrées AWS car il AWS n'a pas accès à la clé utilisée pour déchiffrer les sauvegardes.

Pour chiffrer ses données, le HSM utilise une clé de chiffrement éphémère unique, connue sous le nom de « clé de sauvegarde éphémère (EBK) ». L'EBK est une clé de chiffrement AES 256 bits générée dans le HSM lors AWS CloudHSM d'une sauvegarde. Le HSM génère la clé EBK, puis l'utilise pour chiffrer les données du HSM à l'aide d'une méthode d'encapsulage de clé AES approuvée par FIPS et conforme à la publication spéciale NIST 800-38F. Le HSM transmet ensuite les données chiffrées à AWS CloudHSM. Les données chiffrées incluent une copie chiffrée de la clé EBK.

Pour chiffrer l'EBK, le HSM utilise une autre clé de chiffrement connue sous le nom de « clé de sauvegarde persistante (PBK) ». La clé PBK est également une clé de chiffrement AES de 256 bits. Pour générer la clé PBK, le HSM utilise une fonction de dérivation de clés (KDF) approuvée par FIPS en mode compteur et conforme à la publication spéciale NIST 800-108. Les entrées de cette KDF comprennent les éléments suivants :

- Une clé de sauvegarde de clé de fabricant (MKBK), intégrée de manière permanente au matériel du HSM par le fabricant d'équipement.
- Une AWS clé de sauvegarde (AKBK), installée de manière sécurisée dans le HSM lors de sa configuration initiale par. AWS CloudHSM

Les processus de chiffrement sont résumés dans la figure suivante. La clé de chiffrement de sauvegarde représente la clé de sauvegarde persistante (PBK) et la clé de sauvegarde éphémère (EBK).



AWS CloudHSM peut restaurer des sauvegardes uniquement sur des HSM AWS appartenant au même fabricant. Dans la mesure où chaque sauvegarde contient tous les utilisateurs, toutes les clés et toute la configuration du HSM d'origine, le HSM restauré comporte les mêmes protections et contrôles d'accès que l'original. Les données restaurées remplacent toutes les autres données susceptibles de s'être trouvées sur le HSM avant la restauration.

Une sauvegarde comprend uniquement des données chiffrées. Avant de stocker une sauvegarde dans Amazon S3, le service chiffre à nouveau la sauvegarde à l'aide de AWS Key Management Service (AWS KMS).

Gestion des identités et des accès pour AWS CloudHSM

AWS utilise les informations d'identification de sécurité pour identifier et vous accorder l'accès à vos ressources AWS. Vous pouvez utiliser les fonctionnalités de AWS Identity and Access Management (IAM) pour permettre à d'autres utilisateurs, services et applications d'utiliser vos ressources AWS dans leur intégralité ou de manière limitée. Vous pouvez le faire sans partager vos informations d'identification de sécurité.

Par défaut, les utilisateurs IAM ne sont pas autorisés à créer, afficher ou modifier les ressources AWS. Pour permettre à un utilisateur IAM d'accéder à des ressources, telles qu'un équilibreur de charge, et d'effectuer des tâches, procédez comme suit :

1. Créez une politique IAM qui accorde à l'utilisateur IAM l'autorisation d'utiliser les actions d'API et les ressources spécifiques dont il a besoin.
2. Attachez la stratégie à l'utilisateur IAM ou au groupe auquel cet utilisateur appartient.

Quand vous attachez une politique à un utilisateur ou à un groupe d'utilisateurs, elle accorde ou refuse aux utilisateurs l'autorisation d'exécuter les tâches spécifiées sur les ressources spécifiées.

Par exemple, vous pouvez utiliser IAM pour créer des utilisateurs et des groupes sous votre compte AWS. Un utilisateur IAM peut être une personne, un système ou une application. Vous pouvez ensuite accorder des autorisations aux utilisateurs et aux groupes pour qu'ils exécutent des actions spécifiques sur les ressources spécifiées à l'aide d'une politique IAM.

Utilisation de politiques IAM pour accorder des autorisations

Quand vous attachez une politique à un utilisateur ou à un groupe d'utilisateurs, elle accorde ou refuse aux utilisateurs l'autorisation d'exécuter les tâches spécifiées sur les ressources spécifiées.

Une politique IAM est un document JSON qui se compose d'une ou de plusieurs déclarations. Chaque instruction est structurée comme illustré dans l'exemple suivant.

```
{
  "Version": "2012-10-17",
  "Statement":[{
    "Effect": "effect",
    "Action": "action",
    "Resource": "resource-arn",
    "Condition": {
      "condition": {
        "key":"value"
      }
    }
  }]
}
```

- Effect : effect peut avoir la valeur `Allow` ou `Deny`. Comme, par défaut, les utilisateurs IAM n'ont pas la permission d'utiliser les ressources et les actions d'API, toutes les demandes sont refusées. Une autorisation explicite remplace l'autorisation par défaut. Un refus explicite remplace toute autorisation.
- Action : action désigne l'action d'API spécifique pour laquelle vous accordez ou refusez l'autorisation. Pour plus d'informations sur la spécification d'action, consultez Actions d'API pour AWS CloudHSM.
- Ressource : ressource affectée par l'action. AWS CloudHSM ne prend pas en charge les autorisations au niveau des ressources. Vous devez utiliser le caractère générique* pour spécifier toutes les AWS CloudHSM ressources.

- Condition : elles permettent de contrôler à quel moment votre stratégie est effective. Pour plus d'informations, consultez Clés de condition pour AWS CloudHSM.

Pour plus d'informations, consultez le Guide de l'utilisateur IAM.

Actions d'API pour AWS CloudHSM

Dans l'élément Action de votre déclaration de politique IAM, vous pouvez spécifier toute action d'API qui AWS CloudHSM propose. Vous devez faire précéder le nom de l'action de la chaîne en lettres minuscules `cloudhsm:`, comme illustré dans l'exemple suivant.

```
"Action": "cloudhsm:DescribeClusters"
```

Pour spécifier plusieurs actions dans une même instruction, placez-les entre crochets et séparez-les par une virgule, comme dans l'exemple suivant.

```
"Action": [
    "cloudhsm:DescribeClusters",
    "cloudhsm:DescribeHsm"
]
```

Vous pouvez aussi utiliser le caractère générique * pour spécifier plusieurs actions. L'exemple suivant indique tous les noms d'actions d'API commençant par`List`. AWS CloudHSM

```
"Action": "cloudhsm:List*"
```

Pour spécifier toutes les actions d'API pour AWS CloudHSM, utilisez le caractère générique *, comme indiqué dans l'exemple suivant.

```
"Action": "cloudhsm:*"
```

Pour obtenir la liste des actions d'API pour AWS CloudHSM, consultez la section AWS CloudHSM Actions.

Clés de condition pour AWS CloudHSM

Lorsque vous créez une stratégie, vous pouvez spécifier les conditions qui définissent le moment auquel la stratégie prend effet. Chaque condition contient une ou plusieurs paires clé-valeur. Il existe des clés de condition globales et des clés de condition spécifiques à un service.

AWS CloudHSM ne possède aucune clé de contexte spécifique au service.

Pour obtenir plus d'informations sur les clés de condition globales, consultez Clés de contexte de condition globale AWS dans le Guide de l'utilisateur IAM.

Politiques gérées par AWS prédéfinies pour AWS CloudHSM

Les stratégies gérées créées par AWS octroient les autorisations requises pour les cas d'utilisation courants. Vous pouvez associer ces stratégies à vos utilisateurs IAM, en fonction de l'accès à AWS CloudHSM dont ils ont besoin :

- AWSCloudHSMFullAccess— Accorde l'accès complet requis pour utiliser les AWS CloudHSM fonctionnalités.
- AWSCloudHSMReadOnlyAccess— Accorde un accès en lecture seule aux fonctionnalités. AWS CloudHSM

Politiques gérées par le client pour AWS CloudHSM

Nous vous recommandons de créer un groupe d'administrateurs IAM AWS CloudHSM contenant uniquement les autorisations nécessaires à l'exécution AWS CloudHSM. Associez à ce groupe la stratégie avec les autorisations appropriées. Ajoutez des utilisateurs IAM au groupe. Chaque utilisateur que vous ajoutez hérite de la stratégie du groupe Administrateurs.

En outre, nous vous recommandons de créer des groupes d'utilisateurs supplémentaires en fonction des autorisations dont ils ont besoin. Cela garantit que seuls les utilisateurs de confiance ont accès aux actions d'API critiques. Par exemple, vous pouvez créer un groupe d'utilisateurs auquel vous accordez uniquement un accès en lecture seule aux clusters et aux HSM. Étant donné que ce groupe ne permet pas aux utilisateurs de supprimer des clusters ou des HSM, aucun utilisateur non approuvé ne peut affecter la disponibilité d'une charge de travail de production.

Au fur et à mesure que de nouvelles fonctionnalités de AWS CloudHSM gestion sont ajoutées, vous pouvez vous assurer que seuls les utilisateurs de confiance bénéficient d'un accès immédiat. En attribuant des autorisations limitées aux stratégies à leur création, vous pouvez leur affecter manuellement nouvelles autorisations de fonction ultérieurement.

Vous trouverez ci-dessous des exemples de politiques pour AWS CloudHSM. Pour plus d'informations sur la création d'une stratégie et son association à un groupe d'utilisateurs IAM, consultez Création de stratégies dans l'onglet JSON dans le Guide de l'utilisateur IAM.

Exemples

- Autorisations en lecture seule
- Autorisations d'utilisateur avancé
- Autorisations d'administration

Example Exemple : autorisations en lecture seule

Cette stratégie autorise l'accès aux actions d'API `DescribeClusters` et `DescribeBackups`. Elle inclut également des autorisations supplémentaires pour des actions d'API Amazon EC2 spécifiques. Cependant, elle ne permet pas aux utilisateurs de supprimer des clusters ou des HSM.

```
{
   "Version": "2012-10-17",
   "Statement": {
      "Effect": "Allow",
      "Action": [
         "cloudhsm:DescribeClusters",
         "cloudhsm:DescribeBackups",
         "cloudhsm:ListTags"
      ],
      "Resource": "*"
   }
}
```

Example Exemple : autorisations d'utilisateur avancé

Cette politique permet d'accéder à un sous-ensemble des actions de l' AWS CloudHSM API. Elle inclut également des autorisations supplémentaires pour des actions Amazon EC2 spécifiques. Cependant, elle ne permet pas aux utilisateurs de supprimer des clusters ou des HSM. Vous devez inclure l'`iam:CreateServiceLinkedRole`action permettant de AWS CloudHSM créer automatiquement le rôle AWSServiceRoleForCloudHSMlié au service dans votre compte. Ce rôle permet de AWS CloudHSM consigner les événements. Pour plus d'informations, consultez Rôles liés à un service pour AWS CloudHSM.

> **Note**
>
> Pour voir les autorisations spécifiques pour chaque API, reportez-vous à la section Actions, ressources et clés de condition pour AWS CloudHSM dans la Référence de l'autorisation de service.

```
{
   "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 Exemple : autorisations d'administrateur

Cette politique permet d'accéder à toutes les actions de AWS CloudHSM l'API, y compris les actions de suppression de HSM et de clusters. Elle inclut également des autorisations supplémentaires pour des actions Amazon EC2 spécifiques. Vous devez inclure l'`iam:CreateServiceLinkedRole`action permettant de AWS CloudHSM créer automatiquement le rôle AWSServiceRoleForCloudHSMlié au service dans votre compte. Ce rôle permet de AWS CloudHSM consigner les événements. Pour plus d'informations, consultez Rôles liés à un service pour 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":"*"
   }
}
```

Rôles liés à un service pour AWS CloudHSM

La politique IAM que vous avez créée précédemment Politiques gérées par le client pour AWS CloudHSM inclut l'`iam:CreateServiceLinkedRole`action. AWS CloudHSM définit un rôle lié à un service nommé. AWSServiceRoleForCloudHSM Le rôle est prédéfini par AWS CloudHSM et

inclut des autorisations qui AWS CloudHSM nécessitent d'appeler d'autres AWS services en votre nom. Le rôle permet de configurer votre service plus facilement, car vous n'avez pas besoin d'ajouter manuellement les autorisations de stratégie de rôle et de stratégie d'approbation.

La politique des rôles permet AWS CloudHSM de créer des groupes de CloudWatch journaux et des flux de journaux Amazon Logs et de rédiger des événements de journal en votre nom. Vous pouvez le voir ci-dessous et dans la console IAM.

```
{
    "Version": "2018-06-12",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": [
                "arn:aws:logs:*:*:*"
            ]
        }
    ]
}
```

La politique de confiance associée au AWSServiceRoleForCloudHSMrôle permet AWS CloudHSM d'assumer le rôle.

```
{
  "Version": "2018-06-12",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudhsm.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
```

```
}
```

Création d'un rôle lié à un service (automatique)

AWS CloudHSM crée le AWSServiceRoleForCloudHSMrôle lorsque vous créez un cluster si vous incluez l'`iam:CreateServiceLinkedRole`action dans les autorisations que vous avez définies lors de la création du groupe d' AWS CloudHSM administrateurs. veuillez consulter Politiques gérées par le client pour AWS CloudHSM.

Si vous possédez déjà un ou plusieurs clusters et que vous souhaitez simplement ajouter le AWSServiceRoleForCloudHSMrôle, vous pouvez utiliser la console, la commande create-cluster ou l'opération CreateClusterAPI pour créer un cluster. Utilisez ensuite la console, la commande delete-cluster ou l'opération DeleteClusterAPI pour le supprimer. La création du nouveau cluster entraîne la création du rôle lié à un service et l'applique à tous les clusters de votre compte. Vous pouvez également choisir de créer le rôle manuellement. Consultez la section suivante pour plus d'informations.

> **Note**
>
> Il n'est pas nécessaire de suivre toutes les étapes décrites dans la section Commencer avec AWS CloudHSM pour créer un cluster si vous le créez uniquement pour ajouter le AWSServiceRoleForCloudHSMrôle.

Création d'un rôle lié à un service (manuel)

Vous pouvez utiliser la console IAM ou AWS CLI l'API pour créer le AWSServiceRoleForCloudHSMrôle. Pour plus d'informations, consultez Création d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Modifier le rôle lié à un service

AWS CloudHSM ne vous permet pas de modifier le AWSServiceRoleForCloudHSMrôle. Par exemple, une fois le rôle créé, vous ne pouvez pas modifier son nom car différentes entités peuvent référencer le rôle par nom. De la même façon, vous ne pouvez pas modifier la stratégie de rôle. Néanmoins, vous pouvez utiliser IAM pour modifier la description du rôle. Pour plus d'informations, consultez Modification d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Suppression du rôle lié à un service

Vous ne pouvez pas supprimer un rôle lié à un service tant qu'il existe encore un cluster auquel il a été appliqué. Pour supprimer le rôle, vous devez d'abord supprimer tous les HSM de votre cluster, puis supprimer le cluster. Vous devez supprimer tous les clusters de votre compte. Vous pouvez ensuite utiliser la console IAM ou AWS CLI l'API pour supprimer le rôle. Pour plus d'informations sur la suppression d'un cluster, consultez Supprimer un AWS CloudHSM cluster. Pour plus d'informations, consultez Suppression d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Conformité d'

AWS CloudHSM fournit des politiques de sécurité approuvées par FIPS pour les HSM. En outre, AWS CloudHSM répond aux exigences de conformité PCI-PIN, PCI-3DS et SOC2. L'utilisation d'un HSM validé par la norme FIPS peut vous aider à répondre aux exigences de conformité de l'entreprise, contractuelles et réglementaires en matière de sécurité des données dans le cloud. AWS Pour plus d'informations, veuillez consulter les informations ci-dessous.

Conformité FIPS 140-2

La publication Federal Information Processing Standard (FIPS) 140-2 est une norme de sécurité du gouvernement américain qui spécifie les exigences de sécurité pour les modules cryptographiques qui protègent des informations sensibles. Les HSM fournis par AWS CloudHSM sont certifiés FIPS 140-2 niveau 3 (Certificat #4218). Pour plus d'informations à ce sujet, reportez-vous à la section Validation FIPS pour le matériel.

Conformité DSS d'PCI

La norme PCI (Payment Card Industry) DSS (Data Security Standard) est une norme sur la sécurité des informations propriétaires administrée par le conseil des normes de sécurité dans le secteur des cartes de paiement (Payment Card Industry Security Standards Council). Les HSM fournis par sont AWS CloudHSM conformes à la norme PCI DSS.

Conformité PCI-PIN

Le code PIN PCI fournit des exigences de sécurité et des normes d'évaluation pour la transmission, le traitement et la gestion des données de numéro d'identification personnel (PIN), informations utilisées pour les transactions aux guichets automatiques et aux terminaux point-of-sale (POS). AWS CloudHSM est conforme au code PIN PCI depuis janvier 2023. Pour plus d'informations, consultez l'article AWS CloudHSM est désormais certifié pour la conformité à PCI PIN.

Conformité à PCI 3DS

La norme PCI 3DS (ou Three Domain Secure, 3-D Secure) assure la sécurité des données pour les paiements électroniques sécurisés EMV 3D. La norme PCI 3DS fournit un niveau de sécurité supplémentaire pour les achats en ligne. AWS CloudHSM est conforme à la norme PCI-3DS.

SOC2

SOC2 est un framework destiné à aider les organisations de services à démontrer leurs contrôles de sécurité dans le cloud et les centres de données. AWS CloudHSM a mis en œuvre des contrôles SOC2 dans des domaines critiques afin de respecter les principes de service fiables. Pour plus d'informations, consultez la page FAQ sur AWS SOC.

AWS CloudHSM FAQ sur la conformité PCI-PIN

Le code PIN PCI fournit des exigences de sécurité et des normes d'évaluation pour la transmission, le traitement et la gestion des données de numéro d'identification personnel (PIN), informations utilisées pour les transactions aux guichets automatiques et aux terminaux point-of-sale (POS).

L'attestation de conformité (AOC) PCI-PIN et le résumé des responsabilités sont mis à la disposition des clients via AWS Artifact, un portail en libre-service permettant d'accéder à la demande aux rapports de conformité AWS. Pour plus d'informations, connectez-vous à AWS Artifact dans la console de gestion AWS ou consultez Mise en route avec AWS Artifact.

FAQ

Q : Qu'est-ce que l'attestation de conformité (AOC) et le résumé des responsabilités ?

L'attestation de conformité (AOC) est produite par un évaluateur de code PIN qualifié (QPA) attestant que les contrôles applicables sont conformes AWS CloudHSM à la norme PCI-PIN. La matrice récapitulative des responsabilités décrit les contrôles qui relèvent de la responsabilité respective de AWS CloudHSM et de ses clients.

Q : Comment puis-je obtenir l' AWS CloudHSM attestation de conformité ?

L'attestation de conformité PCI-PIN (AOC) est mise à la disposition des clients via AWS Artifact, un portail en libre-service permettant d'accéder à la demande aux rapports de conformité AWS. Pour plus d'informations, connectez-vous à AWS Artifact dans la console de gestion AWS ou consultez Mise en route avec AWS Artifact.

Q : Comment puis-je savoir de quels contrôles PIN PCI je suis responsable ?

Pour obtenir des informations détaillées, consultez le « Résumé des responsabilités relatives au code PIN AWS CloudHSM PCI » du package AWS de conformité au code PIN PCI, disponible pour les clients via AWS Artifact, un portail en libre-service permettant d'accéder à la demande aux rapports de conformité AWS. Pour plus d'informations, connectez-vous à AWS Artifact dans la console de gestion AWS ou consultez Mise en route avec AWS Artifact.

Q : En tant que AWS CloudHSM client, puis-je me fier à l'attestation de conformité PCI-PIN (AOC) ?

Les clients doivent gérer leur propre conformité à la norme PCI-PIN. Vous devez passer par un processus d'attestation PCI-PIN officiel par l'intermédiaire d'un évaluateur de code PIN qualifié (QPA) afin de vérifier que votre charge de travail de paiement répond à tous les contrôles/exigences PCI-PIN. Toutefois, pour les contrôles dont AWS est responsable, votre QPA peut s'appuyer sur une AWS CloudHSM attestation de conformité (AOC) sans effectuer de tests supplémentaires.

Q : Est-il AWS CloudHSM responsable des exigences PCI-PIN liées au cycle de vie de la gestion des clés ?

AWS CloudHSM est responsable du cycle de vie des appareils physiques des HSM. Les clients sont responsables des principales exigences du cycle de vie de gestion définies dans la norme PCI-PIN.

Q : Quelles AWS CloudHSM commandes sont conformes à la norme PCI-PIN ?

L'AOC résume les AWS CloudHSM contrôles évalués par QPA. Le résumé des responsabilités PCI-PIN est disponible pour les clients via AWS Artifact, un portail en libre-service permettant d'accéder à la demande aux rapports de conformité AWS.

Q : Est-ce que les fonctions de paiement telles que la traduction du code PIN et le DUKPT sont prises en AWS CloudHSM charge ?

Non, AWS CloudHSM fournit des HSM à usage général. Au fil du temps, nous pourrons fournir des fonctions de paiement. Bien que le service n'exécute pas directement les fonctions de paiement, l'attestation de conformité AWS CloudHSM PCI PIN permet aux clients d'obtenir leur propre conformité PCI pour leurs services en cours d'exécution. AWS CloudHSM Si vous souhaitez utiliser les services de cryptographie des paiements AWS pour votre charge de travail, consultez le blog « Transférer le traitement des paiements vers le cloud grâce au chiffrement des paiements AWS. »

Notifications d'obsolescence

De temps à autre, cette fonctionnalité AWS CloudHSM peut être désapprouvée afin de rester conforme aux exigences des normes FIPS 140, PCI-DSS, PCI-PIN, PCI-3DS et SOC2. Cette page répertorie les modifications actuellement applicables.

Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète

Le National Institute of Standards and Technology (NIST) [1] recommande que la prise en charge du chiffrement Triple DES (DESede, 3DES, DES3) du chiffrement et du déchiffrement des clés RSA avec le remplissage des sections PKCS #1 v1.5 ne soit plus autorisée après le 31 décembre 2023. Par conséquent, leur prise en charge prend fin le 1er janvier 2024 dans nos instances conformes à la norme de traitement des informations fédérales (FIPS).

Ce guide s'applique aux opérations cryptographiques suivantes :

- Génération de clés Triple DES
  - `CKM_DES3_KEY_GEN` pour la bibliothèque PKCS#11
  - Keygen `DESede` pour le fournisseur JCE
  - `genSymKey` avec `-t=21` pour le KMU
- Chiffrement avec des clés Triple DES (remarque : les opérations de déchiffrement sont autorisées)
  - Pour la bibliothèque PKCS#11 : chiffrement `CKM_DES3_CBC`, chiffrement `CKM_DES3_CBC_PAD` et chiffrement `CKM_DES3_ECB`
  - Pour le fournisseur JCE : chiffrement `DESede/CBC/PKCS5Padding`, chiffrement `DESede/CBC/NoPadding`, chiffrement `DESede/ECB/Padding` et chiffrement `DESede/ECB/NoPadding`
- Encapsulage, désencapsulage, chiffrement et déchiffrement de clés RSA avec le rembourrage PKCS#1 v1.5
  - encapsulage, désencapsulage, chiffrement et déchiffrement `CKM_RSA_PKCS` pour le SDK PKCS#11
  - encapsulage, désencapsulage, chiffrement et déchiffrement `RSA/ECB/PKCS1Padding` pour le SDK JCE
  - `wrapKey` et un`WrapKey` avec `-m 12` pour le KMU (où `12` la valeur du mécanisme `RSA_PKCS`)

[1] Pour plus de détails sur cette modification, reportez-vous aux tableaux 1 et 5 de la section Transition de l'utilisation des algorithmes de chiffrement et des longueurs de clé.

