



Guide du développeur

Amazon Simple Storage Service



Version de l'API 2006-03-01

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Simple Storage Service: Guide du développeur

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Les marques commerciales et la présentation commerciale d'Amazon ne peuvent pas être utilisées en relation avec un produit ou un service extérieur à Amazon, d'une manière susceptible d'entraîner une confusion chez les clients, ou d'une manière qui dénigre ou discrédite Amazon. Toutes les autres marques commerciales qui ne sont pas la propriété d'Amazon sont la propriété de leurs propriétaires respectifs, qui peuvent ou non être affiliés ou connectés à Amazon, ou sponsorisés par Amazon.

Table of Contents

Qu'est-ce qu'Amazon S3 ?	1
Recommandations d'appels d'API	2
Effectuer des appels directs à l'API REST	3
Permissions	3
Développer avec Amazon S3	5
Demandes	5
À propos des clés d'accès	6
Points de terminaison de demande	8
Envoi de demandes via IPv6	8
Faire des demandes à l'aide du AWS Kits SDK	19
Demandes à l'aide de l'API REST	59
À l'aide du AWS CLI	72
En savoir plus sur le AWS CLI	73
Développement à l'aide de AWS SDK	73
Interfaces de programmation SDK	74
Spécification de la version de signature lors de l'authentification des demandes	75
Obtenir les identifiants de demande Amazon S3 pour Support	85
Utilisation de AWS CLI pour obtenir les numéros de demande	86
Utilisation de Windows PowerShell pour obtenir les ID de demande	86
Utilisation AWS CloudTrail événements de données pour obtenir les identifiants de demande	86
Utilisation de la journalisation des accès au serveur S3 pour obtenir des ID de demande	87
Utilisation de HTTP pour obtenir des ID de demande	87
Utilisation d'un navigateur web pour obtenir des ID de demande	87
Utilisation de AWS SDK pour obtenir les ID de demande	88
Opérations d'API S3 au niveau des objets prises en charge pour les tables S3	93
Exemples de code	98
Amazon S3	104
Principes de base	110
Scénarios	812
Exemples sans serveur	1270
Amazon S3 Control	1282
Principes de base	1282
Compartiments de répertoires S3	1363

Principes de base	1365
Scénarios	1491
Demandes d'authentification (AWS Signature (version 4))	1494
Méthodes d'authentification	1495
Présentation des demandes de signature	1496
Utilisation d'un en-tête d'autorisation	1497
Vue d'ensemble	1498
Calcul des signatures : transférez la charge utile en un seul bloc	1502
Calcul des signatures : transférez la charge utile en plusieurs morceaux	1521
Calcul des signatures : y compris les en-têtes de fin	1534
Utilisation des paramètres de requête	1541
Calculer une signature	1544
Un exemple	1549
Exemple 2	1551
Exemples : calculs de signature	1551
Exemples de calcul de signature à l'aide de Java	1552
Exemples de calcul de signature en C#	1553
Authentification des requêtes HTTP POST	1554
Calculer une signature	1557
Clés de politique d'authentification spécifiques à Amazon S3 Signature version 4	1558
Exemples de politiques relatives aux compartiments utilisant des clés de condition associées à Signature Version 4	1561
Browser-Based Téléversements à l'aide de POST	1565
POST Object	1566
Description	1566
Gestion des versions	1567
Requêtes	1567
Exemples	1587
Ressources connexes	1588
POST Object restore	1589
Description	1589
Interroger les archives à l'aide de requêtes sélectionnées	1589
Restauration des archives	1591
Requêtes	1593
Réponses	1608
Exemples	1609

Plus d'informations	1612
Browser-Based Téléversements à l'aide de HTTP POST	1612
Calculer une signature	1614
Création de formulaires HTML	1615
Déclaration de formulaire HTML	1616
Champs de formulaire HTML	1617
Politique POST	1625
Expiration	1626
Correspondance des conditions	1626
Conditions	1628
Echappement de caractère	1633
Exemple de téléchargement POST	1634
Chargement d'un fichier sur Amazon S3 à l'aide du protocole HTTP POST	1634
Browser-based téléchargements à l'aide d'Amplify AWS	1637
Utilisation de AWS JavaScript Bibliothèque Amplify pour charger des fichiers vers Amazon S3	1637
Plus d'informations	1638
En-têtes de requête courants	1639
En-têtes de réponse courants	1644
Réponses d'erreur	1649
Réponses aux erreurs REST	1650
Liste des codes d'erreur	1652
Liste des codes d'erreur relatifs au contenu de l'objet SELECT	1682
Liste des codes Replication-related d'erreur	1696
Liste des codes Tagging-related d'erreur	1699
Liste des codes d'erreur liés aux annotations Amazon S3	1700
Liste des codes d'erreur Amazon S3 on Outposts	1704
Liste des codes d'erreur d'Amazon S3 Storage Lens	1705
Liste des codes d'erreur Amazon S3 Object Lambda	1715
Liste des codes d'erreur asynchrones Amazon S3	1719
Liste des codes d'erreur Amazon S3 Access Grants	1721
Liste des codes FSx-related d'erreur Amazon	1724
Liste des codes d'erreur d'Amazon S3 Tables	1725
Bonnes pratiques concernant les erreurs Amazon S3	1726
Réessayez InternalErrors	1726
Réglez l'application pour éviter les SlowDown erreurs répétées	1727

Isoler les erreurs	1727
AWS Glossaire	1729
Ressources	1730
Annexe	1732
Annexe : SelectObjectContent Réponse	1733
Description	1733
Réponses	1733
Ressources connexes	1743
Annexe : objet OPTIONS	1745
Description	1745
Requêtes	1745
Réponses	1747
Exemples	1748
Ressources connexes	1748
Annexe : API SOAP	1749
Opérations sur le service (API SOAP)	1749
Opérations sur les buckets (API SOAP)	1751
Opérations sur les objets (API SOAP)	1765
Authentification des demandes SOAP	1790
Configuration d'une stratégie d'accès avec SOAP	1792
Éléments communs	1793
Réponses aux erreurs SOAP	1794
Annexe : Demandes d'authentification (AWS signature (version 2))	1796
Authentification des demandes à l'aide de l'API REST (AWS signature (version 2))	1797
Signature et authentification des requêtes REST (AWS signature (version 2))	1800
Browser-based téléchargements à l'aide de POST (AWS signature version 2)	1815
Annexe : API de configuration du cycle de vie (obsolète)	1841
Cycle de vie du bucket PUT (obsolète)	1842
Cycle de vie du bucket GET (obsolète)	1861
Historique du document	1874
.....	mdccclxxv

Qu'est-ce qu'Amazon S3 ?

Amazon Simple Storage Service (Amazon S3) est un service de stockage d'objets qui offre une capacité de mise à l'échelle, une disponibilité des données, une sécurité et des performances de pointe. Les clients de toutes tailles et secteurs peuvent utiliser Amazon S3 pour stocker et protéger toute quantité de données dans un large éventail de cas d'utilisation, par exemple des lacs de données, des sites Web, des applications mobiles, des sauvegardes et restaurations, des archives, des applications métier, des appareils IoT et l'analytique du big data. Amazon S3 offre des fonctions de gestion qui vous permettent d'optimiser, d'organiser et de configurer l'accès à vos données aux fins de répondre aux exigences spécifiques de votre entreprise, de votre organisation et de votre conformité.

Ce guide fournit des informations destinées aux développeurs sur l'utilisation de l'API REST Amazon S3 pour créer des applications qui stockent et récupèrent des données dans le cloud.

Vous pouvez choisir n'importe quelle boîte à outils prenant en charge HTTP pour utiliser l'API REST. Vous pouvez même utiliser un explorateur pour récupérer des objets, si ceux-ci peuvent être lus de manière anonyme.

L'API REST utilise les codes de statut et en-têtes HTTP standard afin de permettre aux explorateurs et boîtes à outils classiques de fonctionner. Dans certains domaines, nous avons ajouté des fonctionnalités au protocole HTTP (par exemple, nous avons ajouté des en-têtes pour permettre le contrôle d'accès). Dans ces cas, nous avons ajouté la nouvelle fonctionnalité d'une manière qui correspond au style d'utilisation standard du protocole HTTP.

Note

La prise en charge de SOAP sur HTTP est obsolète, mais elle est toujours disponible via HTTPS. Cependant, les nouvelles fonctionnalités d'Amazon S3 ne seront pas prises en charge pour SOAP. Nous vous recommandons d'utiliser cette API REST ou les AWS kits SDK disponibles sur le lien suivant :

<https://aws.amazon.com/developer/tools/>

Ce guide du développeur inclut :

- [En-têtes de requête courants](#) et [En-têtes de réponse courants](#)

- [Réponses d'erreur](#)
- [Browser-Based Téléversements à l'aide de POST \(AWS Signature \(version 4\)\)](#)

 Important

Lisez ce qui suit à propos de l'authentification et du contrôle d'accès avant de passer à des rubriques spécifiques relatives à l'API.

Les demandes adressées à Amazon S3 peuvent être authentifiées ou anonymes. L'accès authentifié nécessite des informations d'identification que AWS peut utiliser pour authentifier vos demandes.

Recommandations d'appels d'API

Effectuer des appels d'API REST directement à partir de votre code peut s'avérer fastidieux. Vous devez écrire le code nécessaire pour calculer une signature valide afin d'authentifier vos demandes. Nous recommandons plutôt les alternatives suivantes :

- Utilisez les [AWS kits de développement logiciel \(SDK\)](#) pour envoyer vos demandes.

Consultez également les [exemples de code et les bibliothèques](#).

Si vous utilisez les SDK, vous n'avez pas besoin d'écrire de code pour calculer une signature pour l'authentification des demandes, car les clients du SDK authentifient vos demandes à l'aide des clés d'accès que vous fournissez. À moins d'avoir une bonne raison de ne pas le faire, vous devez toujours utiliser les AWS SDK.

- Utilisez l'AWS interface de ligne de commande pour effectuer des appels d'API Amazon S3. Pour plus d'informations sur la configuration de l'AWS interface de ligne de commande et des exemples de commandes Amazon S3, consultez les rubriques suivantes :

[Configurez l'AWS interface de ligne de commande](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

[Utilisation d'Amazon S3 avec l'interface de ligne de commande AWS](#) du Guide de l'utilisateur de l'interface de ligne de commande AWS.

Effectuer des appels directs à l'API REST

Note

L'en-tête de la demande PUT est limité à une taille de 8 Ko. Dans l'en-tête de la demande PUT, les métadonnées définies par le système sont limitées à une taille de 2 Ko. La taille des métadonnées définies par le système est mesurée en prenant la somme du nombre d'octets dans le US-ASCII codage de chaque clé et valeur.

Si vous souhaitez effectuer vos propres appels d'API REST au lieu d'utiliser l'une des alternatives ci-dessus, il y a certaines choses à garder à l'esprit.

- Pour effectuer des appels d'API REST directs à partir de votre code, créez une signature à l'aide d'informations d'identification valides et incluez la signature dans votre demande. Pour plus d'informations sur les différentes méthodes d'authentification et les calculs de signature, consultez [Demandes d'authentification \(AWS Signature \(version 4\)\)](#).
- L'API REST utilise des en-têtes et des codes d'état HTTP standard, de sorte que les navigateurs et boîtes à outils standard fonctionnent comme prévu. Dans certains domaines, nous avons ajouté des fonctionnalités au protocole HTTP (par exemple, nous avons ajouté des en-têtes pour permettre le contrôle d'accès). Dans ces cas, nous avons ajouté la nouvelle fonctionnalité d'une manière qui correspond au style d'utilisation standard du protocole HTTP. Pour plus d'informations sur la création de demandes, consultez [Demandes](#).

Permissions

Vous pouvez disposer d'informations d'identification valides pour authentifier vos demandes, mais à moins de disposer des autorisations S3 du propriétaire du compte ou du propriétaire du compartiment, vous ne pouvez pas créer de ressources Amazon S3 ni y accéder. Ces autorisations sont généralement accordées par le biais d'une [politique Gestion des identités et des accès AWS \(IAM\)](#), telle qu'une politique de compartiment. Par exemple, vous devez disposer des autorisations nécessaires pour créer un compartiment S3 ou récupérer un objet dans un compartiment. Pour obtenir la liste complète des autorisations S3, consultez la section [Actions, ressources et clés de condition pour Amazon S3](#).

Pour plus d'informations sur les autorisations relatives aux opérations de l'API S3 par type de ressource S3, consultez la section Autorisations [requisies pour les opérations de l'API Amazon S3](#) dans le guide de l'utilisateur Amazon Simple Storage Service.

Si vous utilisez les informations d'identification de l'utilisateur root de votre Compte AWS, vous disposez de toutes les autorisations. Toutefois, il n'est pas recommandé d'utiliser les informations d'identification de l'utilisateur root. Nous vous recommandons plutôt de créer des rôles Gestion des identités et des accès AWS (IAM) dans votre compte et de gérer les autorisations des utilisateurs. Pour plus d'informations, consultez la section Gestion des [accès](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Développer avec Amazon S3

Cette section couvre les rubriques relatives aux développeurs concernant l'utilisation d'Amazon S3. Pour plus d'informations, consultez les rubriques ci-dessous.

Rubriques

- [Demandes](#)
- [Développement avec Amazon S3 à l'aide du AWS CLI](#)
- [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#)
- [Obtenir les ID de demande Amazon S3 pour AWS Support](#)
- [Opérations d'API Amazon S3 au niveau objet prises en charge pour les tables S3](#)

Demandes

Amazon S3 est un service REST. Vous pouvez envoyer des requêtes à Amazon S3 à l'aide de l'API REST ou du AWS SDK (voir [Exemples de code et bibliothèques](#)), bibliothèques wrapper qui encapsulent l'API REST Amazon S3 sous-jacente, simplifiant ainsi vos tâches de programmation.

Chaque interaction avec Amazon S3 est authentifiée ou anonyme. L'authentification est un processus de vérification de l'identité du demandeur essayant d'accéder à un produit Amazon Web Services (AWS). Les demandes authentifiées doivent inclure une valeur de signature qui authentifie l'expéditeur de la demande. La valeur de signature est, en partie, générée à partir des clés d'AWS accès du demandeur (ID de clé d'accès et clé d'accès secrète). Pour plus d'informations sur l'obtention des clés d'accès, consultez [Comment puis-je obtenir les informations d'identification de sécurité ?](#) dans le Références générales AWS.

Si vous utilisez le AWS SDK, les bibliothèques calculent la signature à partir des clés que vous fournissez. Cependant, si vous faites des appels directs d'API REST dans votre application, vous devez écrire le code pour calculer la signature et l'ajouter à la demande.

Rubriques

- [À propos des clés d'accès](#)
- [Points de terminaison de demande](#)
- [Envoi de demandes à Amazon S3 via IPv6](#)

- [Faire des demandes à l'aide du AWS Kits SDK](#)
- [Demandes à l'aide de l'API REST](#)

À propos des clés d'accès

Les sections suivantes examinent les types de clés d'accès que vous pouvez utiliser pour effectuer des demandes authentifiées.

Compte AWS clés d'accès

Les clés d'accès au compte fournissent un accès complet aux AWS ressources détenues par le compte. Voici quelques exemples de clés d'accès :

- ID de clé d'accès (chaîne alphanumérique de 20 caractères). Par exemple :
AKIAIOSFODNN7EXAMPLE
- Clé d'accès secrète (chaîne de 40 caractères). Par exemple : wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY

L'ID de clé d'accès identifie de manière unique un Compte AWS. Vous pouvez utiliser ces clés d'accès pour envoyer des demandes authentifiées à Amazon S3.

Clés d'accès utilisateur IAM

Vous pouvez en créer un Compte AWS pour votre entreprise ; toutefois, plusieurs employés de l'organisation peuvent avoir besoin d'accéder aux AWS ressources de votre organisation. Le partage de vos clés Compte AWS d'accès réduit la sécurité, et il n'est peut-être pas pratique de créer une clé individuelle Comptes AWS pour chaque employé. De plus, vous ne pouvez pas facilement partager des ressources comme les compartiments et les objets car elles appartiennent à différents comptes. Pour partager des ressources, vous devez accorder des autorisations, ce qui représente du travail en plus.

Dans de tels scénarios, vous pouvez utiliser Gestion des identités et des accès AWS (IAM) pour créer des utilisateurs sous votre nom Compte AWS avec leurs propres clés d'accès et associer des politiques utilisateur IAM qui accordent les autorisations d'accès aux ressources appropriées à ces utilisateurs. Pour mieux gérer ces utilisateurs, IAM vous permet de créer des groupes d'utilisateurs et d'accorder des autorisations au niveau du groupe qui s'appliquent à tous les utilisateurs de ce groupe.

Ces utilisateurs sont appelés utilisateurs IAM dans AWS lesquels vous créez et gérez. Le compte parent contrôle la possibilité de l'utilisateur à accéder à AWS. Toute ressource créée par un utilisateur IAM est sous le contrôle du Compte AWS parent et est payée par celui-ci. Ces utilisateurs IAM peuvent envoyer des demandes authentifiées à Amazon S3 en utilisant leurs propres informations d'identification de sécurité. Pour plus d'informations sur la création et la gestion des utilisateurs sous votre nom Compte AWS, consultez la page détaillée [Gestion des identités et des accès AWS du produit](#).

Informations d'identification de sécurité temporaires

Outre la création d'utilisateurs IAM dotés de leurs propres clés d'accès, IAM vous permet également d'accorder des informations d'identification de sécurité temporaires (clés d'accès temporaires et jeton de sécurité) à n'importe quel utilisateur IAM afin qu'il puisse accéder à vos AWS services et ressources. Vous pouvez aussi gérer des utilisateurs dans votre système hors d AWS. Ces dernières sont appelées utilisateurs fédérés. En outre, les utilisateurs peuvent être des applications que vous créez pour accéder à vos AWS ressources.

IAM fournit l' AWS Security Token Service API qui vous permet de demander des informations d'identification de sécurité temporaires. Vous pouvez utiliser l'API AWS STS ou le AWS SDK pour demander ces informations d'identification. L'API retourne des informations d'identification de sécurité temporaires (ID de clé d'accès et clé Secret Access Key), ainsi qu'un jeton de sécurité. Ces informations d'identification sont uniquement valides pour la durée que vous spécifiez lorsque vous les demandez. Vous utilisez l'ID de clé d'accès et la clé secrète de la même façon que vous les utilisez lorsque vous envoyez des demandes en utilisant votre Compte AWS ou les clés d'accès utilisateur IAM. De plus, vous devez inclure le jeton dans chaque demande que vous envoyez à Amazon S3.

Un utilisateur IAM peut demander ces informations d'identification de sécurité temporaires pour sa propre utilisation ou les transmettre à des utilisateurs fédérés ou à des applications. Lors d'une demande d'informations d'identification de sécurité temporaires pour des utilisateurs fédérés, vous devez fournir un nom d'utilisateur et une stratégie IAM qui définit les autorisations que vous souhaitez associer à ces informations. L'utilisateur fédéré ne peut pas obtenir plus d'autorisations que l'utilisateur IAM parent qui a demandé les informations d'identification temporaires.

Vous pouvez utiliser ces informations d'identification de sécurité temporaires pour effectuer des demandes auprès d'Amazon S3. Les bibliothèques d'API calculent la valeur de signature nécessaire en utilisant ces informations d'identification pour authentifier votre demande. Si vous envoyez des demandes en utilisant des informations d'identification expirées, Amazon S3 les rejette.

Pour plus d'informations sur la signature de demandes en utilisant des informations d'identification de sécurité temporaires dans vos demandes d'API REST, consultez [Signature et authentification des requêtes REST \(AWS signature \(version 2\)\)](#). Pour plus d'informations sur l'envoi de demandes à l'aide de kits de AWS développement logiciel (SDK), consultez [Faire des demandes à l'aide du AWS Kits SDK](#).

Pour de plus amples informations sur la prise en charge des informations d'identification de sécurité temporaires par IAM, veuillez consulter [Informations d'identification de sécurité temporaires](#) dans le Guide de l'utilisateur IAM.

Pour plus de sécurité, vous pouvez demander une authentification multi-facteur (MFA) lors de l'accès à vos ressources Amazon S3 en configurant une stratégie de compartiment. Pour plus d'informations, consultez la section [Exemples de politiques relatives aux compartiments : Exiger l'authentification multifacteur](#). Une fois que vous avez demandé une authentification multi-facteur (MFA) pour l'accès à vos ressources Amazon S3, la seule façon d'accéder à ces ressources est de fournir des informations d'identification temporaires créées avec une clé MFA. Pour plus d'informations, consultez la page [détaillée de l'AWS Multi-Factor authentication](#) et la [configuration de MFA-Protected l'accès à l'API](#) dans le guide de l'utilisateur IAM.

Points de terminaison de demande

Vous envoyez des demandes REST au point de terminaison prédéfini du service. Pour obtenir la liste de tous les AWS services et de leurs points de terminaison correspondants, accédez à [Régions et points de terminaison](#) dans le. Références générales AWS

Envoi de demandes à Amazon S3 via IPv6

Amazon Simple Storage Service (Amazon S3) prend en charge l'accès aux compartiments S3 à l'aide du Protocole Internet version 6 (IPv6), en plus du protocole IPv4. Les points de terminaison à double pile Amazon S3 prennent en charge les demandes envoyées aux compartiments S3 via IPv6 et IPv4. Il n'y a aucuns frais supplémentaires pour accéder à Amazon S3 via IPv6. Pour de plus amples informations sur la tarification, veuillez consulter [Tarification Amazon S3](#).

Note

Les informations suivantes concernent les points de terminaison IPv6 Amazon S3 publics. Pour plus d'informations sur les points de terminaison du AWS Virtual Private Cloud (VPC), reportez-vous [AWS PrivateLink à Amazon S3](#).

Rubriques

- [Mise en route de l'envoi des demandes via IPv6](#)
- [Utilisation d'adresses IPv6 dans les stratégies IAM](#)
- [Test de compatibilité d'adresses IP](#)
- [Utilisation des points de terminaison Dual-Stack Amazon S3](#)

Mise en route de l'envoi des demandes via IPv6

Pour envoyer une demande à un compartiment S3 via IPv6, vous devez utiliser un point de terminaison Dual-Stack. La section suivante décrit comment envoyer des demandes via IPv6 à l'aide de points de terminaison Dual-Stack.

Voici quelques éléments à connaître avant de tenter d'accéder à un compartiment via IPv6 :

- Le client et le réseau accédant au compartiment doivent être autorisés à utiliser le protocole IPv6.
- Les demandes de type hébergement virtuel et type chemin sont prises en charge pour un accès via IPv6. Pour de plus amples informations, veuillez consulter [Points de terminaison Amazon S3 Dual-Stack](#).
- Si vous utilisez le filtrage des adresses IP source dans vos politiques d'utilisateur ou de compartiment Gestion des identités et des accès AWS (IAM), vous devez mettre à jour les politiques pour inclure des plages d'adresses IPv6. Pour de plus amples informations, veuillez consulter [Utilisation d'adresses IPv6 dans les stratégies IAM](#).
- Lorsque vous utilisez le protocole IPv6, les fichiers journaux d'accès au serveur génèrent les adresses IP au format IPv6. Vous devez mettre à jour les outils, scripts et logiciels existants utilisés pour analyser les fichiers journaux Amazon S3, de manière à ce qu'ils puissent également analyser les adresses Remote IP au format IPv6. Pour plus d'informations, consultez la section [Journalisation des demandes avec journalisation des accès au serveur](#).

Note

Si vous rencontrez des problèmes liés à la présence d'adresses IPv6 dans les fichiers journaux, contactez [AWS Support](#).

Envoi de demandes via IPv6 à l'aide de points de terminaison Dual-Stack

Pour envoyer des demandes à l'aide d'appels d'API Amazon S3 via IPv6, vous devez utiliser des points de terminaison Dual-Stack. Le fonctionnement des opérations d'API Amazon S3 est le même, que vous accédez à Amazon S3 via IPv6 ou via IPv4. Les performances sont normalement également identiques.

Lorsque vous utilisez l'API REST, vous accédez directement à un point de terminaison Dual-Stack. Pour de plus amples informations, veuillez consulter [Dual-stack points de terminaison](#).

Lorsque vous utilisez le AWS Command Line Interface (AWS CLI) et AWS les SDK, vous pouvez utiliser un paramètre ou un indicateur pour passer à un point de terminaison à double pile. Vous pouvez également spécifier directement le point de terminaison Dual-Stack en remplaçant le point de terminaison Amazon S3 dans le fichier de configuration.

Un point de terminaison Dual-Stack peut être utilisé pour accéder à un compartiment via IPv6 à partir de chacun des éléments suivants :

- Le AWS CLI, tu vois [À l'aide de points de terminaison à double pile provenant du AWS CLI](#).
- Les AWS kits de développement logiciel (SDK), voir [À l'aide de points de terminaison à double pile provenant du AWS Kits SDK](#).
- L'API REST (consultez [Envoi de demandes à des points de terminaison Dual-Stack à l'aide de l'API REST](#)).

Fonctions non disponibles via IPv6

La fonctionnalité suivante n'est actuellement pas prise en charge lors de l'accès à un compartiment S3 via IPv6 : Hébergement de site web statique à partir d'un compartiment S3.

Utilisation d'adresses IPv6 dans les stratégies IAM

Avant de tenter d'accéder à un compartiment à l'aide du protocole IPv6, vous devez vérifier que toutes les stratégies d'utilisateur IAM ou de compartiment S3 utilisées pour le filtrage des adresses IP ont été mises à jour et comprennent bien les plages d'adresses IPv6. En cas d'absence de mise à jour des stratégies de filtrage des adresses IP pour gérer les adresses IPv6, les clients risquent de ne pas pouvoir accéder correctement au compartiment lorsqu'ils commenceront à utiliser le protocole IPv6. Pour plus d'informations sur la gestion des autorisations d'accès avec IAM, consultez [Gestion des identités et des accès pour Amazon S3](#).

Les stratégies IAM qui permettent de filtrer les adresses IP utilisent des [opérateurs de condition d'adresse IP](#). La stratégie de compartiment suivante identifie la plage d'adresses 54.240.143.* d'adresses IPv4 autorisées à l'aide d'opérateurs de condition d'adresse IP. Toute adresse IP se trouvant hors de cette plage se verra refuser l'accès au compartiment (examplebucket). Comme toutes les adresses IPv6 se trouvent hors de la plage autorisée, cette stratégie empêche les adresses IPv6 d'accéder à examplebucket.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "IPAllow",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket/*",
      "Condition": {
        "IpAddress": {"aws:SourceIp": "54.240.143.0/24"}
      }
    }
  ]
}
```

Vous pouvez modifier l'élément Condition de la stratégie de compartiment afin d'autoriser les plages d'adresses IPv4 (54.240.143.0/24) et IPv6 (2001:DB8:1234:5678::/64), comme illustré dans l'exemple suivant. Vous pouvez utiliser le même type de bloc Condition que celui indiqué dans l'exemple pour mettre à jour vos stratégies de compartiment et d'utilisateur IAM.

```
"Condition": {
  "IpAddress": {
    "aws:SourceIp": [
      "54.240.143.0/24",
      "2001:DB8:1234:5678::/64"
    ]
  }
}
```

Avant d'utiliser IPv6, vous devez mettre à jour toutes les politiques IAM relatives aux utilisateurs et aux compartiments qui utilisent le filtrage des adresses IP. Évitez d'utiliser le filtrage des adresses IP dans les politiques de bucket.

Vous pouvez consulter vos politiques utilisateur IAM à l'aide de la console IAM à l'adresse. <https://console.aws.amazon.com/iam/> Pour de plus amples informations sur IAM, consultez le [Guide de l'utilisateur IAM](#). Pour plus d'informations sur la modification des politiques de compartiment S3, consultez la section [Ajouter une politique de compartiment](#).

Test de compatibilité d'adresses IP

Si vous utilisez Linux/Unix ou Mac OS X, vous pouvez vérifier si vous pouvez accéder à un point de terminaison à double pile via IPv6 à l'aide de la `curl` commande illustrée dans l'exemple suivant :

Exemple

```
curl -v http://s3.dualstack.us-west-2.amazonaws.com/
```

Les informations que vous obtenez doivent ressembler à celles de l'exemple ci-dessous. Si vous êtes connecté via IPv6, l'adresse IP connectée est une adresse IPv6.

```
* About to connect() to s3-us-west-2.amazonaws.com port 80 (#0)
* Trying IPv6 address... connected
* Connected to s3.dualstack.us-west-2.amazonaws.com (IPv6 address) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.18.1 (x86_64-unknown-linux-gnu) libcurl/7.18.1 OpenSSL/1.0.1t
zlib/1.2.3
> Host: s3.dualstack.us-west-2.amazonaws.com
```

Si vous utilisez Microsoft 7 ou 10, vous pouvez tester si vous pouvez accéder à un point de terminaison à double pile via IPv6 ou IPv4 en utilisant la `ping` commande comme indiqué dans l'exemple suivant.

```
ping ipv6.s3.dualstack.us-west-2.amazonaws.com
```

Utilisation des points de terminaison Dual-Stack Amazon S3

Les points de terminaison à double pile Amazon S3 prennent en charge les demandes envoyées aux compartiments S3 via IPv6 et IPv4. Cette section décrit comment utiliser les points de terminaison Dual-Stack (double pile).

Rubriques

- [Points de terminaison Amazon S3 Dual-Stack](#)
- [À l'aide de points de terminaison à double pile provenant du AWS CLI](#)
- [À l'aide de points de terminaison à double pile provenant du AWS Kits SDK](#)
- [Utilisation de points de terminaison Dual-Stack avec l'API REST](#)

Points de terminaison Amazon S3 Dual-Stack

Lorsque vous envoyez une demande à un point de terminaison Dual-Stack, l'URL du compartiment est résolue en une adresse IPv6 ou IPv4. Pour plus d'informations sur l'accès à un compartiment via IPv6, consultez [Envoi de demandes à Amazon S3 via IPv6](#).

Lorsque vous utilisez l'API REST, vous accédez directement à un point de terminaison Amazon S3 à l'aide du nom de point de terminaison (URI). Vous pouvez accéder à un compartiment S3 via un point de terminaison Dual-Stack en utilisant un nom de point de terminaison de type hébergement virtuel ou chemin. Amazon S3 prend uniquement en charge les noms de points de terminaison Dual-Stack Régionaux ; vous devez donc spécifier la Région dans le nom.

Utilisez les conventions de dénomination suivantes pour les points de terminaison Dual-Stack de type hébergement virtuel ou chemin :

- Point de terminaison Dual-Stack de type hébergement virtuel :

bucketname.s3. dualstack. *aws-region*.amazonaws.com

- Path-style point de terminaison à double pile :

S3. DualStack. *aws-region*.amazonaws.com/ *bucketname*

Pour plus d'informations sur le style de nom du point de terminaison, consultez la section [Accès à un compartiment Amazon S3 et sa mise en vente](#). Pour obtenir la liste des points de terminaison Amazon S3, consultez [Régions et points de terminaison](#) dans la Références générales AWS.

 Important

Vous pouvez utiliser Transfer Acceleration avec les points de terminaison double pile (« Dual-Stack »). Pour plus d'informations, consultez [Mise en route avec Amazon S3 Transfer Acceleration](#).

 Note

Les deux types de points de terminaison de cloud privé virtuel (VPC) qui accèdent à Amazon S3 (points de terminaison VPC d'interface et points de terminaison VPC de passerelle) sont désormais compatibles avec la double pile. Pour plus d'informations sur les points de terminaison VPC pour Amazon S3, consultez [AWS PrivateLink Amazon S3](#).

Lorsque vous utilisez le AWS Command Line Interface (AWS CLI) et AWS les SDK, vous pouvez utiliser un paramètre ou un indicateur pour passer à un point de terminaison à double pile. Vous pouvez également spécifier directement le point de terminaison Dual-Stack en remplaçant le point de terminaison Amazon S3 dans le fichier de configuration. Les sections suivantes décrivent comment utiliser les points de terminaison à double pile à partir du SDK AWS CLI et des AWS SDK.

À l'aide de points de terminaison à double pile provenant du AWS CLI

Cette section fournit des exemples de AWS CLI commandes utilisées pour envoyer des requêtes à un point de terminaison à double pile. Pour obtenir des instructions sur la configuration du AWS CLI, consultez [Développement avec Amazon S3 à l'aide du AWS CLI](#).

Vous définissez la valeur de configuration `use_dualstack_endpoint` sur `true` dans un profil de votre AWS Config fichier afin de diriger toutes les requêtes Amazon S3 effectuées par les `s3api` AWS CLI commandes `s3` et vers le point de terminaison à double pile pour la région spécifiée. Vous indiquez la Région dans le fichier de configuration ou dans une commande à l'aide de l'option `--region`.

Lorsque vous utilisez des points de terminaison à double pile avec les styles d'virtualadressage AWS CLI, les deux `path` et `style` sont pris en charge. Le type d'adressage défini dans le fichier de

configuration vérifie que le nom du compartiment se trouve bien dans le nom d'hôte ou fait partie de l'URL. Par défaut, l'interface de ligne de commande tente, si possible, d'utiliser le style virtuel, mais revient au type chemin si nécessaire. Pour plus d'informations, consultez [Configuration de la AWS CLI d'Amazon S3](#).

Vous pouvez également modifier la configuration à l'aide d'une commande, comme indiqué dans l'exemple ci-dessous, qui définit `use_dualstack_endpoint` sur `true` et `addressing_style` sur `virtual` dans le profil par défaut.

```
$ aws configure set default.s3.use_dualstack_endpoint true
$ aws configure set default.s3.addressing_style virtual
```

Si vous souhaitez utiliser un point de terminaison à double pile pour des AWS CLI commandes spécifiques uniquement (pas toutes les commandes), vous pouvez utiliser l'une des méthodes suivantes :

- Vous pouvez utiliser le point de terminaison Dual-Stack par commande en définissant le paramètre `--endpoint-url` sur `https://s3.dualstack.aws-region.amazonaws.com` ou sur `http://s3.dualstack.aws-region.amazonaws.com` pour toute commande `s3` ou `s3api`.

```
$ aws s3api list-objects --bucket bucketname --endpoint-url https://s3.dualstack.aws-region.amazonaws.com
```

- Vous pouvez créer des profils distincts dans votre AWS Config dossier. Par exemple, vous pouvez créer un profil qui définit `use_dualstack_endpoint` sur `true` et un profil qui ne définit pas `use_dualstack_endpoint`. Lorsque vous exécutez une commande, spécifiez le profil adéquat selon que vous comptez utiliser ou non le point de terminaison Dual-Stack.

Note

Lorsque vous utilisez le, AWS CLI vous ne pouvez actuellement pas utiliser l'accélération des transferts avec des points de terminaison à double pile. Cependant, le support pour AWS CLI le sera bientôt. Pour plus d'informations, consultez la section [Activation et utilisation de l'accélération de transfert S3](#).

À l'aide de points de terminaison à double pile provenant du AWS Kits SDK

Cette section fournit des exemples d'accès à un terminal à double pile à l'aide des AWS kits SDK.

AWS SDK pour Java exemple de point de terminaison à double pile

L'exemple suivant montre comment activer les points de terminaison Dual-Stack lors de la création d'un client Amazon S3 à l'aide du kit AWS SDK pour Java.

Pour obtenir des instructions sur la création et le test d'un exemple Java fonctionnel, consultez [Getting Started](#) dans le Guide du AWS SDK pour Java développeur.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;

public class DualStackEndpoints {

    public static void main(String[] args) {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "*** Bucket name ***";

        try {
            // Create an Amazon S3 client with dual-stack endpoints enabled.
            AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .withDualstackEnabled(true)
                .build();

            s3Client.listObjects(bucketName);
        } catch (AmazonServiceException e) {
            // The call was transmitted successfully, but Amazon S3 couldn't process
            // it, so it returned an error response.
            e.printStackTrace();
        } catch (SdkClientException e) {
            // Amazon S3 couldn't be contacted for a response, or the client
            // couldn't parse the response from Amazon S3.
            e.printStackTrace();
        }
    }
}
```

Si vous utilisez AWS SDK pour Java le sous Windows, vous devrez peut-être définir la propriété de machine virtuelle Java (JVM) suivante :

```
java.net.preferIPv6Addresses=true
```

AWS Exemple de point de terminaison à double pile du SDK .NET

Lorsque vous utilisez le AWS SDK pour .NET, vous utilisez la `AmazonS3Config` classe pour activer l'utilisation d'un point de terminaison à double pile, comme indiqué dans l'exemple suivant.

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class DualStackEndpointTest
    {
        private const string bucketName = "**** bucket name ****";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
        private static IAmazonS3 client;

        public static void Main()
        {
            var config = new AmazonS3Config
            {
                UseDualstackEndpoint = true,
                RegionEndpoint = bucketRegion
            };
            client = new AmazonS3Client(config);
            Console.WriteLine("Listing objects stored in a bucket");
            ListingObjectsAsync().Wait();
        }

        private static async Task ListingObjectsAsync()
        {
            try
            {
                var request = new ListObjectsV2Request
                {

```

```

        BucketName = bucketName,
        MaxKeys = 10
    };
    ListObjectsV2Response response;
    do
    {
        response = await client.ListObjectsV2Async(request);

        // Process the response.
        foreach (S3Object entry in response.S3Objects)
        {
            Console.WriteLine("key = {0} size = {1}",
                              entry.Key, entry.Size);
        }
        Console.WriteLine("Next Continuation Token: {0}",
response.NextContinuationToken);
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated == true);
    }
    catch (AmazonS3Exception amazonS3Exception)
    {
        Console.WriteLine("An AmazonS3Exception was thrown. Exception: " +
amazonS3Exception.ToString());
    }
    catch (Exception e)
    {
        Console.WriteLine("Exception: " + e.ToString());
    }
}
}
}

```

Pour plus d'informations sur la configuration et l'exécution des exemples de code, consultez la section [Getting Started with the AWS SDK for .NET](#) Developer Guide.

Utilisation de points de terminaison Dual-Stack avec l'API REST

Pour obtenir des informations sur la procédure permettant d'envoyer des demandes à des points de terminaison Dual-Stack à l'aide de l'API REST, consultez la section [Envoi de demandes à des points de terminaison Dual-Stack à l'aide de l'API REST](#).

Faire des demandes à l'aide du AWS Kits SDK

Rubriques

- [Faire des demandes à l'aide de Compte AWS ou informations d'identification utilisateur IAM](#)
- [Envoi de demandes à l'aide des informations d'identification temporaires des utilisateurs IAM](#)
- [Demandes grâce aux informations d'identification de sécurité temporaires de l'utilisateur fédéré](#)

Vous pouvez envoyer des demandes authentifiées à Amazon S3 à l'aide du AWS SDK ou en effectuant les appels d'API REST directement dans votre application. L'API du AWS SDK utilise les informations d'identification que vous fournissez pour calculer la signature à des fins d'authentification. Si vous utilisez l'API REST directement dans vos applications, vous devez écrire le code nécessaire pour calculer la signature d'authentification de votre demande. Pour obtenir la liste des AWS kits de développement logiciel disponibles, rendez-vous sur [Exemples de code et bibliothèques](#).

Faire des demandes à l'aide de Compte AWS ou informations d'identification utilisateur IAM

Vous pouvez utiliser vos informations de sécurité Compte AWS ou celles de l'utilisateur IAM pour envoyer des demandes authentifiées à Amazon S3. Cette section fournit des exemples de la manière dont vous pouvez envoyer des demandes authentifiées à l'aide des AWS SDK pour Java touches AWS SDK pour .NET, et AWS SDK pour PHP. Pour obtenir la liste des AWS SDK disponibles, consultez la section [Exemples de code et bibliothèques](#).

Chacun de ces AWS kits SDK utilise une chaîne de fournisseurs SDK-specific d'informations d'identification pour rechercher et utiliser les informations d'identification et effectuer des actions pour le compte du propriétaire des informations d'identification. Le point commun de toutes ces chaînes de fournisseurs d'informations d'identification est qu'elles recherchent toutes votre fichier AWS d'informations d'identification local.

Pour plus d'informations, consultez les rubriques ci-dessous :

Rubriques

- [Pour créer un local AWS fichier d'informations d'identification](#)
- [Envoi de demandes authentifiées à l'aide du AWS Kits SDK](#)
- [Ressources connexes](#)

Pour créer un local AWS fichier d'informations d'identification

Le moyen le plus simple de configurer les informations d'identification pour vos AWS kits SDK consiste à utiliser un fichier d' AWS informations d'identification. Si vous utilisez le AWS Command Line Interface (AWS CLI), il se peut qu'un fichier d'informations d' AWS identification local soit déjà configuré. Sinon, suivez la procédure ci-dessous pour configurer en :

1. Connectez-vous au Console de gestion AWS et ouvrez la console IAM à <https://console.aws.amazon.com/iam/> l'adresse.
2. Créez un utilisateur doté d'autorisations limitées aux services et actions auxquels votre code sera autorisé à accéder. Pour en savoir plus sur la création d'un utilisateur, veuillez consulter [Création d'utilisateurs IAM \(Console\)](#), puis suivez les instructions jusqu'à l'étape 8.
3. Choisissez Download .csv (Télécharger .csv) pour enregistrer une copie locale de vos informations d'identification AWS .
4. Sur votre ordinateur, créez le répertoire .aws dans le répertoire de base. Sur Unix-based des systèmes tels que Linux ou OS X, il se trouve à l'emplacement suivant :