Résilience dans AWS CloudHSM

L'infrastructure AWS mondiale est construite autour des AWS régions et des zones de disponibilité. AWS Les régions fournissent plusieurs zones de disponibilité physiquement séparées et isolées, connectées par un réseau à faible latence, à haut débit et hautement redondant. Avec les zones

de disponibilité, vous pouvez concevoir et exploiter des applications et des bases de données qui basculent automatiquement d'une zone à l'autre sans interruption. Les zones de disponibilité sont davantage disponibles, tolérantes aux pannes et ont une plus grande capacité de mise à l'échelle que les infrastructures traditionnelles à un ou plusieurs centres de données.

Pour plus d'informations sur AWS les régions et les zones de disponibilité, consultez la section Infrastructure AWS mondiale. Pour plus d'informations sur les fonctionnalités AWS CloudHSM qui contribuent à la résilience, consultez Haute disponibilité et équilibrage de charge du cluster.

Sécurité de l'infrastructure dans AWS CloudHSM

En tant que service géré, AWS CloudHSM il est protégé par les procédures de sécurité du réseau AWS mondial décrites dans le livre blanc Amazon Web Services : présentation des processus de sécurité.

Vous utilisez des appels d'API AWS publiés pour accéder AWS CloudHSM via le réseau. En outre, les demandes doivent être signées à l'aide d'un ID de clé d'accès et d'une clé d'accès secrète associée à un principal IAM. Vous pouvez également utiliser AWS Security Token Service (AWS STS) pour générer des informations d'identification de sécurité temporaires et signer les demandes.

Isolement de réseau

Un cloud privé virtuel (VPC) est un réseau virtuel situé dans votre propre zone logiquement isolée dans le cloud AWS. Vous pouvez créer un cluster dans un sous-réseau privé de votre VPC. Vous pouvez créer des sous-réseaux privés lorsque vous créez un VPC. Pour plus d'informations, consultez Création d'un cloud privé virtuel (VPC).

Lorsque vous créez un HSM, AWS CloudHSM insérez une Elastic Network Interface (ENI) dans votre sous-réseau afin de pouvoir interagir avec vos HSM. Pour plus d'informations, consultez Architecture du cluster.

AWS CloudHSM crée un groupe de sécurité qui autorise les communications entrantes et sortantes entre les HSM de votre cluster. Vous pouvez utiliser ce groupe de sécurité pour permettre à vos instances EC2 de communiquer avec les HSM de votre cluster. Pour plus d'informations, consultez Configuration des groupes de sécurité de l'instance Amazon EC2 cliente.

Autorisation des utilisateurs

Avec AWS CloudHSM, les opérations effectuées sur le HSM nécessitent les informations d'identification d'un utilisateur HSM authentifié. Pour plus d'informations, consultez the section called "Comprendre les utilisateurs HSM".

AWS CloudHSM et points de terminaison VPC

Vous pouvez établir une connexion privée entre votre VPC et créer un point de AWS CloudHSM terminaison VPC d'interface. Les points de terminaison de l'interface sont alimentés par AWS PrivateLinkune technologie qui vous permet d'accéder à AWS CloudHSM des API de manière privée sans passerelle Internet, appareil NAT, connexion VPN ou connexion AWS Direct Connect. Les instances de votre VPC n'ont pas besoin d'adresses IP publiques pour communiquer avec AWS CloudHSM les API. Le trafic entre votre VPC et AWS CloudHSM ne quitte pas le réseau Amazon.

Chaque point de terminaison d'interface est représenté par une ou plusieurs interfaces réseau Elastic dans vos sous-réseaux.

Pour plus d'informations, consultez la section Interface VPC endpoints (AWS PrivateLink) dans le guide de l'utilisateur Amazon VPC.

Considérations relatives aux points de AWS CloudHSM terminaison VPC

Avant de configurer un point de terminaison VPC d'interface pour AWS CloudHSM, assurez-vous de consulter les propriétés et les limites du point de terminaison d'interface dans le guide de l'utilisateur Amazon VPC.

- AWS CloudHSM permet d'appeler toutes ses actions d'API depuis votre VPC.

Création d'un point de terminaison de VPC d'interface pour AWS CloudHSM

Vous pouvez créer un point de terminaison VPC pour le AWS CloudHSM service à l'aide de la console Amazon VPC ou de la (AWS Command Line Interface CLI). Pour plus d'informations, consultez Création d'un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

Pour créer un point de terminaison VPC pour AWS CloudHSM, utilisez le nom de service suivant :

```
com.amazonaws.region.cloudhsmv2
```

Par exemple, dans la région USA Ouest (Oregon) (`us-west-2`), le nom du service serait :

```
com.amazonaws.us-west-2.cloudhsmv2
```

Pour faciliter l'utilisation du point de terminaison de VPC, vous pouvez activer un nom d'hôte DNS privé pour votre point de terminaison de VPC. Si vous sélectionnez l'option Activer le nom DNS privé, le nom d'hôte AWS CloudHSM DNS standard (`https://cloudhsmv2.<region>.amazonaws.com`) correspond à votre point de terminaison VPC.

Cette option facilite l'utilisation du point de terminaison d'un VPC. Les AWS SDK et la CLI utilisent le nom d'hôte AWS CloudHSM DNS standard par défaut. Il n'est donc pas nécessaire de spécifier l'URL du point de terminaison du VPC dans les applications et les commandes.

Pour plus d'informations, consultez Accès à un service via un point de terminaison d'interface dans le Guide de l'utilisateur Amazon VPC.

Création d'une politique de point de terminaison VPC pour AWS CloudHSM

Vous pouvez attacher une stratégie de point de terminaison à votre point de terminaison d'un VPC qui contrôle l'accès à AWS CloudHSM. La politique spécifie les informations suivantes :

- Le principal qui peut exécuter des actions.
- Les actions qui peuvent être effectuées.
- Les ressources sur lesquelles les actions peuvent être exécutées.

Pour plus d'informations, consultez Contrôle de l'accès aux services avec points de terminaison d'un VPC dans le Guide de l'utilisateur Amazon VPC.

Exemple : politique de point de terminaison VPC pour les actions AWS CloudHSM

Voici un exemple de politique de point de terminaison pour AWS CloudHSM. Lorsqu'elle est attachée à un point de terminaison, cette politique accorde l'accès aux AWS CloudHSM actions répertoriées pour tous les principaux sur toutes les ressources. Consultez Gestion des identités et des accès pour AWS CloudHSM les autres AWS CloudHSM actions et les autorisations IAM correspondantes.

```
{
```

```
   "Statement":[
      {
         "Principal":"*",
         "Effect":"Allow",
         "Action":[
            "cloudhsm:DescribeBackups",
            "cloudhsm:DescribeClusters",
            "cloudhsm:ListTags",
         ],
         "Resource":"*"
      }
   ]
}
```

Gestion des mises à jour dans AWS CloudHSM

AWS gère le microprogramme. La maintenance du microprogramme est effectuée par un tiers et sa conformité à la norme FIPS 140-2 niveau 3 doit être évaluée par le NIST (National Institute of Standards and Technology). Seuls les microprogrammes qui ont été signés de manière chiffrée à l'aide d'une clé FIPS (à laquelle AWS n'a pas accès) peuvent être installés.

Résolution des problèmes AWS CloudHSM

Si vous rencontrez des problèmes avec AWS CloudHSM, les rubriques suivantes peuvent vous aider à les résoudre.

Rubriques

Problèmes connus

AWS CloudHSM présente les problèmes connus suivants. Choisissez une rubrique pour en savoir plus.

Rubriques

Problèmes connus pour toutes les instances HSM

Les problèmes suivants concernent tous les AWS CloudHSM utilisateurs, qu'ils utilisent l'outil de ligne de commande key_mgmt_util, le SDK PKCS #11, le SDK JCE ou le SDK OpenSSL.

Rubriques

Problème : l'encapsulage de clé AES utilise le remplissage PKCS#5 au lieu de fournir une implémentation conforme aux normes de l'encapsulage de clé avec remplissage avec des zéros

En outre, l'encapsulage de clé sans remplissage ni remplissage avec des zéros n'est pas pris en charge.

- Conséquence : il n'y a aucun impact si vous encapsulez et déballez à l'aide de cet algorithme intégré. AWS CloudHSM Cependant, les clés encapsulées AWS CloudHSM ne peuvent pas être déballées dans d'autres HSM ou logiciels qui s'attendent à être conformes à la spécification de non-rembourrage. Cela est dû au fait que huit octets de données de remplissage peuvent être ajoutés à la fin de vos données clés lors d'un désencapsulage conforme aux normes. Les clés encapsulées en externe ne peuvent pas être correctement déballées dans une AWS CloudHSM instance.
- Solution : pour désencapsuler en externe une clé encapsulée à l'aide de la fonction AES Key Wrap avec PKCS #5 Padding sur une instance AWS CloudHSM, enlevez le remplissage supplémentaire avant d'essayer d'utiliser la clé. Pour ce faire, vous pouvez essayer de réduire les octets supplémentaires dans un éditeur de fichier ou copier uniquement les octets de la clé dans une nouvelle mémoire tampon dans votre code.
- État de la résolution : avec la version client et logiciel 3.1.0, AWS CloudHSM fournit des options conformes aux normes pour l'encapsulage des clés AES. Pour plus d'informations, consultez Encapsulage des clés AES.

Problème : Le démon client nécessite au moins une adresse IP valide dans son fichier de configuration pour pouvoir se connecter au cluster

- Impact : Si vous supprimez tous les HSM de votre cluster, puis que vous ajoutez un autre HSM qui obtient une nouvelle adresse IP, le démon client continue de rechercher vos HSM à leurs adresses IP d'origine.
- Solution : Si vous exécutez une charge de travail intermittente, nous vous recommandons d'utiliser l'`IpAddress`argument de la CreateHsmfonction pour rétablir la valeur d'origine de l'Elastic Network Interface (ENI). Notez qu'une ENI est spécifique à une zone de disponibilité (AZ). L'alternative consiste à supprimer le fichier `/opt/cloudhsm/daemon/1/cluster.info`, puis à réinitialiser la configuration du client par l'adresse IP de votre nouveau HSM. Vous pouvez utiliser la commande `client -a <IP address>`. Pour plus d'informations, voir Installer et configurer le AWS CloudHSM client (Linux) ou Installer et configurer le AWS CloudHSM client (Windows).

Problème : les données pouvant être hachées et signées à l' AWS CloudHSM aide du SDK client 3 étaient limitées à 16 Ko

- État de résolution : les données inférieures à 16 Ko continuent d'être envoyées au HSM pour hachage. Nous avons ajouté de la capacité pour hacher en local, dans les logiciels, les données

comprises entre 16 Ko et 64 Ko. Le SDK client 5 échouera explicitement si la mémoire tampon de données est supérieure à 64 Ko. Vous devez mettre à jour votre client et votre ou vos SDK vers une version supérieure à 5.0.0 ou supérieure pour bénéficier du correctif.

Problème : les clés importées ne peuvent pas être spécifiées comme non exportables

- État de résolution : le problème est résolu. Aucune action n'est requise de votre part pour tirer parti du correctif.

Problème : le mécanisme par défaut pour le WrapKey et les unWrapKey commandes du key_mgmt_util a été supprimé

- Résolution : Lorsque vous utilisez le WrapKey ou unWrapKey les commandes, vous devez utiliser l'`-m`option pour spécifier le mécanisme. Consultez les exemples fournis dans le WrapKey ou les unWrapKeyarticles pour plus d'informations.

Problème : si vous avez un seul HSM dans votre cluster, le basculement HSM ne se produit pas correctement

- Impact : si l'instance HSM unique de votre cluster perd la connectivité, le client ne se reconnectera pas à elle, même si l'instance HSM est restaurée ultérieurement.
- Solution de contournement : nous recommandons d'au moins deux instances HSM dans n'importe quel cluster de production. Si vous utilisez cette configuration, vous ne serez pas affecté par ce problème. Pour les clusters à un seul HSM, renvoyez le démon client à l'expéditeur pour restaurer la connectivité.
- État de résolution : ce problème a été résolu dans la version client 1.1.2 de AWS CloudHSM . Vous devez effectuer une mise à niveau vers ce client afin de bénéficier de la correction.

Problème : si vous dépassez la capacité en clés des HSM de votre cluster dans un court laps de temps, le client entre en état d'erreur non gérée

- Impact : lorsque le client rencontre une erreur non gérée, il se bloque et doit être redémarré.
- Solution de contournement : testez votre débit pour vous assurer que vous ne créez pas de clés de session plus rapidement que ce que le client est en mesure de traiter. Vous pouvez réduire votre débit en ajoutant un HSM au cluster ou en ralentissant la création de clé de session.

- État de résolution : ce problème a été résolu dans la version client 1.1.2 de AWS CloudHSM . Vous devez effectuer une mise à niveau vers ce client afin de bénéficier de la correction.

Problème : les opérations de digest avec clés HMAC de taille supérieure à 800 octets ne sont pas prises en charge

- Impact : les clés HMAC de plus de 800 octets peuvent être générées ou importées dans le HSM. Toutefois, si vous utilisez cette clé dans une opération de digest via JCE ou key_mgmt_util, l'opération échoue. Notez que si vous utilisez PKCS11, les clés HMAC sont limitées à une taille de 64 octets.
- Solution de contournement : si vous utiliserez des clés HMAC pour les opérations de digest sur le HSM, assurez-vous que la taille est inférieure à 800 octets.
- État de résolution : Aucun pour l'instant.

Problème : L'outil client_info, distribué avec le SDK client 3, supprime le contenu du chemin spécifié par l'argument de sortie facultatif

- Conséquence : tous les fichiers et sous-répertoires existants situés sous le chemin de sortie spécifié risquent d'être définitivement perdus.
- Solution : n'utilisez pas l'argument facultatif `-output` *`path`* lorsque vous utilisez l'outil `client_info`.
- État de résolution : ce problème a été résolu dans la version SDK client 3.3.2. Vous devez effectuer une mise à niveau vers ce client afin de bénéficier de la correction.

Problème : vous recevez une erreur lors de l'exécution de l'outil de configuration du SDK 5 à l'aide de l'argument **`--cluster-id`** dans des environnements conteneurisés

Le message d'erreur suivant s'affiche lorsque vous utilisez l'argument --cluster-id avec l'outil de configuration :

```
No credentials in the property bag
```

Cette erreur est due à une mise à jour de la version 2 du service des métadonnées d'instance (IMDSv2). Pour en savoir plus, consultez la documentation IMDSv2.

- Conséquence : ce problème aura un impact sur les utilisateurs exécutant l'outil de configuration sur les versions 5.5.0 et ultérieures du SDK dans des environnements conteneurisés et utilisant les métadonnées d'instance EC2 pour fournir des informations d'identification.
- Solution : définissez la limite de sauts de réponse PUT à au moins deux. Pour savoir comment procéder, voir Configurer les options de métadonnées de l'instance.

Problème : vous recevez le message d'erreur « Impossible de créer le cert/la clé à partir du fichier pfx fourni. Erreur : NotPkcs 8 pouces

- Conséquence : les utilisateurs du SDK 5.11.0 qui reconfigurent le protocole SSL à l'aide d'un certificat et d'une clé privée échoueront si leurs clés privées ne sont pas au format PKCS8.
- Solution : vous pouvez convertir la clé privée SSL personnalisée au format PKCS8 à l'aide de la commande openssl : `openssl pkcs8 -topk8 -inform PEM -outform PEM -in ssl_private_key -out ssl_private_key_pkcs8`
- État de la résolution : ce problème a été résolu dans la version 5.12.0 du SDK client. Vous devez effectuer une mise à niveau vers cette version du client ou une version ultérieure pour bénéficier du correctif.

Problèmes connus pour la bibliothèque PKCS#11

Rubriques

- Problème : l'encapsulage de la clé AES dans la version 3.0.0 de la bibliothèque PKCS#11 ne valide pas les IV avant utilisation
- Problème : Le kit SDK PKCS #11 2.0.4 et versions antérieures utilisaient toujours le vecteur d'initialisation par défaut de 0xA6A6A6A6A6A6A6A6 pour l'encapsulage et le désencapsulage des clés AES.
- Problème : l'attribut CKA_DERIVE n'est pas pris en charge et n'est pas géré.
- Problème : l'attribut CKA_SENSITIVE n'est pas pris en charge et n'est pas géré.
- Problème : Le hachage et la signature en plusieurs parties ne sont pas pris en charge.
- Problème : C_GenerateKeyPair ne gère pas CKA_MODULUS_BITS ou CKA_PUBLIC_EXPONENT dans le modèle privé d'une manière conforme aux normes.
- Problème : Les mémoires tampons des opérations d'API C_Encrypt et C_Decrypt ne doivent pas avoir une taille de plus de 16 Ko lors de l'utilisation du mécanisme CKM_AES_GCM.

Problème : l'encapsulage de la clé AES dans la version 3.0.0 de la bibliothèque PKCS#11 ne valide pas les IV avant utilisation

Si vous spécifiez un vecteur d'initialisation d'une longueur inférieure à 8 octets, celui-ci est complété par des octets imprévisibles avant utilisation.

> **Note**
>
> Cela a un impact sur `C_WrapKey` uniquement avec un mécanisme `CKM_AES_KEY_WRAP`.

- Impact : Si vous spécifiez un vecteur d'initialisation d'une longueur inférieure à 8 octets dans la version 3.0.0 de la bibliothèque PKCS#11, vous risquez de ne pas pouvoir désencapsuler la clé.
- Solutions de contournement :
  - Nous vous recommandons fortement de procéder à une mise à niveau vers la version 3.0.1 ou ultérieure de la bibliothèque PKCS#11, qui applique correctement la longueur des vecteurs d'initialisation pendant l'encapsulage des clés AES. Modifiez votre code d'encapsulage pour transmettre un vecteur d'initialisation NULL, ou spécifiez le vecteur d'initialisation par défaut de `0xA6A6A6A6A6A6A6A6`. Pour plus d'informations, consultez les vecteurs d'initialisation personnalisés de longueur non conforme pour l'encapsulage des clés AES.
  - Si vous avez encapsulé des clés avec la version 3.0.0 de la bibliothèque PKCS #11 en utilisant un vecteur d'initialisation d'une longueur inférieure à 8 octets, contactez-nous pour obtenir de l'aide.
- Statut de la résolution : ce problème a été résolu dans la version 3.0.1 de la bibliothèque PKCS #11. Pour encapsuler des clés à l'aide de l'encapsulage de clés AES, spécifiez un vecteur d'initialisation de valeur NULL ou d'une longueur égale à 8 octets.

Problème : Le kit SDK PKCS #11 2.0.4 et versions antérieures utilisaient toujours le vecteur d'initialisation par défaut de **`0xA6A6A6A6A6A6A6A6`** pour l'encapsulage et le désencapsulage des clés AES.

Les vecteurs d'initialisation fournis par l'utilisateur étaient ignorés.

> **Note**
>
> Cela a un impact sur `C_WrapKey` uniquement avec un mécanisme `CKM_AES_KEY_WRAP`.

- Impact :
  - Si vous avez utilisé le kit SDK PKCS #11 2.0.4 ou version antérieure et un vecteur d'initialisation fourni par l'utilisateur, vos clés sont encapsulées avec le vecteur d'initialisation par défaut de `0xA6A6A6A6A6A6A6A6`.
  - Si vous avez utilisé le kit PKCS #11 3.0.0 ou version ultérieure et un vecteur d'initialisation fourni par l'utilisateur, vos clés sont encapsulées avec le vecteur d'initialisation fourni par l'utilisateur.
- Solutions de contournement :
  - Pour désencapsuler des clés encapsulées avec le kit SDK PKCS #11 2.0.4 ou version antérieure, utilisez le vecteur d'initialisation par défaut de `0xA6A6A6A6A6A6A6A6`.
  - Pour désencapsuler des clés encapsulées avec le kit SDK PKCS #11 3.0.0 ou version ultérieure, utilisez le vecteur d'initialisation fourni par l'utilisateur.
- État de résolution : Nous vous recommandons fortement de modifier votre code d'encapsulage et de désencapsulage pour transmettre un vecteur d'initialisation NULL, ou de spécifier le vecteur d'initialisation par défaut de `0xA6A6A6A6A6A6A6A6`.

Problème : l'attribut **`CKA_DERIVE`** n'est pas pris en charge et n'est pas géré.

- Statut de résolution : nous avons implémenté des correctifs pour accepter `CKA_DERIVE` s'il a la valeur `FALSE`. `CKA_DERIVE` défini sur `TRUE` ne sera pas pris en charge tant que nous n'aurons pas ajouté la prise en charge de la fonction de dérivation de clés à AWS CloudHSM. Vous devez mettre à jour votre client et les kits SDK vers la version 1.1.1 ou ultérieur pour tirer parti du correctif.

Problème : l'attribut **CKA_SENSITIVE** n'est pas pris en charge et n'est pas géré.

- État de résolution : nous avons implémenté les correctifs pour accepter et honorer correctement l'attribut `CKA_SENSITIVE`. Vous devez mettre à jour votre client et les kits SDK vers la version 1.1.1 ou ultérieur pour tirer parti du correctif.

Problème : Le hachage et la signature en plusieurs parties ne sont pas pris en charge.

- Impact : `C_DigestUpdate` et `C_DigestFinal` ne sont pas implémentés. `C_SignFinal` n'est pas implémenté non plus et échoue avec l'erreur `CKR_ARGUMENTS_BAD` pour un tampon non-`NULL`.
- Solution : hachez vos données dans votre application et utilisez-les AWS CloudHSM uniquement pour signer le hachage.
- État de résolution : Nous corrigeons actuellement le client et les kits SDK afin qu'ils mettent en œuvre correctement le hachage en plusieurs parties. Les mises à jour seront annoncées sur le forum AWS CloudHSM et sur la page de l'historique des versions.

Problème : **C_GenerateKeyPair** ne gère pas **CKA_MODULUS_BITS** ou **CKA_PUBLIC_EXPONENT** dans le modèle privé d'une manière conforme aux normes.

- Impact : `C_GenerateKeyPair` doit renvoyer `CKA_TEMPLATE_INCONSISTENT` lorsque le modèle privé contient `CKA_MODULUS_BITS` ou `CKA_PUBLIC_EXPONENT`. Au lieu de cela, il génère une clé privée dont tous les champs d'utilisation sont définis sur `FALSE`. La clé ne peut pas être utilisée.
- Solution : Nous recommandons que votre application vérifie les valeurs des champs d'utilisation en plus du code d'erreur.
- État de résolution : Nous sommes en train de mettre en place des correctifs pour renvoyer le message d'erreur adéquat lorsqu'un modèle de clé privée incorrect est utilisé. La bibliothèque PKCS #11 utilisée sera annoncée sur la page d'historique des versions.