```
~/ .aws
```

Sur les systèmes Windows, il se trouve à l'emplacement suivant :

```
%HOMEPATH%\ .aws
```

5. Dans le répertoire .aws, créez un fichier appelé `credentials`.
6. Ouvrez le fichier .csv des informations d'identification que vous avez téléchargé à partir de la console IAM, puis copiez-en le contenu dans le fichier , en respectant le format suivant :

```
[default]
aws_access_key_id = your_access_key_id
aws_secret_access_key = your_secret_access_key
```

7. Enregistrez le fichier `credentials`, puis supprimez le fichier .csv que vous avez téléchargé à l'étape 3.

Votre fichier d'informations d'identification partagé est maintenant configuré sur votre ordinateur local et il est prêt à être utilisé avec les AWS kits SDK.

Envoi de demandes authentifiées à l'aide du AWS Kits SDK

Utilisez les AWS kits SDK pour envoyer des demandes authentifiées. Pour plus d'informations sur l'envoi de demandes authentifiées, consultez [Informations d'identification de sécurité AWS](#) ou [Authentification dans IAM Identity Center](#).

Java

Pour plus d'informations sur l'authentification des demandes à l'aide du AWS SDK pour Java, consultez les sections [Utilisation de fichiers de configuration et d'informations d'identification partagés pour configurer globalement les AWS SDK et les outils](#) et [Authentification et accès à l'aide des AWS SDK et des outils](#) dans le Guide de référence des kits de AWS développement logiciel et des outils.

.NET

Pour envoyer des demandes authentifiées à l'aide de vos informations d'identification Compte AWS ou de celles de l'utilisateur IAM, procédez comme suit :

- Créez une instance de la classe `AmazonS3Client`.
- Exécutez l'une des méthodes `AmazonS3Client` pour envoyer des demandes à Amazon S3. Le client génère la signature nécessaire à partir des informations d'identification fournies et l'inclut dans la demande envoyée à Amazon S3.

Pour plus d'informations, consultez la section [Création de demandes à l'aide des informations d'identification d'un AWS compte ou d'un utilisateur IAM](#) >.

Note

- Vous pouvez créer la classe `AmazonS3Client` sans fournir d'informations d'identification de sécurité. Les demandes envoyées via ce client sont anonymes et ne comportent pas de signature. Amazon S3 renvoie un message d'erreur si vous envoyez des demandes anonymes pour une ressource qui n'est pas disponible publiquement.
- Vous pouvez créer un utilisateur Compte AWS et créer les utilisateurs requis. Vous pouvez également gérer les informations d'identification pour ces utilisateurs. Vous avez besoin de ces informations d'identification pour exécuter la tâche dans l'exemple suivant. Pour de plus amples informations, veuillez consulter la section [Configurer les informations d'identification AWS](#) du Guide du développeur pour le kit SDK pour .NET .

Vous pouvez ensuite configurer votre application pour récupérer activement les profils et les informations d'identification, puis utiliser explicitement ces informations d'identification lors de la création d'un client AWS de service. Pour plus d'informations, consultez la section [Accès aux informations d'identification et aux profils dans une application](#) du Guide du développeur pour le kit SDK pour .NET .

L'exemple C# suivant montre comment exécuter les tâches précédentes. Pour plus d'informations sur la configuration et l'exécution des exemples de code, consultez la section [Getting Started with the AWS SDK for .NET](#) Developer Guide.

Exemple

```
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;
using System;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
    class MakeS3RequestTest
    {
        private const string bucketName = "**** bucket name ****";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
        RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            using (client = new AmazonS3Client(bucketRegion))
            {
                Console.WriteLine("Listing objects stored in a bucket");
                ListingObjectsAsync().Wait();
            }
        }

        static async Task ListingObjectsAsync()
        {
            try
```



```
{
    ListObjectsRequest request = new ListObjectsRequest
    {
        BucketName = bucketName,
        MaxKeys = 2
    };
    do
    {
        ListObjectsResponse response = await
client.ListObjectsAsync(request);
        // Process the response.
        foreach (S3Object entry in response.S3Objects)
        {
            Console.WriteLine("key = {0} size = {1}",
                entry.Key, entry.Size);
        }

        // If the response is truncated, set the marker to get the next
        // set of keys.
        if (response.IsTruncated)
        {
            request.Marker = response.NextMarker;
        }
        else
        {
            request = null;
        }
    } while (request != null);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine("Error encountered on server. Message:'{0}' when
writing an object", e.Message);
}
catch (Exception e)
{
    Console.WriteLine("Unknown encountered on server. Message:'{0}' when
writing an object", e.Message);
}
}
}
```

PHP

Cette section explique comment utiliser une classe de la version 3 du pour envoyer des demandes authentifiées AWS SDK pour PHP à l'aide de vos informations d'identification Compte AWS ou de celles de l'utilisateur IAM. Pour plus d'informations sur l'API AWS SDK pour Ruby, consultez [AWS SDK pour Ruby - Version 2](#).

L'exemple PHP suivant montre comment le client effectue une demande à l'aide de vos informations d'identification de sécurité pour répertorier tous les compartiments pour votre compte.

Exemple

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;

$bucket = '*** Your Bucket Name ***';

$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
]);

// Retrieve the list of buckets.
$result = $s3->listBuckets();

try {
    // Retrieve a paginator for listing objects.
    $objects = $s3->getPaginator('ListObjects', [
        'Bucket' => $bucket
    ]);

    echo "Keys retrieved!" . PHP_EOL;

    // Print the list of objects to the page.
    foreach ($objects as $object) {
        echo $object['Key'] . PHP_EOL;
    }
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
```

```
}
```

Note

Vous pouvez créer la classe `S3Client` sans fournir d'informations d'identification de sécurité. Les demandes envoyées via ce client sont anonymes et ne comportent pas de signature. Amazon S3 renvoie un message d'erreur si vous envoyez des demandes anonymes pour une ressource qui n'est pas disponible publiquement. Pour plus d'informations, consultez [Création de clients anonymes](#) dans la [documentation AWS SDK pour PHP](#).

Ruby

Avant de pouvoir utiliser la version 3 du AWS SDK pour Ruby pour passer des appels vers Amazon S3, vous devez définir les informations d'AWS accès que le SDK utilise pour vérifier votre accès à vos compartiments et à vos objets. Si vous avez configuré des informations d'identification partagées dans le profil AWS d'informations d'identification de votre système local, la version 3 du SDK pour Ruby peut utiliser ces informations d'identification sans que vous ayez à les déclarer dans votre code. Pour plus d'informations sur la configuration des informations d'identification partagées, consultez la section [Effectuer des demandes à l'aide des informations d'identification](#) d'un AWS compte ou d'un utilisateur IAM.

L'extrait de code Ruby suivant utilise les informations d'identification contenues dans un fichier AWS d'informations d'identification partagé sur un ordinateur local pour authentifier une demande visant à obtenir tous les noms de clés d'objets d'un compartiment spécifique. Il exécute les opérations suivantes :

1. Crée une instance de la classe `Aws::S3::Client`.
2. Envoie une demande à Amazon S3 en énumérant les objets d'un compartiment à l'aide de la méthode `list_objects_v2` de `Aws::S3::Client`. Le client génère la valeur de signature nécessaire à partir des informations d'AWS identification figurant dans le fichier d'informations d'identification de votre ordinateur et l'inclut dans la demande qu'il envoie à Amazon S3.
3. Imprime le tableau des noms de clés d'objet vers le terminal.

Example

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'amzn-s3-demo-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts 'The object keys in this bucket are (first 50 objects):'
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts 'No objects found in this bucket.'
  end

  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
  false
end

# Example usage:
def run_me
  region = 'us-west-2'
  bucket_name = 'BUCKET_NAME'
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end
```

```
run_me if $PROGRAM_NAME == __FILE__
```

Si vous ne disposez pas d'un fichier AWS d'informations d'identification local, vous pouvez toujours créer la `Aws::S3::Client` ressource et exécuter du code sur des compartiments et des objets Amazon S3. Les demandes envoyées avec la version 3 du kit SDK pour Ruby sont anonymes et ne comportent aucune signature par défaut. Amazon S3 renvoie une erreur si vous envoyez des demandes anonymes pour une ressource qui n'est pas disponible publiquement.

Vous pouvez utiliser et développer l'extrait de code précédent pour les applications du kit SDK pour Ruby, comme illustré dans l'exemple suivant, plus complexe.

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'

# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if all operations succeed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_bucket_objects?(s3_client, 'amzn-s3-demo-bucket')
def list_bucket_objects?(s3_client, bucket_name)
  puts "Accessing the bucket named '#{bucket_name}'..."
  objects = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if objects.count.positive?
    puts 'The object keys in this bucket are (first 50 objects):'
    objects.contents.each do |object|
      puts object.key
    end
  else
    puts 'No objects found in this bucket.'
  end

  true
rescue StandardError => e
```

```

    puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
    false
end

# Example usage:
def run_me
  region = 'us-west-2'
  bucket_name = 'BUCKET_NAME'
  s3_client = Aws::S3::Client.new(region: region)

  exit 1 unless list_bucket_objects?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__

```

Go

Example

L'exemple suivant utilise des AWS informations d'identification chargées automatiquement par le SDK pour Go à partir du fichier d'informations d'identification partagé.

```

package main

import (
    "context"
    "errors"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {

```

```
    fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
    fmt.Println(err)
    return
}
s3Client := s3.NewFromConfig(sdkConfig)
count := 10
fmt.Printf("Let's list up to %v buckets for your account.\n", count)
result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
if err != nil {
    var ae smithy.APIError
    if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
        fmt.Println("You don't have permission to list buckets for this account.")
    } else {
        fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
    }
    return
}
if len(result.Buckets) == 0 {
    fmt.Println("You don't have any buckets!")
} else {
    if count > len(result.Buckets) {
        count = len(result.Buckets)
    }
    for _, bucket := range result.Buckets[:count] {
        fmt.Printf("\t\t%v\n", *bucket.Name)
    }
}
}
```

Ressources connexes

- [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#)
- [AWS SDK pour PHP pour la classe Amazon S3 Aws\S3\S3Client](#)
- [Documentation AWS SDK pour PHP](#)

Envoi de demandes à l'aide des informations d'identification temporaires des utilisateurs IAM

Un utilisateur Compte AWS ou un utilisateur IAM peut demander des informations d'identification de sécurité temporaires et les utiliser pour envoyer des demandes authentifiées à Amazon S3. Cette section fournit des exemples d'utilisation des kits AWS SDK pour Java SDK pour Java, .NET, and PHP afin d'obtenir des informations d'identification de sécurité temporaires et de les utiliser pour authentifier vos demandes à Amazon S3.

Java

Un ou plusieurs utilisateurs IAM Compte AWS peuvent demander des informations d'identification de sécurité temporaires (voir [Demandes](#)) à l'aide du AWS SDK pour Java et les utiliser pour accéder à Amazon S3. Ces informations d'identification expirent à la fin de session spécifiée.

Par défaut, la session dure une heure. Si vous utilisez les informations d'identification des utilisateurs IAM, vous pouvez spécifier la durée (de 15 minutes à la durée maximale de session pour le rôle) lors de la demande d'informations d'identification de sécurité temporaires. Pour en savoir plus sur les informations d'identification de sécurité temporaires, veuillez consulter [Informations d'identification de sécurité temporaires](#) dans le Guide de l'utilisateur IAM. Pour plus d'informations sur la création de demandes, consultez [Demandes](#).

Pour obtenir des informations d'identification de sécurité temporaires et accéder à Amazon S3

1. Créez une instance de la classe `AWSSecurityTokenService`.
2. Récupérez les informations d'identification de sécurité temporaires pour le rôle souhaité en appelant la méthode `assumeRole()` du client STS (Security Token Service).
3. Regroupez les informations d'identification de sécurité temporaires dans un objet `BasicSessionCredentials`. Vous utilisez cet objet pour fournir les informations d'identification de sécurité temporaires à votre client Amazon S3.
4. Créez une instance de la classe `AmazonS3Client` à l'aide des informations d'identification de sécurité temporaires. Vous envoyez des demandes à Amazon S3 grâce à ce client. Si vous envoyez des demandes à l'aide d'informations d'identification expirées, Amazon S3 renvoie une erreur.

 Note

Si vous obtenez les informations d'identification de sécurité temporaires à l'aide des informations d'identification de sécurité du Compte AWS, les informations d'identification de sécurité temporaires ne sont valides que pendant une heure. Vous pouvez spécifier la durée de session uniquement si vous utilisez les informations d'identification des utilisateurs IAM pour demander une session.

L'exemple suivant répertorie un ensemble de clés d'objet dans le compartiment spécifié. L'exemple obtient les informations d'identification de sécurité temporaires pour une session et les utilise pour envoyer une demande authentifiée à Amazon S3.

Si vous souhaitez tester l'exemple à l'aide des informations d'identification de l'utilisateur IAM, vous devez créer un utilisateur IAM sous votre Compte AWS. Pour de plus amples informations sur la création d'un utilisateur IAM, veuillez consulter [Création de votre premier groupe d'utilisateurs et d'administrateurs IAM](#) dans le Guide de l'utilisateur IAM.

Pour obtenir des instructions sur la création et le test d'un échantillon de travail, voir [Getting Started](#) dans le Guide du AWS SDK pour Java développeur.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.AssumeRoleRequest;
import com.amazonaws.services.securitytoken.model.AssumeRoleResult;
import com.amazonaws.services.securitytoken.model.Credentials;

public class MakingRequestsWithIAMTempCredentials {
    public static void main(String[] args) {
        String clientRegion = "**** Client region ****";
        String roleARN = "**** ARN for role to be assumed ****";
        String roleSessionName = "**** Role session name ****";
```

```
String bucketName = "*** Bucket name ***";

try {
    // Creating the STS client is part of your trusted code. It has
    // the security credentials you use to obtain temporary security
credentials.
    AWSSecurityTokenService stsClient =
AWSecurityTokenServiceClientBuilder.standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(clientRegion)
        .build();

    // Obtain credentials for the IAM role. Note that you cannot assume the
role of
    // an AWS root account;
    // Amazon S3 will deny access. You must use credentials for an IAM user
or an
    // IAM role.
    AssumeRoleRequest roleRequest = new AssumeRoleRequest()
        .withRoleArn(roleARN)
        .withRoleSessionName(roleSessionName);
    AssumeRoleResult roleResponse = stsClient.assumeRole(roleRequest);
    Credentials sessionCredentials = roleResponse.getCredentials();

    // Create a BasicSessionCredentials object that contains the credentials
you
    // just retrieved.
    BasicSessionCredentials awsCredentials = new BasicSessionCredentials(
        sessionCredentials.getAccessKeyId(),
        sessionCredentials.getSecretAccessKey(),
        sessionCredentials.getSessionToken());

    // Provide temporary security credentials so that the Amazon S3 client
    // can send authenticated requests to Amazon S3. You create the client
    // using the sessionCredentials object.
    AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
        .withCredentials(new
AWSStaticCredentialsProvider(awsCredentials))
        .withRegion(clientRegion)
        .build();

    // Verify that assuming the role worked and the permissions are set
correctly
    // by getting a set of object keys from the bucket.
```

```
        ObjectListing objects = s3Client.listObjects(bucketName);
        System.out.println("No. of Objects: " +
objects.getObjectSummaries().size());
    } catch (AmazonServiceException e) {
        // The call was transmitted successfully, but Amazon S3 couldn't process
        // it, so it returned an error response.
        e.printStackTrace();
    } catch (SdkClientException e) {
        // Amazon S3 couldn't be contacted for a response, or the client
        // couldn't parse the response from Amazon S3.
        e.printStackTrace();
    }
}
```

.NET

Un ou plusieurs utilisateurs IAM Compte AWS peuvent demander des informations d'identification de sécurité temporaires à l'aide du AWS SDK pour .NET et les utiliser pour accéder à Amazon S3. Ces informations d'identification expirent à la fin de la session.

Par défaut, la session dure une heure. Si vous utilisez les informations d'identification des utilisateurs IAM, vous pouvez spécifier la durée (de 15 minutes à la durée maximale de session pour le rôle) lors de la demande d'informations d'identification de sécurité temporaires. Pour en savoir plus sur les informations d'identification de sécurité temporaires, veuillez consulter [Informations d'identification de sécurité temporaires](#) dans le Guide de l'utilisateur IAM. Pour plus d'informations sur la création de demandes, consultez [Demandes](#).

Pour obtenir des informations d'identification de sécurité temporaires et accéder à Amazon S3

1. Créez une instance du AWS Security Token Service client, `AmazonSecurityTokenServiceClient`.
2. Lancez une session en appelant la méthode `GetSessionToken` du client STS que vous avez créé dans l'étape précédente. Vous fournissez des informations de session à cette méthode grâce à un objet `GetSessionTokenRequest`.

La méthode renvoie les informations d'identification de sécurité temporaires.

3. Regroupez les informations d'identification de sécurité temporaires dans une instance de l'objet `SessionAWSCredentials`. Vous utilisez cet objet pour fournir les informations d'identification de sécurité temporaires à votre client Amazon S3.

4. Créez une instance de la classe `AmazonS3Client` en fournissant les informations d'identification de sécurité temporaires. Vous envoyez des demandes à Amazon S3 grâce à ce client. Si vous envoyez des demandes à l'aide d'informations d'identification expirées, Amazon S3 renvoie une erreur.

 Note

Si vous obtenez les informations d'identification de sécurité temporaires à l'aide des informations d'identification de sécurité de votre Compte AWS, ces informations d'identification de sécurité temporaires ne sont valides que pendant une heure. Vous pouvez spécifier une durée de session uniquement si vous utilisez les informations d'identification des utilisateurs IAM pour demander une session.

L'exemple C# suivant répertorie les clés d'objet dans le compartiment spécifié. À titre d'illustration, l'exemple obtient des informations d'identification de sécurité temporaires pour une session d'une heure par défaut et les utilise pour envoyer une demande authentifiée à Amazon S3.

Si vous souhaitez tester l'exemple à l'aide des informations d'identification de l'utilisateur IAM, vous devez créer un utilisateur IAM sous votre Compte AWS. Pour de plus amples informations sur la création d'un utilisateur IAM, veuillez consulter [Création de votre premier groupe d'utilisateurs et d'administrateurs IAM](#) dans le Guide de l'utilisateur IAM. Pour plus d'informations sur la création de demandes, consultez [Demandes](#).

Pour plus d'informations sur la configuration et l'exécution des exemples de code, consultez la section [Getting Started with the AWS SDK for .NET](#) Developer Guide.

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;

namespace Amazon.DocSamples.S3
{
```

```

class TempCredExplicitSessionStartTest
{
    private const string bucketName = "*** bucket name ***";
    // Specify your bucket region (an example region is shown).
    private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
    private static IAmazonS3 s3Client;
    public static void Main()
    {
        ListObjectsAsync().Wait();
    }

    private static async Task ListObjectsAsync()
    {
        try
        {
            // Credentials use the default AWS SDK for .NET credential search
chain.
            // On local development machines, this is your default profile.
            Console.WriteLine("Listing objects stored in a bucket");
            SessionAWSCredentials tempCredentials = await
GetTemporaryCredentialsAsync();

            // Create a client by providing temporary security credentials.
            using (s3Client = new AmazonS3Client(tempCredentials, bucketRegion))
            {
                var listObjectRequest = new ListObjectsRequest
                {
                    BucketName = bucketName
                };
                // Send request to Amazon S3.
                ListObjectsResponse response = await
s3Client.ListObjectsAsync(listObjectRequest);
                List<S3Object> objects = response.S3Objects;
                Console.WriteLine("Object count = {0}", objects.Count);
            }
        }
        catch (AmazonS3Exception s3Exception)
        {
            Console.WriteLine(s3Exception.Message, s3Exception.InnerException);
        }
        catch (AmazonSecurityTokenServiceException stsException)
        {

```

```

        Console.WriteLine(stsException.Message,
stsException.InnerException);
    }
}

private static async Task<SessionAWSCredentials>
GetTemporaryCredentialsAsync()
{
    using (var stsClient = new AmazonSecurityTokenServiceClient())
    {
        var getSessionTokenRequest = new GetSessionTokenRequest
        {
            DurationSeconds = 7200 // seconds
        };

        GetSessionTokenResponse sessionTokenResponse =
            await
stsClient.GetSessionTokenAsync(getSessionTokenRequest);

        Credentials credentials = sessionTokenResponse.Credentials;

        var sessionCredentials =
            new SessionAWSCredentials(credentials.AccessKeyId,
                                     credentials.SecretAccessKey,
                                     credentials.SessionToken);

        return sessionCredentials;
    }
}
}
}

```

PHP

Pour plus d'informations sur l'API AWS SDK pour Ruby, consultez [AWS SDK pour Ruby - Version 2.](#)

Un utilisateur IAM ou un utilisateur Compte AWS peut demander des informations d'identification de sécurité temporaires à l'aide de la version 3 du AWS SDK pour PHP. Puis, il utilise les informations d'identification temporaires pour accéder à Amazon S3. Les informations d'identification expirent à la fin de la session.

Par défaut, la session dure une heure. Si vous utilisez les informations d'identification des utilisateurs IAM, vous pouvez spécifier la durée (de 15 minutes à la durée maximale de session

pour le rôle) lors de la demande d'informations d'identification de sécurité temporaires. Pour en savoir plus sur les informations d'identification de sécurité temporaires, veuillez consulter [Informations d'identification de sécurité temporaires](#) dans le Guide de l'utilisateur IAM. Pour plus d'informations sur la création de demandes, consultez [Demandes](#).

 Note

Si vous obtenez des informations d'identification de sécurité temporaires grâce aux informations d'identification de sécurité du Compte AWS, les informations d'identification de sécurité temporaires ne sont valides que pendant une heure. Vous pouvez spécifier la durée de session uniquement si vous utilisez les informations d'identification des utilisateurs IAM pour demander une session.

Exemple

L'exemple de PHP suivant liste les clés d'objet dans le compartiment spécifié grâce aux informations d'identification de sécurité temporaires. L'exemple obtient des informations d'identification de sécurité temporaires pour une session d'une heure par défaut, et les utilise pour envoyer une demande authentifiée à Amazon S3. Pour plus d'informations sur l'API AWS SDK pour Ruby, consultez [AWS SDK pour Ruby - Version 2](#).

Si vous souhaitez tester l'exemple à l'aide des informations d'identification utilisateur IAM, vous devez créer un utilisateur IAM sous votre nom. Compte AWS Pour plus d'informations sur la création d'un utilisateur IAM, consultez la section [Création de votre premier utilisateur IAM et d'un groupe d'administrateurs](#) dans le Guide de l'utilisateur IAM. Pour obtenir des exemples de configuration de la durée de la session lors de l'utilisation des informations d'identification des utilisateurs IAM pour demander une session, veuillez consulter [Envoi de demandes à l'aide des informations d'identification temporaires des utilisateurs IAM](#).

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

$sts = new StsClient([
    'version' => 'latest',
```

```

        'region' => 'us-east-1'
    ]);

    $sessionToken = $sts->getSessionToken();

    $s3 = new S3Client([
        'region' => 'us-east-1',
        'version' => 'latest',
        'credentials' => [
            'key'      => $sessionToken['Credentials']['AccessKeyId'],
            'secret'   => $sessionToken['Credentials']['SecretAccessKey'],
            'token'    => $sessionToken['Credentials']['SessionToken']
        ]
    ]);

    $result = $s3->listBuckets();

    try {
        // Retrieve a paginator for listing objects.
        $objects = $s3->getPaginator('ListObjects', [
            'Bucket' => $bucket
        ]);

        echo "Keys retrieved!" . PHP_EOL;

        // List objects
        foreach ($objects as $object) {
            echo $object['Key'] . PHP_EOL;
        }
    } catch (S3Exception $e) {
        echo $e->getMessage() . PHP_EOL;
    }
}

```

Ruby

Un ou plusieurs utilisateurs IAM Compte AWS peuvent demander des informations d'identification de sécurité temporaires AWS SDK pour Ruby et les utiliser pour accéder à Amazon S3. Ces informations d'identification expirent à la fin de la session.

Par défaut, la session dure une heure. Si vous utilisez les informations d'identification des utilisateurs IAM, vous pouvez spécifier la durée (de 15 minutes à la durée maximale de session pour le rôle) lors de la demande d'informations d'identification de sécurité temporaires. Pour

en savoir plus sur les informations d'identification de sécurité temporaires, veuillez consulter [Informations d'identification de sécurité temporaires](#) dans le Guide de l'utilisateur IAM. Pour plus d'informations sur la création de demandes, consultez [Demandes](#).

 Note

Si vous obtenez des informations d'identification de sécurité temporaires grâce aux informations d'identification de sécurité de votre Compte AWS, les informations d'identification de sécurité temporaires ne sont valides que pendant une heure. Vous pouvez spécifier une durée de session uniquement si vous utilisez les informations d'identification des utilisateurs IAM pour demander une session.

L'exemple de code Ruby suivant crée un utilisateur temporaire afin de répertorier les éléments d'un compartiment spécifié pendant une heure. Pour utiliser cet exemple, vous devez disposer AWS d'informations d'identification dotées des autorisations nécessaires pour créer de nouveaux AWS Security Token Service (AWS STS) clients et répertorier les compartiments Amazon S3.

```
# Prerequisites:
# - A user in AWS Identity and Access Management (IAM). This user must
#   be able to assume the following IAM role. You must run this code example
#   within the context of this user.
# - An existing role in IAM that allows all of the Amazon S3 actions for all of the
#   resources in this code example. This role must also trust the preceding IAM
#   user.
# - An existing S3 bucket.

require 'aws-sdk-core'
require 'aws-sdk-s3'
require 'aws-sdk-iam'

# Checks whether a user exists in IAM.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Boolean] true if the user exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless user_exists?(iam_client, 'my-user')
def user_exists?(iam_client, user_name)
```

```

    response = iam_client.get_user(user_name: user_name)
    return true if response.user.user_name
  rescue Aws::IAM::Errors::NoSuchEntity
    # User doesn't exist.
  rescue StandardError => e
    puts 'Error while determining whether the user ' \
        "'#{user_name}' exists: #{e.message}"
  end

# Creates a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS::IAM::Types::User] The new user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = create_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def create_user(iam_client, user_name)
  response = iam_client.create_user(user_name: user_name)
  response.user
rescue StandardError => e
  puts "Error while creating the user '#{user_name}': #{e.message}"
end

# Gets a user in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [AWS::IAM::Types::User] The existing user.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam_client, 'my-user')
#   exit 1 unless user.user_name
def get_user(iam_client, user_name)
  response = iam_client.get_user(user_name: user_name)
  response.user
rescue StandardError => e
  puts "Error while getting the user '#{user_name}': #{e.message}"
end

# Checks whether a role exists in IAM.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.

```

```

# @param role_name [String] The role's name.
# @return [Boolean] true if the role exists; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-west-2')
#   exit 1 unless role_exists?(iam_client, 'my-role')
def role_exists?(iam_client, role_name)
  response = iam_client.get_role(role_name: role_name)
  return true if response.role.role_name
rescue StandardError => e
  puts 'Error while determining whether the role ' \
    "'#{role_name}' exists: #{e.message}"
end

# Gets credentials for a role in IAM.
#
# @param sts_client [Aws::STS::Client] An initialized AWS STS client.
# @param role_arn [String] The role's Amazon Resource Name (ARN).
# @param role_session_name [String] A name for this role's session.
# @param duration_seconds [Integer] The number of seconds this session is valid.
# @return [AWS::AssumeRoleCredentials] The credentials.
# @example
#   sts_client = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_credentials(
#     sts_client,
#     'arn:aws:iam::123456789012:role/AmazonS3ReadOnly',
#     'ReadAmazonS3Bucket',
#     3600
#   )
#   exit 1 if credentials.nil?
def get_credentials(sts_client, role_arn, role_session_name, duration_seconds)
  Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: role_session_name,
    duration_seconds: duration_seconds
  )
rescue StandardError => e
  puts "Error while getting credentials: #{e.message}"
end

# Checks whether a bucket exists in Amazon S3.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The name of the bucket.

```

```

# @return [Boolean] true if the bucket exists; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless bucket_exists?(s3_client, 'amzn-s3-demo-bucket')
def bucket_exists?(s3_client, bucket_name)
  response = s3_client.list_buckets
  response.buckets.each do |bucket|
    return true if bucket.name == bucket_name
  end
rescue StandardError => e
  puts "Error while checking whether the bucket '#{bucket_name}' " \
    "exists: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'amzn-s3-demo-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."
  response = s3_client.list_objects_v2(
    bucket: bucket_name,
    max_keys: 50
  )

  if response.count.positive?
    puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
    puts 'Name => ETag'
    response.contents.each do |obj|
      puts "#{obj.key} => #{obj.etag}"
    end
  else
    puts "No objects in the bucket named '#{bucket_name}'."
  end
  true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end

```

Demandes grâce aux informations d'identification de sécurité temporaires de l'utilisateur fédéré

Vous pouvez demander des informations d'identification de sécurité temporaires et les fournir à vos utilisateurs fédérés ou à vos applications qui ont besoin d'accéder à vos AWS ressources. Cette section fournit des exemples de la manière dont vous pouvez utiliser le AWS SDK pour obtenir des informations d'identification de sécurité temporaires pour vos utilisateurs ou applications fédérés et envoyer des demandes authentifiées à Amazon S3 à l'aide de ces informations d'identification. Pour obtenir la liste des AWS SDK disponibles, consultez la section [Exemples de code et bibliothèques](#).

Note

L'utilisateur Compte AWS et un utilisateur IAM peuvent demander des informations d'identification de sécurité temporaires pour les utilisateurs fédérés. Toutefois, par mesure de sécurité, seul un utilisateur IAM muni des autorisations nécessaires peut demander ces informations d'identification de sécurité temporaires, pour veiller à ce que l'utilisateur fédéré obtienne autant d'autorisations que l'utilisateur IAM. Dans certaines applications, vous choisirez peut-être de créer un utilisateur IAM avec des autorisations spécifiques, afin d'accorder uniquement des informations d'identification de sécurité temporaires à vos utilisateurs fédérés et vos applications.

Java

Vous pouvez fournir des informations d'identification de sécurité temporaires à vos utilisateurs et applications fédérés afin qu'ils puissent envoyer des demandes authentifiées pour accéder à vos AWS ressources. Lorsque vous demandez ces informations d'identification de sécurité temporaires, vous devez fournir un nom d'utilisateur et une stratégie IAM décrivant les autorisations de ressources que vous souhaitez accorder. Par défaut, la session dure une heure. Vous pouvez explicitement définir une valeur de durée différente lorsque vous demandez des informations d'identification de sécurité temporaires pour les utilisateurs fédérés et les applications.

Note

Lorsque vous demandez des informations d'identification de sécurité temporaires pour des utilisateurs fédérés et des applications, nous vous recommandons, pour plus de sécurité, d'utiliser un utilisateur IAM dédié disposant uniquement des autorisations

d'accès nécessaires. L'utilisateur temporaire que vous créez ne peut pas obtenir plus d'autorisations que l'utilisateur IAM ayant demandé les informations d'identification de sécurité temporaires. Pour de plus amples informations, veuillez consulter [Questions fréquentes \(FAQ\) Gestion des identités et des accès AWS](#).

Pour fournir des informations d'identification de sécurité et envoyer une demande authentifiée pour accéder aux ressources, procédez comme suit :

- Créez une instance de la classe `AWSSecurityTokenServiceClient`.
- Lancez une session en appelant la méthode `getFederationToken()` du client STS (Security Token Service). Fournissez des informations de session, y compris le nom d'utilisateur et une stratégie IAM, que vous souhaitez attacher aux informations d'identification temporaires. Vous pouvez renseigner une durée de session optionnelle. Cette méthode renvoie les informations d'identification de sécurité temporaires.
- Regroupez les informations d'identification de sécurité temporaires dans une instance de l'objet `BasicSessionCredentials`. Vous utilisez cet objet pour fournir les informations d'identification de sécurité temporaires à votre client Amazon S3.
- Créez une instance de la classe `AmazonS3Client` à l'aide des informations d'identification de sécurité temporaires. Vous envoyez des demandes à Amazon S3 grâce à ce client. Si vous envoyez des demandes à l'aide d'informations d'identification expirées, Amazon S3 renvoie une erreur.

Exemple

L'exemple répertorie les clés dans le compartiment S3 spécifié. Dans l'exemple, vous obtenez des informations d'identification de sécurité temporaires pour une session de deux heures pour votre utilisateur fédéré et les utilisez pour envoyer des demandes authentifiées à Amazon S3. Pour exécuter cet exemple, vous devez créer un utilisateur IAM avec une politique associée qui lui permet de demander des informations d'identification de sécurité temporaires et de répertorier vos AWS ressources. La stratégie suivante permet d'effectuer ceci :

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"]
  },
```

```
        "Effect": "Allow",
        "Resource": "*"
    }
]
```

Pour de plus amples informations sur la création d'un utilisateur IAM, veuillez consulter [Création de votre premier groupe d'utilisateurs et d'administrateurs IAM](#) dans le Guide de l'utilisateur IAM.

Après avoir créé un utilisateur IAM et attaché la stratégie précédente, vous pouvez exécuter l'exemple suivant. Pour obtenir des instructions sur la création et le test d'un échantillon de travail, voir [Getting Started](#) dans le Guide du AWS SDK pour Java développeur.

```
import com.amazonaws.AmazonServiceException;
import com.amazonaws.SdkClientException;
import com.amazonaws.auth.AWSSStaticCredentialsProvider;
import com.amazonaws.auth.BasicSessionCredentials;
import com.amazonaws.auth.policy.Policy;
import com.amazonaws.auth.policy.Resource;
import com.amazonaws.auth.policy.Statement;
import com.amazonaws.auth.policy.Statement.Effect;
import com.amazonaws.auth.policy.actions.S3Actions;
import com.amazonaws.auth.profile.ProfileCredentialsProvider;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.securitytoken.AWSSecurityTokenService;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClientBuilder;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;

import java.io.IOException;

public class MakingRequestsWithFederatedTempCredentials {

    public static void main(String[] args) throws IOException {
        Regions clientRegion = Regions.DEFAULT_REGION;
        String bucketName = "**** Specify bucket name ****";
        String federatedUser = "**** Federated user name ****";
        String resourceARN = "arn:aws:s3:::" + bucketName;
```

```
try {
    AWSSecurityTokenService stsClient = AWSSecurityTokenServiceClientBuilder
        .standard()
        .withCredentials(new ProfileCredentialsProvider())
        .withRegion(clientRegion)
        .build();

    GetFederationTokenRequest getFederationTokenRequest = new
GetFederationTokenRequest();
    getFederationTokenRequest.setDurationSeconds(7200);
    getFederationTokenRequest.setName(federatedUser);

    // Define the policy and add it to the request.
    Policy policy = new Policy();
    policy.withStatements(new Statement(Effect.Allow)
        .withActions(S3Actions.ListObjects)
        .withResources(new Resource(resourceARN)));
    getFederationTokenRequest.setPolicy(policy.toJson());

    // Get the temporary security credentials.
    GetFederationTokenResult federationTokenResult =
stsClient.getFederationToken(getFederationTokenRequest);
    Credentials sessionCredentials = federationTokenResult.getCredentials();

    // Package the session credentials as a BasicSessionCredentials
    // object for an Amazon S3 client object to use.
    BasicSessionCredentials basicSessionCredentials = new
BasicSessionCredentials(
        sessionCredentials.getAccessKeyId(),
        sessionCredentials.getSecretAccessKey(),
        sessionCredentials.getSessionToken());
    AmazonS3 s3Client = AmazonS3ClientBuilder.standard()
        .withCredentials(new
AWSStaticCredentialsProvider(basicSessionCredentials))
        .withRegion(clientRegion)
        .build();

    // To verify that the client works, send a listObjects request using
    // the temporary security credentials.
    ObjectListing objects = s3Client.listObjects(bucketName);
    System.out.println("No. of Objects = " +
objects.getObjectSummaries().size());
} catch (AmazonServiceException e) {
```



```
// The call was transmitted successfully, but Amazon S3 couldn't process
// it, so it returned an error response.
e.printStackTrace();
} catch (SdkClientException e) {
    // Amazon S3 couldn't be contacted for a response, or the client
    // couldn't parse the response from Amazon S3.
    e.printStackTrace();
}
}
```

.NET

Vous pouvez fournir des informations d'identification de sécurité temporaires à vos utilisateurs et applications fédérés afin qu'ils puissent envoyer des demandes authentifiées pour accéder à vos AWS ressources. Lorsque vous demandez ces informations d'identification de sécurité temporaires, vous devez fournir un nom d'utilisateur et une stratégie IAM décrivant les autorisations de ressources que vous souhaitez accorder. Par défaut, la session dure une heure. Vous pouvez explicitement définir une valeur de durée différente lorsque vous demandez des informations d'identification de sécurité temporaires pour les utilisateurs fédérés et les applications. Pour obtenir des informations sur l'envoi de demandes authentifiées, consultez [Demandes](#).

Note

Lorsque vous demandez des informations d'identification de sécurité temporaires pour des applications et des utilisateurs fédérés, pour plus de sécurité, utilisez un utilisateur IAM dédié disposant uniquement des autorisations d'accès nécessaires. L'utilisateur temporaire que vous créez ne peut pas obtenir plus d'autorisations que l'utilisateur IAM ayant demandé les informations d'identification de sécurité temporaires. Pour de plus amples informations, veuillez consulter [Questions fréquentes \(FAQ\) Gestion des identités et des accès AWS](#).

Vous effectuez les actions suivantes :

- Créez une instance du AWS Security Token Service client, `AmazonSecurityTokenServiceClient` classe.

- Lancez une session en invoquant la méthode `GetFederationToken` du client STS. Vous devez fournir des informations de session, y compris le nom d'utilisateur et une stratégie IAM, que vous souhaitez attacher aux informations d'identification temporaires. Vous pouvez également renseigner une durée de session. Cette méthode renvoie les informations d'identification de sécurité temporaires.
- Regroupez les informations d'identification de sécurité temporaires dans une instance de l'objet `SessionAWSCredentials`. Vous utilisez cet objet pour fournir les informations d'identification de sécurité temporaires à votre client Amazon S3.
- Créez une instance de la classe `AmazonS3Client` en fournissant les informations d'identification de sécurité temporaires. Vous utilisez ce client pour envoyer des demandes à Amazon S3. Si vous envoyez des demandes à l'aide d'informations d'identification expirées, Amazon S3 renvoie une erreur.

Exemple

L'exemple C# suivant répertorie les clés dans le compartiment spécifié. Dans l'exemple, vous obtenez des informations d'identification de sécurité temporaires pour une session de deux heures pour votre utilisateur fédéré (Utilisateur 1), et les utilisez pour envoyer des demandes authentifiées à Amazon S3.

- Pour cet exercice, vous créez un utilisateur IAM disposant d'autorisations minimales. À l'aide des informations d'identification de cet utilisateur IAM, vous demandez des informations d'identification temporaires pour d'autres utilisateurs. Cet exemple répertorie uniquement les objets contenus dans un compartiment spécifique. Créez un utilisateur IAM auquel est attachée la stratégie suivante :

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

La politique permet à l'utilisateur IAM de demander des informations d'identification de sécurité temporaires et des autorisations d'accès uniquement pour répertorier vos AWS ressources. Pour de plus amples informations sur la création d'un utilisateur IAM, veuillez consulter [Création de votre groupe d'utilisateurs et d'administrateurs IAM](#) dans le Guide de l'utilisateur IAM.