Problème : Les mémoires tampons des opérations d'API **C_Encrypt** et **C_Decrypt** ne doivent pas avoir une taille de plus de 16 Ko lors de l'utilisation du mécanisme **CKM_AES_GCM**.

AWS CloudHSM ne prend pas en charge le chiffrement AES-GCM en plusieurs parties.

- Impact : Vous ne pouvez pas utiliser le mécanisme `CKM_AES_GCM` pour chiffrer des données de plus de 16 Ko.
- Solution : vous pouvez utiliser un autre mécanisme tel que `CKM_AES_CBC`CKM_AES_CBC_PAD, ou vous pouvez diviser vos données en plusieurs parties et chiffrer chaque élément individuellement. `AES_GCM` Si vous en utilisez`AES_GCM`, vous devez gérer la division de vos données et le chiffrement ultérieur. AWS CloudHSM n'effectue pas de chiffrement AES-GCM en plusieurs parties pour vous. Notez que FIPS nécessite que le vecteur d'initialisation (IV) `AES-GCM` soit généré sur le HSM. Par conséquent, le vecteur d'initialisation de chaque portion de données AES GCM chiffrées sera différent.
- État de résolution : Nous sommes en train de corriger le kit SDK afin qu'il échoue de façon explicite si le tampon de données est trop volumineux. Nous retournons `CKR_MECHANISM_INVALID` pour les opérations d'API `C_EncryptUpdate` et `C_DecryptUpdate`. Nous évaluons les alternatives possibles pour prendre en charge des mémoires tampons de plus grande taille sans devoir s'appuyer sur le chiffrement en plusieurs parties. Les mises à jour seront annoncées sur le AWS CloudHSM forum et sur la page d'historique des versions.

Problème : La dérivation de clé Diffie-Hellman à courbe elliptique (ECDH) est exécutée partiellement dans le HSM.

Votre clé privée EC reste dans le HSM à tout moment, mais le processus de dérivation de clés est effectué en plusieurs étapes. Par conséquent, les résultats intermédiaires de chaque étape sont disponibles sur le client.

- Conséquence : dans le SDK client 3, la clé dérivée à l'aide du `CKM_ECDH1_DERIVE` mécanisme est d'abord disponible sur le client, puis importée dans le HSM. Un handle de clé est ensuite retourné à votre application.
- Solution : Si vous mettez en service le déchargement SSL/TLS dans AWS CloudHSM, cette limitation n'est pas nécessairement un problème. Si votre application nécessite que votre clé reste dans une limite FIPS à tout moment, envisagez d'utiliser un autre protocole qui ne s'appuie pas sur la dérivation de clés ECDH.
- État de résolution : Nous développons actuellement la possibilité d'effectuer la dérivation de clés ECDH entièrement dans le HSM. L'implémentation mise à jour sera annoncée sur la page de l'historique des versions.

Problème : La vérification des signatures secp256k1 échoue sur les plateformes EL6 telles que CentOS6 et RHEL 6

En effet, la bibliothèque PKCS #11 CloudHSM permet d'éviter un appel réseau lors de l'initialisation de l'opération de vérification en utilisant OpenSSL pour vérifier les données de courbe EC. Secp256k1 n'étant pas pris en charge par le package OpenSSL par défaut sur les plateformes EL6, l'initialisation échoue.

- Impact : La vérification de signature Secp256k1 échoue sur les plateformes EL6. L'appel de vérification échoue avec une erreur `CKR_HOST_MEMORY`.
- Solution de contournement : Nous vous conseillons d'utiliser Amazon Linux 1 ou une plateforme EL7 si votre application PKCS #11 a besoin de vérifier des signatures secp256k1. Vous pouvez également effectuer une mise à niveau vers une version du package OpenSSL qui prend en charge la courbe secp256k1.
- État de résolution : Nous implémentons des correctifs pour revenir au HSM si la validation de courbe locale n'est pas disponible. La bibliothèque PKCS #11 mise à jour sera annoncée sur la page de l' historique des versions.

Problème : une séquence incorrecte d'appels de fonction donne des résultats indéfinis au lieu d'échouer

- Conséquence : si vous appelez une séquence de fonctions incorrecte, le résultat final est incorrect même si chaque appel de fonction renvoie un résultat positif. Par exemple, les données déchiffrées peuvent ne pas correspondre au texte brut d'origine ou les signatures peuvent ne pas être vérifiées. Ce problème concerne à la fois les opérations en une seule partie et en plusieurs parties.

  Exemples de séquences de fonctions incorrectes :

  - `C_EncryptInit`/`C_EncryptUpdate` suivi de `C_Encrypt`
  - `C_DecryptInit`/`C_DecryptUpdate` suivi de `C_Decrypt`
  - `C_SignInit`/`C_SignUpdate` suivi de `C_Sign`
  - `C_VerifyInit`/`C_VerifyUpdate` suivi de `C_Verify`
  - `C_FindObjectsInit` suivi de `C_FindObjectsInit`
- Solution : Votre application doit, conformément à la spécification PKCS #11, utiliser la bonne séquence d'appels de fonction pour les opérations en une ou plusieurs parties. Dans ce cas, votre

application ne doit pas s'appuyer sur la bibliothèque CloudHSM PKCS #11 pour renvoyer une erreur.

Problème : la session en lecture seule n'est pas prise en charge dans le SDK 5

- Problème : le SDK 5 ne prend pas en charge l'ouverture de sessions en lecture seule avec `C_OpenSession`.
- Conséquence : si vous tentez d'appeler `C_OpenSession` sans fournir `CKF_RW_SESSION`, l'appel échouera avec le message d'erreur `CKR_FUNCTION_FAILED`.
- Solution : Lorsque vous ouvrez une session, vous devez transmettre les indicateurs `CKF_SERIAL_SESSION | CKF_RW_SESSION` à l'appel de fonction `C_OpenSession`.

Problème : le fichier **`cryptoki.h`** d'en-tête est réservé à Windows

- Problème : avec les versions 5.0.0 à 5.4.0 du SDK AWS CloudHSM client 5 sous Linux, le fichier d'en-tête n'`/opt/cloudhsm/include/pkcs11/cryptoki.h`est compatible qu'avec les systèmes d'exploitation Windows.
- Conséquence : vous pouvez rencontrer des problèmes lorsque vous essayez d'inclure ce fichier d'en-tête dans votre application sur des systèmes d'exploitation basés sur Linux.
- État de résolution : mise à niveau vers la version 5.4.1 ou supérieure du SDK AWS CloudHSM client 5, qui inclut une version compatible avec Linux de ce fichier d'en-tête.

Problèmes connus pour le kit SDK JCE

Rubriques

- Problème : le SDK client 5 émet l'avertissement « Une opération d'accès réflexif illégale s'est produite »
- Problème : le pool de sessions JCE est épuisé
- Problème : fuite de mémoire du SDK client 5 avec les opérations GetKey

Problème : lorsque vous utilisez des paires de clés asymétriques, vous voyez la capacité de clé occupée même lorsque vous ne créez pas ou n'importez pas explicitement des clés

- Impact : ce problème peut entraîner un manque d'espace de clé pour vos HSM et se produire lorsque votre application utilise un objet clé JCE standard pour les opérations de chiffrement au lieu d'un objet `CaviumKey`. Lorsque vous utilisez un objet de clé JCE standard, `CaviumProvider` importe implicitement cette clé dans le HSM, car une clé de session ne supprime pas la clé tant que l'application n'est pas finie. Par conséquent, les clés s'accumulent pendant que l'application est en cours d'exécution et peuvent faire en sorte que vos HSM manquent d'espace de clé libre, ce qui bloque votre application.
- Solution : lorsque vous utilisez la classe `CaviumSignature`, `CaviumCipher`, `CaviumMac` ou `CaviumKeyAgreement`, vous devez fournir la clé comme un objet de clé `CaviumKey` au lieu d'un objet de clé JCE standard.

  Vous pouvez convertir manuellement une clé normale en une clé `CaviumKey` à l'aide de la classe `ImportKey`, puis supprimer manuellement la clé une fois l'opération terminée.
- Statut de résolution : nous mettons à jour `CaviumProvider` pour gérer correctement les importations implicites. Une fois disponible, la mise à jour sera annoncée sur la page de l'historique des versions.

Problème : Le JCE KeyStore est en lecture seule

- Impact : À l'heure actuelle, vous ne pouvez pas stocker un type d'objet de type qui n'est pas pris en charge par le HSM dans le JCE Keystore. Plus particulièrement, vous ne pouvez pas stocker des certificats dans le magasin de clés. Cela empêche l'interopérabilité avec des outils tels que jarsigner qui s'attendent à trouver le certificat dans le keystore.
- Solution : Vous pouvez retravailler votre code pour charger des certificats à partir de fichiers locaux ou d'un emplacement du compartiment S3 plutôt que depuis le magasin de clés.

- Statut de résolution : Nous ajoutons actuellement la prise en charge du stockage de certificats dans le magasin de clés. Une fois disponible, la fonction sera annoncée sur la page de l'historique des versions.

Problème : Les mémoires tampons de chiffrement AES GCM ne peuvent pas dépasser 16 000 octets.

Le chiffrement AES GCM en plusieurs parties n'est pas pris en charge.

- Impact : Vous ne pouvez pas utiliser AES-GCM pour chiffrer des données dont la taille est supérieure à 16 000 octets.
- Solution : Vous pouvez utiliser un autre mécanisme tel que AES-CBC ou diviser vos données en plusieurs parties et chiffrer chaque partie individuellement. Si vous divisez les données, vous devez gérer le texte chiffré divisé et son déchiffrement. Comme la norme FIPS nécessite que le vecteur d'initialisation (IV) pour AES GCM soit généré sur le HSM, le vecteur d'initialisation de chaque portion de données AES GCM chiffrées sera différent.
- État de résolution : Nous sommes en train de corriger le kit SDK afin qu'il échoue de façon explicite si le tampon de données est trop volumineux. Nous évaluons les alternatives possibles pour prendre en charge des mémoires tampons de plus grande taille sans devoir s'appuyer sur le chiffrement en plusieurs parties. Les mises à jour seront annoncées sur le forum AWS CloudHSM et sur la page de l'historique des versions.

Problème : La dérivation de clé Diffie-Hellman à courbe elliptique (ECDH) est exécutée partiellement dans le HSM.

Votre clé privée EC reste dans le HSM à tout moment, mais le processus de dérivation de clés est effectué en plusieurs étapes. Par conséquent, les résultats intermédiaires de chaque étape sont disponibles sur le client. Un exemple de dérivation de clé ECDH est disponible dans les exemples de code Java.

- Conséquence : le SDK client 3 ajoute des fonctionnalités ECDH au JCE. Lorsque vous utilisez la `KeyAgreement` classe pour dériver un SecretKey, elle est d'abord disponible sur le client, puis importée dans le HSM. Un handle de clé est ensuite retourné à votre application.
- Solution : si vous implémentez le déchargement SSL/TLS dans AWS CloudHSM, cette limitation ne posera peut-être aucun problème. Si votre application nécessite que votre clé reste dans une limite

FIPS à tout moment, envisagez d'utiliser un autre protocole qui ne s'appuie pas sur la dérivation de clés ECDH.

- État de résolution : Nous développons actuellement la possibilité d'effectuer la dérivation de clés ECDH entièrement dans le HSM. Lorsqu'elle sera disponible, nous annoncerons l'implémentation mise à jour sur la page de l'historique des versions.

Problème : KeyGenerator et interprète KeyAttribute incorrectement le paramètre de taille de clé en nombre d'octets au lieu de bits

Lors de la génération d'une clé à l'aide de la `init` fonction de la KeyGenerator classe ou de l'`SIZE`attribut de l'AWS CloudHSM KeyAttribute énumération, l'API s'attend à tort à ce que l'argument soit le nombre d'octets clés, alors qu'il devrait plutôt être le nombre de bits de clé.

- Conséquence : les versions 5.4.0 à 5.4.2 du SDK client s'attendent à tort à ce que la taille de clé soit fournie aux API spécifiées en octets.
- Solution : convertissez la taille de la clé en bits en octets avant d'utiliser la KeyGenerator classe ou KeyAttribute l'énumération pour générer des clés à l'aide du fournisseur AWS CloudHSM JCE si vous utilisez les versions 5.4.0 à 5.4.2 du SDK client.
- État de résolution : mettez à niveau la version du SDK client vers la version 5.5.0 ou ultérieure, qui inclut un correctif permettant de s'attendre correctement à la taille des clés en bits lors de l'utilisation de la KeyGenerator classe ou de l' KeyAttribute énumération pour générer des clés.

Problème : le SDK client 5 émet l'avertissement « Une opération d'accès réflexif illégale s'est produite »

Lorsque vous utilisez le SDK client 5 avec Java 11, CloudHSM émet l'avertissement Java suivant :

```
```
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
```
```

```
```

Ces avertissements n'ont aucun impact. Nous sommes conscients de ce problème et nous nous efforçons de le résoudre. Aucune résolution ni solution de contournement n'est nécessaire.

Problème : le pool de sessions JCE est épuisé

Conséquence : il se peut que vous ne puissiez pas effectuer d'opérations dans JCE après avoir reçu le message suivant :

```
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
```

Solutions de contournement :

- Redémarrez votre application JCE en cas de problème.
- Lorsque vous effectuez une opération, vous devrez peut-être terminer l'opération JCE avant de perdre toute référence à l'opération.

> **Note**
>
> En fonction de l'opération, une méthode de complétion peut être nécessaire.

Opération	Méthode(s) de complétion
Chiffrement	`doFinal()` en mode chiffrement ou déchiffrement `wrap()` en mode d'encapsulage `unwrap()` en mode de désencapsulage
KeyAgreement	`generateSecret()` ou `generateSecret(String)`

Opération	Méthode(s) de complétion
KeyPairGenerator	`generateKeyPair()` , `genKeyPair()` ou `reset()`
KeyStore	Aucune méthode n'est nécessaire
MAC	`doFinal()` ou `reset()`
MessageDigest	`digest()` ou `reset()`
SecretKeyFactory	Aucune méthode n'est nécessaire
SecureRandom	Aucune méthode n'est nécessaire
Signature	`sign()` en mode signature `verify()` en mode vérification

État de la résolution : nous avons résolu ce problème dans le SDK client 5.9.0 et versions ultérieures. Pour résoudre ce problème, mettez à niveau votre SDK client vers l'une de ces versions.

Problème : fuite de mémoire du SDK client 5 avec les opérations GetKey

- Conséquence : l'`getKey`opération d'API présente une fuite de mémoire dans JCE dans les versions 5.10.0 et antérieures du SDK client. Si vous utilisez l'`getKey`API plusieurs fois dans votre application, cela entraînera une augmentation de la mémoire et, par conséquent, une augmentation de l'empreinte mémoire de votre application. Au fil du temps, cela peut provoquer des erreurs de régulation ou nécessiter le redémarrage de l'application.
- Solution : nous vous recommandons de passer au SDK client 5.11.0. Si cela n'est pas possible, nous vous recommandons de ne pas appeler l'`getKey`API plusieurs fois dans votre application. Réutilisez plutôt autant que possible la clé précédemment renvoyée `getKey` lors de l'opération précédente.
- État de la résolution : mettez à niveau la version du SDK client vers la version 5.11.0 ou ultérieure, qui inclut un correctif pour ce problème.

Problèmes connus pour le moteur dynamique OpenSSL

Voici les problèmes connus pour le moteur dynamique OpenSSL

Rubriques

Problème : vous ne pouvez pas installer AWS CloudHSM OpenSSL Dynamic Engine sur RHEL 6 et CentOS6

- Impact : OpenSSL Dynamic Engine prend uniquement en charge OpenSSL 1.0.2 [f+]. Par défaut, RHEL 6 et CentOS 6 sont livrés avec OpenSSL 1.0.1.
- Solution : mettez à niveau la bibliothèque OpenSSL sur RHEL 6 et CentOS 6 vers la version 1.0.2 [f+].

Problème : Par défaut, seul le déchargement RSA vers le HSM est pris en charge.

- Impact : Pour optimiser les performances, le kit SDK n'est pas configuré pour décharger des fonctions supplémentaires, telles que la génération de nombres aléatoires ou les opérations EC-DH.
- Solution : Veuillez nous contacter au moyen d'une demande de support si vous avez besoin de décharger des opérations supplémentaires.

- Statut de résolution : Nous ajoutons actuellement la prise en charge de la configuration des options de déchargement pour le kit SDK au moyen d'un fichier de configuration. Une fois disponible, la mise à jour sera annoncée sur la page de l'historique des versions.

Problème : Le chiffrement et le déchiffrement RSA avec remplissage OAEP en utilisant une clé sur le HSM ne sont pas pris en charge.

- Conséquence : tout appel au chiffrement et au déchiffrement RSA avec rembourrage OAEP échoue avec une erreur. divide-by-zero Cela se produit car le moteur dynamique OpenSSL appelle l'opération localement à l'aide du fichier PEM factice au lieu de décharger l'opération vers le HSM.
- Solution : Vous pouvez effectuer cette procédure à l'aide de Bibliothèque PKCS #11 ou de Fournisseur JCE.
- Résolution d'état : Nous sommes en train d'ajouter la prise en charge pour le kit SDK pour décharger correctement cette opération. Une fois disponible, la mise à jour sera annoncée sur la page de l'historique des versions.

Problème : seule la génération de clé privée des clés RSA et ECC est transférée sur le HSM.

Pour tout autre type de clé, le moteur AWS CloudHSM OpenSSL n'est pas utilisé pour le traitement des appels. C'est le moteur dynamique OpenSSL local qui est utilisé à la place. Cela génère une clé localement dans le logiciel.

- Impact : comme le basculement est silencieux, rien n'indique que vous n'avez pas reçu une clé qui a été générée de façon sécurisée sur le HSM. Vous verrez une sortie de trace contenant la chaîne `"..........++++++"` si la clé est générée localement par OpenSSL dans le logiciel. Cette trace est absente lorsque l'opération est déchargée sur le HSM. Comme la clé n'est pas générée ou stockée sur le HSM, elle sera indisponible pour une utilisation future.
- Solution de contournement : utilisez uniquement le moteur OpenSSL pour les Types de clé qu'il prend en charge. Pour tous les autres types de clés, utilisez PKCS #11 ou JCE dans les applications, ou utilisez-le `key_mgmt_util` dans la CLI.

Problème : vous ne pouvez pas installer le moteur dynamique OpenSSL pour le SDK client 3 sur RHEL 8, CentOS 8 ou Ubuntu 18.04 LTS

- Conséquence : par défaut, RHEL 8, CentOS 8 et Ubuntu 18.04 LTS fournissent une version d'OpenSSL qui n'est pas compatible avec OpenSSL Dynamic Engine for Client SDK 3.
- Solution : utilisez une plateforme Linux qui prend en charge le moteur dynamique OpenSSL. Pour plus d'informations sur les plateformes prises en charge, consultez Plateformes prises en charge.
- État de résolution : AWS CloudHSM prend en charge ces plateformes avec OpenSSL Dynamic Engine for Client SDK 5. Pour plus d'informations, consultez Plateformes prises en charge et OpenSSL Dynamic Engine.

Problème : Obsolète SHA-1 Sign and Verify sur RHEL 9 (9.2+)

- Conséquence : l'utilisation du condensé de message SHA-1 à des fins cryptographiques est devenue obsolète dans RHEL 9 (version 9.2+). Par conséquent, les opérations de signature et de vérification avec SHA-1 à l'aide du moteur dynamique OpenSSL échoueront.
- Solution : si votre scénario nécessite l'utilisation de SHA-1 pour signer/vérifier des signatures cryptographiques existantes ou tierces, consultez les notes de version relatives à l'amélioration de la sécurité RHEL : comprendre la dépréciation de SHA-1 sur RHEL 9 (9.2+) et RHEL 9 (9.2+) pour plus de détails.

Problème : AWS CloudHSM OpenSSL Dynamic Engine est incompatible avec le fournisseur FIPS pour OpenSSL v3.x

- Conséquence : vous recevrez un message d'erreur si vous tentez d'utiliser le moteur dynamique AWS CloudHSM OpenSSL alors que le fournisseur FIPS est activé pour les versions 3.x d'OpenSSL.
- Solution : pour utiliser le moteur dynamique AWS CloudHSM OpenSSL avec les versions 3.x d'OpenSSL, assurez-vous que le fournisseur « par défaut » est configuré. Pour en savoir plus sur le fournisseur par défaut, consultez le site Web d'OpenSSL.

Problèmes connus pour les instances Amazon EC2 exécutant Amazon Linux 2

Problème : Amazon Linux 2 version 2018.07 utilise un **`ncurses`** package mis à jour (version 6) qui est actuellement incompatible avec les SDK AWS CloudHSM

L'erreur suivante s'affiche lors de l'exécution de AWS CloudHSMcloudhsm-mgmt_util ou key_mgmt_util :

```
/opt/cloudhsm/bin/cloudhsm_mgmt_util: error while loading shared libraries:
 libncurses.so.5: cannot open shared object file: No such file or directory
```

- Conséquence : les instances exécutées sur Amazon Linux 2 version 2018.07 ne pourront pas utiliser tous les AWS CloudHSM utilitaires.
- Solution : émettez la commande suivante sur vos instances Amazon Linux 2 EC2 pour installer le package `ncurses` pris en charge (version 5) :

  ```
  sudo yum update && yum install ncurses-compat-libs
  ```

- État de résolution : ce problème a été résolu dans la version client 1.1.2 de AWS CloudHSM . Vous devez effectuer une mise à niveau vers ce client afin de bénéficier de la correction.

Problèmes connus pour l'intégration d'applications tierces

Problème : Oracle définit l'attribut PKCS #11 **`CKA_MODIFIABLE`** lors de la génération de la clé principale, mais le SDK client 3 ne le prend pas en charge.

Cette limite est définie dans la bibliothèque PKCS #11. Pour plus d'informations, consultez l'annotation 1 sur les attributs PKCS #11 pris en charge.

- Impact : échec de la création de la clé principale Oracle.
- Solution : définissez la variable d'environnement spéciale `CLOUDHSM_IGNORE_CKA_MODIFIABLE_FALSE` sur TRUE lors de la création d'une clé principale. Cette variable d'environnement n'est nécessaire que pour la génération d'une clé principale. Vous n'avez pas besoin de l'utiliser dans d'autres circonstances. Par exemple, vous utiliseriez cette variable pour la première clé principale que vous créez, puis vous n'utiliserez à nouveau cette

variable d'environnement que si vous souhaitez faire pivoter votre édition de clé principale. Pour plus d'informations, consultez Génération de la clé de chiffrement principale Oracle TDE.