- Utilisez les informations d'identification de sécurité des utilisateurs IAM pour tester l'exemple suivant. Cet exemple envoie une demande authentifiée à Amazon S3 à l'aide d'informations d'identification de sécurité temporaires. L'exemple indique la stratégie suivante lors de la demande des informations d'identification de sécurité temporaires pour l'utilisateur fédéré (User1), qui limite l'accès aux objets répertoriés dans un compartiment spécifique (YourBucketName). Vous devez mettre à jour la stratégie et fournir votre propre nom de compartiment existant.

```
{
  "Statement": [
    {
      "Sid": "1",
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": "arn:aws:s3:::YourBucketName"
    }
  ]
}
```

- Example

Mettez à jour l'exemple suivant et fournissez le nom de compartiment que vous avez spécifié dans la stratégie d'accès d'utilisateur fédéré précédente. Pour plus d'informations sur la configuration et l'exécution des exemples de code, consultez la section [Getting Started with the AWS SDK for .NET](#) Developer Guide.

```
using Amazon;
using Amazon.Runtime;
using Amazon.S3;
using Amazon.S3.Model;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
```

```

namespace Amazon.DocSamples.S3
{
    class TempFederatedCredentialsTest
    {
        private const string bucketName = "*** bucket name ***";
        // Specify your bucket region (an example region is shown).
        private static readonly RegionEndpoint bucketRegion =
RegionEndpoint.USWest2;
        private static IAmazonS3 client;

        public static void Main()
        {
            ListObjectsAsync().Wait();
        }

        private static async Task ListObjectsAsync()
        {
            try
            {
                Console.WriteLine("Listing objects stored in a bucket");
                // Credentials use the default AWS SDK for .NET credential search
chain.
                // On local development machines, this is your default profile.
                SessionAWSCredentials tempCredentials =
                    await GetTemporaryFederatedCredentialsAsync();

                // Create a client by providing temporary security credentials.
                using (client = new AmazonS3Client(bucketRegion))
                {
                    ListObjectsRequest listObjectRequest = new
ListObjectsRequest();
                    listObjectRequest.BucketName = bucketName;

                    ListObjectsResponse response = await
client.ListObjectsAsync(listObjectRequest);
                    List<S3Object> objects = response.S3Objects;
                    Console.WriteLine("Object count = {0}", objects.Count);

                    Console.WriteLine("Press any key to continue...");
                    Console.ReadKey();
                }
            }
            catch (AmazonS3Exception e)
            {

```

```

        Console.WriteLine("Error encountered ***. Message:'{0}' when
writing an object", e.Message);
    }
    catch (Exception e)
    {
        Console.WriteLine("Unknown encountered on server. Message:'{0}'
when writing an object", e.Message);
    }
}

private static async Task<SessionAWSCredentials>
GetTemporaryFederatedCredentialsAsync()
{
    AmazonSecurityTokenServiceConfig config = new
AmazonSecurityTokenServiceConfig();
    AmazonSecurityTokenServiceClient stsClient =
        new AmazonSecurityTokenServiceClient(
            config);

    GetFederationTokenRequest federationTokenRequest =
        new GetFederationTokenRequest();
    federationTokenRequest.DurationSeconds = 7200;
    federationTokenRequest.Name = "User1";
    federationTokenRequest.Policy = @"{
        ""Statement"":
        [
            {
                ""Sid"":""Stmnt1311212314284"",
                ""Action"":[""s3:ListBucket""],
                ""Effect"":""Allow"",
                ""Resource"":""arn:aws:s3:::" + bucketName + @""
            }
        ]
    }
";

    GetFederationTokenResponse federationTokenResponse =
        await
stsClient.GetFederationTokenAsync(federationTokenRequest);
    Credentials credentials = federationTokenResponse.Credentials;

    SessionAWSCredentials sessionCredentials =
        new SessionAWSCredentials(credentials.AccessKeyId,
            credentials.SecretAccessKey,

```

```
        credentials.SessionToken);  
    return sessionCredentials;  
}  
}  
}
```

PHP

Cette rubrique explique comment utiliser les classes de la version 3 du AWS SDK pour PHP pour demander des informations d'identification de sécurité temporaires pour les utilisateurs fédérés et les utiliser pour accéder aux ressources stockées dans Amazon S3. Pour plus d'informations sur l'API AWS SDK pour Ruby, consultez [AWS SDK pour Ruby - Version 2](#).

Vous pouvez fournir des informations d'identification de sécurité temporaires à vos utilisateurs et applications fédérés afin qu'ils puissent envoyer des demandes authentifiées pour accéder à vos AWS ressources. Lorsque vous demandez ces informations d'identification de sécurité temporaires, vous devez fournir un nom d'utilisateur et une stratégie IAM décrivant les autorisations de ressources que vous souhaitez accorder. Ces informations d'identification expirent à la fin de la session. Par défaut, la session dure une heure. Vous pouvez explicitement définir une valeur de durée différente lorsque vous demandez des informations d'identification de sécurité temporaires pour les utilisateurs fédérés et les applications. Pour en savoir plus sur les informations d'identification de sécurité temporaires, veuillez consulter [Informations d'identification de sécurité temporaires](#) dans le Guide de l'utilisateur IAM. Pour plus d'informations sur la transmission d'informations d'identification de sécurité temporaires à vos utilisateurs fédérés et aux applications, consultez [Demandes](#).

Lorsque vous demandez des informations d'identification de sécurité temporaires pour des utilisateurs fédérés et des applications, nous vous recommandons, pour plus de sécurité, d'utiliser un utilisateur IAM dédié disposant uniquement des autorisations d'accès nécessaires. L'utilisateur temporaire que vous créez ne peut pas obtenir plus d'autorisations que l'utilisateur IAM ayant demandé les informations d'identification de sécurité temporaires. Pour de plus amples informations sur la fédération d'identité, veuillez consulter [FAQ Gestion des identités et des accès AWS](#).

Pour plus d'informations sur l'API AWS SDK pour Ruby, consultez [AWS SDK pour Ruby - Version 2](#).

Exemple

L'exemple de PHP suivant répertorie les clés dans le compartiment spécifié. Dans l'exemple, vous obtenez des informations d'identification de sécurité temporaires pour une session d'une heure pour votre utilisateur fédéré (User1). Vous utilisez ensuite les informations d'identification de sécurité temporaires pour envoyer les demandes authentifiées à Amazon S3.

Pour plus de sécurité, lorsque vous demandez des informations d'identification de sécurité temporaires pour d'autres utilisateurs, vous utilisez les informations d'identification d'un utilisateur IAM disposant des autorisations pour demander des informations d'identification de sécurité temporaires. Pour veiller à ce que cet utilisateur IAM accorde uniquement à l'utilisateur fédéré les autorisations minimales propres aux applications, vous pouvez également limiter les autorisations d'accès de cet utilisateur IAM. Cet exemple répertorie uniquement les objets contenus dans un compartiment spécifique. Créez un utilisateur IAM auquel est attachée la stratégie suivante :

```
{
  "Statement": [{
    "Action": ["s3:ListBucket",
      "sts:GetFederationToken*"],
    "Effect": "Allow",
    "Resource": "*"
  }]
}
```

La politique permet à l'utilisateur IAM de demander des informations d'identification de sécurité temporaires et des autorisations d'accès uniquement pour répertorier vos AWS ressources. Pour de plus amples informations sur la création d'un utilisateur IAM, veuillez consulter [Création de votre premier groupe d'utilisateurs et d'administrateurs IAM](#) dans le Guide de l'utilisateur IAM.

Vous pouvez désormais utiliser les informations d'identification de sécurité des utilisateurs IAM pour tester l'exemple suivant. Cet exemple envoie une demande authentifiée à Amazon S3 à l'aide d'informations d'identification de sécurité temporaires. Lors d'une demande d'informations d'identification de sécurité temporaires pour l'utilisateur fédéré (User1), cet exemple spécifie la stratégie suivante, qui limite l'accès pour répertorier les objets contenus dans un compartiment spécifique. Mettez à jour la stratégie avec le nom de votre compartiment.

```
{
  "Statement": [
```

```
{
    "Sid": "1",
    "Action": ["s3:ListBucket"],
    "Effect": "Allow",
    "Resource": "arn:aws:s3:::YourBucketName"
}
]
```

Dans l'exemple suivant, remplacez `YourBucketName` par le nom de votre compartiment lorsque vous spécifiez une ressource de stratégie.

```
require 'vendor/autoload.php';

use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;

$bucket = '*** Your Bucket Name ***';

// In real applications, the following code is part of your trusted code. It has
// the security credentials that you use to obtain temporary security credentials.
$sts = new StsClient([
    'version' => 'latest',
    'region' => 'us-east-1'
]);

// Fetch the federated credentials.
$sessionToken = $sts->getFederationToken([
    'Name' => 'User1',
    'DurationSeconds' => '3600',
    'Policy' => json_encode([
        'Statement' => [
            'Sid' => 'randomstatementid' . time(),
            'Action' => ['s3:ListBucket'],
            'Effect' => 'Allow',
            'Resource' => 'arn:aws:s3:::' . $bucket
        ]
    ])
]);

// The following will be part of your less trusted code. You provide temporary
// security credentials so the code can send authenticated requests to Amazon S3.
```



```
$s3 = new S3Client([
    'region' => 'us-east-1',
    'version' => 'latest',
    'credentials' => [
        'key' => $sessionToken['Credentials']['AccessKeyId'],
        'secret' => $sessionToken['Credentials']['SecretAccessKey'],
        'token' => $sessionToken['Credentials']['SessionToken']
    ]
]);

try {
    $result = $s3->listObjects([
        'Bucket' => $bucket
    ]);
} catch (S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
```

Ruby

Vous pouvez fournir des informations d'identification de sécurité temporaires à vos utilisateurs et applications fédérés afin qu'ils puissent envoyer des demandes authentifiées pour accéder à vos AWS ressources. Lorsque vous demandez des informations d'identification de sécurité temporaires auprès du service IAM, vous devez fournir un nom d'utilisateur et une stratégie IAM décrivant les autorisations de ressources que vous souhaitez accorder. Par défaut, la session dure une heure. Toutefois, si vous demandez des informations d'identification temporaires à l'aide des informations d'identification des utilisateurs IAM, vous pouvez explicitement définir une valeur de durée différente lorsque vous demandez les informations d'identification de sécurité temporaires pour des utilisateurs fédérés et des applications. Pour plus d'informations sur les informations d'identification de sécurité temporaires pour vos utilisateurs fédérés et applications, consultez [Demandes](#).

Note

Lorsque vous demandez des informations d'identification de sécurité temporaires pour des utilisateurs fédérés et des applications, vous souhaitez peut-être utiliser un utilisateur IAM dédié disposant uniquement des autorisations d'accès nécessaires. L'utilisateur temporaire que vous créez ne peut pas obtenir plus d'autorisations que l'utilisateur IAM ayant demandé les informations d'identification de sécurité temporaires.

Pour de plus amples informations, veuillez consulter [Questions fréquentes \(FAQ\) Gestion des identités et des accès AWS](#).

Exemple

L'exemple de code Ruby suivant autorise un utilisateur fédéré disposant d'un ensemble limité d'autorisations à répertorier des clés dans le compartiment spécifié.

```
# Prerequisites:
# - An existing Amazon S3 bucket.

require 'aws-sdk-s3'
require 'aws-sdk-iam'
require 'json'

# Checks to see whether a user exists in IAM; otherwise,
# creates the user.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param user_name [String] The user's name.
# @return [Aws::IAM::Types::User] The existing or new user.
# @example
#   iam = Aws::IAM::Client.new(region: 'us-west-2')
#   user = get_user(iam, 'my-user')
#   exit 1 unless user.user_name
#   puts "User's name: #{user.user_name}"
def get_user(iam, user_name)
  puts "Checking for a user with the name '#{user_name}'..."
  response = iam.get_user(user_name: user_name)
  puts "A user with the name '#{user_name}' already exists."
  response.user
# If the user doesn't exist, create them.
rescue Aws::IAM::Errors::NoSuchEntity
  puts "A user with the name '#{user_name}' doesn't exist. Creating this user..."
  response = iam.create_user(user_name: user_name)
  iam.wait_until(:user_exists, user_name: user_name)
  puts "Created user with the name '#{user_name}'."
  response.user
rescue StandardError => e
  puts "Error while accessing or creating the user named '#{user_name}':
  #{e.message}"
end
```

```

# Gets temporary AWS credentials for an IAM user with the specified permissions.
#
# @param sts [Aws::STS::Client] An initialized AWS STS client.
# @param duration_seconds [Integer] The number of seconds for valid credentials.
# @param user_name [String] The user's name.
# @param policy [Hash] The access policy.
# @return [Aws::STS::Types::Credentials] AWS credentials for API authentication.
# @example
#   sts = Aws::STS::Client.new(region: 'us-west-2')
#   credentials = get_temporary_credentials(sts, duration_seconds, user_name,
#     {
#       'Version' => '2012-10-17',
#       'Statement' => [
#         'Sid' => 'Stmt1',
#         'Effect' => 'Allow',
#         'Action' => 's3:ListBucket',
#         'Resource' => 'arn:aws:s3:::amzn-s3-demo-bucket'
#       ]
#     }
#   )
#   exit 1 unless credentials.access_key_id
#   puts "Access key ID: #{credentials.access_key_id}"
def get_temporary_credentials(sts, duration_seconds, user_name, policy)
  response = sts.get_federation_token(
    duration_seconds: duration_seconds,
    name: user_name,
    policy: policy.to_json
  )
  response.credentials
rescue StandardError => e
  puts "Error while getting federation token: #{e.message}"
end

# Lists the keys and ETags for the objects in an Amazon S3 bucket.
#
# @param s3_client [Aws::S3::Client] An initialized Amazon S3 client.
# @param bucket_name [String] The bucket's name.
# @return [Boolean] true if the objects were listed; otherwise, false.
# @example
#   s3_client = Aws::S3::Client.new(region: 'us-west-2')
#   exit 1 unless list_objects_in_bucket?(s3_client, 'amzn-s3-demo-bucket')
def list_objects_in_bucket?(s3_client, bucket_name)
  puts "Accessing the contents of the bucket named '#{bucket_name}'..."

```

```

response = s3_client.list_objects_v2(
  bucket: bucket_name,
  max_keys: 50
)

if response.count.positive?
  puts "Contents of the bucket named '#{bucket_name}' (first 50 objects):"
  puts 'Name => ETag'
  response.contents.each do |obj|
    puts "#{obj.key} => #{obj.etag}"
  end
else
  puts "No objects in the bucket named '#{bucket_name}'."
end
true
rescue StandardError => e
  puts "Error while accessing the bucket named '#{bucket_name}': #{e.message}"
end

# Example usage:
def run_me
  region = "us-west-2"
  user_name = "my-user"
  bucket_name = "amzn-s3-demo-bucket"

  iam = Aws::IAM::Client.new(region: region)
  user = get_user(iam, user_name)

  exit 1 unless user.user_name

  puts "User's name: #{user.user_name}"
  sts = Aws::STS::Client.new(region: region)
  credentials = get_temporary_credentials(sts, 3600, user_name,
    {
      'Version' => '2012-10-17',
      'Statement' => [
        'Sid' => 'Stmt1',
        'Effect' => 'Allow',
        'Action' => 's3:ListBucket',
        'Resource' =>
"arn:aws:s3:::#{bucket_name}"
      ]
    })

```

```
exit 1 unless credentials.access_key_id

puts "Access key ID: #{credentials.access_key_id}"
s3_client = Aws::S3::Client.new(region: region, credentials: credentials)

exit 1 unless list_objects_in_bucket?(s3_client, bucket_name)
end

run_me if $PROGRAM_NAME == __FILE__
```

Demandes à l'aide de l'API REST

Cette section contient des informations expliquant comment envoyer des demandes à des points de terminaison Amazon S3 à l'aide de l'API REST. Pour obtenir la liste des points de terminaison Amazon S3, consultez [Régions et points de terminaison](#) dans la Références générales AWS.

Construction de noms d'hôte S3 pour les demandes d'API REST

Les points de terminaison Amazon S3 suivent la structure indiquée ci-dessous :

```
s3.Region.amazonaws.com
```

Les points de terminaison de points d'accès Amazon S3 et les points de terminaison Dual-Stack (double pile) suivent également la structure standard :

- Points d'accès Amazon S - s3-accesspoint.*Region*.amazonaws.com
- Dual-stack - s3.dualstack.*Region*.amazonaws.com

Pour obtenir la liste complète des régions et points de terminaison Amazon S3, consultez [Points de terminaison et quotas Amazon S3](#) dans le Référence générale d'Amazon Web Services.

Demandes de type hébergement virtuel et de type chemin

Lorsque vous envoyez des demandes à l'aide de l'API REST, vous pouvez utiliser des URI de types hébergement virtuel ou chemin pour les points de terminaison Amazon S3. Pour plus d'informations, consultez la section [Path-style Demandes](#).

Exemple Demande de type hébergement virtuel

Voici un exemple de demande de type hébergement virtuel visant à supprimer le fichier `puppy.jpg` du compartiment `examplebucket` dans la Région USA Ouest (Oregon). Pour plus d'informations sur les requêtes de type hébergé virtuel, consultez la section Demandes. [Path-style](#)

```
DELETE /puppy.jpg HTTP/1.1
Host: examplebucket.s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Exemple Path-style demande

Voici un exemple d'une version de type chemin de la même demande.

```
DELETE /examplebucket/puppy.jpg HTTP/1.1
Host: s3.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Amazon S3 prend actuellement en charge l'accès aux URL de type hébergement virtuel et chemin d'accès dans toutes les Régions AWS. Toutefois, les URL de type chemin d'accès seront abandonnées à l'avenir. Pour plus d'informations, consultez la remarque importante suivante.

Pour plus d'informations sur les requêtes de type chemin, consultez [Path-style la section requêtes](#).

Important

Mise à jour (23 septembre 2020) – Pour laisser aux clients le temps nécessaire pour passer à des URL de type hébergement virtuel, nous avons décidé de retarder l'obsolescence des URL de type chemin d'accès. Pour plus d'informations, consultez [Amazon S3 Path Deprecation Plan – The Rest of the Story](#) dans le blog dédié aux actualités d'AWS .

Envoi de demandes à des points de terminaison Dual-Stack à l'aide de l'API REST

Lorsque vous utilisez l'API REST, vous pouvez accéder directement à un point de terminaison Dual-Stack (double pile) si vous utilisez un nom de point de terminaison (URI) de type hébergement virtuel

ou chemin. Tous les noms de points de terminaison Dual-Stack (double pile) Amazon S3 incluent la Région concernée. Contrairement aux points de IPv4-only terminaison standard, les points de terminaison de style hébergé virtuel et de type chemin utilisent des noms de points de terminaison spécifiques à la région.

Exemple Demande de point de terminaison Dual-Stack (double pile) de type hébergement virtuel

Vous pouvez utiliser un point de terminaison de type hébergement virtuel dans votre demande REST, comme indiqué dans l'exemple suivant, qui permet de récupérer l'objet `puppy.jpg` à partir du compartiment `examplebucket` dans la Région USA Ouest (Oregon).

```
GET /puppy.jpg HTTP/1.1
Host: examplebucket.s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Exemple Path-style demande de point de terminaison à double pile

Vous pouvez également utiliser un point de terminaison de type chemin dans votre demande, comme indiqué dans l'exemple ci-dessous.