- Statut de la résolution : nous améliorons actuellement le microprogramme HSM pour prendre en charge entièrement l'attribut CKA_MODIFIABLE. Les mises à jour seront annoncées sur le AWS CloudHSM forum et sur la page d'historique des versions

Défaillances de synchronisation des clés du SDK client 3

Dans le SDK client 3, si la synchronisation côté client échoue, faites de AWS CloudHSM votre mieux pour nettoyer toutes les clés indésirables qui ont pu être créées (et qui le sont désormais). Ce processus consiste à retirer immédiatement les éléments de clés indésirables ou à marquer les éléments indésirables pour un retrait ultérieur. Dans les deux cas, la résolution ne nécessite aucune action de votre part. Dans les rares cas où vous AWS CloudHSM ne pouvez pas retirer ou marquer un élément clé indésirable, vous devez supprimer le contenu clé.

Problème : vous tentez de générer, d'importer ou de désencapsuler une clé de jeton et vous constatez des erreurs indiquant un échec de Tombstone.

```
2018-12-24T18:28:54Z liquidSecurity ERR: print_node_ts_status:
[create_object_min_nodes]Key: 264617 failed to tombstone on node:1
```

Cause : AWS CloudHSM échec du retrait et du marquage des éléments clés indésirables.

Résolution : un HSM de votre cluster contient des éléments de clés indésirables qui ne sont pas marqués comme indésirables. Vous devez retirer manuellement les éléments de clé. Pour supprimer manuellement les éléments de clés indésirables, utilisez key_mgmt_util (KMU) ou une API de la bibliothèque PKCS #11 ou du fournisseur JCE. Pour plus d'informations, consultez deleteKey ou SDK clients.

Pour rendre les clés à jeton plus durables, les opérations de création de clés qui n'aboutissent pas au nombre minimum de HSM spécifié dans les paramètres de synchronisation côté client AWS CloudHSM échouent. Pour de plus amples informations, veuillez consulter la Synchronisation des clés dans AWS CloudHSM.

SDK client 3 : vérifier les performances du HSM à l'aide de l'outil pkpspeed

Cette rubrique décrit comment vérifier les performances du HSM avec le SDK client 3.

Pour vérifier les performances des HSM de votre AWS CloudHSM cluster, vous pouvez utiliser l'outil pkpspeed (Linux) ou pkpspeed_blocking (Windows) inclus dans le SDK client 3. L'outil pkpspeed s'exécute dans des conditions idéales et appelle directement le HSM pour exécuter des opérations sans passer par un SDK tel que PKCS11. Nous vous recommandons de tester la charge de votre application de manière indépendante afin de déterminer vos besoins en matière de mise à l'échelle. Nous vous déconseillons d'exécuter les tests suivants : Random (I), ModExp (R) et EC point mul (Y).

Pour plus d'informations sur l'installation du client sur une instance Linux EC2, consultez Installation et configuration du AWS CloudHSM client (Linux). Pour plus d'informations sur l'installation du client sur une instance Windows, consultez Installation et configuration du AWS CloudHSM client (Windows).

Après avoir installé et configuré le AWS CloudHSM client, exécutez la commande suivante pour le démarrer.

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

- Pour le Client Windows version 1.1.2 et ultérieure :

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Pour les clients Windows version 1.1.1 et antérieure :

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe C:
\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Si vous avez déjà installé le logiciel client, vous devrez télécharger et installer la dernière version pour obtenir pkpspeed. L'outil pkpspeed est disponible dans `/opt/cloudhsm/bin/pkpspeed` sous Linux ou dans `C:\Program Files\Amazon\CloudHSM\` sous Windows.

Pour utiliser pkpspeed, exécutez la commande pkpspeed ou pkpspeed_blocking.exe, en spécifiant le nom d'utilisateur et le mot de passe d'un utilisateur de chiffrement sur le HSM. Ensuite, définissez les options à utiliser tout en prenant en compte les recommandations suivantes.

Recommandations de test

- Pour tester les performances des opérations de signature et de vérification RSA, choisissez le chiffrement `RSA_CRT` sous Linux ou l'option B sous Windows. Ne choisissez pas `RSA` (option A sous Windows). Les chiffrements sont équivalents, mais `RSA_CRT` est optimisé pour les performances.

- Commencez par un petit nombre de threads. Pour tester les performances AES, un seul thread suffit généralement pour afficher des performances maximales. Pour tester les performances RSA (RSA_CRT), trois ou quatre threads suffisent généralement.

Options configurables pour l'outil pkpspeed

- Mode FIPS : AWS CloudHSM est toujours en mode FIPS (voir les AWS CloudHSM FAQ pour plus de détails). Cela peut être vérifié en utilisant les outils CLI tels que décrits dans le guide de AWS CloudHSM l'utilisateur et en exécutant la getHSMInfo commande qui indiquera l'état du mode FIPS.
- Type de test (bloquant ou non bloquant) : indique comment les opérations sont effectuées de manière progressive. Vous obtiendrez probablement de meilleurs numéros en utilisant le non-blocage. Cela est dû au fait qu'ils utilisent des threads et la simultanéité.
- Nombre de fils : nombre de fils avec lesquels exécuter le test.
- Durée en secondes d'exécution du test (max = 600) : pkpspeed produit des résultats mesurés en « opérations/seconde » et indique cette valeur pour chaque seconde pendant laquelle le test est exécuté. Par exemple, si le test est exécuté pendant 5 secondes, le résultat peut ressembler aux exemples de valeurs suivants :
  - `OPERATIONS/second 821/1`
  - `OPERATIONS/second 833/1`
  - `OPERATIONS/second 845/1`
  - `OPERATIONS/second 835/1`
  - `OPERATIONS/second 837/1`

Tests pouvant être exécutés avec l'outil pkpspeed

- AES GCM : teste le chiffrement en mode AES GCM.
- Basic 3DES CBC : teste le chiffrement en mode 3DES CBC. Voir la note 1 ci-dessous pour un changement à venir.
- Basic AES : teste le chiffrement AES CBC/ECB.
- Digest : teste le condensé de hachage.
- ECDSA Sign : teste la signature ECDSA.
- ECDSA Verify : teste la vérification ECDSA.

- FIPS Random : teste la génération d'un nombre aléatoire conforme à la norme FIPS (Remarque : cela ne peut être utilisé qu'en mode blocage).
- HMAC : Teste le HMAC.
- Random : Ce test n'est pas pertinent car nous utilisons des HSM FIPS 140-2.
- Comparaison entre RSA non-CRT et RSA_CRT : teste les opérations de signature et de vérification RSA.
- RSA OAEP Enc : teste le chiffrement RSA OAEP.
- RSA OAEP Dec : teste le déchiffrement RSA OAEP.
- RSA private dec non-CRT : teste le chiffrement par clé privée RSA (non optimisé).
- RSA private key dec CRT : teste le chiffrement par clé privée RSA (optimisé).
- RSA PSS Sign : teste la signature RSA PSS.
- RSA PSS Verify : teste la vérification RSA PSS.
- RSA public key enc : teste le chiffrement par clé publique RSA.

Le chiffrement par clé publique RSA, le déchiffrement privé RSA sans CRT et le déchiffrement par clé privée RSA invitent également l'utilisateur à répondre aux questions suivantes :

```
Do you want to use static key [y/n]
```

Si y est saisi, une clé pré-calculée est importée dans le HSM.

Si n est saisi, une nouvelle clé est générée.

[1] Interdit après 2023 pour conformité à la norme FIPS conformément aux directives du NIST. Consultez Conformité à la norme FIPS 140 : mécanisme 2024 rendu obsolète pour plus de détails.

Exemples

Les exemples suivants montrent les options que vous pouvez choisir avec pkpspeed (Linux) ou pkpspeed_blocking (Windows) pour tester les performances des modules HSM par rapport aux opérations RSA et AES.

Example - Utilisation de pkpspeed pour tester les performances RSA

Vous pouvez exécuter cet exemple sous Windows, Linux et les systèmes d'exploitation compatibles.

Linux

Utilisez ces instructions pour Linux et les systèmes d'exploitation compatibles.

```
/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 - Utilisation de pkpspeed pour tester les performances AES

Linux

Utilisez ces instructions pour Linux et les systèmes d'exploitation compatibles.

```
/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/...
```

L'utilisateur du SDK client 5 contient des valeurs incohérentes

La commande `user list` renvoie une liste de tous les utilisateurs et de leurs propriétés dans votre cluster. Si l'une des propriétés d'un utilisateur possède la valeur « incohérente », cet utilisateur n'est pas synchronisé au sein de votre cluster. Cela signifie que l'utilisateur existe avec des propriétés différentes sur les différents HSM du cluster. En fonction de la propriété incohérente, différentes étapes de réparation peuvent être effectuées.

Le tableau suivant indique les étapes permettant de résoudre les incohérences pour un seul utilisateur. Si un même utilisateur présente plusieurs incohérences, résolvez-les en suivant ces étapes de haut en bas. Si plusieurs utilisateurs présentent des incohérences, parcourez cette liste pour chaque utilisateur, en résolvant complètement les incohérences pour cet utilisateur avant de passer au suivant.

> **Note**
>
> Pour effectuer ces étapes, vous devez idéalement être connecté en tant qu'administrateur. Si votre compte administrateur n'est pas cohérent, suivez ces étapes en vous connectant à l'administrateur et en répétant les étapes jusqu'à ce que toutes les propriétés soient cohérentes. Une fois que votre compte administrateur est cohérent, vous pouvez continuer à utiliser cet administrateur pour synchroniser les autres utilisateurs du cluster.

Propriété incohérente	Exemple de sortie d'une liste d'utilisateurs	Implication	Méthode de récupération
Le « rôle » de l'utilisateur est « incohérent »	`{ "username": "test_user", "role": "inconsistent ", "locked": "false", "mfa": [],`	Cet utilisateur est un utilisateur CryptoUser sur certains HSM, et un administrateur sur d'autres HSM. Cela peut se produire si deux SDK tentent de créer le même utilisateur, en même temps, avec des rôles	1. Connectez-vous en tant qu'administrateur. 2. Supprimez l'utilisateur sur tous les HSM : user delete --username

Propriété incohérente	Exemple de sortie d'une liste d'utilisateurs	Implication	Méthode de récupération
	"cluster-coverage": "full"	différents. Vous devez supprimer cet utilisateur et le recréer avec le rôle souhaité.	**<user's name>** --role admin user delete --username **<user's name>** --role crypto-user 3. Créez l'utilisateur avec le rôle souhaité : user create --username **<user's name>** --role **<desired role>**

Propriété incohérente	Exemple de sortie d'une liste d'utilisateurs	Implication	Méthode de récupération

La « couverture du cluster » de l'utilisateur est « incohérente »	<pre>{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [],
	"cluster-coverage":
	"**inconsistent** "
	}</pre>
		Cet utilisateur existe sur un sous-ensemble de HSM du cluster. Cela peut se produire en cas de réussite partielle de user create ou de réussite partielle de user delete.
		
		Vous devez terminer votre opération précédente, en créant ou en supprimant cet utilisateur de votre cluster.
			Si l'utilisateur ne doit pas exister, procédez comme suit :
			
			1. Connectez-vous en tant qu'administrateur.
			2. Exécutez cette commande :
			
			user delete --username**<user's name>** --role admin
			3. Maintenant, exécutez la commande suivante :
			
			user delete --username**<user's name>** --role crypto-user
			
			Si l'utilisateur doit exister, procédez comme suit :
			
			1. Connectez-vous en tant qu'administrateur.
			2. Exécutez la commande suivante :

Propriété incohérente	Exemple de sortie d'une liste d'utilisateurs	Implication	Méthode de récupération
			user create --username **<user's name>** --role **<desired role>**

Propriété incohérente	Exemple de sortie d'une liste d'utilisateurs	Implication	Méthode de récupération

Le paramètre « verrouillé » de l'utilisateur est « incohérent » ou « vrai »	<pre>{
	"username": "test_user",
	"role": "crypto-user",
	"locked": **inconsistent** ,
	
	"mfa": [],
	"cluster-coverage": "full"
	}</pre>
		Cet utilisateur est bloqué sur un sous-ensemble de HSM.
		
		Cela peut se produire si un utilisateur utilise le mauvais mot de passe et se connecte uniquement à un sous-ensemble de HSM du cluster.
		
		Vous devez modifier les informations d'identification de l'utilisateur pour garantir la cohérence au sein du cluster.
			Si l'utilisateur a activé la MFA, procédez comme suit :
			
			1. Connectez-vous en tant qu'administrateur.
			2. Exécutez la commande suivante pour désactiver temporairement la MFA :
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			
			3. Modifiez le mot de passe de l'utilisateur afin qu'il puisse se connecter à tous les HSM :
			
			user change-password --username **<user's name>** --role **<desired role>**

Propriété incohérente	Exemple de sortie d'une liste d'utilisateurs	Implication	Méthode de récupération
			Si la MFA doit être activée pour l'utilisateur, procédez comme suit : 1. Demandez à l'utilisateur de se connecter et de réactiver la MFA (cela l'obligera à signer des jetons et à fournir sa clé publique dans un fichier PEM) : user change-mfa token-sign --username **<user's name>** --role **<desired role>** —token <File>

Propriété incohérente	Exemple de sortie d'une liste d'utilisateurs	Implication	Méthode de récupération

Le statut de la MFA est « incohérent »	{
	"username":
	"test_user",
	
	"role": "crypto-user",
	"locked":
	"false",
	"mfa": [
	{
	"strategy":
	"token-sign",
	"status":
	"**inconsistent** "
	}
	],
	"cluster-coverage":
	"full"
	}
		Cet utilisateur possède différents indicateurs MFA sur les différents HSM du cluster.
		
		Cela peut se produire si une opération MFA ne s'est terminée que sur un sous-ensemble de HSM.
		
		Vous devez réinitialiser le mot de passe de l'utilisateur et l'autoriser à réactiver la MFA.
			Si l'utilisateur a activé la MFA, procédez comme suit :
			
			1. Connectez-vous en tant qu'administrateur.
			2. Exécutez la commande suivante pour désactiver temporairement la MFA :
			
			user change-mfa token-sign --username **<user's name>** --role **<desired role>** --disable
			3. Vous devrez également modifier le mot de passe de l'utilisateur afin qu'il puisse se connecter à tous les HSM :
			
			user change-password --username **<user's name>** --role **<desired role>**

Propriété incohérente	Exemple de sortie d'une liste d'utilisateurs	Implication	Méthode de récupération
			Si la MFA doit être activée pour l'utilisateur, procédez comme suit : 1. Demandez à l'utilisateur de se connecter et de réactiver la MFA (cela l'obligera à signer des jetons et à fournir sa clé publique dans un fichier PEM) : user change-mfa token-sign --username **<user's name>** --role **<desired role>** —token <File>

Erreur détectée lors de la vérification de la disponibilité des clés

Problème : un HSM renvoie le message d'erreur suivant :

```
Key <KEY HANDLE> does not meet the availability requirements - The key must be
 available on at least 2 HSMs before being used.
```

Cause : Les contrôles de disponibilité des clés visent à détecter les clés qui, dans de rares cas, pourraient être perdues. Cette erreur se produit généralement dans les clusters avec un seul HSM ou dans les clusters avec deux HSM pendant une période au cours de laquelle l'un d'entre eux

est remplacé. Dans ces situations, les opérations client suivantes sont probablement à l'origine de l'erreur ci-dessus :

- Une nouvelle clé a été générée à l'aide d'une commande telle que Clé generate-symmetric ouclé generate-asymmetric-pair .
- Une opération liste des clés a été lancée.
- Une nouvelle instance du SDK a été démarrée.

> **Note**
>
> OpenSSL crée fréquemment de nouvelles instances du SDK.

Résolution/recommandation : Choisissez l'une des actions suivantes pour éviter que cette erreur ne se produise :

- Utilisez le paramètre --disable-key-availability-check pour définir la disponibilité des clés sur false dans le fichier de configuration de votre outil de configuration. Pour plus d'informations, consultez la section Paramètres de l'outil de configuration.
- Si vous utilisez un cluster avec deux HSM, évitez d'utiliser les opérations à l'origine de l'erreur, sauf pendant le code d'initialisation.
- Augmentez le nombre de HSM dans votre cluster à au moins trois.

Extraction de clés à l'aide de JCE

GetEncoded ou GetS getPrivateExponent renvoie la valeur nulle

`getEncoded`, `getPrivateExponent` et `getS` renverront une valeur nulle car ils sont désactivés par défaut. Pour les activer, reportez-vous à Extraction de clés à l'aide de JCE.

Si `getEncoded`, `getPrivateExponent` et `getS` renvoient ne valeur nulle après avoir été activés, votre clé ne répond pas aux bons prérequis. Pour plus d'informations, consultez Extraction de clés à l'aide de JCE.

GetEncoded ou GETS renvoient des octets clés en dehors du HSM getPrivateExponent

Vous ou une personne ayant accès à votre système avez activé l'extraction claire des clés. Consultez les pages suivantes pour plus d'informations, notamment pour savoir comment rétablir l'état désactivé par défaut de cette configuration.

- Extraction de clés à l'aide de JCE
- Protection et extraction des clés d'un HSM

Limitation du HSM

Lorsque votre charge de travail dépasse la capacité HSM de votre cluster, vous recevez des messages d'erreur indiquant que les HSM sont occupés ou limités. Dans ce cas, vous pouvez constater une réduction du débit ou une augmentation du taux de rejet des demandes de la part des HSM. En outre, les HSM peuvent envoyer les erreurs d'occupation suivantes.

Pour le SDK client 5

- Dans PKCS11, les erreurs d'occupation correspondent à `CKR_FUNCTION_FAILED`. Cette erreur peut se produire pour plusieurs raisons, mais si la limitation HSM est à l'origine de cette erreur, les lignes de journal suivantes apparaîtront dans votre journal :
  - `[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`
- Dans JCE, les erreurs d'occupation correspondent à `com.amazonaws.cloudhsm.jce.jni.exception.InternalException: Unexpected error with the Provider: The HSM could not queue the request for processing.`
- Les erreurs d'occupation des autres SDK impriment le message suivant : `Received error response code from Server. Response Code: 187`.

Pour le SDK client 3

- Dans PKCS11, les erreurs d'occupation sont associées à des erreurs `CKR_OPERATION_ACTIVE`.
- Dans JCE, les erreurs d'occupation correspondent à `CFM2Exception` avec l'état de `0xBB (187)`. Les applications peuvent utiliser la fonction `getStatus()` sur `CFM2Exception` pour vérifier l'état renvoyé par le HSM.
- Les autres erreurs liées à l'état d'occupation du SDK entraîneront l'affichage du message suivant : `HSM Error: HSM is already busy generating the keys(or random bytes) for another request.`

Résolution

Vous pouvez résoudre ces problèmes en prenant une ou plusieurs des mesures suivantes :

- Ajoutez des commandes de nouvelle tentative pour les opérations HSM rejetées dans votre couche d'application. Avant d'activer les commandes de nouvelle tentative, assurez-vous que votre cluster est correctement dimensionné pour répondre aux pics de charge.

  > **Note**
  >
  > Pour le SDK client 5.8.0 et versions ultérieures, les commandes de nouvelle tentative sont activées par défaut. Pour plus de détails sur la configuration de la commande de nouvelle tentative de chaque SDK, reportez-vous à Configurations avancées pour l'outil de configuration du SDK client 5.

- Ajoutez d'autres HSM à votre cluster en suivant les instructions de Ajouter ou supprimer des HSM dans un cluster AWS CloudHSM.

  > **Important**
  >
  > Nous vous recommandons de tester la charge de votre cluster pour déterminer le pic de charge auquel vous devez vous attendre, puis d'y ajouter un ou plusieurs HSM supplémentaires pour garantir une haute disponibilité.

Conserver la synchronisation des utilisateurs HSM sur tous les HSM du cluster

Pour gérer les utilisateurs de votre HSM, vous utilisez un outil de ligne de AWS CloudHSM commande appelé cloudhsm-mgmt_util. Il communique uniquement avec les HSM qui se trouvent dans le fichier de configuration de l'outil. Il ne peut savoir s'il existe d'autres HSM dans le cluster si ceux-ci ne figurent pas dans le fichier de configuration.

AWS CloudHSM synchronise les clés de vos HSM sur tous les autres HSM du cluster, mais ne synchronise pas les utilisateurs ou les politiques du HSM. Lorsque vous utiliser l'outil cloudhsm_mgmt_util pour gérer les utilisateurs HSM, les modifications apportées à ces utilisateurs peuvent affecter uniquement certains des HSM du cluster (ceux qui figurent dans le fichier de configuration cloudhsm_mgmt_util). Cela peut entraîner des problèmes lors de AWS CloudHSM la synchronisation des clés entre les HSM du cluster, car les utilisateurs propriétaires des clés peuvent ne pas exister sur tous les HSM du cluster.

Pour éviter ces problèmes, modifiez le fichier de configuration cloudhsm_mgmt_util avant de gérer les utilisateurs. Pour plus d'informations, voir ???.

Connexion au cluster perdue

Lorsque vous avez configuré le AWS CloudHSM client, vous avez fourni l'adresse IP du premier HSM de votre cluster. Cette adresse IP est enregistrée dans le fichier de configuration du AWS CloudHSM client. Lorsque le client démarre, il essaie de se connecter à cette adresse IP. Si ce n'est pas le cas, par exemple parce que le HSM a échoué ou que vous l'avez supprimé, vous risquez de voir apparaître des erreurs telles que les suivantes :

```
LIQUIDSECURITY: Daemon socket connection error
```

```
LIQUIDSECURITY: Invalid Operation
```

Pour résoudre ces erreurs, mettez à jour le fichier de configuration avec l'adresse IP d'un HSM actif et accessible dans le cluster.

Pour mettre à jour le fichier de configuration du AWS CloudHSM client

1. Utilisez l'une des procédures suivantes pour rechercher l'adresse IP d'un HSM actif dans votre cluster.

- Affichez l'onglet HSM sur la page des détails sur le cluster dans la console AWS CloudHSM.
- Utilisez la AWS Command Line Interface (CLI) pour émettre la describe-clusterscommande.

Cette adresse IP vous sera utile dans une autre étape.

2. Utilisez la commande suivante pour arrêter le client.

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

- Pour le Client Windows version 1.1.2 et ultérieure :

```
C:\Program Files\Amazon\CloudHSM>net.exe stop AWSCloudHSMClient
```

- Pour les clients Windows version 1.1.1 et antérieure :

  Utilisez Ctrl + C dans la fenêtre de commande dans laquelle vous avez démarré le AWS CloudHSM client.

3. Utilisez la commande suivante pour mettre à jour le fichier de configuration du client, en fournissant l'adresse IP que vous avez trouvée los d'une précédente étape.

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

4. Utilisez la commande suivante pour démarrer le client.

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

- Pour le Client Windows version 1.1.2 et ultérieure :

```
C:\Program Files\Amazon\CloudHSM>net.exe start AWSCloudHSMClient
```

- Pour les clients Windows version 1.1.1 et antérieure :