```
GET /examplebucket/puppy.jpg HTTP/1.1
Host: s3.dualstack.us-west-2.amazonaws.com
Date: Mon, 11 Apr 2016 12:00:00 GMT
x-amz-date: Mon, 11 Apr 2016 12:00:00 GMT
Authorization: authorization string
```

Pour de plus amples informations sur les points de terminaison Dual-Stack, consultez la section [Utilisation des points de terminaison Dual-Stack Amazon S3](#).

Pour plus d'informations sur l'envoi de requêtes à l'aide de l'API REST, consultez les rubriques suivantes.

Rubriques

- [Redirection de demande et API REST](#)
- [Demande de routage](#)

Redirection de demande et API REST

Rubriques

- [Redirections et agents utilisateurs HTTP](#)
- [Redirections et 100-continue](#)
- [Exemple de redirection](#)

Cette section décrit comment traiter les redirections HTTP à l'aide de l'API REST Amazon S3. Pour obtenir des informations générales sur les redirections Amazon S3, veuillez consulter [Demandes](#) dans la Référence d'API Amazon Simple Storage Service.

Redirections et agents utilisateurs HTTP

Les programmes qui utilisent l'API REST Amazon S3 doivent traiter les redirections au niveau de la couche applicative ou de la couche HTTP. De nombreux agents utilisateurs et bibliothèques client HTTP peuvent être configurés pour traiter automatiquement correctement les redirections. Toutefois, de nombreux autres présentent des implémentations de redirection incorrectes ou incomplètes.

Avant de faire confiance à une bibliothèque pour s'acquitter de l'obligation de redirection, testez les cas suivants :

- Vérifiez que tous les en-têtes de demande HTTP sont correctement inclus dans la demande redirigée (la seconde demande après réception d'une redirection), y compris les standards HTTP tels que Authorization et Date.
- Vérifiez que les redirections autres que GET, telles que PUT et DELETE, fonctionnent correctement.
- Vérifiez que les demandes PUT de grande taille suivent correctement les redirections.
- Vérifiez que les demandes PUT suivent correctement les redirections si la réponse 100-continue met longtemps à arriver.

Les agents utilisateurs HTTP qui respectent scrupuleusement la RFC 2616 peuvent nécessiter une confirmation explicite avant de suivre une redirection lorsque la méthode de demande HTTP n'est pas GET ni HEAD. Il est généralement sans danger de suivre les redirections générées automatiquement par Amazon S3, car le système émet des redirections uniquement vers les hôtes situés au sein du domaine amazonaws.com et l'effet de la demande redirigée est identique à celui de la demande d'origine.

Redirections et 100-continue

Pour simplifier le traitement de la redirection, améliorer l'efficacité et éviter les coûts associés à l'envoi répété du corps d'une demande redirigée, configurez l'application pour qu'elle utilise des expressions 100-continue pour les opérations PUT. Lorsque votre application utilise 100-continue, elle n'envoie pas le corps de la demande tant qu'elle ne reçoit pas d'accusé de réception. Si le message est rejeté sur la base des en-têtes, le corps du message n'est pas envoyé. Pour plus d'informations sur 100-continue, consultez la [RFC 2616, section 8.2.3](#)

Note

Selon la RFC 2616, lorsque vous utilisez Expect: Continue avec un serveur HTTP inconnu, vous ne devez pas attendre un temps indéfini avant d'envoyer le corps de la demande. Cela est dû au fait que certains serveurs HTTP ne reconnaissent pas l'expression 100-continue. Toutefois, Amazon S3 reconnaît que la demande contient une expression Expect: Continue et répond avec un statut 100-continue provisoire ou un code de statut final. En outre, aucune erreur de redirection ne surviendra après la réception de l'autorisation provisoire 100-continue. Cela vous aidera à éviter de recevoir une réponse de redirection alors que vous écrivez encore le corps de la demande.

Exemple de redirection

Cette section fournit un exemple d'interaction client-serveur utilisant des redirections HTTP et 100-continue.

Voici un exemple de commande PUT pour le compartiment `quotes.s3.amazonaws.com`.

```
PUT /nelson.txt HTTP/1.1
Host: quotes.s3.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 retourne les informations suivantes :

```
HTTP/1.1 307 Temporary Redirect
```

```
Location: http://quotes.s3-4c25d83b.amazonaws.com/nelson.txt?rk=8d47490b
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Mon, 15 Oct 2007 22:18:46 GMT
```

Server: AmazonS3

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the
    specified temporary endpoint. Continue to use the
    original request endpoint for future requests.
  </Message>
  <Endpoint>quotes.s3-4c25d83b.amazonaws.com</Endpoint>
  <Bucket>quotes</Bucket>
</Error>
```

Le client suit la réponse de redirection et émet une nouvelle demande adressée au point de terminaison temporaire `quotes.s3-4c25d83b.amazonaws.com`.

```
PUT /nelson.txt?rk=8d47490b HTTP/1.1
Host: quotes.s3-4c25d83b.amazonaws.com
Date: Mon, 15 Oct 2007 22:18:46 +0000

Content-Length: 6
Expect: 100-continue
```

Amazon S3 retourne une expression 100-continue indiquant que le client doit à présent envoyer le corps de la demande.

```
HTTP/1.1 100 Continue
```

Le client envoie le corps de la demande.

```
ha ha\n
```

Amazon S3 retourne la réponse finale.

```
HTTP/1.1 200 OK
Date: Mon, 15 Oct 2007 22:18:48 GMT
```

```
ETag: "a2c8d6b872054293afd41061e93bc289"  
Content-Length: 0  
Server: AmazonS3
```

Demande de routage

Programmes qui adressent des requêtes à des buckets créés à l'aide de l'[CreateBucket](#) API et qui incluent des redirections [CreateBucketConfiguration](#) indispensables. De plus, certains clients qui ne respectent pas la durée de vie des demandes DNS risquent de rencontrer des problèmes.

Cette section décrit les demandes de routage et les erreurs DNS à prendre en compte lors de la conception de votre service ou de votre application à utiliser avec Amazon S3.

Redirection de demande et API REST

Amazon S3 utilise le système de noms de domaine (DNS) pour diriger les demandes vers les installations susceptibles de les traiter. Ce système fonctionne de façon efficace, mais des erreurs de routage temporaires peuvent survenir. Si une demande arrive à une destination Amazon S3 erronée, Amazon S3 répond par une redirection temporaire qui indique au demandeur de renvoyer la demande à un nouveau point de terminaison. Si une demande est mal formulée, Amazon S3 utilise des redirections permanentes pour fournir des instructions sur la manière de formuler correctement la demande.

Important

Pour utiliser cette fonction, vous devez disposer d'une application capable de gérer les réponses de redirection Amazon S3. La seule exception concerne les applications qui fonctionnent avec des compartiments qui ont été créés sans `<CreateBucketConfiguration>`. Pour plus d'informations sur les contraintes de localisation, consultez la section [Accès à un compartiment Amazon S3 et sa mise en vente](#). Pour toutes les Régions lancées après le 20 mars 2019, si une demande arrive à l'emplacement Amazon S3 incorrect, Amazon S3 renvoie une erreur HTTP 400 Requête incorrecte.

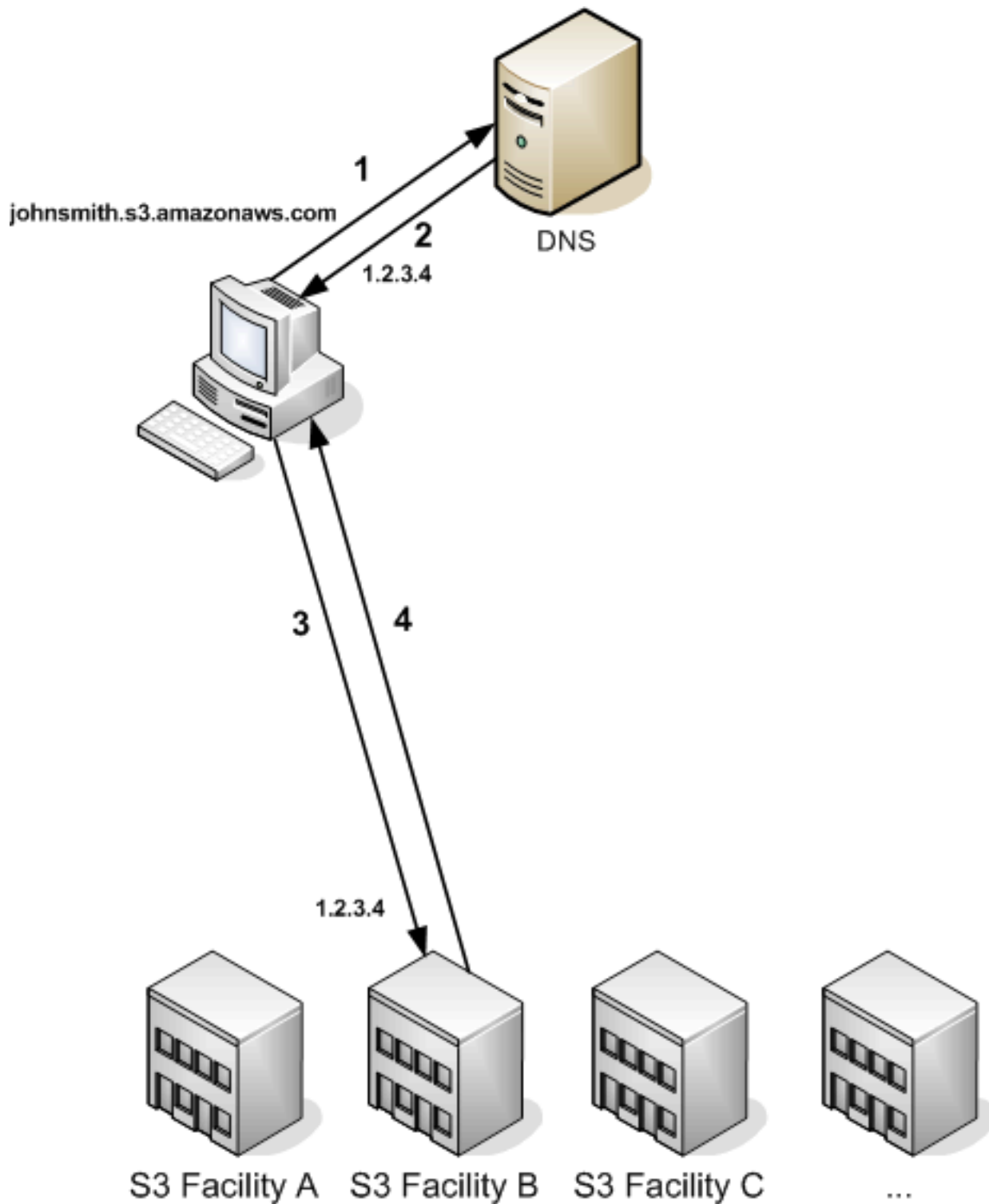
Pour plus d'informations sur l'activation ou la désactivation d'un Région AWS, consultez [Régions AWS et Points de terminaison](#) dans le. Références générales AWS

Rubriques

- [routage DNS](#)
- [Redirection de demande temporaire](#)
- [Redirection de demande permanente](#)
- [Exemples de redirection de demande](#)

routage DNS

Le routage DNS achemine les demandes vers les installations Amazon S3 appropriées. La figure et la procédure suivantes montrent un exemple de routage DNS.



Étapes des demandes de routage DNS

1. Le client effectue une demande DNS pour obtenir un objet stocké sur Amazon S3.

2. Le client reçoit une ou plusieurs adresses IP des installations qui peuvent traiter la demande. Dans cet exemple, l'adresse IP est pour l'installation B.
3. Le client effectue une demande adressée à l'installation B Amazon S3.
4. L'installation B renvoie une copie de l'objet au client.

Redirection de demande temporaire

Une redirection temporaire est un type de réponse d'erreur qui indique au demandeur qu'il doit renvoyer la demande à un autre point de terminaison. En raison de la nature distribuée d'Amazon S3, les demandes peuvent être acheminées temporairement vers la mauvaise installation. Cela a plus de chances de se produire immédiatement après la création ou la suppression de compartiments.

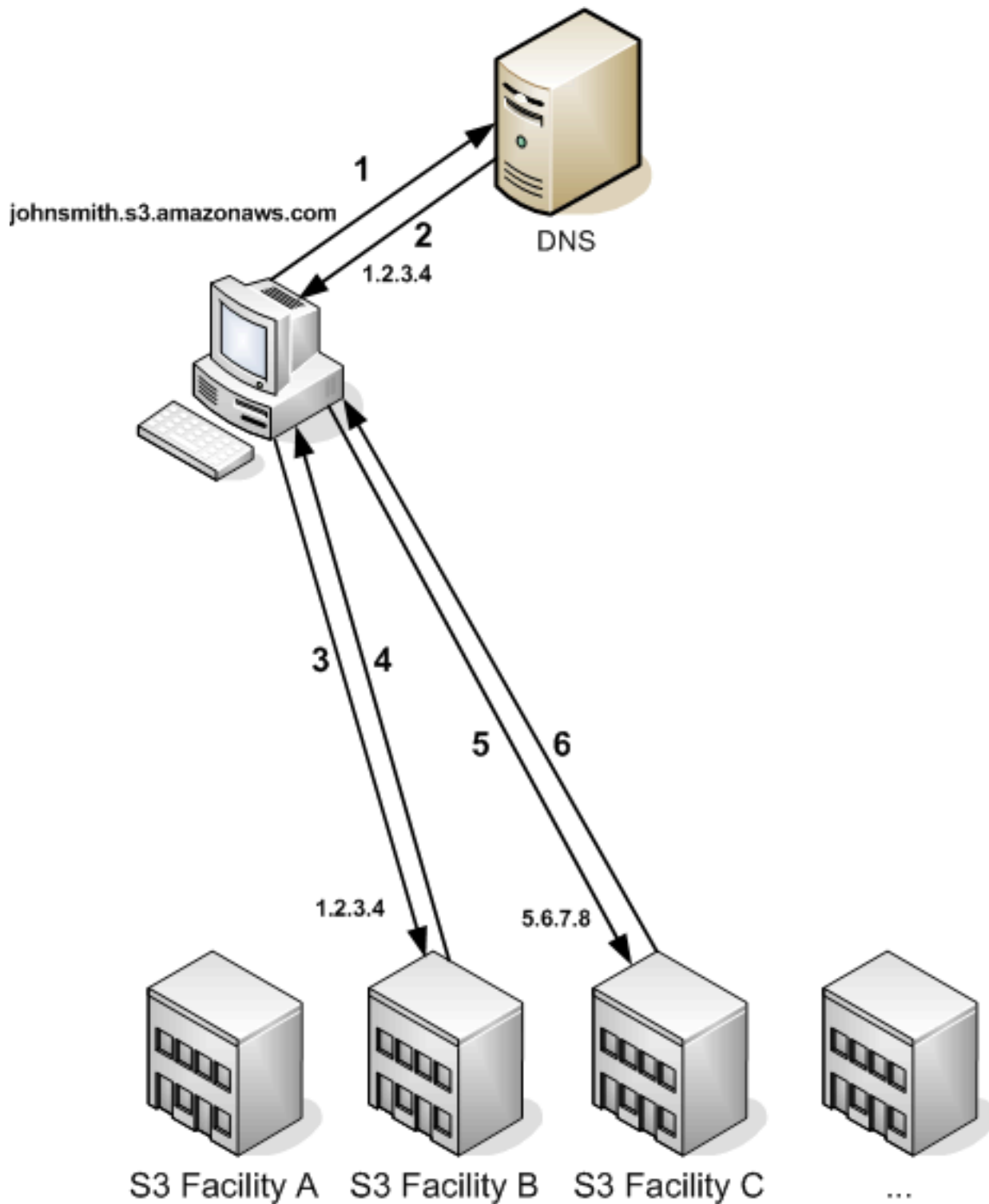
Par exemple, si vous créez un compartiment et adressez immédiatement une demande à ce compartiment, vous pouvez recevoir une redirection temporaire, selon la contrainte d'emplacement du compartiment. Si vous avez créé le bucket dans l'est des États-Unis (Virginie du Nord) Région AWS, vous ne verrez pas la redirection car il s'agit également du point de terminaison Amazon S3 par défaut.

Toutefois, si le compartiment est créé dans une autre Région, toutes les demandes adressées au compartiment iront au point de terminaison par défaut lors de la propagation de l'entrée DNS du compartiment. Le point de terminaison par défaut redirige la demande vers le point de terminaison correct avec une réponse HTTP 302. Les redirections temporaires contiennent un URI vers l'installation correcte, que vous pouvez utiliser pour renvoyer immédiatement la demande.

Important

Ne réutilisez pas un point de terminaison fourni par une réponse de redirection précédente. Cela peut sembler fonctionner (même pendant de longues périodes), mais cela peut fournir des résultats imprévisibles, voire entraîner un échec sans préavis.

La figure et la procédure suivantes montrent un exemple de redirection temporaire.



Étapes d'une redirection de demande temporaire

1. Le client effectue une demande DNS pour obtenir un objet stocké sur Amazon S3.
2. Le client reçoit une ou plusieurs adresses IP des installations qui peuvent traiter la demande.

3. Le client effectue une demande adressée à l'installation B Amazon S3.
4. L'installation B renvoie une redirection indiquant que l'objet est disponible à partir de l'emplacement C.
5. Le client renvoie la demande à l'installation C.
6. L'installation C renvoie une copie de l'objet.

Redirection de demande permanente

Une redirection permanente indique que votre demande s'adressait de façon inappropriée à une ressource. Par exemple, des redirections permanentes se produisent si vous utilisez une demande de type chemin pour accéder à un compartiment qui a été créé avec `<CreateBucketConfiguration>`. Pour plus d'informations, consultez la section [Accès à un compartiment Amazon S3 et sa mise en vente](#).

Pour vous aider à détecter ces erreurs au cours du développement, ce type de redirection ne contient pas d'en-tête HTTP d'emplacement vous permettant de suivre automatiquement la demande vers l'emplacement correct. Veuillez consulter le document d'erreur XML obtenu comme aide pour utiliser le point de terminaison Amazon S3 correct.

Exemples de redirection de demande

Voici des exemples de réponses à des redirections de demande temporaires.

Réponse de redirection temporaire d'API REST

```
HTTP/1.1 307 Temporary Redirect
Location: http://awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com/photos/puppy.jpg?rk=e2c69a31
Content-Type: application/xml
Transfer-Encoding: chunked
Date: Fri, 12 Oct 2007 01:12:56 GMT
Server: AmazonS3