```
C:\Program Files\Amazon\CloudHSM>start "cloudhsm_client" cloudhsm_client.exe
 C:\ProgramData\Amazon\CloudHSM\data\cloudhsm_client.cfg
```

Connexion AWS CloudHSM d'audit manquante CloudWatch

Si vous avez créé un cluster avant le 20 janvier 2018, vous devez configurer manuellement un rôle lié à un service pour activer la livraison des journaux d'audit de ce cluster. Pour savoir comment activer un rôle lié à un service sur un cluster HSM, consultez Présentation des rôles liés à un service, ainsi que Création d'un rôle lié à un service dans le Guide d'utilisateur IAM.

IV personnalisés avec une longueur non conforme pour l'encapsulage de clés AES

Cette rubrique de résolution des problèmes vous aide à déterminer si votre application génère des clés encapsulées irrécupérables. Si ce problème vous concerne, consultez cette rubrique pour le résoudre.

Rubriques

Déterminez si votre code génère des clés encapsulées irrécupérables

Vous êtes concerné uniquement si vous remplissez toutes les conditions ci-dessous :

Condition	Comment le savoir ?
Votre application utilise la bibliothèque PKCS #11	La bibliothèque PKCS #11 est installée sous forme de fichier `libpkcs11.so` dans votre dossier `/opt/cloudhsm/lib` . Les applications écrites en langage C utilisent généralement directement la bibliothèque PKCS #11, tandis que les applications écrites en Java peuvent utiliser la bibliothèque indirectement via une couche d'abstraction Java. Si vous utilisez Windows, vous n'êtes PAS concerné, car la bibliothèque PKCS #11 n'est actuellement pas disponible pour Windows.
Votre application utilise spécifiquement la version 3.0.0 de la bibliothèque PKCS #11	Si vous avez reçu un e-mail de l' AWS CloudHSM équipe, vous utilisez probablement la version 3.0.0 de la bibliothèque PKCS #11. Pour vérifier la version du logiciel sur vos instances d'application, utilisez cette commande :

Condition	Comment le savoir ?
	`rpm -qa | grep ^cloudhsm`
Vous encapsulez les clés à l'aide de l'encapsulage de clés AES	L'encapsulage des touches AES signifie que vous utilisez une clé AES pour encapsuler une autre clé. Le nom du mécanisme correspondant est `CKM_AES_KEY_WRAP` . Il est utilisé avec la fonction `C_WrapKey` . Les autres mécanismes d'encapsulage basés sur AES qui utilisent des vecteurs d'initialisation (IV), tels que `CKM_AES_GCM` et `CKM_CLOUDHSM_AES_GCM` , ne sont pas concernés par ce problème. En savoir plus sur les fonctions et les mécanismes.
Vous spécifiez un IV personnalisé lorsque vous appelez l'encapsulage de clés AES, et la longueur de cet IV est inférieure à 8	L'encapsulage des clés AES est généralement initialisée à l'aide d'une structure `CK_MECHANISM` comme suit : `CK_MECHANISM mech = {CKM_AES_KEY_WRAP, IV_POINTER, IV_LENGTH };` Ce problème s'applique à vous uniquement si : • IV_POINTER n'est pas NULL • IV_LENGTH est inférieur à 8 octets

Si vous ne remplissez pas toutes les conditions ci-dessus, vous pouvez arrêter de lire maintenant. Vos clés encapsulées peuvent être désencapsulées correctement, et ce problème ne vous concerne pas. Sinon, consultez the section called "Mesures à prendre si votre code génère des clés encapsulées irrécupérables".

Mesures à prendre si votre code génère des clés encapsulées irrécupérables

Vous devez suivre les trois étapes suivantes :

1. Mettez immédiatement à niveau votre bibliothèque PKCS #11 vers une version plus récente
   - Dernière bibliothèque PKCS #11 pour Amazon Linux, CentOS 6 et RHEL 6
   - Dernière bibliothèque PKCS #11 pour Amazon Linux 2, CentOS 7 et RHEL 7
   - Dernière bibliothèque PKCS #11 pour Ubuntu 16.04 LTS
2. Mettez à jour votre logiciel pour utiliser un IV conforme aux normes

   Nous vous recommandons vivement de suivre notre exemple de code et de simplement spécifier un IV NULL, ce qui oblige le HSM à utiliser le IV par défaut conforme aux normes. Vous pouvez également spécifier explicitement l'IV comme `0xA6A6A6A6A6A6A6A6` avec une longueur d'IV correspondante de 8. Nous ne recommandons pas d'utiliser un autre IV pour l'encapsulage des clés AES, et nous désactiverons explicitement les IV personnalisés pour l'encapsulage des clés AES dans une future version de la bibliothèque PKCS #11.

   Un exemple de code permettant de spécifier correctement l'IV apparaît dans aes_wrapping.c on. GitHub
3. Identifiez et récupérez les clés encapsulées existantes

   Vous devez identifier toutes les clés que vous avez encapsulées à l'aide de la version 3.0.0 de la bibliothèque PKCS #11, puis contacter le support pour obtenir de l'aide (https://aws.amazon.com/support) pour récupérer ces clés.

> **Important**
>
> Ce problème concerne uniquement les clés encapsulées avec la version 3.0.0 de la bibliothèque PKCS #11. Vous pouvez encapsuler les clés à l'aide de versions antérieures (packages 2.0.4 et numéros inférieurs) ou de versions ultérieures (packages 3.0.1 et numéros supérieurs) de la bibliothèque PKCS #11.

Résolution des échecs de création de cluster

Lorsque vous créez un cluster, AWS CloudHSM crée le rôle AWSServiceRoleForCloudHSM lié au service, s'il n'existe pas déjà. Si vous AWS CloudHSM ne parvenez pas à créer le rôle lié au service, votre tentative de création d'un cluster peut échouer.

Cette rubrique explique comment résoudre les problèmes les plus courants, afin de vous permettre de créer un cluster. Vous devez créer ce rôle une seule fois. Lorsque le rôle lié au service est créé dans votre compte, vous pouvez utiliser l'une des méthodes prises en charge pour créer et gérer des clusters supplémentaires.

Les sections suivantes présentent des suggestions pour résoudre les échecs de création de cluster relatifs au rôle lié au service. Si vous ne parvenez toujours pas à créer un cluster après avoir essayé d'appliquer ces suggestions, contactez AWS Support. Pour plus d'informations sur le rôle AWSServiceRoleForCloudHSM lié à un service, consultez. Rôles liés à un service pour AWS CloudHSM

Rubriques

Ajout de l'autorisation manquante

Pour créer un rôle lié au service, l'utilisateur doit disposer de l'autorisation `iam:CreateServiceLinkedRole`. Si l'utilisateur IAM qui crée le cluster n'a pas cette autorisation, le processus de création du cluster échoue lorsqu'il essaie de créer le rôle lié au service dans votre compte. AWS

Lorsqu'une autorisation manquante est à l'origine de l'échec, le message d'erreur inclut le texte suivant.

```
This operation requires that the caller have permission to call
 iam:CreateServiceLinkedRole to create the CloudHSM Service Linked Role.
```

Pour résoudre cette erreur, donnez à l'utilisateur IAM qui crée le cluster l'autorisation `AdministratorAccess`, ou ajoutez l'autorisation `iam:CreateServiceLinkedRole` à la politique

IAM de l'utilisateur. Pour obtenir des instructions, consultez Ajout d'autorisations à un utilisateur existant ou nouvellement créé.

Ensuite, réessayez de créer le cluster.

Création manuelle du rôle lié à un service

Vous pouvez utiliser la console, la CLI ou l'API IAM pour créer le rôle lié au AWSServiceRoleForCloudHSM service. Pour plus d'informations, consultez Création d'un rôle lié à un service dans le Guide de l'utilisateur IAM.

Faire appel à un utilisateur non fédéré

Les utilisateurs fédérés, dont les informations d'identification proviennent de l'extérieur AWS, peuvent effectuer de nombreuses tâches d'un utilisateur non fédéré. Toutefois, AWS n'autorise pas les utilisateurs à effectuer les appels d'API pour créer un rôle lié au service à partir d'un point de terminaison fédéré.

Pour résoudre ce problème, créez un utilisateur non fédéré avec l'autorisation `iam:CreateServiceLinkedRole`, ou donnez à un utilisateur non fédéré existant l'autorisation `iam:CreateServiceLinkedRole`. Demandez ensuite à cet utilisateur de créer un cluster à partir de la CLI. Cela permet de créer le rôle lié au service dans votre compte.

Lorsque le rôle lié au service est créé, vous pouvez, si vous le souhaitez, supprimer le cluster créé par l'utilisateur non fédéré. La suppression du cluster n'affecte pas le rôle. Par la suite, tout utilisateur disposant des autorisations requises, y compris les utilisateurs fédérés, peut créer des AWS CloudHSM clusters dans votre compte.

Pour vérifier que le rôle a été créé, ouvrez la console IAM à l'adresse https://console.aws.amazon.com/iam/ et choisissez Rôles. Vous pouvez également utiliser la commande IAM get-role dans la CLI.

```
$  aws iam get-role --role-name AWSServiceRoleForCloudHSM
{
    "Role": {
        "Description": "Role for CloudHSM service operations",
        "AssumeRolePolicyDocument": {
            "Version": "2012-10-17",
            "Statement": [
```

```
                {
                    "Action": "sts:AssumeRole",
                    "Effect": "Allow",
                    "Principal": {
                        "Service": "cloudhsm.amazonaws.com"
                    }
                }
            ]
        },
        "RoleId": "AROAJ4I6WN5QVGG5G7CBY",
        "CreateDate": "2017-12-19T20:53:12Z",
        "RoleName": "AWSServiceRoleForCloudHSM",
        "Path": "/aws-service-role/cloudhsm.amazonaws.com/",
        "Arn": "arn:aws:iam::111122223333:role/aws-service-role/cloudhsm.amazonaws.com/
AWSServiceRoleForCloudHSM"
    }
}
```

Récupération des journaux de configuration du client

AWS CloudHSM propose des outils pour le SDK client 3 et le SDK client 5 afin de recueillir des informations sur votre environnement afin que le AWS Support puisse résoudre les problèmes.

Rubriques

- Outil de support du SDK client 5
- Outil de support du SDK client 3

Outil de support du SDK client 5

Le script extrait les informations suivantes :

- Le fichier de configuration pour le composant du SDK client 5
- Fichiers journaux disponibles
- La version actuelle du système d'exploitation
- Informations sur les packages

Exécution de l'outil d'information pour le SDK client 5

Le SDK client 5 inclut un outil de support client pour chaque composant, mais tous les outils fonctionnent de la même manière. L'exécution de l'outil crée un fichier de sortie avec toutes les informations collectées.

Les outils utilisent une syntaxe comme celle-ci :

```
[ pkcs11 | dyn | jce ]_info
```

Par exemple, pour recueillir des informations d'assistance auprès d'un hôte Linux exécutant la bibliothèque PKCS #11 et demander au système d'écrire dans le répertoire par défaut, vous devez exécuter cette commande :

```
/opt/cloudhsm/bin/pkcs11_info
```

L'outil crée le fichier de sortie dans le répertoire `/tmp`.

PKCS #11 library

Pour recueillir des données de support pour la bibliothèque PKCS #11 sous Linux

- Utilisez l'outil d'assistance pour collecter des données.

  ```
  /opt/cloudhsm/bin/pkcs11_info
  ```

Pour recueillir des données de support pour la bibliothèque PKCS #11 sous Windows

- Utilisez l'outil d'assistance pour collecter des données.

  ```
  C:\Program Files\Amazon\CloudHSM\bin\pkcs11_info.exe
  ```

OpenSSL Dynamic Engine

Pour recueillir des données de support pour OpenSSL Dynamic Engine sous Linux

- Utilisez l'outil d'assistance pour collecter des données.

```
/opt/cloudhsm/bin/dyn_info
```

JCE provider

Pour recueillir des données de support pour le fournisseur JCE sous Linux

- Utilisez l'outil d'assistance pour collecter des données.

```
/opt/cloudhsm/bin/jce_info
```

Pour recueillir des données de support pour le fournisseur JCE sous Windows

- Utilisez l'outil d'assistance pour collecter des données.

```
C:\Program Files\Amazon\CloudHSM\bin\jce_info.exe
```

Récupération des journaux depuis un environnement sans serveur

Pour effectuer une configuration pour des environnements sans serveur, tels que Fargate ou Lambda, nous vous recommandons de configurer votre type de journal sur. AWS CloudHSM `term` Une fois configuré pour`term`, l'environnement sans serveur pourra produire vers CloudWatch.

Pour obtenir les journaux des clients CloudWatch, consultez la section Utilisation des groupes de journaux et des flux de CloudWatch journaux dans le guide de l'utilisateur Amazon Logs.

Outil de support du SDK client 3

Le script extrait les informations suivantes :

- Système d'exploitation et sa version actuelle
- Informations de configuration du client à partir des fichiers `cloudhsm_client.cfg`, `cloudhsm_mgmt_util.cfg` et `application.cfg`
- Journaux client à partir de l'emplacement spécifique à la plate-forme
- Informations de cluster et de HSM à l'aide de cloudhsm_mgmt_util
- Informations OpenSSL

- Version actuelle du client et de la génération
- Version du programme d'installation

Exécution de l'outil d'information pour le SDK client 3

Le script crée un fichier de sortie avec toutes les informations collectées. Le script crée le fichier de sortie à l'intérieur du répertoire `/tmp`.

Linux : `/opt/cloudhsm/bin/client_info`

Windows : `C:\Program Files\Amazon\CloudHSM\client_info`

> **Warning**
>
> Ce script présente un problème connu pour les versions 3.1.0 à 3.3.1 du SDK client 3. Nous vous recommandons vivement de passer à la version 3.3.2 qui inclut un correctif pour ce problème. Reportez-vous à la page Problèmes connus pour plus d'informations avant d'utiliser cet outil.

AWS CloudHSM quotas

Les quotas, anciennement appelés limites, sont les valeurs attribuées aux AWS ressources. Les quotas suivants s'appliquent à vos AWS CloudHSM ressources par AWS région et par AWS compte. Le quota par défaut est la valeur initiale appliquée par AWS, et ces valeurs sont répertoriées dans le tableau ci-dessous. Un quota ajustable peut être augmenté au-dessus du quota par défaut.

Quotas de service

Ressource	Quota par défaut	Ajustable?
Clusters	4	Oui
HSM	6	Oui
HSM par cluster	28	Non

La méthode recommandée pour demander une augmentation de quota consiste à ouvrir la console Quotas de service. Dans la console, choisissez votre service et votre quota, puis soumettez votre demande. Pour de plus amples informations, veuillez consulter la documentation sur les quotas de service.

Les quotas du tableau Quotas système suivant ne sont pas ajustables.

Quotas système

Ressource	Quota
Nombre maximum de clés par cluster	3 300
Nombre maximum d'utilisateurs par cluster	1,024
Longueur maximale d'un nom d'utilisateur.	31 caractères
Longueur du mot de passe requis	7 à 32 caractères
Nombre maximal de connexions client simultanées par cluster[1]	900

Ressource	Quota
Nombre maximum de sessions PKCS#11 par application	1,024

[1] Une connexion client pour le SDK client 3 est un démon client. Pour le SDK client 5, une connexion client est une application.

Pour plus d'informations, voir Ressources système.

Ressources système

Les quotas de ressources système sont des quotas relatifs à ce que le AWS CloudHSM client est autorisé à utiliser lors de son exécution.

Les descripteurs de fichier sont un mécanisme du système d'exploitation permettant d'identifier et de gérer les fichiers ouverts par processus.

Le démon client CloudHSM utilise les descripteurs de fichier pour gérer les connexions entre les applications et le client, ainsi qu'entre le client et le serveur.

Par défaut, la configuration du client CloudHSM alloue 3 000 descripteurs de fichier. Cette valeur par défaut est conçue pour générer une capacité de session et de thread optimale entre le démon client et vos HSM.

Dans de rares cas, si vous exécutez votre client dans un environnement à ressources restreintes, il peut s'avérer nécessaire de modifier ces valeurs par défaut.

> **Note**
>
> En modifiant ces valeurs, les performances de votre client CloudHSM peuvent pâtir et/ou votre application peut devenir inutilisable.

1. Modifiez le fichier `/etc/security/limits.d/cloudhsm.conf`.