<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>TemporaryRedirect</Code>
  <Message>Please re-send this request to the specified temporary endpoint. Continue to use the original request endpoint for future requests.</Message>
  <Endpoint>awsexamplebucket1.s3-gztb4pa9sq.amazonaws.com</Endpoint>
```


</Error>

Réponse de redirection temporaire d'API SOAP

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les kits de AWS développement logiciel (SDK).

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.TemporaryRedirect</Faultcode>
    <Faultstring>Please re-send this request to the specified temporary endpoint.
    Continue to use the original request endpoint for future requests.</Faultstring>
    <Detail>
      <Bucket>images</Bucket>
      <Endpoint>s3-gztlb4pa9sq.amazonaws.com</Endpoint>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

Considérations DNS

Une disponibilité extrêmement élevée est l'un des prérequis pour la conception d'Amazon S3. Afin de répondre à cette exigence, nous devons actualiser les adresses IP associées au point de terminaison Amazon S3 dans le DNS si nécessaire. Ces changements se répercutent automatiquement sur les clients de courte durée, mais pas sur certains clients de longue durée. Long-lived les clients devront prendre des mesures spéciales pour résoudre périodiquement le point de terminaison Amazon S3 afin de bénéficier de ces modifications. Pour plus d'informations sur les machines virtuelles, consultez les rubriques suivantes :

- Pour Java, la JVM de Sun met en cache les recherches DNS pour toujours par défaut ; consultez la section « InetAddress Mise en cache » de [la InetAddress documentation](#) pour savoir comment modifier ce comportement.
- Pour PHP, l'ordinateur virtuel persistant qui utilise les configurations de développement les plus couramment utilisées met en cache les recherches jusqu'à ce que l'ordinateur virtuel redémarre. Accédez à [la documentation de téléchargement de HostByName PHP](#).

Développement avec Amazon S3 à l'aide du AWS CLI

Les AWS CLI commandes Amazon S3 sont organisées en différents ensembles dans AWS CLI Command Reference et chacun possède ses propres commandes disponibles. Si vous ne trouvez pas la commande que vous recherchez dans un ensemble, cochez l'un des autres ensembles. Les différents sets sont les suivants :

- **s3**— Décrit les commandes de haut niveau permettant de travailler avec des objets et des compartiments. Il s'agit notamment des actions de copie, de liste et de déplacement. Pour une liste complète des commandes de cet ensemble, voir [s3](#) dans la référence des AWS CLI commandes.
- **s3api**— Décrit les commandes de bas niveau qui gèrent les ressources S3 telles que les compartiments, les objets, les sessions et les téléchargements partitionnés. Ces commandes correspondent aux [opérations de l'API](#) Amazon S3. Pour une liste complète des commandes CLI de cet ensemble, consultez [s3api](#) dans la référence des AWS CLI commandes.
- **s3control**— Décrit les commandes de bas niveau qui gèrent toutes les autres ressources S3 telles que Access Grants, Storage Lens groups et les buckets Amazon S3 on Outposts. Ces commandes correspondent aux opérations de l'[API](#) Amazon S3 Control. Pour une liste complète des commandes CLI de cet ensemble, consultez [s3control](#) dans la référence des AWS CLI commandes.

Si vous rencontrez des problèmes lors de la configuration de AWS CLI, consultez [Comment résoudre l'erreur « Impossible de localiser les informations d'identification » dans Amazon S3](#) dans le centre de AWS re:Post connaissances.

Note

Les services dans AWS, tels qu'Amazon S3, nécessitent que vous fournissiez des informations d'identification lorsque vous y accédez. Le service peut ainsi déterminer si vous avez les autorisations pour accéder à ses ressources. La console exige votre mot de passe. Vous pouvez créer des clés d'accès pour accéder Compte AWS à l'API AWS CLI or. Cependant, n'accédez pas à AWS l'aide des informations d'identification de votre Compte AWS. Nous vous recommandons plutôt d'utiliser Gestion des identités et des accès AWS (IAM). Créez un utilisateur IAM, ajoutez-le à un groupe IAM avec des autorisations administratives, puis attribuez-lui ces autorisations. Vous pouvez alors accéder à AWS à l'aide d'une URL spéciale et des informations d'identification de cet utilisateur IAM. Pour

obtenir des instructions, veuillez accéder à [Création de votre premier groupe d'utilisateurs et d'administrateurs IAM](#) dans le Guide de l'utilisateur IAM.

En savoir plus sur le AWS CLI

Pour en savoir plus sur le AWS, consultez les ressources suivantes :

- [AWS Command Line Interface](#)
- [AWS Command Line Interface Guide de l'utilisateur pour la version 2](#)

Développement avec Amazon S3 à l'aide du AWS Kits SDK

AWS des kits de développement logiciel (SDK) sont disponibles pour de nombreux langages de programmation courants. Chaque kit SDK fournit une API, des exemples de code et de la documentation qui facilitent la création d'applications par les développeurs dans leur langage préféré.

Note

Vous pouvez l'utiliser AWS Amplify pour le développement complet d'applications Web et mobiles de bout en bout. Amplify Storage intègre de manière fluide les fonctionnalités de stockage et de gestion de fichiers dans les applications Web et mobiles frontales, basées sur Amazon S3. Pour plus d'informations, consultez la section [Stockage](#) dans le guide de l'utilisateur d'Amplify.

Documentation SDK	Exemples de code
AWS SDK pour C++	AWS SDK pour C++ exemples de code
AWS CLI	AWS CLI exemples de code
AWS SDK pour Go	AWS SDK pour Go exemples de code
AWS SDK pour Java	AWS SDK pour Java exemples de code
AWS SDK pour JavaScript	AWS SDK pour JavaScript exemples de code

Documentation SDK	Exemples de code
AWS SDK pour Kotlin	AWS SDK pour Kotlin exemples de code
AWS SDK pour .NET	AWS SDK pour .NET exemples de code
AWS SDK pour PHP	AWS SDK pour PHP exemples de code
Outils AWS pour PowerShell	Outils pour des exemples PowerShell de code
AWS SDK pour Python (Boto3)	AWS SDK pour Python (Boto3) exemples de code
AWS SDK pour Ruby	AWS SDK pour Ruby exemples de code
AWS SDK pour Rust	AWS SDK pour Rust exemples de code
AWS SDK pour SAP ABAP	AWS SDK pour SAP ABAP exemples de code
AWS SDK pour Swift	AWS SDK pour Swift exemples de code

Pour plus d'exemples, consultez [Exemples de code pour Amazon S3 utilisant AWS Kits SDK](#).

Interfaces de programmation SDK

Chaque AWS SDK fournit une ou plusieurs interfaces programmatiques permettant de travailler avec Amazon S3. Chaque SDK fournit une interface de bas niveau pour Amazon S3, avec des méthodes qui ressemblent étroitement aux opérations d'API. Certains kits SDK fournissent des interfaces de haut niveau pour Amazon S3, qui sont des abstractions destinées à simplifier les cas d'utilisation courants.

Par exemple, lorsque vous effectuez un chargement en plusieurs parties à l'aide des opérations d'API de bas niveau, vous utilisez une opération pour lancer le téléchargement, une autre opération pour télécharger des parties et une opération finale pour terminer le téléchargement. Une opération d'API de téléchargement partitionné de haut niveau vous permet d'effectuer toutes les opérations requises pour le téléchargement en un seul appel d'API. Pour des exemples, consultez la section [Chargement d'un objet à l'aide du chargement partitionné](#) dans le Guide de l'utilisateur Amazon S3.

Low-level Les opérations d'API permettent de mieux contrôler le téléchargement. Nous vous recommandons d'utiliser les opérations d'API de bas niveau si vous devez suspendre et reprendre

des téléchargements, modifier la taille des parties pendant le téléchargement ou commencer des téléchargements alors que vous ne connaissez pas la taille des données à l'avance.

Spécification de la version de signature lors de l'authentification des demandes

Amazon S3 ne prend en charge que AWS la version 4 de Signature dans la plupart des cas Régions AWS. Dans certaines versions plus anciennes Régions AWS, Amazon S3 prend en charge Signature Version 4 et Signature Version 2. Cependant, Signature Version 2 est en cours de désactivation (obsolète). Pour de plus amples informations sur la fin de la prise en charge de Signature Version 2, veuillez consulter [AWS Signature Version 2 désactivée \(obsolète\) pour Amazon S3](#).

Pour obtenir la liste de toutes les régions Amazon S3 et savoir quelles sont les versions de signatures qu'elles prennent en charge, veuillez consulter [Régions et points de terminaisons](#) dans les Références générales AWS .

Dans tous les Régions AWS cas, les AWS SDK utilisent Signature Version 4 par défaut pour authentifier les demandes. Si vous utilisez AWS des SDK publiés avant mai 2016, vous devrez peut-être demander Signature Version 4, comme indiqué dans le tableau suivant.

Kit SDK	Demande de la Signature Version 4 pour l'authentification des demandes
AWS CLI	Pour le profil par défaut, exécutez la commande suivante : <pre data-bbox="609 1318 1507 1423">\$ aws configure set default.s3.signature_version s3v4</pre> Pour un profil personnalisé, exécutez la commande suivante : <pre data-bbox="609 1535 1507 1640">\$ aws configure set profile.your_profile_name.s3.signature_version s3v4</pre>
Kit SDK Java	Ajoutez la chaîne suivante à votre code : <pre data-bbox="609 1766 1507 1871">System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");</pre>

Kit SDK	Demande de la Signature Version 4 pour l'authentification des demandes
	Sinon, vous pouvez spécifier la commande ci-dessous dans la ligne de commande : <pre data-bbox="597 380 1507 457">-Dcom.amazonaws.services.s3.enableV4</pre>
Kit SDK PHP	Définissez le paramètre <code>signature</code> sur <code>v4</code> lors de la création du client de service Amazon S3 pour le kit SDK PHP v2 : <pre data-bbox="597 617 1507 894"><?php \$client = S3Client::factory(['region' => 'YOUR-REGION', 'version' => 'latest', 'signature' => 'v4']);</pre> Lorsque vous utilisez le kit SDK PHP v3, définissez le paramètre <code>signature_version</code> sur <code>v4</code> lors de la création du client de service Amazon S3 : <pre data-bbox="597 1100 1507 1377"><?php \$s3 = new Aws\S3\S3Client(['version' => '2006-03-01', 'region' => 'YOUR-REGION', 'signature_version' => 'v4']);</pre>
Python-Boto SDK	Spécifiez la chaîne suivante dans le fichier de configuration par défaut boto: <pre data-bbox="597 1541 1507 1612">[s3] use-sigv4 = True</pre>

Kit SDK	Demande de la Signature Version 4 pour l'authentification des demandes
Kit SDK Ruby	Kit SDK Ruby – Version 1 : définissez le paramètre <code>:s3_signature_version</code> sur <code>:v4</code> lors de la création du client: <pre>s3 = AWS::S3::Client.new(:s3_signature_version => :v4)</pre> Kit SDK Ruby – Version 3 : définissez le paramètre <code>signature_version</code> sur <code>v4</code> lors de la création du client: <pre>s3 = Aws::S3::Client.new(signature_version: 'v4')</pre>
Kit SDK .NET	Ajoutez la chaîne suivante au code avant de créer le client Amazon S3 : <pre>AWSConfigsS3.UseSignatureVersion4 = true;</pre> Sinon, vous pouvez ajouter le code ci-dessous au fichier de configuration : <pre><appSettings> <add key="AWS.S3.UseSignatureVersion4" value="true" /> </appSettings></pre>

AWS Signature Version 2 désactivée (obsolète) pour Amazon S3

Signature Version 2 est en cours de désactivation (obsolète) dans Amazon S3. Une fois cette version devenue obsolète, Amazon S3 n'acceptera plus que les demandes d'API signées avec Signature Version 4.

Cette section fournit des réponses aux questions fréquentes concernant la fin de la prise en charge de Signature Version 2.

Qu'est-ce que la version 2/4 de signature et que signifie signer des demandes ?

Le processus de signature de Signature Version 2 ou Signature Version 4 est utilisé pour authentifier vos demandes d'API Amazon S3. La signature des demandes permet à Amazon S3 d'identifier l'auteur de l'envoi de la demande et protège vos demandes contre les utilisateurs malveillants.

Pour plus d'informations sur les AWS demandes de signature, voir [Signer des demandes d' AWS API](#) dans le Références générales AWS.

En quoi consiste la mise à jour ?

Nous prenons actuellement en charge les demandes d'API Amazon S3 qui sont signées avec les processus Signature Version 2 et Signature Version 4. Une fois la version 2 devenue obsolète, Amazon S3 n'acceptera plus que les demandes signées avec Signature Version 4.

Pour plus d'informations sur les AWS demandes de signature, consultez la section [Modifications apportées à la version 4 de Signature](#) dans le Références générales AWS.

Quel est l'objectif de cette mise à jour ?

Signature Version 4 améliore la sécurité grâce à l'utilisation d'une clé de signature au lieu d'une clé d'accès secrète. La version 4 de Signature est actuellement prise en charge dans tous les pays Régions AWS, tandis que la version 2 de Signature n'est prise en charge que dans les régions qui ont été lancées avant janvier 2014. Cette mise à jour nous permet d'offrir une expérience plus homogène dans toutes les régions.

Comment vérifier que j'utilise Signature Version 4 et quelles sont les mises à jour dont j'ai besoin ?

La version de signature utilisée pour signer les demandes est généralement définie par l'outil ou le kit SDK du côté du client. Par défaut, les dernières versions de nos AWS SDK utilisent Signature Version 4. Pour les logiciels tiers, contactez l'équipe d'assistance appropriée de votre logiciel pour confirmer la version dont vous avez besoin. Si vous envoyez des appels REST directs à Amazon S3, vous devez modifier votre application pour qu'elle utilise le processus de signature Signature Version 4.

Pour plus d'informations sur la version des AWS SDK à utiliser lors du passage à la version 4 de Signature, consultez [Passage de Signature Version 2 à Signature Version 4](#).

Pour de plus amples informations sur l'utilisation de Signature Version 4 avec l'API REST Amazon S3, veuillez consulter [Authentification des demandes \(AWS Signature Version 4\)](#) dans la Référence API Amazon Simple Storage Service.

Que se passe-t-il si je n'effectue pas les mises à niveau ?

L'authentification des demandes signées avec Signature Version 2 échouera avec Amazon S3. Les demandeurs verront des erreurs indiquant que la demande doit être signée avec Signature Version 4.

Dois-je effectuer des modifications même si j'utilise une URL présignée qui exige que je signe pour plus de 7 jours ?

Si vous utilisez une URL présignée qui exige que vous signiez pour plus de 7 jours, aucune action n'est requise de votre part. Vous pouvez continuer à utiliser AWS Signature Version 2 pour signer et authentifier l'URL présignée. Pour plus de détails sur la manière de migrer vers Signature Version 4 pour un scénario d'URL présignée, consultez la documentation Amazon S3.

Plus d'informations

- Pour plus d'informations sur l'utilisation de Signature Version 4, consultez [Signing AWS API Requests](#).
- Consultez la liste des modifications entre Signature Version 2 et Signature Version 4 dans [Modification dans Signature Version 4](#).
- Consultez l'article [AWS Signature Version 4 pour remplacer AWS Signature Version 2 pour signer les demandes d'API Amazon S3](#) dans les AWS forums.
- Si vous avez des questions, veuillez contacter le [Support](#).

Passage de Signature Version 2 à Signature Version 4

Si vous utilisez actuellement Signature Version 2 pour authentifier vos demandes d'API Amazon S3, vous devez passer à Signature Version 4. Signature Version 2 ne sera bientôt plus pris en charge, comme décrit dans [AWS Signature Version 2 désactivée \(obsolète\) pour Amazon S3](#).

Pour de plus amples informations sur l'utilisation de Signature Version 4 avec l'API REST Amazon S3, veuillez consulter [Authentification des demandes \(AWS Signature Version 4\)](#) dans la Référence API Amazon Simple Storage Service.

La liste suivante répertorie les kits SDK avec la version minimum nécessaire pour utiliser Signature Version 4 (SigV4). Si vous utilisez des URL présignées avec les SDK AWS Java, JavaScript (Node.js) ou Python (Boto/CLI), vous devez définir les URL correctes Région AWS et définir Signature Version 4 dans la configuration du client. Pour de plus amples informations sur la définition

de SigV4 dans la configuration client, veuillez consulter [Spécification de la version de signature lors de l'authentification des demandes](#).

Si vous utilisez ce SDK/Product	Passez à cette version de SDK	Modification de code requise dans le client pour utiliser Sigv4 ?	Lien vers la documentation SDK
AWS SDK pour Java v1	Passez à Java 1.11.201+ ou v2.	Oui	Spécification de la version de signature lors de l'authentification des demandes
AWS SDK pour Java v2	Pas de mise à niveau de SDK nécessaire.	Non	AWS SDK pour Java
AWS SDK pour .NET v1	Passez à 3.1.10 ou version ultérieure.	Oui	AWS SDK pour .NET
AWS SDK pour .NET v2	Passez à 3.1.10 ou version ultérieure.	Non	AWS SDK pour .NET v2
AWS SDK pour .NET v3	Passez à 3.3.0.0 ou version ultérieure.	Oui	AWS SDK pour .NET v3
AWS SDK pour JavaScript v1	Passez à 2.68.0 ou	Oui	AWS SDK pour JavaScript

Si vous utilisez ce SDK/Product	Passez à cette version de SDK	Modification de code requise dans le client pour utiliser Sigv4 ?	Lien vers la documentation SDK
	version ultérieure.		
AWS SDK pour JavaScript v2	Passez à 2.68.0 ou version ultérieure.	Oui	AWS SDK pour JavaScript
AWS SDK pour JavaScript v3	Aucune autre action n'est requise. Passez à la version majeure V3 au cours du 3e trimestre 2019.	Non	AWS SDK pour JavaScript

Si vous utilisez ce SDK/Product	Passez à cette version de SDK	Modification de code requise dans le client pour utiliser Sigv4 ?	Lien vers la documentation SDK
AWS SDK pour PHP v1	Recommandation de mise à niveau vers la version la plus récente de PHP ou, au moins, vers v2.7.4 avec le paramètre de signature défini sur v4 dans la configuration du client S3.	Oui	AWS SDK pour PHP

Si vous utilisez ce SDK/Product	Passez à cette version de SDK	Modification de code requise dans le client pour utiliser Sigv4 ?	Lien vers la documentation SDK
AWS SDK pour PHP v2	Recommandation de mise à niveau vers la version la plus récente de PHP ou, au moins, vers v2.7.4 avec le paramètre de signature défini sur v4 dans la configuration du client S3.	Non	AWS SDK pour PHP
AWS SDK pour PHP v3	Pas de mise à niveau de SDK nécessaire.	Non	AWS SDK pour PHP
Boto2	Passez à Boto2 v2.49.0.	Oui	Mise à niveau Boto 2
Boto3	Passez à 1.5.71 (Botocore), 1.4.6 (Boto3).	Oui	Boto 3 - AWS SDK pour Python

Si vous utilisez ce SDK/Product	Passez à cette version de SDK	Modification de code requise dans le client pour utiliser Sigv4 ?	Lien vers la documentation SDK
AWS CLI	Passez à 1.11.108.	Oui	AWS Command Line Interface
AWS CLI v2 (aperçu)	Pas de mise à niveau de SDK nécessaire.	Non	AWS Command Line Interface version 2
AWS SDK pour Ruby v1	Passez à Ruby V3.	Oui	Ruby V3 pour AWS
AWS SDK pour Ruby v2	Passez à Ruby V3.	Oui	Ruby V3 pour AWS
AWS SDK pour Ruby v3	Pas de mise à niveau de SDK nécessaire.	Non	Ruby V3 pour AWS
Go	Pas de mise à niveau de SDK nécessaire.	Non	AWS SDK pour Go
C++	Pas de mise à niveau de SDK nécessaire.	Non	AWS SDK pour C++

AWS Tools for Windows PowerShell or AWS Tools for PowerShell Core

Si vous utilisez des versions de module antérieures à 3.3.0.0, vous devez passer à 3.3.0.0.

Pour obtenir les informations de version, utilisez le cmdlet `Get-Module` :

```
Get-Module -Name AWSPowerShell  
Get-Module -Name AWSPowerShell.NetCore
```

Pour mettre à jour la version 3.3.0.0, utilisez le cmdlet `Update-Module` :

```
Update-Module -Name AWSPowerShell  
Update-Module -Name AWSPowerShell.NetCore
```

Vous pouvez utiliser des URL présignées valides pendant plus de 7 jours sur lesquelles vous pouvez envoyer le trafic Signature Version 2.

Obtenir les ID de demande Amazon S3 pour AWS Support

Chaque fois que vous contactez AWS Support parce que vous avez rencontré des erreurs ou un comportement inattendu dans Amazon S3, vous devez fournir les identifiants de demande associés à l'action qui a échoué. AWS Support utilise ces identifiants de demande pour résoudre les problèmes que vous rencontrez.

Les ID de demande viennent par paires, sont renvoyés dans chaque réponse traitée par Amazon S3 (même ceux contenant des erreurs), et sont accessibles via des journaux verbeux. Il existe un certain nombre de méthodes courantes pour obtenir vos identifiants de demande, notamment les journaux et AWS CloudTrail événements d'accès au serveur S3 ou les événements liés aux données.

Après avoir récupéré ces journaux, copiez et conservez ces deux valeurs, car vous en aurez besoin lors de votre contact AWS Support. Pour plus d'informations sur la prise de [contact AWS Support](#), [consultez la section Contact AWS](#) ou la [AWS Support documentation](#).

Rubriques

- [Utilisation de AWS CLI pour obtenir les numéros de demande](#)
- [Utilisation de Windows PowerShell pour obtenir les ID de demande](#)

- [Utilisation AWS CloudTrail événements de données pour obtenir les identifiants de demande](#)
- [Utilisation de la journalisation des accès au serveur S3 pour obtenir des ID de demande](#)
- [Utilisation de HTTP pour obtenir des ID de demande](#)
- [Utilisation d'un navigateur web pour obtenir des ID de demande](#)
- [Utilisation de AWS SDK pour obtenir les ID de demande](#)

Utilisation de AWS CLI pour obtenir les numéros de demande

Pour obtenir vos identifiants de demande lorsque vous utilisez le AWS Command Line Interface (AWS CLI), ajoutez-les `--debug` à votre commande. Par exemple, vous devriez voir `x-amz-request-id` et `x-amz-id-2` dans le journal de débogage comme indiqué ci-dessous :

```
...
2025-04-30 14:35:28,572 - MainThread - botocore.parsers - DEBUG - Response headers:
{'x-amz-id-2': 'a+zm50vDJH3LLmCiXvwEo0u0PtPS/qCJaBvB2ZMH9dzyzTiJhiLZkBFFoRfsPf0KztUKT/
garCI=', 'x-amz-request-id': 'N4NMN0MJ4VDFZMX9', 'Date': 'Wed, 30 Apr 2025 21:35:29
GMT', 'Content-Type': 'application/xml', 'Transfer-Encoding': 'chunked', 'Server':
'AmazonS3'}
...
```

Utilisation de Windows PowerShell pour obtenir les ID de demande

Pour plus d'informations sur la restauration des journaux avec Windows PowerShell, consultez la section relative AWS Tools for Windows PowerShell [à la connexion](#) aux réponses sur le blog AWS Developer Tools.

Utilisation AWS CloudTrail événements de données pour obtenir les identifiants de demande

Un compartiment Amazon S3 configuré avec CloudTrail des événements de données pour enregistrer les opérations d'API S3 au niveau de l'objet fournit des informations détaillées sur les actions entreprises par un utilisateur, un rôle ou un AWS service dans Amazon S3. Vous pouvez identifier les ID de requête S3 en interrogeant les CloudTrail événements avec Athena.

Pour plus d'informations, consultez la section [Identification des requêtes Amazon S3 CloudTrail](#) à l'aide du guide de l'utilisateur d'Amazon Simple Storage Service.

Utilisation de la journalisation des accès au serveur S3 pour obtenir des ID de demande

Un compartiment Amazon S3 configuré pour journaliser les accès au serveur S3 fournit des enregistrements détaillés des demandes qui lui sont soumises. Vous pouvez identifier les ID de demande S3 en interrogeant les journaux d'accès au serveur à l'aide d'Athena.

Pour plus d'informations, consultez la section [Interroger les journaux d'accès pour les demandes Amazon Athena](#) à l'aide du guide de l'utilisateur d'Amazon Simple Storage Service.

Utilisation de HTTP pour obtenir des ID de demande

Vous pouvez obtenir les ID de demande, `x-amz-request-id` et `x-amz-id-2` en consignnant les bits d'une demande HTTP avant qu'elle n'atteigne l'application cible. Plusieurs outils tiers peuvent être utilisés pour récupérer les journaux verbeux pour les demandes HTTP. Choisissez un outil fiable, exécutez-le pour écouter le port sur lequel le trafic Amazon S3 circule, lorsque vous envoyez une autre demande HTTP Amazon S3.

Pour les demandes HTTP, la paire d'ID de demande ressemble à ce qui suit :

```
x-amz-request-id: 79104EXAMPLEB723
x-amz-id-2: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km
```

Note

Les demandes HTTPS sont chiffrées et masquées dans la plupart des captures de paquet.

Utilisation d'un navigateur web pour obtenir des ID de demande

La plupart des navigateurs web possèdent des outils de développement que vous pouvez utiliser pour consulter les en-têtes de demande.

Pour les demandes basées sur navigateur Web qui renvoient une erreur, la paire d'ID de demande ressemble aux exemples suivants.