```
#
```

```
# DO NOT EDIT THIS FILE
#
hsmuser soft nofile 3000
hsmuser hard nofile 3000
```

2. Modifiez les valeurs numériques, si nécessaire.

> **Note**
>
> Le quota `soft` doit être inférieur ou égal au quota `hard`.

3. Redémarrez votre processus de démon client CloudHSM.

> **Note**
>
> Cette option de configuration n'est pas disponible sur les plateformes Microsoft Windows.

Téléchargements pour AWS CloudHSM le SDK client

Téléchargements

En mars 2021, AWS CloudHSM a publié la version 5.0.0 du SDK client, qui introduit un tout nouveau SDK client avec différentes exigences, fonctionnalités et support de plate-forme.

Le SDK client 5 est entièrement compatible avec les environnements de production et offre les mêmes composants et le même niveau de support que le SDK client 3, à l'exception du support pour les fournisseurs CNG et KSP. Pour plus d'informations, consultez Comparaison des composants du SDK client.

> **Note**
>
> Pour plus d'informations sur les plateformes prises en charge par chaque SDK client, reportez-vous à Plateformes prises en charge par le SDK client 5 etPlateformes prises en charge par le SDK client 3.

Dernière parution

Cette section inclut la dernière version du SDK client.

Sortie du SDK client 5 : version 5.12.0

Amazon Linux 2

Téléchargez la version 5.12.0 du logiciel pour Amazon Linux 2 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL Dynamic Engine (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- Fournisseur JCE(SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)

- Interface de ligne de commande CloudHSM (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

Téléchargez la version 5.12.0 du logiciel pour Amazon Linux 2 sur l'architecture ARM64 :

- Bibliothèque PKCS #11 (SHA256 checksum c28a1f27e23e6ab1550dab6a353c6c9338a391a84d57f4ac99a1a3a9810c753f)
- OpenSSL Dynamic Engine (SHA256 checksum 7d2e864c31c13f55443c1b1d04589fbdd4558fe103954de4384691e2c429a872)
- Fournisseur JCE(SHA256 checksum e9a35eb87b2f257c47fb083d286deb835da45858b2d89759ca7d5bb4ef747b4b)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum 28b6f918912b5c63bf10018824b642a805b309c21947a1d0ebbdc44647e80554)

Amazon Linux 2023

Téléchargez la version 5.12.0 du logiciel pour Amazon Linux 2023 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 02801365cba449c5238a4e5ad3df1ddf7edd00ade976f47e956e885286503f3f)
- OpenSSL Dynamic Engine (SHA256 checksum 0abed69a7c6acaafdaabdcc5fab7d56611ffd94f5480cade6f8beace9aeae056)
- Fournisseur JCE(SHA256 checksum 3d5d9a903d3a216eca40f92dbb0b4030b7a86ad7ceee8d62241c97a6e1881e25)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum f96671d882b862033bba0b3633448dc6a26e45a25063e29b79a5cd4b7fc4945c)

Téléchargez la version 5.12.0 du logiciel pour Amazon Linux 2023 sur l'architecture ARM64 :

- Bibliothèque PKCS #11 (SHA256 checksum 53d05006b46bda8e9c1dd76e8307a780bfe0a67b10a9a87723c97f94e29f5b8e)
- OpenSSL Dynamic Engine (SHA256 checksum ec1cca8e01b3303ff9473eeef6b33dc85b6affac7a47387b098905f9f2fc85ba)

- Fournisseur JCE(SHA256 checksum c828ae56f46233215b9f35798b5859ebdac962af442acbc457081c3baaa44f11)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum ddd5dcd68d01f4fafaf13dc0b4ddcf98e3731ed51bdd51f85535b29353644a9f)

CentOS 7 (7.8+)

Téléchargez la version 5.12.0 du logiciel pour CentOS 7 sur l'architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL Dynamic Engine (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- Fournisseur JCE(SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 7 (7.8+)

Téléchargez la version 5.12.0 du logiciel pour RHEL 7 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 383baed4a861391eb0923c0d9cf451851c6dd02d7d6a9e9cc3638c60bf300ef2)
- OpenSSL Dynamic Engine (SHA256 checksum f7aba68787a4c975f3e9f4ead28c2c28adc787ca0babebc070a928d226ff330a)
- Fournisseur JCE(SHA256 checksum 1f75f1a5d428b18ce2dc6ce8e17923009895c2545e2d04d76dafd6da914c0b4e)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum 4c27fae1ef5fd1642c04514ec84ad4cab78f59a32eb3fce59b51805c44b25295)

RHEL 8 (8.3+)

Téléchargez la version 5.12.0 du logiciel pour RHEL 8 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 6e51e95122fd0991278888287f0c408808b26fb5f1196c46168477b9090fc478)
- OpenSSL Dynamic Engine (SHA256 checksum 1f1d52ff7af6c537d8cfeb5973c691a9d90a518accd685ff9b66cd78daf98928)
- Fournisseur JCE(SHA256 checksum 156944607de987d6b39bd8a2d21ccd294c01377a9e35f9f15f8b0f4c8bb90033)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum 351e802f79dd2d0b5f7d23bb74c146be05e5169b603c9aace24189094a45a35d)

RHEL 9 (9.2+)

Téléchargez la version 5.12.0 du logiciel pour RHEL 9 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum d1b2f4ac7e6e0c18e788512e7726bc68b571d99a1442ce2f2e80f4b0f9956266)
- OpenSSL Dynamic Engine (SHA256 checksum cf86a3f17cd6c51969d4ce80c1e3ea6513b995611be7e2e72e5e5233c71d6add)
- Fournisseur JCE(SHA256 checksum ae89e256eb89ec6b4fa0f001e7a4e1d8f1c08530423e81aa74d69a17b25d9a99)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum dfe6fe5d890c33b2f5d38f906ade113b06c8c05f3427a327744c454e7302f1a5)

Téléchargez la version 5.12.0 du logiciel pour RHEL 9 sur l'architecture ARM64 :

- Bibliothèque PKCS #11 (SHA256 checksum cad72a6ab2232b4c38b90d7c62147520b975d646773dd90d7be897fa0a537d2d)
- OpenSSL Dynamic Engine (SHA256 checksum ad751f756530a2317c3c64380ea3a07865b13e1874fab0e61ac530b21487c7fb)
- Fournisseur JCE(SHA256 checksum d204e69acfbb90996fb08ae3573607b65630b1124fb379e078c002d55ac07766)

  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum c0f412cc59bafd235e046cdc1a0c5d330f2d72f7d6434672e9522f86bc945090)

Ubuntu 20.04 LTS

Téléchargez la version 5.12.0 du logiciel pour Ubuntu 20.04 LTS sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum d37b1f872eb2b1ab34303d5b8b803daa925902b645c57c6e15a28bb6321e0f42)
- OpenSSL Dynamic Engine (SHA256 checksum cdc6e737652556b57d26d8816b2bc9820128cb3919360660b6f7fe65f9d39e3f)
- Fournisseur JCE(SHA256 checksum f567a08344414a4776e1c5a9715657476925ca32695c4c2dd84a4f3fc5dc1615)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum f2ee5ad01c5018fc3670f602228fd71087228cd3923bf5b9bc73e4d7084dac6c)

Ubuntu 22.04 LTS

Téléchargez la version 5.12.0 du logiciel pour Ubuntu 22.04 LTS sur l'architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 0e78928acd7a1662e4b07b15d5c3ccb88714ff89e47b991c8ab6e4c2229ee5aa)
- OpenSSL Dynamic Engine (SHA256 checksum 4f3168745edc5592234891a7b1d82b179a4947e87c72fade1be3bad58b7ed1a3)
- Fournisseur JCE(SHA256 checksum d4c3655cdc2b00d1ab5ceafac94dfbc5c5244ed20e10fdd9db9f4e741e013733)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum d00bbacb6f2e57bd92d832a2bd11cadede972f8e82cc402ec0684b9c6b23123c)

Téléchargez la version 5.12.0 du logiciel pour Ubuntu 22.04 LTS sur l'architecture ARM64 :

- Bibliothèque PKCS #11 (SHA256 checksum 0c1121535c523acb864215338292bab32acee438357878b5fc0b6d268713b86f)
- OpenSSL Dynamic Engine (SHA256 checksum dc7a219302021570bc8c36674d2bd33165557bb2f9a0af8fdf114f1b85a70d84)
- Fournisseur JCE(SHA256 checksum af3834a10081f1e4e7894275c8b9c7b7649b8de3b6f0aeb0781a3358183a9046)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum baa253ac62c2fbcc5712561e0fb0feb25461efc3ce68cf86d4c7bf0af0f14a34)

Windows Server 2016

Téléchargez la version 5.12.0 du logiciel pour Windows Server 2016 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- Fournisseur JCE(SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

Windows Server 2019

Téléchargez la version 5.12.0 du logiciel pour Windows Server 2019 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 11c3255fcc90b47810cfe4b2f71d56a006d295efccdd90f0d3f2dec5d2bab893)
- Fournisseur JCE(SHA256 checksum 09001458196590f54352c0c8986f442003bfc2db71bac6392ce512899d386806)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 7158bc80e3b5b0915d83c39d4c060060a43a79cc407b1f783383b9e20bc5ff43)
- Interface de ligne de commande CloudHSM (SHA256 checksum b446ad1387fe406dcc0a12b6de86fa98e9db4a18f9829b745efb87750c6e31ea)

Le SDK client 5.12.0 ajoute le support ARM à plusieurs plateformes et améliore les performances de tous les SDK. De nouvelles fonctionnalités ont été ajoutées à la CLI CloudHSM et au fournisseur JCE.

Plateformes prises en charge

- Ajout du support pour Amazon Linux 2023 sur l'architecture ARM64 pour tous les SDK.
- Ajout du support pour Red Hat Enterprise Linux 9 (9.2+) sur architecture ARM64 pour tous les SDK.
- Ajout du support pour Ubuntu 22.04 LTS sur l'architecture ARM64 pour tous les SDK.

CLI CloudHSM

- La commande suivante a été ajoutée :
  - répliquer la clé
- Ajout de la prise en charge de la connexion à plusieurs clusters. Pour plus d'informations, consultez Connexion à plusieurs clusters à l'aide de la CLI.

Fournisseur JCE

- Ajouté `KeyReferenceSpec` pour récupérer des clés à l'aide `KeyStoreWithAttributes` de.
- Ajouté `getKeys` pour récupérer plusieurs clés à la fois en utilisant`KeyStoreWithAttributes`.

Améliorations des performances

- Améliorations des performances du NoPadding fonctionnement AES CBC pour tous les SDK.

Versions précédentes du SDK client

Cette section répertorie les versions précédentes du SDK client.

La version 5.11.0

Amazon Linux 2

Téléchargez la version 5.11.0 du logiciel pour Amazon Linux 2 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL Dynamic Engine (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- Fournisseur JCE(SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs pour AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- Interface de ligne de commande CloudHSM (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

Téléchargez la version 5.11.0 du logiciel pour Amazon Linux 2 sur l'architecture ARM64 :

- Bibliothèque PKCS #11 (SHA256 checksum 5ac16449ec149c9b5e7776865803245ab17d0f1ad56df80173840c5e8d257b19)
- OpenSSL Dynamic Engine (SHA256 checksum 28c2eb7f3f60172b0186e5c25f71bb7341537058a71f288673936766048083c1)
- Fournisseur JCE(SHA256 checksum 06c9d9d281c12b1d2bd9a7b601d6317e46cedf175706bbfa3e4dcaed6ba05448)
  - Javadocs pour AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- Interface de ligne de commande CloudHSM (SHA256 checksum 218982bb17aa751969a7866b0a9ff27e7aa5007a07817627d9cc1f7d60a78160)

Amazon Linux 2023

Téléchargez la version 5.11.0 du logiciel pour Amazon Linux 2023 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 55310ab333d18bcfabdc4b74115b040386b4508934bdff93e1d054c4c4a6f9ea)
- OpenSSL Dynamic Engine (SHA256 checksum f3d4934dc872a9b5212a180b9814ca2af3eca01ee228a8725563f1770add0dce)
- Fournisseur JCE(SHA256 checksum 757d3abb515aeb08f4b1c83970ee0979399efee00ee78c9a9dbec05f4ed9768d)
  - Javadocs pour AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- Interface de ligne de commande CloudHSM (SHA256 checksum 22af8f0501ff9a45a9e0683a408a63771c2c06c66abf5478d310d6d32e013555)

CentOS 7 (7.8+)

Téléchargez la version 5.11.0 du logiciel pour CentOS 7 sur l'architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL Dynamic Engine (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- Fournisseur JCE(SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs pour AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- Interface de ligne de commande CloudHSM (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 7 (7.8+)

Téléchargez la version 5.11.0 du logiciel pour RHEL 7 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 9fc0cd7cf003a7cb7e42dbd19671d58a97fc3b3d871d284dc6ae7fd226598772)
- OpenSSL Dynamic Engine (SHA256 checksum 1df6669c971440d446890b0fbeb74125a423df7b14e7ac4577347be7ef176572)
- Fournisseur JCE(SHA256 checksum 148a3f1de55a68e3bb525fb2994645333a52c2e9e46946dd8d90fcbc90ab64fd)
  - Javadocs pour AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- Interface de ligne de commande CloudHSM (SHA256 checksum a68f4a56d4c539cfcc8a1e56e19b5ff385bb24936ea5f349255b4e9bfbee9aab)

RHEL 8 (8.3+)

Téléchargez la version 5.11.0 du logiciel pour RHEL 8 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum b95b9f588656fb14fd08bb66ce0e0da807b96daa38348dec07a508c9bef7403a)
- OpenSSL Dynamic Engine (SHA256 checksum 7bb437b91a52e863b2b00ff7f427ce22522026daf757be873ee031ec6ffffd88)

- Fournisseur JCE(SHA256 checksum e0db887e05eb535314f4d99f21da12d87d35ebb8baf9726f4ce8f01d9df0ea01)
  - Javadocs pour AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- Interface de ligne de commande CloudHSM (SHA256 checksum 8485b5a6d679767ca9b4f611718159a643cf3e85090a8e4d20fe53c3707e25c3)

RHEL 9 (9.2+)

Téléchargez la version 5.11.0 du logiciel pour RHEL 9 sur l'architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 87b56a20accf67df53a203b7f115655b2acfaec4516682d4976d9475b10bec8e)
- OpenSSL Dynamic Engine (SHA256 checksum 83a6b58572e985df937beede4b10e867b0ac6050ace8010dc8d535be365d2747)
- Fournisseur JCE(SHA256 checksum ee95213d02d913250478d0793d6dd578e5c54d765e635c7468a49bdff4c2a6f3)
  - Javadocs pour AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- Interface de ligne de commande CloudHSM (SHA256 checksum 7e168ed3bef8e9c5110645e9960680e9a57f7b94e16aec71422e3c67ebc58fb5)

Ubuntu 20.04 LTS

Téléchargez la version 5.11.0 du logiciel pour Ubuntu 20.04 LTS sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum abc3a339d1fe5850db65620804e9a910f8b4f913624ef9b7189f2f0df1825c01)
- OpenSSL Dynamic Engine (SHA256 checksum 075fc3f9974d552f27ad67fa92c8abff31b756b9add875b8cd4957e6801583a4)
- Fournisseur JCE(SHA256 checksum 5de45c519133a0dae8da3ac01809db7974be25c14c15eb773fc5c972c0178c13)
  - Javadocs pour AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- Interface de ligne de commande CloudHSM (SHA256 checksum 83e0e4505a063792c19feb3d4cfd032b9089091916168d92b0f51a967a007734)

Ubuntu 22.04 LTS

Téléchargez la version 5.11.0 du logiciel pour Ubuntu 22.04 LTS sur l'architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum b8f20be125c8530b2a7bd945956e9c04296fba5634af408b40be4e03bdbad72a)
- OpenSSL Dynamic Engine (SHA256 checksum d728c156eb4ee5c67159e57d6b092785800baa5fb61c14d64f460a8b8f53a778)
- Fournisseur JCE(SHA256 checksum 44e943b8cd1176ad666e249342687744a280c6222df58b5a9f084c932f628284)
  - Javadocs pour AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- Interface de ligne de commande CloudHSM (SHA256 checksum 8ccf5389d459611be813e42d7f9d040090f94f3fe88f9d110bcfb25e9619e4a7)

Windows Server 2016

Téléchargez la version 5.11.0 du logiciel pour Windows Server 2016 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- Fournisseur JCE(SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - Javadocs pour AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)
- Interface de ligne de commande CloudHSM (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

Windows Server 2019

Téléchargez la version 5.11.0 du logiciel pour Windows Server 2019 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum aa4bce5be15bbe0978b7205c619bb91c55a8e0f1f4636be311f24878f7709e07)
- Fournisseur JCE(SHA256 checksum 004cdb9ecb4a4d72458084997de7f562fb76a4e2f0567009f1dfafa7b2bded47)
  - Javadocs pour AWS CloudHSM (SHA256 checksum fb469ae53b516338f3326b402b15b7b84912801a8c25a28cd31a5da0631cd3c5)

- Interface de ligne de commande CloudHSM (SHA256 checksum 679795db759fda4823232142297a281e21a7d6f32cb5ddd6ac4c479866fa33b7)

Le SDK client 5.11.0 ajoute de nouvelles fonctionnalités, améliore la stabilité et inclut des corrections de bogues pour tous les SDK.

Plateformes prises en charge

- Ajout du support pour Amazon Linux 2023 et RHEL 9 (9.2+) pour tous les SDK.
- Suppression du support pour Ubuntu 18.04 LTS en raison de sa récente fin de vie.
- Suppression du support pour Amazon Linux en raison de sa récente fin de vie.

CLI CloudHSM

- Les commandes suivantes ont été ajoutées :
  - signe cryptographique
  - vérification cryptographique
  - clé d'importation pem
  - déballage des clés
  - étui pour clés
- key generate-fileprend désormais en charge l'exportation de clés publiques.

OpenSSL Dynamic Engine

- Le moteur dynamique AWS CloudHSM OpenSSL est désormais pris en charge sur les plateformes installées avec une version de bibliothèque OpenSSL 3.x. Cela inclut Amazon Linux 2023, RHEL 9 (9.2+) et Ubuntu 22.04.

JCE

- Ajout du support pour JDK 17 et JDK 21.
- Ajout de la prise en charge des clés AES à utiliser pour les opérations HMAC.
- Le nouvel attribut clé a été ajouté`ID`.
- Introduction d'une nouvelle `DataExceptionCause` variante pour l'épuisement des clés :`DataExceptionCause.KEY_EXHAUSTED`.

Correctifs de bogues et améliorations :

- La longueur maximale de l'`label`attribut a été augmentée de 126 à 127 caractères.
- Correction d'un bug qui empêchait le déballage des clés EC à l'aide du RsaOaep mécanisme.
- Résolution d'un problème connu lié à l'opération GetKey dans le fournisseur JCE. Pour plus d'informations, consultez Problème : fuite de mémoire du SDK client 5 avec les opérations GetKey.
- Amélioration de la journalisation de tous les SDK pour les clés Triple DES ayant atteint leur limite maximale de blocs de chiffrement, conformément à la norme FIPS 140-2.
- Ajout de problèmes connus pour le moteur dynamique OpenSSL. Consultez Problèmes connus pour le moteur dynamique OpenSSL pour plus de détails.

La version 5.10.0

Amazon Linux

Téléchargez la version 5.10.0 du logiciel pour Amazon Linux sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum d63adf3e96c19c2d894b2defcbadd916dbb0398993050b1358bd93a36aa5acab)
- OpenSSL Dynamic Engine (SHA256 checksum 4daa3e591ffd5f7ce8ef3759c41deaa38867f5e5d21f15927aea83afb1678ac5)
- Fournisseur JCE(SHA256 checksum 6c1ac94d3080f1c609d9dafbcb14480911beef3a488c4ed6f2b11b377da9b477)
  - Javadocs pour AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- Interface de ligne de commande CloudHSM (SHA256 checksum c12617fcd7990ba53e96f477979b410e3a5f17842ca7a912861b8b820809b5b5)

Amazon Linux 2

Téléchargez la version 5.10.0 du logiciel pour Amazon Linux 2 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL Dynamic Engine (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)

- Fournisseur JCE(SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - Javadocs pour AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- Interface de ligne de commande CloudHSM (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

Téléchargez la version 5.10.0 du logiciel pour Amazon Linux 2 sur une architecture ARM64 :

- Bibliothèque PKCS #11 (SHA256 checksum 5d8dfd835f1ed5a7f5a4fcc8ecf81cfa29883aca7e2985de69b5db723ab663db)
- OpenSSL Dynamic Engine (SHA256 checksum 91fb8efe2646bf0dbd9087554baa09554714e9d56e9bfd5c0dc3023a9f485574)
- Fournisseur JCE(SHA256 checksum 99f6e55c37fdf00085a816d46835aeff54470797b3b71f4d28a70dc79c9caf44)
  - Javadocs pour AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- Interface de ligne de commande CloudHSM (SHA256 checksum 4a88ba9b4cf0dd5573f3dd88ab9dc257e4c486069cb529c5d554979ee2dd83af)

CentOS 7 (7.8+)

Téléchargez la version 5.10.0 du logiciel pour CentOS 7 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL Dynamic Engine (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- Fournisseur JCE(SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - Javadocs pour AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- Interface de ligne de commande CloudHSM (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 7 (7.8+)

Téléchargez la version 5.10.0 du logiciel pour RHEL 7 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum fc47e705e57a0bfd433f7b46c9477a70df5c442a8ad9c2969bcef38e328e4933)
- OpenSSL Dynamic Engine (SHA256 checksum 0aca262df6780995c9b884fcb8765bbd64acaf21b2286ec4d05a9a90edb3d4cb)
- Fournisseur JCE(SHA256 checksum b5be7f73c4bcffc5da6f89f324e6b3db5b091610464c8bd38dbddfff0484b2c2)
  - Javadocs pour AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- Interface de ligne de commande CloudHSM (SHA256 checksum e8cf09966890b88a61e695dc034874a445093300359d5d6a86b5a546803920bb)

RHEL 8 (8.3+)

Téléchargez la version 5.10.0 du logiciel pour RHEL 8 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 96afb7042a148ddc7a60ab6235b49e176d0460d1c2957bd76ca3d8406ac1cb03)
- OpenSSL Dynamic Engine (SHA256 checksum 2caad2bffea8aef73c91ad422d09772ef830fe7f80a7be19020e6a107eadfbe8)
- Fournisseur JCE(SHA256 checksum 3543551f08fbe3900821ea2d4ea148b4e86e2334bc94d7ffef6f3b831457cd71)
  - Javadocs pour AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- Interface de ligne de commande CloudHSM (SHA256 checksum 812eccaadfc490f13bcd0b0a835ef58f3a3d4344ad7e0a237de476dd24509525)

Ubuntu 18.04 LTS

Téléchargez la version 5.10.0 du logiciel pour Ubuntu 18.04 LTS sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum be4c61766b8b46e1f6c14c3dcf90aaab9f38240fcd9c68b4009704276c5f6f4a)
- OpenSSL Dynamic Engine (SHA256 checksum 64bd8af827b6dc3786e8ad28858cbc4ef6a0fd42164a0945f427eddcf5f02858)
- Fournisseur JCE(SHA256 checksum 9fcbdf08e93641468588b608173f26f18781bbc029ed95b2e086da29a968cc00)
  - Javadocs pour AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)

- Interface de ligne de commande CloudHSM (SHA256 checksum 13808bddddb7eedeb2b8486d23a9976c7fa8d9220149a6b9400626bcaff3b513)

> **Note**
>
> En raison de la récente fin de vie d'Ubuntu 18.04 LTS, AWS CloudHSM nous ne serons plus en mesure de prendre en charge cette plate-forme dans la prochaine version.

Ubuntu 20.04 LTS

Téléchargez la version 5.10.0 du logiciel pour Ubuntu 20.04 LTS sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 99ae96504580ff85ed4958a582903a847f666bdaafafbe887a5a76db58f24500)
- OpenSSL Dynamic Engine (SHA256 checksum 13e3f6fe086acf9617b163f66e3941f973daa583fb9322d16c396aa29fc3611d)
- Fournisseur JCE(SHA256 checksum 44562cebd9af1aa965840cd9bcb237e518d24c715b3c8bca1405c9c1871835e2)
  - Javadocs pour AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- Interface de ligne de commande CloudHSM (SHA256 checksum ab71b4ec531c5e6d05c91539c7edc1c07e6c748052ebf6200f148cb6812538c5)

Ubuntu 22.04 LTS

Téléchargez la version 5.10.0 du logiciel pour Ubuntu 22.04 LTS sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum ee331a44fbe4936ec98a3ae55d58e67ed38e8bbff0a4f4ce8b1bd8239b75877b)
- Pour l'instant, OpenSSL Dynamic Engine n'est pas disponible pour cette plateforme.
- Fournisseur JCE(SHA256 checksum 9e44d14dd33624f6fe36711633013e47e4a93f4d4635e08900546113ded56e3d)
  - Javadocs pour AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- Interface de ligne de commande CloudHSM (SHA256 checksum 2df361546848cd3f8965b1007dca42a0c959eb10d9e3f4995e8e1c852406751d)

Windows Server 2016

Téléchargez la version 5.10.0 du logiciel pour Windows Server 2016 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- Fournisseur JCE(SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)
  - Javadocs pour AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- Interface de ligne de commande CloudHSM (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

Windows Server 2019

Téléchargez la version 5.10.0 du logiciel pour Windows Server 2019 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 7aae9bfd99a6dd0f4d376c227c206c01847f83a9efd774d1063d76cc6fdaa89f)
- Fournisseur JCE(SHA256 checksum 1c58fd651e51be2ba59051a87aceca0452990b29837b8a7efabcd510ccbf8c1f)
  - Javadocs pour AWS CloudHSM (SHA256 checksum dcbb870c6bd58c6770ba7a2b616c6103a5efb3bdeab831ce8f9c82cc09a9870f)
- Interface de ligne de commande CloudHSM (SHA256 checksum f745a2236c9eb9f6f128313eddc35795bd5e47fdf67332bedeb2554201b61a24)

Le SDK client 5.10.0 améliore la stabilité et inclut des corrections de bogues pour tous les SDK.

Interface de ligne de commande CloudHSM

- Ajout de nouvelles commandes permettant aux clients de gérer les clés à l'aide de l'interface de ligne de commande CloudHSM, notamment :
  - Création de clés symétriques et de paires de clés asymétriques
  - Partage et annulation du partage de clés
  - Liste et filtre des clés à l'aide des attributs clés
  - Affichage des attributs des clés
  - Génération des fichiers de référence clés
  - Suppression des clés

- Amélioration de la journalisation des erreurs
- Ajout du support pour les commandes Unicode multilignes en mode interactif

Correctifs de bogues et améliorations :

- Performances améliorées pour l'importation, le déballage, la dérivation et la création de clés de session pour tous les SDK.
- Correction d'un bogue dans le fournisseur JCE qui empêchait la suppression des fichiers temporaires à la sortie.
- Correction d'un bogue qui provoquait une erreur de connexion dans certaines conditions après le remplacement des HSM du cluster.
- Format de sortie `getVersion` JCE modifié pour gérer les numéros de version mineurs importants et inclure le numéro de correctif.

Plateformes prises en charge

- Ajout du support pour Ubuntu 22.04 avec JCE, PKCS #11 et interface de ligne de commande CloudHSM (la prise en charge pour OpenSSL Dynamic Engine n'est pas encore disponible).

version 5.9.0

Amazon Linux

Téléchargez la version 5.9.0 du logiciel pour Amazon Linux sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 4f368be41f006b751ac41b14e1435c27841f60bbde0f032ec02a359fea637dcf)
- OpenSSL Dynamic Engine (SHA256 checksum 81af0d34683825cd6ff844ccacf9c8f4842a4ba76e3875a89121d09a286b4490)
- Fournisseur JCE(SHA256 checksum e8e5bc09d8e0b3cb24f30ab420fe08902a19073012335ac94382ec55fcc45abd)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- Interface de ligne de commande CloudHSM (SHA256 checksum 17284144b45043204ce012fe8b62b1973f10068950abedbd9c2c6172ed0979c6)

Amazon Linux 2

Téléchargez la version 5.9.0 du logiciel pour Amazon Linux 2 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL Dynamic Engine (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- Fournisseur JCE(SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- Interface de ligne de commande CloudHSM (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

Téléchargez la version 5.9.0 du logiciel pour Amazon Linux 2 sur une architecture ARM64 :

- Bibliothèque PKCS #11 (SHA256 checksum 4337dca5a08c5194b1118fa197bb4a4f7988df4e1b961e6f2e367295ba99d61d)
- OpenSSL Dynamic Engine (SHA256 checksum 4f08689934e877662a7ce64554fb04eb4b2c213b936018609ff187d100e34a85)
- Fournisseur JCE(SHA256 checksum b337b80271a2d308949d5911971fe6ad35df4e34876a481fcac347f1d897fe39)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- Interface de ligne de commande CloudHSM (SHA256 checksum a4d466e6b5f74dcd283ba32c9dd87441941d5e5a05936b7c2b4cc7ef85eb1071)

CentOS 7 (7.8+)

Téléchargez la version 5.9.0 du logiciel pour CentOS 7 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL Dynamic Engine (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- Fournisseur JCE(SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)

  - Javadocs pour AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- Interface de ligne de commande CloudHSM (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 7 (7.8+)

Téléchargez la version 5.9.0 du logiciel pour RHEL 7 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum e5affca37abc4ff76369237649830feb32fccd3fa05199cc2021230137093c56)
- OpenSSL Dynamic Engine (SHA256 checksum 848a2e31550bbc2b0223468877baa2a8cda3131ef8537856b31db226d55c4170)
- Fournisseur JCE(SHA256 checksum 884f483ef3e9c7def92e3ff01b226e5cbf276d96dcb2f6f56009516f19d41dc0)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- Interface de ligne de commande CloudHSM (SHA256 checksum 2e62d5a27cff46d9fb47d656afeccd9dbfb5413bfd2267dd3c8fb7960fef7f26)

RHEL 8 (8.3+)

Téléchargez la version 5.9.0 du logiciel pour RHEL 8 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 081887f6ea1d9df9d1e409b2b5bde83e965c42229acbeb1f950c8fe478361edc)
- OpenSSL Dynamic Engine (SHA256 checksum 6b0500a42fd57c39f076f14e5079f80145b6ebd2c441395761eb04600c07bda5)
- Fournisseur JCE(SHA256 checksum 2bc7ac26b259af92a65fbd5a30d5eb2a92ce0e70efe41feb53bf82f168aa90bb)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- Interface de ligne de commande CloudHSM (SHA256 checksum 79ecbe9b4c5316ccf447d8c59b76b5ac2cc854bd79cd50c1f29197aa8cb080db)

Ubuntu 18.04 LTS

Téléchargez la version 5.9.0 du logiciel pour Ubuntu 18.04 LTS sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum bc6d2227edd7b5a83fed32741fbacbb1756d5df89ebb3435d96f0609a180db65)
- OpenSSL Dynamic Engine (SHA256 checksum 2d6a26434fa6faf337f1dfb42de033220fa405a82d4540e279639a03b3ee6e9d)
- Fournisseur JCE(SHA256 checksum e12aef122f490e9026452ce31c25625b1accb9a5866b3d470488f10f047f1873)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- Interface de ligne de commande CloudHSM (SHA256 checksum f0bcabe594db3e8ff86cc0f65c2a10858d34452eb6b9fc33d7aac05c0f5f4f30)

Ubuntu 20.04 LTS

Téléchargez la version 5.9.0 du logiciel pour Ubuntu 20.04 LTS sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum 15dde8182f432de9e7d369b05e384e1f2d80dcca85db3b16ecc26cdef1a34bb9)
- OpenSSL Dynamic Engine (SHA256 checksum c8ba94a999038af87d4905b7c1feb4cc87e20d1776a32ef6f6d11ee000b5a896)
- Fournisseur JCE(SHA256 checksum de33cd3e8130a06d9da5207079533aac8276a1319ac435a3737b4f65bd8fb972)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- Interface de ligne de commande CloudHSM (SHA256 checksum cfa31535ad9a99a5113496c06fbace38e9593491aca9bb031a18b51075973e68)

Windows Server 2016

Téléchargez la version 5.9.0 du logiciel pour Windows Server 2016 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- Fournisseur JCE(SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- Interface de ligne de commande CloudHSM (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

Windows Server 2019

Téléchargez la version 5.9.0 du logiciel pour Windows Server 2019 sur une architecture x86_64 :

- Bibliothèque PKCS #11 (SHA256 checksum ab5380805b0e17dd89dbbefd3fbda8b54da3c140f82e9f3d021850c31837bbe3)
- Fournisseur JCE(SHA256 checksum f0941d7a20193818133de8a742d3b848ea19abaf25f5a71ac65949ce5a37c533)
  - Javadocs pour AWS CloudHSM (SHA256 checksum 6343427177180c8f61eec0341e827fbba29420ed2033c0e4b4803d49a3df7763)
- Interface de ligne de commande CloudHSM (SHA256 checksum 131530ffe5caff963d483f440d06dcfb41dc11b0f8d78f1dd07bb07f76aeb6d2)

Le SDK client 5.9.0 améliore la stabilité et inclut des corrections de bogues pour tous les SDK. Une optimisation a été réalisée pour tous les SDK afin d'informer les applications d'un échec de fonctionnement immédiatement lorsqu'un HSM est déterminé indisponible. Cette version inclut des améliorations de performances pour JCE.

Fournisseur JCE

- Performances améliorées
- Correction d'un problème connu d'épuisement du pool de sessions

La version 3.4.4

Pour mettre à niveau le SDK client 3 sur les plateformes Linux, vous devez utiliser une commande par lots qui met à niveau le démon client et toutes les bibliothèques en même temps. Pour de plus amples informations sur les mises à niveau, consultez Mise à niveau du SDK client 3.

Pour télécharger le logiciel, choisissez l'onglet correspondant au système d'exploitation de votre choix, puis le lien de chaque package logiciel.

Amazon Linux

Téléchargez la version 3.4.4 du logiciel pour Amazon Linux :

- AWS CloudHSM Cliente (SHA256 checksum 900de424d70f41e661aa636f256a6a79cc43bea6b0fe6eb95c2aaa63e5289505)

- Bibliothèque PKCS #11 (SHA256 checksum a3f93f084d59fee5d7c859292bc02cb7e7f15fb06e971171ebf9b52bbd229c30)
- OpenSSL Dynamic Engine (SHA256 checksum 8db07b9843d49016b0b6fec46d39881d94e426fcaae1cee2747be14af9313bb0)
- Fournisseur JCE (SHA256 checksum 360617c55bf4caa8e6e78ede079ca68cf9ef11473e7918154c22ba908a219843)
- AWS CloudHSM utilitaire de gestion (SHA256 checksum c9961ffe38921131bd6f3702e10d73588e68b8ab10fbb241723e676f4fa8c4fa)

Amazon Linux 2

Téléchargez la version 3.4.4 du logiciel pour Amazon Linux 2 :

- AWS CloudHSM Cliente (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- Bibliothèque PKCS #11 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL Dynamic Engine (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- Fournisseur JCE (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM utilitaire de gestion (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 6

AWS CloudHSM ne prend pas en charge CentOS 6 avec la version 3.4.4 du SDK client.

Utilisez the section called "Version 3.2.1" pour CentOS 6 ou choisissez une plateforme compatible.

CentOS 7 (7.8+)

Téléchargez la version 3.4.4 du logiciel pour CentOS 7 :

- AWS CloudHSM Cliente (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)
- Bibliothèque PKCS #11 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)

- OpenSSL Dynamic Engine (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- Fournisseur JCE (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM utilitaire de gestion (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

CentOS 8

Téléchargez la version 3.4.4 du logiciel pour CentOS 8 :

- AWS CloudHSM Cliente (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)
- Bibliothèque PKCS #11 (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)
- Fournisseur JCE(SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM utilitaire de gestion (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

> **Note**
>
> En raison de la récente fin de vie de CentOS 8, nous ne serons plus en mesure de prendre en charge cette plate-forme dans la prochaine version.

RHEL 6

AWS CloudHSM ne prend pas en charge RedHat Enterprise Linux 6 avec la version 3.4.4 du SDK client.

the section called "Version 3.2.1"Utilisez-le pour RedHat Enterprise Linux 6 ou choisissez une plate-forme prise en charge.

RHEL 7 (7.8+)

Téléchargez la version 3.4.4 du logiciel pour RedHat Enterprise Linux 7 :

- AWS CloudHSM Cliente (SHA256 checksum 7d61d835ae38c6ce121d102b516527f342a76ac31733768097d5cab8bc482610)

- Bibliothèque PKCS #11 (SHA256 checksum 2099f324ff625e1a46d96c1d5084263ca1d650424d7465ead43fe767d6687f36)
- OpenSSL Dynamic Engine (SHA256 checksum 6d8e81ad1208652904fe4b6abc4f174e866303f2302a6551c3fbef617337e663)
- Fournisseur JCE (SHA256 checksum 70e3cdce143c45a76e155ffb5969841e0153e011f59eb9f2c6e6be0707030abf)
- AWS CloudHSM utilitaire de gestion (SHA256 checksum 5a702fe5e50dc6055daa723df71a0874317c9ff5844eea30104587a61097ecf4)

RHEL 8 (8.3+)

Téléchargez la version 3.4.4 du logiciel pour RedHat Enterprise Linux 8 :

- AWS CloudHSM Cliente (SHA256 checksum 81639c9ec83e501709c4117ba9d98b23dea7838a206ed244c9c6cc0d65130f8c)
- Bibliothèque PKCS #11 (SHA256 checksum 9a15daa87b8616cf03a6bf6b375f53451ef448dbc54bf2c27fbc2be7823fc633)
- Fournisseur JCE(SHA256 checksum 2b1c4208992903cf7bcc669c1392c59a64fbfc82e010c626ffa58d0cb8e9126b)
- AWS CloudHSM utilitaire de gestion (SHA256 checksum 3adbecc802e0854c23aa4b8d80540d1748903c8dba93b6c8042fb7885051c360)

Ubuntu 16.04 LTS

Téléchargez la version 3.4.4 du logiciel pour Ubuntu 16.04 LTS :

- AWS CloudHSM Cliente (SHA256 checksum 317c92c2e0b5d60afab1beb947f053d13ddaacb994cccc2c2b898e997ece29b9)
- Bibliothèque PKCS #11 (SHA256 checksum 91451c420c51488a022569fd32f052a3b988a2883ea4c2ac952acb61a2fea37c)
- OpenSSL Dynamic Engine (SHA256 checksum 4098771ad0e38df9bf14d50520ca49b9395f819f0387e2bc3b0e61abb5888e66)
- Fournisseur JCE (SHA256 checksum e136ff183271c2f9590a9fccb8261a7eb809506686b070e3854df1b8686c6641)
- AWS CloudHSM utilitaire de gestion (SHA256 checksum cbf24a4032f393a913a9898b1b27036392104e8e05d911cab84049b2bcca2541)

> **Note**
>
> En raison de la fin de vie imminente d'Ubuntu 16.04, nous avons l'intention de supprimer le support de cette plate-forme dans la prochaine version.

Ubuntu 18.04 LTS

Téléchargez la version 3.4.4 du logiciel pour Ubuntu 18.04 LTS :

- AWS CloudHSM Cliente (SHA256 checksum cf57d5e0e95efbf032aac8887aebd59ac8cc80e97c69e7c39fdad40873374fe8)
- Bibliothèque PKCS #11 (SHA256 checksum 428f8bdad7925db5401112f707942ee8f3ca554f4ab53fa92237996e69144d2f)
- Fournisseur JCE(SHA256 checksum 1ff17b8f7688e84f7f0bfc96383564dca598a1cab2f2c52c888d0361682f2b9e)
- AWS CloudHSM utilitaire de gestion (SHA256 checksum afe253046146ed6177c520b681efc680dac1048c4a95b3d8ad0f305e79bbe93e)

Windows Server

AWS CloudHSM prend en charge les versions 64 bits de Windows Server 2012, Windows Server 2012 R2, Windows Server 2016 et Windows Server 2019. Le logiciel client AWS CloudHSM 3.4.4 pour Windows Server inclut les fournisseurs CNG et KSP requis. Pour plus de détails, voir Installer et configurer le AWS CloudHSM client (Windows). Téléchargez la dernière version (3.4.4) du logiciel pour Windows Server :

- AWS CloudHSM pour Windows Server (SHA256 checksum d51a7db588e9121d8f0b0351606bd986e1c4de6547f2c8235200dc8a5ffbe53e)
- AWS CloudHSM utilitaire de gestion (SHA256 checksum 0c12d7da9086735cdf189535937a8e036163009c5018dcaf2ee9cdbb6bd4c06f)

La version 3.4.4 ajoute des mises à jour au fournisseur JCE.

AWS CloudHSM Logiciel client

- Mise à jour de la version à des fins de cohérence

Bibliothèque PKCS #11

- Mise à jour de la version à des fins de cohérence

OpenSSL Dynamic Engine

- Mise à jour de la version à des fins de cohérence

Fournisseur JCE

- Mise à jour de log4j vers la version 2.17.1.

Windows (fournisseurs KSP et CNG)

- Mise à jour de la version à des fins de cohérence

Versions obsolètes

Les versions 5.8.0 et antérieures sont obsolètes. Nous vous déconseillons d'utiliser des versions obsolètes dans les charges de travail de production. Nous ne fournissons pas de mises à jour rétrocompatibles pour les versions obsolètes, et nous n'hébergeons pas de versions obsolètes à télécharger. Si vous subissez un impact sur la production lors de l'utilisation de versions obsolètes, vous devez effectuer une mise à niveau pour obtenir des correctifs logiciels.

Versions obsolètes du SDK client 5

Cette section répertorie les versions obsolètes du SDK client 5.

Version 5.8.0

La version 5.8.0 introduit l'authentification par quorum pour l’interface de ligne de commande CloudHSM, le déchargement SSL/TLS avec JSSE, la prise en charge de plusieurs emplacements pour PKCS #11, la prise en charge de plusieurs clusters/utilisateurs pour JCE, l'extraction de clés avec JCE, la prise en charge de KeyFactory pour JCE, de nouvelles configurations de nouvelle tentative pour les codes de retour non terminaux, et inclut une stabilité améliorée et des corrections de bogues pour tous les SDK.

Bibliothèque PKCS #11

- Ajout de la prise en charge de la configuration à emplacements multiples.

Fournisseur JCE

- Ajout de l'extraction de clés basée sur la configuration.
- Ajout de la prise en charge des configurations multi-clusters et multi-utilisateurs.
- Ajout du support pour le déchargement SSL et TLS avec JSSE.
- Ajout du support de déballage pour NoPadding AES/CBC/.
- Ajout de nouveaux types d'usines clés : SecretKeyFactory et KeyFactory.

Interface de ligne de commande CloudHSM

- Ajout de la prise en charge de l'authentification par quorum

Version 5.7.0

La version 5.7.0 introduit l'interface de ligne de commande CloudHSM et inclut un nouvel algorithme de code d'authentification de message basé sur le chiffrement (CMAC). Cette version ajoute l'architecture ARM sur Amazon Linux 2. Les Javadocs du fournisseur JCE sont désormais disponibles pour AWS CloudHSM.

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.
- Désormais pris en charge sur l'architecture ARM avec Amazon Linux 2.
- Algorithmes
  - CKM_AES_CMAC (signer et vérifier)

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.
- Désormais pris en charge sur l'architecture ARM avec Amazon Linux 2.

Fournisseur JCE

- Correctifs et amélioration de la stabilité.
- Algorithmes
  - AESCMAC

Version 5.6.0

La version 5.6.0 inclut un nouveau support de mécanisme pour la bibliothèque PKCS #11 et le fournisseur JCE. De plus, la version 5.6 prend en charge Ubuntu 20.04.

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.
- Mécanismes
  - CKM_RSA_X_509, pour les modes de chiffrement, de déchiffrement, de signature et de vérification

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.

Fournisseur JCE

- Correctifs et amélioration de la stabilité.
- Chiffrements
  - RSA/ECB/, pour les modes de chiffrement et de NoPadding déchiffrement

Clés prises en charge

- EC avec les courbes secp224r1 et secp521r1

Plateformes prises en charge

- Ajout de la prise en charge d'Ubuntu 20.04

Version 5.5.0

La version 5.5.0 ajoute le support pour l'intégration d'OpenJDK 11, Keytool et Jarsigner, ainsi que des mécanismes supplémentaires au fournisseur JCE. Résout un problème connu concernant une KeyGenerator classe interprétant incorrectement le paramètre de taille de clé en nombre d'octets au lieu de bits.

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.

Fournisseur JCE

- Support pour les utilitaires Keytool et Jarsigner
- Support d'OpenJDK 11 sur toutes les plateformes
- Chiffrements
  - Mode de chiffrement et de déchiffrement AES/CBC/ NoPadding
  - Mode de chiffrement et de déchiffrement AES/ECB/PKCS5Padding
  - Mode de chiffrement et de déchiffrement AES/CTR/ NoPadding
  - Mode d'encapsulation et de déballage AES/GCM/ NoPadding
  - Mode de chiffrement et de déchiffrement DESede/ECB/PKCS5Padding
  - NoPadding Desede/CBC/ Mode de chiffrement et de déchiffrement
  - Mode NoPadding AESWrap/ECB/Wrap et Unwrap
  - Mode d'encapsulage et de désencapsulage AESWrap/ECB/PKCS5Padding
  - Mode ZeroPadding AESWrap/ECB/Wrap et Unwrap
  - Mode d'encapsulage et de désencapsulage RSA/ECB/PKCS1Padding
  - Mode d'encapsulage et de désencapsulage RSA/ECB/OAEPPadding
  - Mode d'encapsulage et de désencapsulage RSA/ECB/OAEPWithSHA-1ANDMGF1Padding
  - Mode d'encapsulage et de désencapsulage RSA/ECB/OAEPWithSHA-224ANDMGF1Padding
  - Mode d'encapsulage et de désencapsulage RSA/ECB/OAEPWithSHA-256ANDMGF1Padding

  - Mode d'encapsulage et de désencapsulage RSA/ECB/OAEPWithSHA-384ANDMGF1Padding
  - Mode d'encapsulage et de désencapsulage RSA/ECB/OAEPWithSHA-512ANDMGF1Padding
  - Mode d'encapsulage et de désencapsulage RSAAESWrap/ECB/OAEPPadding
  - Mode d'encapsulage et de désencapsulage RSAAESWrap/ECB/OAEPWithSHA-1ANDMGF1Padding
  - Mode d'encapsulage et de désencapsulage RSAAESWrap/ECB/OAEPWithSHA-224ANDMGF1Padding
  - Mode d'encapsulage et de désencapsulage RSAAESWrap/ECB/OAEPWithSHA-256ANDMGF1Padding
  - Mode d'encapsulage et de désencapsulage RSAAESWrap/ECB/OAEPWithSHA-384ANDMGF1Padding
  - Mode d'encapsulage et de désencapsulage RSAAESWrap/ECB/OAEPWithSHA-512ANDMGF1Padding
- KeyFactory et SecretKeyFactory
  - RSA – Clés RSA de 2 048 bits à 4 096 bits, par incréments de 256 bits
  - AES – Clés AES de 128, 192 et 256 bits
  - Paires de clés EC pour courbes NIST secp256r1 (P-256), secp384r1 (P-384) et secp256k1
  - DESede (3DES)
  - GenericSecret
  - HMAC — avec prise en charge du hachage SHA1, SHA224, SHA256, SHA384, SHA512
- Signature et vérification
  - RSASSA-PSS
  - SHA1withRSA/PSS
  - SHA224withRSA/PSS
  - SHA256withRSA/PSS
  - SHA384withRSA/PSS
  - SHA512withRSA/PSS
  - SHA1withRSAandMGF1
  - SHA224withRSAandMGF1
  - SHA256withRSAandMGF1
  - SHA384withRSAandMGF1

- SHA512withRSAandMGF1

Version 5.4.2

La version 5.4.2 inclut une stabilité améliorée et des corrections de bogues pour tous les SDK. Il s'agit également de la dernière version de la plateforme CentOS 8. Pour plus d'informations, veuillez consulter le site Web CentOS.

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.

Fournisseur JCE

- Correctifs et amélioration de la stabilité.

Version 5.4.1

La version 5.4.1 résout un problème connu avec la bibliothèque PKCS #11. Il s'agit également de la dernière version de la plateforme CentOS 8. Pour plus d'informations, veuillez consulter le site Web CentOS.

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.

Fournisseur JCE

- Correctifs et amélioration de la stabilité.

Version 5.4.0

La version 5.4.0 ajoute le support initial du fournisseur JCE pour toutes les plateformes. Le fournisseur JCE est compatible avec OpenJDK 8.

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.

Fournisseur JCE

- Types de clé
  - RSA – Clés RSA de 2 048 bits à 4 096 bits, par incréments de 256 bits.
  - AES – Clés AES de 128, 192 et 256 bits.
  - Paires de clés ECC pour courbes NIST secp256r1 (P-256), secp384r1 (P-384) et secp256k1.
  - DESede (3DES)
  - HMAC — avec prise en charge du hachage SHA1, SHA224, SHA256, SHA384, SHA512.
- Chiffrements (chiffrement et déchiffrement uniquement)
  - AES/GCM/ NoPadding
  - AES/BCE/ NoPadding
  - AES/CBC/PKCS5Padding
  - Desède/BCE/ 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
- Récapitulatifs
  - SHA-1
  - SHA-224
  - SHA-256
  - SHA-384
  - SHA-512
- Signature et vérification
  - NONEwithRSA
  - SHA1withRSA
  - SHA224withRSA
  - SHA256withRSA
  - SHA384withRSA
  - SHA512withRSA
  - NONEwithECDSA
  - SHA1withECDSA
  - SHA224withECDSA
  - SHA256withECDSA
  - SHA384withECDSA
  - SHA512withECDSA
- Intégration avec Java KeyStore

Version 5.3.0

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.

OpenSSL Dynamic Engine

- Ajout de la prise en charge de la signature/vérification ECDSA avec les courbes P-256, P-384 et secp256k1.

- Ajoutez le support pour les plateformes : Amazon Linux, Amazon Linux 2, Centos 7.8+, RHEL 7 (7.8+).
- Ajout de la prise en charge pour OpenSSL version 1.0.2
- Correctifs et amélioration de la stabilité.

Fournisseur JCE

- Types de clé
  - RSA – Clés RSA de 2 048 bits à 4 096 bits, par incréments de 256 bits.
  - AES – Clés AES de 128, 192 et 256 bits.
  - Paires de clés EC pour courbes NIST secp256r1 (P-256), secp384r1 (P-384) et secp256k1.
  - DESede (3DES)
  - HMAC — avec prise en charge du hachage SHA1, SHA224, SHA256, SHA384, SHA512.
- Chiffrements (chiffrement et déchiffrement uniquement)
  - AES/GCM/ NoPadding
  - AES/BCE/ NoPadding
  - AES/CBC/PKCS5Padding
  - Desède/BCE/ 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
- Récapitulatifs
  - SHA-1
  - SHA-224
  - SHA-256

  - SHA-384
  - SHA-512
- Signature et vérification
  - NONEwithRSA
  - SHA1withRSA
  - SHA224withRSA
  - SHA256withRSA
  - SHA384withRSA
  - SHA512withRSA
  - NONEwithECDSA
  - SHA1withECDSA
  - SHA224withECDSA
  - SHA256withECDSA
  - SHA384withECDSA
  - SHA512withECDSA
- Intégration avec Java KeyStore

Version 5.2.1

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.

Version 5.2.0

La version 5.2.0 ajoute la prise en charge de Types de clé et de mécanismes supplémentaires à la bibliothèque PKCS #11.

Bibliothèque PKCS #11

Types de clé

- ECDSA — Courbes P-224, P-256, P-384, P-521 et secp256k1
- Triple DES (3DES)

Mécanismes

- 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
- CKM_RSA_PKCS pour chiffrer/déchiffrer

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.

Version 5.1.0

La version 5.1.0 ajoute la prise en charge de mécanismes supplémentaires à la bibliothèque PKCS #11.

Bibliothèque PKCS #11

Mécanismes

- CKM_RSA_PKCS pour l'encapsulage/le désencapsulage
- CKM_RSA_PKCS_PSS
- CKM_SHA1_RSA_PKCS

- CKM_SHA224_RSA_PKCS_PSS
- CKM_SHA256_RSA_PKCS_PSS
- CKM_SHA384_RSA_PKCS_PSS
- CKM_SHA512_RSA_PKCS_PSS
- CKM_AES_ECB
- CKM_AES_CTR
- CKM_AES_CBC
- CKM_AES_CBC_PAD
- CKM_SP800_108_COUNTER_KDF
- CKM_GENERIC_SECRET_KEY_GEN
- CKM_SHA_1_HMAC
- CKM_SHA224_HMAC
- CKM_SHA256_HMAC
- CKM_SHA384_HMAC
- CKM_SHA512_HMAC
- CKM_RSA_PKCS_OAEP encapsulage/désencapsulage uniquement
- CKM_RSA_AES_KEY_WRAP
- CKM_CLOUDHSM_AES_KEY_WRAP_NO_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP_PKCS5_PAD
- CKM_CLOUDHSM_AES_KEY_WRAP_ZERO_PAD

Opérations d'API

- C_ CreateObject
- C_ DeriveKey
- C_ WrapKey
- C_ UnWrapKey

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.

Version 5.0.1

La version 5.0.1 ajoute le support initial pour OpenSSL Dynamic Engine.

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.

OpenSSL Dynamic Engine

- Version initiale d'OpenSSL Dynamic Engine.
- Cette version propose une prise en charge initiale des Types de clé et des API OpenSSL :
  - Génération de clés RSA de 2 048, 3 072 et 4 096 bits
  - Les API OpenSSL :
    - Signature RSA à l'aide de RSA PKCS avec SHA1/224/256/384/512 et RSA PSS
    - Génération de clés RSA

  Pour de plus amples informations, veuillez consulter OpenSSL Dynamic Engine.
- Plateformes prises en charge : CentOS 8.3+, Red Hat Enterprise Linux (RHEL) 8.3+, and Ubuntu 18.04 LTS
  - Pré-requis : OpenSSL 1.1.1

  Pour plus d'informations, consultez Plateformes prises en charge.
- Support pour SSL/TLS sur CentOS 8.3+, Red Hat Enterprise Linux (RHEL) 8.3 et Ubuntu 18.04 LTS, y compris NGINX 1.19 (pour certaines suites de chiffrement).

  Pour plus d'informations, consultez la section Utilisation du déchargement SSL/TLS sur Linux.

Version 5.0.0

La version 5.0.0 est la première.

Bibliothèque PKCS #11

- Il s'agit de la version initiale.

Introduction à la prise en charge de la bibliothèque PKCS #11 dans le SDK client version 5.0.0

Cette section décrit la prise en charge des Types de clé, des mécanismes, des opérations d'API et des attributs du SDK client version 5.0.0.

Types de clé :

- AES – Clés AES de 128, 192 et 256 bits
- RSA – Clés RSA de 2 048 bits à 4 096 bits, par incréments de 256 bits

Mécanismes :

- CKM_AES_GCM
- CKM_EC_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

Opérations d'API :

- 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

Attributs :

- GenerateKeyPair
  - Tous les attributs clés RSA
- GenerateKey
  - Tous les attributs clés AES
- GetAttributeValue
  - Tous les attributs clés RSA
  - Tous les attributs clés AES

Exemples :

- Générer des clés (AES, RSA, EC)
- Afficher les attributs des clés
- Chiffrer et déchiffrer les données avec AES GCM
- Signer et vérifier les données avec RSA

Versions obsolètes du SDK client 3

Cette section répertorie les versions obsolètes du SDK client 3.

Version 3.4.3

La version 3.4.3 ajoute des mises à jour au fournisseur JCE.

AWS CloudHSM Logiciel client

- Mise à jour de la version à des fins de cohérence

Bibliothèque PKCS #11

- Mise à jour de la version à des fins de cohérence

OpenSSL Dynamic Engine

- Mise à jour de la version à des fins de cohérence

Fournisseur JCE

- Mise à jour vers la version 2.17.0 de log4j

Windows (fournisseurs KSP et CNG)

- Mise à jour de la version à des fins de cohérence

Version 3.4.2

La version 3.4.2 ajoute des mises à jour au fournisseur JCE.

AWS CloudHSM Logiciel client

- Mise à jour de la version à des fins de cohérence

Bibliothèque PKCS #11

- Mise à jour de la version à des fins de cohérence

OpenSSL Dynamic Engine

- Mise à jour de la version à des fins de cohérence

Fournisseur JCE

- Mise à jour de log4j vers la version 2.16.0.

Windows (fournisseurs KSP et CNG)

- Mise à jour de la version à des fins de cohérence

Version 3.4.1

La version 3.4.1 ajoute des mises à jour au fournisseur JCE.

AWS CloudHSM Logiciel client

- Mise à jour de la version à des fins de cohérence

Bibliothèque PKCS #11

- Mise à jour de la version à des fins de cohérence

OpenSSL Dynamic Engine

- Mise à jour de la version à des fins de cohérence

Fournisseur JCE

- Mise à jour de log4j vers la version 2.15.0.

Windows (fournisseurs KSP et CNG)

- Mise à jour de la version à des fins de cohérence

Version 3.4.0

La version 3.4.0 ajoute des mises à jour à tous les composants.

AWS CloudHSM Logiciel client

- Correctifs et amélioration de la stabilité.

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.

Fournisseur JCE

- Correctifs et amélioration de la stabilité.

Windows (fournisseurs KSP et CNG)

- Correctifs et amélioration de la stabilité.

Version 3.3.2

La version 3.3.2 résout un problème avec le script client_info.

AWS CloudHSM Logiciel client

- Mise à jour de la version à des fins de cohérence

Bibliothèque PKCS #11

- Mise à jour de la version à des fins de cohérence

OpenSSL Dynamic Engine

- Mise à jour de la version à des fins de cohérence

Fournisseur JCE

- Mise à jour de la version à des fins de cohérence

Windows (fournisseurs KSP et CNG)

- Mise à jour de la version à des fins de cohérence

Version 3.3.1

La version 3.3.1 ajoute des mises à jour à tous les composants.

AWS CloudHSM Logiciel client

- Correctifs et amélioration de la stabilité.

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.

Fournisseur JCE

- Correctifs et amélioration de la stabilité.

Windows (fournisseurs KSP et CNG)

- Correctifs et amélioration de la stabilité.

Version 3.3.0

La version 3.3.0 ajoute l'authentification à deux facteurs (2FA) et d'autres améliorations.

AWS CloudHSM Logiciel client

- Ajout de l'authentification 2FA pour les responsables de chiffrement (CO). Pour de plus amples informations, consultez Gestion de l'authentification à deux facteurs pour les responsables de chiffrement.
- Suppression du support de plate-forme pour RedHat Enterprise Linux 6 et CentOS 6. Pour en savoir plus, consultez Prise en charge de Linux.

- Ajout d'une version autonome de CMU à utiliser avec le SDK client 5 ou le SDK client 3. Il s'agit de la même version de CMU incluse dans le démon client de la version 3.3.0. Vous pouvez désormais télécharger le CMU sans télécharger le démon client.

Bibliothèque PKCS #11

- Correctifs et amélioration de la stabilité.
- Suppression du support de plate-forme pour RedHat Enterprise Linux 6 et CentOS 6. Pour en savoir plus, consultez Prise en charge de Linux.

OpenSSL Dynamic Engine

- Mise à jour de la version à des fins de cohérence
- Suppression du support de plate-forme pour RedHat Enterprise Linux 6 et CentOS 6. Pour en savoir plus, consultez Prise en charge de Linux.

Fournisseur JCE

- Correctifs et amélioration de la stabilité.
- Suppression du support de plate-forme pour RedHat Enterprise Linux 6 et CentOS 6. Pour en savoir plus, consultez Prise en charge de Linux.

Windows (fournisseurs KSP et CNG)

- Mise à jour de la version à des fins de cohérence

Version 3.2.1

La version 3.2.1 ajoute une analyse de conformité entre l' AWS CloudHSM implémentation de la bibliothèque PKCS #11 et la norme PKCS #11, les nouvelles plateformes et d'autres améliorations.

AWS CloudHSM Logiciel client

- Ajoutez le support de plateforme pour CentOS 8, RHEL 8 et Ubuntu 18.04 LTS. Pour plus d'informations, consultez ???.

Bibliothèque PKCS #11

- Rapport de conformité de la bibliothèque PKCS #11 pour le SDK client 3.2.1
- Ajoutez le support de plateforme pour CentOS 8, RHEL 8 et Ubuntu 18.04 LTS. Pour plus d'informations, consultez ???.

OpenSSL Dynamic Engine

- Aucun support pour CentOS 8, RHEL 8 et Ubuntu 18.04 LTS. Pour plus d'informations, consultez ???.

Fournisseur JCE

- Ajoutez le support de plateforme pour CentOS 8, RHEL 8 et Ubuntu 18.04 LTS. Pour plus d'informations, consultez ???.

Windows (fournisseurs KSP et CNG)

- Correctifs et amélioration de la stabilité.

Version 3.2.0

La version 3.2.0 ajoute la prise en charge du masquage des mots de passe et d'autres améliorations.

AWS CloudHSM Logiciel client

- Permet de masquer votre mot de passe lorsque vous utilisez des outils de ligne de commande. Pour plus d'informations, consultez loginHSM et logoutHSM (cloudhsm-mgmt_util) et loginHSM et logoutHSM (key_mgmt_util).

Bibliothèque PKCS #11

- Ajoute la prise en charge du hachage de données volumineuses dans le logiciel pour certains mécanismes PKCS #11 qui n'étaient pas pris en charge auparavant. Pour plus d'informations, consultez Versions prises en charge.

OpenSSL Dynamic Engine

- Correctifs et amélioration de la stabilité.

Fournisseur JCE

- Mise à jour de la version à des fins de cohérence

Windows (fournisseurs KSP et CNG)

- Correctifs et amélioration de la stabilité.

Version 3.1.2

La version 3.1.2 ajoute des mises à jour au fournisseur JCE.

AWS CloudHSM Logiciel client

- Mise à jour de la version à des fins de cohérence

Bibliothèque PKCS #11

- Mise à jour de la version à des fins de cohérence

OpenSSL Dynamic Engine

- Mise à jour de la version à des fins de cohérence

Fournisseur JCE

- Mise à jour de log4j vers la version 2.13.3

Windows (fournisseurs KSP et CNG)

- Mise à jour de la version à des fins de cohérence

Version 3.1.1

AWS CloudHSM Logiciel client

- Mise à jour de la version à des fins de cohérence

Bibliothèque PKCS #11

- Mise à jour de la version à des fins de cohérence

OpenSSL Dynamic Engine

- Mise à jour de la version à des fins de cohérence

Fournisseur JCE

- Correctifs de bogues et améliorations de performances

Windows (CNG, KSP)

- Mise à jour de la version à des fins de cohérence

Version 3.1.0

La version 3.1.0 ajoute un encapsulage de clés AES conforme aux normes.

AWS CloudHSM Logiciel client

- Nouvelle exigence pour la mise à niveau : la version de votre client doit correspondre à la version de toutes les bibliothèques logicielles que vous utilisez. Pour mettre à niveau le client, vous devez utiliser une commande par lots qui met à niveau le client et toutes les bibliothèques en même temps. Pour plus d'informations, consultez Mise à niveau du SDK client 3.
- Key_mgmt_util (KMU) inclut les mises à jour suivantes :
  - Ajout de deux nouvelles méthodes d'encapsulage de clés AES : encapsulage de clés AES conforme aux normes avec remplissage à l'aide de zéros et encapsulage de clés AES sans remplissage. Pour de plus amples informations, veuillez consulter wrapKey et unwrapKey.
  - Désactivation de la possibilité de spécifier un vecteur d'initialisation personnalisé lors de l'encapsulage d'une clé à l'aide d'AES_KEY_WRAP_PAD_PKCS5. Pour plus d'informations, consultez Encapsulage des clés AES.

Bibliothèque PKCS #11

- Ajout de deux nouvelles méthodes d'encapsulage de clés AES : encapsulage de clés AES conforme aux normes avec remplissage à l'aide de zéros et encapsulage de clés AES sans remplissage. Pour plus d'informations, consultez Encapsulage des clés AES.
- Vous pouvez configurer la longueur du salt pour les signatures RSA-PSS. Pour savoir comment utiliser cette fonctionnalité, consultez la section Longueur de sel configurable pour les signatures RSA-PSS activées. GitHub

OpenSSL Dynamic Engine

- CHANGEMENT IMPORTANT : les suites de chiffrement TLS 1.0 et 1.2 avec SHA1 ne sont pas disponibles dans OpenSSL Engine 3.1.0. Ce problème sera résolu sous peu.
- Si vous avez l'intention d'installer la bibliothèque OpenSSL Dynamic Engine sur RHEL 6 ou CentOS 6, consultez un problème connu concernant la version OpenSSL par défaut installée sur ces systèmes d'exploitation.
- Correctifs et améliorations de la stabilité

Fournisseur JCE

- CHANGEMENT IMPORTANT : pour résoudre un problème avec la conformité JCE (Java Cryptography Extension), la fonction d'encapsulage et de désencapsulage des clés AES utilise désormais correctement l'algorithme AesWrap au lieu de l'algorithme AES. Cela signifie que `Cipher.WRAP_MODE` et `Cipher.UNWRAP_MODE` ne conviennent plus aux mécanismes AES/BCE et AES/CBC.

  Pour mettre à niveau vers la version 3.1.0 du client, vous devez mettre à jour votre code. Si vous avez déjà des clés encapsulées, vous devez porter une attention particulière au mécanisme que vous utilisez pour désencapsuler et à la façon dont les valeurs IV par défaut ont changé. Si vous avez encapsulé des clés avec la version 3.0.0 ou antérieure du client, vous devez utiliser AESWrap/ECB/PKCS5Padding dans 3.1.1 pour désencapsuler vos clés existantes. Pour plus d'informations, consultez Encapsulage des clés AES.
- Vous pouvez répertorier plusieurs clés avec la même étiquette à partir du fournisseur Java. Pour savoir comment parcourir toutes les touches disponibles, voir Rechercher toutes les touches activées GitHub.

- Vous pouvez définir des valeurs plus restrictives pour les attributs lors de la création des clés, notamment en spécifiant différentes étiquettes pour les clés publiques et privées. Pour plus d'informations, consultez Attributs Java pris en charge.

Windows (CNG, KSP)

- Correctifs et amélioration de la stabilité.

nd-of-life Versions électroniques

AWS CloudHSM annonce la fin de vie des versions qui ne sont plus compatibles avec le service. Pour préserver la sécurité de votre application, nous nous réservons le droit de refuser activement les connexions dès les end-of-life versions.

- Aucune version du SDK client n'est actuellement publiée end-of-life .

Historique du document

Cette rubrique décrit les mises à jour importantes du Guide de l'utilisateur AWS CloudHSM .

Rubriques

Mises à jour récentes

Le tableau suivant décrit les modifications importantes apportées à cette documentation depuis avril 2018. En plus des principales modifications répertoriées ici, nous mettons fréquemment à jour la documentation pour améliorer les descriptions et les exemples, et pour répondre aux commentaires que vous nous envoyez. Pour recevoir une notification concernant les modifications importantes, utilisez le lien figurant dans le coin supérieur droit pour vous abonner aux flux RSS.

Pour plus de détails sur les nouvelles versions, consultez Téléchargements pour AWS CloudHSM le SDK client

Modification	Description	Date
Ajout d'une nouvelle version	A publié la version 5.12.0 du AWS CloudHSM client.	20 mars 2024
Ajout d'une nouvelle version	A publié la version 5.11.0 du AWS CloudHSM client.	17 janvier 2024
Ajout d'une nouvelle version	A publié la version 5.10.0 du AWS CloudHSM client.	28 juillet 2023
Ajout d'une nouvelle version	Sortie de la version AWS CloudHSM client 5.9.0.	23 mai 2023
Ajout d'une nouvelle version	Sortie de la version AWS CloudHSM client 5.8.0.	16 mars 2023

Ajout d'une nouvelle version	A publié la version 5.7.0 du AWS CloudHSM client.	16 novembre 2022
Ajout d'une nouvelle version	A publié la version 5.6.0 du AWS CloudHSM client.	1er septembre 2022
Ajout d'une nouvelle version	A publié la version 5.5.0 du AWS CloudHSM client.	13 mai 2022
Ajout d'une nouvelle version	Sortie de la version AWS CloudHSM client 5.4.2.	18 mars 2022
Ajout d'une nouvelle version	A publié la version 5.4.1 du AWS CloudHSM client.	10 février 2022
Ajout d'une nouvelle version	A publié la version 5.4.0 du fournisseur AWS CloudHSM JCE pour les plateformes Windows.	1er février 2022
Ajout d'une nouvelle version	Sortie de la version AWS CloudHSM client 5.4.0, qui ajoute le support initial du fournisseur JCE pour toutes les plateformes Linux.	28 janvier 2022
Ajout d'une nouvelle version	Sortie de la version AWS CloudHSM client 5.3.0.	3 janvier 2022
Ajout d'une nouvelle version	A publié la version 3.4.4 du AWS CloudHSM client.	3 janvier 2022
Ajout d'une nouvelle version	Sortie de la version AWS CloudHSM client 3.4.3.	20 décembre 2021
Ajout d'une nouvelle version	A publié la version 3.4.2 du AWS CloudHSM client.	15 décembre 2021

Ajout d'une nouvelle version	A publié la version 3.4.1 du AWS CloudHSM client.	10 décembre 2021
Ajout d'une nouvelle version	A publié la version 5.2.1 du AWS CloudHSM client.	4 octobre 2021
Ajout d'une nouvelle version	A publié la version 3.4.0 du AWS CloudHSM client.	25 août 2021
Ajout d'une nouvelle version	A publié la version 5.2.0 du AWS CloudHSM client.	3 août 2021
Ajout d'une nouvelle version	Sortie de la version AWS CloudHSM client 3.3.2.	2 juillet 2021
Ajout d'une nouvelle version	A publié la version 5.1.0 du AWS CloudHSM client.	1er juin 2021
Ajout d'une nouvelle version	A publié la version 3.3.1 du AWS CloudHSM client.	26 avril 2021
Ajout d'une nouvelle version	A publié la version 5.0.1 du AWS CloudHSM client.	08 avril 2021
Ajout d'une nouvelle version	A publié la version 5.0.0 du AWS CloudHSM client.	12 mars 2021
Ajout de nouveau contenu	Ajout d'une interface VPC Endpoint, une fonctionnalité AWS qui vous permet de créer une connexion privée entre votre VPC AWS CloudHSM sans avoir besoin d'un accès via Internet ou via un appareil NAT, une connexion VPN ou une connexion. AWS Direct Connect	10 février 2021

Ajout d'une nouvelle version	Sortie de la version AWS CloudHSM client 3.3.0.	3 février 2021
Ajout de nouveau contenu	Ajout de la conservation gérée des sauvegardes, une fonctionnalité qui supprime automatiquement les anciennes sauvegardes.	18 novembre 2020
Ajout de nouveau contenu	Ajout d'un rapport de conformité qui analyse l'implémentation du SDK AWS CloudHSM client 3.2.1 de la bibliothèque PKCS #11 avec la norme PKCS #11.	29 octobre 2020
Ajout d'une nouvelle version	A publié la version 3.2.1 du AWS CloudHSM client.	8 octobre 2020
Ajout de nouveau contenu	Ajout d'une documentation décrivant les principaux paramètres de synchronisation dans AWS CloudHSM.	1 septembre 2020
Ajout d'une nouvelle version	A publié la version 3.2.0 du AWS CloudHSM client.	31 août 2020
Ajout d'une nouvelle version	A publié la version 3.1.2 du AWS CloudHSM client.	30 juillet 2020
Ajout d'une nouvelle version	A publié la version 3.1.1 du AWS CloudHSM client.	3 juin 2020
Ajout d'une nouvelle version	Sortie de la version AWS CloudHSM client 3.1.0.	21 mai 2020
Ajout d'une nouvelle version	A publié la version 3.0.1 du AWS CloudHSM client.	20 avril 2020

Ajout d'une nouvelle version	Publication de la version AWS CloudHSM client 3.0.0 pour la plate-forme Windows Server.	30 octobre 2019
Ajout d'une nouvelle version	Sortie de la version AWS CloudHSM client 3.0.0 pour toutes les plateformes, à l'exception de Windows.	22 octobre 2019
Ajout d'une nouvelle version	A publié la version 2.0.4 du AWS CloudHSM client.	26 août 2019
Ajout d'une nouvelle version	A publié la version 2.0.3 du AWS CloudHSM client.	13 mai 2019
Ajout d'une nouvelle version	A publié la version 2.0.1 du AWS CloudHSM client.	21 mars 2019
Ajout d'une nouvelle version	A publié la version 2.0.0 du AWS CloudHSM client.	6 février 2019
Prise en charge de régions supplémentaires	Ajout du AWS CloudHSM support pour les régions UE (Stockholm) et AWS GovCloud (US-Est).	19 décembre 2018
Ajout d'une nouvelle version	Sortie de la version 1.1.2 du AWS CloudHSM client pour Windows.	20 novembre 2018
Mise à jour des problèmes connus	Du contenu a été ajouté au guide de dépannage.	le 8 novembre 2018
Ajout d'une nouvelle version	Publication de la version 1.1.2 du AWS CloudHSM client pour les plateformes Linux.	le 8 novembre 2018

Prise en charge de régions supplémentaires	AWS CloudHSM Support supplémentaire pour les régions de l'UE (Paris) et de l'Asie-Pacifique (Séoul).	24 octobre 2018
Ajout de nouveau contenu	Ajout de la possibilité de supprimer et de restaurer AWS CloudHSM les sauvegardes.	10 septembre 2018
Ajout de nouveau contenu	Ajout de la livraison automatique des journaux d'audit à Amazon CloudWatch Logs.	13 août 2018
Ajout de nouveau contenu	Ajout de la possibilité de copier une sauvegarde de AWS CloudHSM cluster entre les régions.	30 juillet 2018
Prise en charge de régions supplémentaires	AWS CloudHSM Support supplémentaire pour la région de l'UE (Londres).	13 juin 2018
Ajout de nouveau contenu	Ajout de la prise en charge des AWS CloudHSM clients et des bibliothèques pour Amazon Linux 2, Red Hat Enterprise Linux (RHEL) 6, Red Hat Enterprise Linux (RHEL) 7, CentOS 6, CentOS 7 et Ubuntu 16.04 LTS.	10 mai 2018
Ajout d'une nouvelle version	Ajout d'un AWS CloudHSM client Windows.	30 avril 2018

Mises à jour antérieures

Le tableau suivant décrit les modifications importantes apportées par rapport à la AWS CloudHSM période antérieure à 2018.

Modification	Description	Date
Nouveau contenu	Ajout de l'authentification par quorum (contrôle d'accès M sur N) pour les responsab les de chiffrement. Pour plus d'informations, consultez Utilisation de l'Utilitaire de gestion CloudHSM (CMU) pour gérer l'authentification par quorum (contrôle d'accès M sur N).	9 novembre 2017
Mettre à jour	Ajout de la documentation sur l'utilisation de l'outil de ligne de commande key_mgmt_ util. Pour plus d'informations, consultez Référence de commande key_mgmt_util.	9 novembre 2017
Nouveau contenu	Ajout d'Oracle Transparent Data Encryption. Pour plus d'informations, consultez Chiffrement des bases de données Oracle.	25 octobre 2017
Nouveau contenu	Ajout de déchargement SSL. Pour plus d'informations, consultez Déchargement SSL/ TLS.	12 octobre 2017

Modification	Description	Date
Nouveau guide	Cette version introduit AWS CloudHSM	14 août 2017

Les traductions sont fournies par des outils de traduction automatique. En cas de conflit entre le contenu d'une traduction et celui de la version originale en anglais, la version anglaise prévaudra.