```
<Error><Code>AccessDenied</Code><Message>Access Denied</Message>
```

```
<RequestId>79104EXAMPLEB723</RequestId><HostId>IOWQ4fDEXAMPLEQM  
+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km</HostId></Error>
```

Pour obtenir la paire d'ID de demande à partir de demandes réussies, utilisez les outils de développeur de votre navigateur pour consulter les en-têtes de réponse HTTP.

Utilisation de AWS SDK pour obtenir les ID de demande

Les sections suivantes contiennent des informations sur la configuration de la journalisation à l'aide de différents AWS kits de développement logiciel.

Note

Bien que vous puissiez activer la journalisation détaillée pour chaque demande et réponse, il est déconseillé d'activer la journalisation dans les systèmes de production, car les demandes ou réponses volumineuses peuvent considérablement ralentir une application.

Pour les demandes du AWS SDK, la paire d'ID de demande ressemblera à ce qui suit :

```
Status Code: 403, AWS Service: Amazon S3, AWS Request ID: 79104EXAMPLEB723  
AWS Error Code: AccessDenied AWS Error Message: Access Denied  
S3 Extended Request ID: IOWQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK  
+Jd1vEXAMPLEa3Km
```

C++

Le type d'enregistreur et la verbosité sont spécifiés lors de l'initialisation du SDK dans l'argument. `SDKOptions` L'exemple suivant spécifie le niveau de verbosité comme suit. `LogLevel::Debug`

L'enregistreur par défaut écrit dans le système de fichiers et le fichier est nommé selon la convention suivante. `aws_sdk_YYYY-MM-DD-HH.log` L'enregistreur crée un nouveau fichier chaque heure.

```
Aws::SDKOptions options;  
options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;  
Aws::InitAPI(options);  
// ...  
Aws::ShutdownAPI(options);
```

Pour plus d'informations, voir [Comment activer la journalisation ?](#) dans le wiki du AWS SDK pour C++ sur GitHub.

Go

Vous pouvez configurer la journalisation à l'aide du SDK for Go. Pour plus d'informations, consultez la section [Logging](#) in the AWS SDK pour Go v2 Developer Guide.

Java

Vous pouvez activer la journalisation pour des demandes ou des réponses spécifiques afin de récupérer et de renvoyer uniquement les en-têtes pertinents. Pour ce faire, importez la classe `com.amazonaws.services.s3.S3ResponseMetadata`. Ensuite, vous pouvez stocker la demande dans une variable avant de faire la demande réelle. Appelez `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()` pour obtenir la demande ou réponse journalisée.

```
PutObjectRequest req = new PutObjectRequest(bucketName, key, createSampleFile());
s3.putObject(req);
S3ResponseMetadata md = s3.getCachedResponseMetadata(req);
System.out.println("Host ID: " + md.getHostId() + " RequestID: " +
    md.getRequestId());
```

Sinon, vous pouvez utiliser une journalisation verbeuse de chaque demande ou réponse Java. Pour plus d'informations, consultez [Verbose Wire Logging](#) (Journalisation du réseau filaire détaillée) dans le Guide du développeur AWS SDK pour Java .

JavaScript

AWS SDK pour JavaScript II dispose d'un enregistreur intégré qui vous permet d'enregistrer les appels d'API que vous passez avec celui-ci. Pour activer l'enregistreur et imprimer les entrées du journal dans la console, ajoutez l'instruction suivante à votre code :

```
AWS.config.logger = console;
```

Pour plus d'informations, consultez la section [Journalisation AWS SDK pour JavaScript des appels](#) dans le Guide du AWS SDK pour JavaScript développeur.

Kotlin

Avec le AWS SDK pour Kotlin, vous pouvez spécifier le mode de journalisation pour les messages filaires à l'aide du code ou des paramètres d'environnement. Vous pouvez définir le mode journal pour les requêtes HTTP et les réponses HTTP.

Pour activer la journalisation supplémentaire, définissez la `logMode` propriété lorsque vous créez un client de service :

```
import aws.smithy.kotlin.runtime.client.LogMode

// ...

val client = S3Client {
    // ...
    logMode = LogMode.LogRequestWithBody + LogMode.LogResponse
}
```

Vous pouvez également définir le mode journal à l'aide d'une variable d'environnement :

```
export SDK_LOG_MODE=LogRequestWithBody|LogResponse
```

Pour plus d'informations, consultez la section [Logging](#) in the AWS SDK pour Kotlin Developer Guide.

.NET

Vous pouvez configurer la journalisation à l'aide SDK pour .NET de l'outil de `System.Diagnostics` journalisation intégré. Pour plus d'informations, consultez l'article de blog [Logging with the SDK pour .NET](#) AWS Developer.

Note

Par défaut, le journal renvoyé contient uniquement des informations sur les erreurs. Pour obtenir les ID de demande, le fichier de configuration doit avoir l'élément `AWSLogMetrics` (et si vous le souhaitez, `AWSResponseLogging`) ajouté.

PHP

Vous pouvez obtenir des informations de débogage, y compris les données envoyées par câble, en définissant l'option de débogage sur `true` dans un constructeur client.

```
$s3Client = new Aws\S3\S3Client([
    'region' => 'us-standard',
    'version' => '2006-03-01',
    'debug'   => true
]);
```

```
]);
```

Pour plus d'informations, consultez [How can I see what data is sent over the wire?](#) (Comment savoir quelles données sont envoyées sur le réseau ?) dans le Guide du développeur AWS SDK pour PHP .

Python (Boto3)

Avec le AWS SDK pour Python (Boto3), vous pouvez enregistrer des réponses spécifiques. Vous pouvez utiliser cette fonction pour capturer uniquement les en-têtes pertinents. Le code suivant vous montre comment journaliser des parties de la réponse à un fichier :

```
import logging
import boto3
logging.basicConfig(filename='logfile.txt', level=logging.INFO)
logger = logging.getLogger(__name__)
s3 = boto3.resource('s3')
response = s3.Bucket(bucket_name).Object(object_key).put()
logger.info("HTTPStatusCode: %s", response['ResponseMetadata']['HTTPStatusCode'])
logger.info("RequestId: %s", response['ResponseMetadata']['RequestId'])
logger.info("HostId: %s", response['ResponseMetadata']['HostId'])
logger.info("Date: %s", response['ResponseMetadata']['HTTPHeaders']['date'])
```

Vous pouvez également intercepter les exceptions et consigner les informations pertinentes lorsqu'une exception est déclenchée. Pour plus d'informations, consultez [Discerning useful information from error responses](#) (Obtenir des informations utiles à partir d'erreurs de réponse) dans la Référence d'API du kit SDK AWS for Python (Boto).

En outre, vous pouvez configurer Boto3 pour générer des journaux de débogage verbeux à l'aide du code suivant :

```
import logging
import boto3
boto3.set_stream_logger('', logging.DEBUG)
```

Pour plus d'informations, consultez [set_stream_logger](#) dans la Référence d'API du kit SDK AWS for Python (Boto).

Ruby

Vous pouvez obtenir vos identifiants de demande à l'aide du SDK pour Ruby version 3. Vous pouvez activer la journalisation par câble HTTP dans votre client à l'aide du code suivant :

```
s3 = Aws::S3::Client.new(http_wire_trace: true)
```

Note

L'enregistrement des virements peut inclure des informations sensibles, telles que l'identifiant de votre clé d'accès. Les informations sensibles doivent être nettoyées avant de les partager avec le AWS support.

Vous pouvez également trouver l'ID de demande de l'objet de contexte de demande dans la réponse ou l'erreur de demande :

```
# Finding the request ID from an error:
begin
  s3.put_object(bucket: 'bucket', key: 'key', body: 'test')
rescue Aws::S3::Errors::ServiceError => e
  puts e.context[:request_id]
  puts e.context[:s3_host_id]
end

# Finding the request ID from a successful call:
resp = s3.put_object(bucket: 'bucket', key: 'key', body: 'test')
puts resp.context[:request_id]
puts resp.context[:s3_host_id]
```

Pour plus d'informations, consultez la section [Débogage à l'aide des informations de traçage de fils provenant d'un AWS SDK pour Ruby client](#) dans le Guide du AWS SDK pour Ruby développeur.

Rust

Pour activer la journalisation, ajoutez la `tracing-subscriber` caisse et initialisez-la dans votre application Rust.

Ajoutez la bibliothèque de traçage à votre `Cargo.toml` fichier :

```
tracing-subscriber = { version = "0.3", features = ["env-filter"] }
```

Ensuite, dans votre code Rust, initialisez l'enregistreur dans la fonction principale avant d'appeler une opération du SDK :

```
tracing_subscriber::fmt::init();
```

Pour plus d'informations, consultez [la section Configuration et utilisation de la connexion AWS SDK pour Rust](#) dans le Guide du AWS SDK pour Rust développeur.

Opérations d'API Amazon S3 au niveau objet prises en charge pour les tables S3

Le tableau suivant inclut les opérations d'API S3 au niveau des objets prises en charge et les en-têtes correspondants pour les tables S3. Pour obtenir la liste des API Amazon S3 Tables, consultez [Amazon S3 Tables](#). Pour plus d'informations sur les tables Amazon S3, consultez la section [Utilisation des tables et des compartiments de tables Amazon S3](#) dans le guide de l'utilisateur Amazon S3.

Object-level API	En-têtes pris en charge	Remarques	
AbortMultipartUpload	x-amz-expected-bucket-owner	Aucune	
CompleteMultipartUpload	x-amz-checksum-crc32	Aucun	
	x-amz-checksum-crc32c	Aucun	
	x-amz-checksum-sha1	Aucun	
	x-amz-checksum-sha256	Aucun	
	x-amz-expected-bucket-owner	Aucun	
	If-Match	Aucun	

Object-level API	En-têtes pris en charge	Remarques	
CreateMultipartUpload	If-None-Match	Aucune	
	x-amz-acl: ACL	Pour les tables S3, la valeur par défaut est bucket-owner-full-control et elle ne peut pas être modifiée.	
	Cache-Control	Aucune	
	Content-Disposition	Aucun	
	Content-Encoding	Aucun	
	Content-Language	Aucun	
	Content-Type	Aucun	
	Expires	Aucun	
	x-amz-checksum-algorithm	Aucune	
	x-amz-storage-class	Pour les tables S3, la valeur par défaut est STANDARD et elle ne peut pas être modifiée.	
	x-amz-server-side-encryption	Pour les tables S3, la valeur par défaut est (SSE-S3) (AES256) et elle ne peut pas être modifiée.	

Object-level API	En-têtes pris en charge	Remarques	
GetObject	If-Match	Aucune	
	If-Modified-Since	Aucun	
	If-None-Match	Aucun	
	If-Unmodified-Since	Aucun	
	Range	Aucun	
	x-amz-expected-bucket-owner	Aucun	
	x-amz-checksum-mode	Aucun	
HeadObject	If-Match	Aucun	
	If-Modified-Since	Aucun	
	If-None-Match	Aucun	
	If-Unmodified-Since	Aucun	
	Range	Aucun	
	x-amz-checksum-mode	Aucun	
	x-amz-expected-bucket-owner	Aucun	
ListParts	x-amz-expected-bucket-owner	Aucune	

Object-level API	En-têtes pris en charge	Remarques	
PutObject	x-amz-acl: ACL	Pour les tables S3, la valeur par défaut est <code>bucket-owner-full-control</code> et elle ne peut pas être modifiée.	
	Cache-Control	Aucune	
	Content-Disposition	Aucun	
	Content-Encoding	Aucun	
	Content-Language	Aucun	
	Content-Length	Aucun	
	Content-MD5	Aucun	
	Content-Type	Aucun	
	x-amz-sdk-checksum-algorithm	Aucun	
	x-amz-checksum-crc32	Aucun	
	x-amz-checksum-crc32c	Aucun	
	x-amz-checksum-sha1	Aucun	
	x-amz-checksum-sha256	Aucun	

Object-level API	En-têtes pris en charge	Remarques	
	Expires	Aucun	
	If-None-Match	Aucun	
	x-amz-expected-bucket-owner	Aucune	
	x-amz-storage-class	Pour les tables S3, la valeur par défaut est STANDARD et elle ne peut pas être modifiée.	
	x-amz-server-side-encryption	Pour les tables S3, la valeur par défaut est (SSE-S3) (AES256) et elle ne peut pas être modifiée.	
UploadPart	Content-Length	Aucune	
	Content-MD5	Aucun	
	x-amz-checksum-crc32	Aucun	
	x-amz-checksum-crc32c	Aucun	
	x-amz-checksum-sha1	Aucun	
	x-amz-checksum-sha256	Aucun	
	x-amz-expected-bucket-owner	Aucune	

Exemples de code pour Amazon S3 utilisant AWS Kits SDK

Les exemples de code suivants montrent comment utiliser Amazon S3 avec un kit de développement AWS logiciel (SDK).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit de développement logiciel (SDK).

Exemples de code

- [Exemples de code pour Amazon S3 utilisant AWS Kits SDK](#)
 - [Exemples de base illustrant l'utilisation d'Amazon S3 AWS Kits SDK](#)
 - [Hello Amazon S3](#)
 - [Découvrez les bases d'Amazon S3 grâce à AWS Kit SDK](#)
 - [Actions pour Amazon S3 utilisant AWS Kits SDK](#)
 - [À utiliser AbortMultipartUploadavec un AWS SDK ou CLI](#)
 - [À utiliser CompleteMultipartUploadavec un AWS SDK ou CLI](#)
 - [À utiliser CopyObjectavec un AWS SDK ou CLI](#)
 - [À utiliser CreateBucketavec un AWS SDK ou CLI](#)
 - [À utiliser CreateMultiRegionAccessPointavec un AWS Kit SDK](#)
 - [À utiliser CreateMultipartUploadavec un AWS SDK ou CLI](#)
 - [À utiliser CreatePresignedPostavec un AWS Kit SDK](#)
 - [À utiliser DeleteBucketavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketAnalyticsConfiguration avec une CLI](#)
 - [À utiliser DeleteBucketCorsavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketEncryption avec une CLI](#)
 - [Utilisation de DeleteBucketInventoryConfiguration avec une CLI](#)
 - [À utiliser DeleteBucketLifecycleavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketMetricsConfiguration avec une CLI](#)
 - [À utiliser DeleteBucketPolicyavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketReplication avec une CLI](#)

- [Utilisation de DeleteBucketTagging avec une CLI](#)
- [À utiliser DeleteBucketWebsiteavec un AWS SDK ou CLI](#)
- [À utiliser DeleteObjectavec un AWS SDK ou CLI](#)
- [Utilisation de DeleteObjectTagging avec une CLI](#)
- [À utiliser DeleteObjectsavec un AWS SDK ou CLI](#)
- [Utilisation de DeletePublicAccessBlock avec une CLI](#)
- [Utilisation de GetBucketAccelerateConfiguration avec une CLI](#)
- [À utiliser GetBucketAclavec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketAnalyticsConfiguration avec une CLI](#)
- [À utiliser GetBucketCorsavec un AWS SDK ou CLI](#)
- [À utiliser GetBucketEncryptionavec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketInventoryConfiguration avec une CLI](#)
- [À utiliser GetBucketLifecycleConfigurationavec un AWS SDK ou CLI](#)
- [À utiliser GetBucketLocationavec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketLogging avec une CLI](#)
- [Utilisation de GetBucketMetricsConfiguration avec une CLI](#)
- [Utilisation de GetBucketNotification avec une CLI](#)
- [À utiliser GetBucketPolicyavec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketPolicyStatus avec une CLI](#)
- [À utiliser GetBucketReplicationavec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketRequestPayment avec une CLI](#)
- [Utilisation de GetBucketTagging avec une CLI](#)
- [Utilisation de GetBucketVersioning avec une CLI](#)
- [À utiliser GetBucketWebsiteavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectAclavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectAttributesavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectLegalHoldavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectLockConfigurationavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectRetentionavec un AWS SDK ou CLI](#)

- [Utilisation de GetObjectTagging avec une CLI](#)
- [Utilisation de GetPublicAccessBlock avec une CLI](#)
- [À utiliser HeadBucket avec un AWS SDK ou CLI](#)
- [À utiliser HeadObject avec un AWS SDK ou CLI](#)
- [Utilisation de ListBucketAnalyticsConfigurations avec une CLI](#)
- [Utilisation de ListBucketInventoryConfigurations avec une CLI](#)
- [À utiliser ListBuckets avec un AWS SDK ou CLI](#)
- [À utiliser ListMultipartUploads avec un AWS SDK ou CLI](#)
- [À utiliser ListObjectVersions avec un AWS SDK ou CLI](#)
- [Utilisation de ListObjects avec une CLI](#)
- [Utilisez la ListObjectsV2 avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketAccelerateConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketAcl avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketCors avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketEncryption avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketLifecycleConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketLogging avec un AWS SDK ou CLI](#)
- [Utilisation de PutBucketNotification avec une CLI](#)
- [À utiliser PutBucketNotificationConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketPolicy avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketReplication avec un AWS SDK ou CLI](#)
- [Utilisation de PutBucketRequestPayment avec une CLI](#)
- [Utilisation de PutBucketTagging avec une CLI](#)
- [À utiliser PutBucketVersioning avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketWebsite avec un AWS SDK ou CLI](#)
- [À utiliser PutObject avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectAcl avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectLegalHold avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectLockConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectRetention avec un AWS SDK ou CLI](#)

- [À utiliser RestoreObject avec un AWS SDK ou CLI](#)
- [À utiliser SelectObjectContent avec un AWS SDK ou CLI](#)
- [À utiliser UploadPart avec un AWS SDK ou CLI](#)
- [À utiliser UploadPartCopy avec un AWS SDK ou CLI](#)
- [Scénarios pour Amazon S3 utilisant AWS Kits SDK](#)
 - [Vérification de l'existence d'un compartiment](#)
 - [Vérifiez si un objet existe](#)
 - [Convertissez du texte en parole et de nouveau en texte à l'aide d'un AWS Kit SDK](#)
 - [Créez une URL présignée pour Amazon S3 à l'aide d'un AWS Kit SDK](#)
 - [Création d'une application de gestion des ressources photographiques permettant aux utilisateurs de gérer les photos à l'aide d'étiquettes](#)
 - [Une page Web répertoriant les objets Amazon S3 à l'aide d'un AWS Kit SDK](#)
 - [Créer une application Amazon Textract Explorer](#)
 - [Supprimez tous les objets d'un compartiment Amazon S3 donné à l'aide d'un AWS Kit SDK](#)
 - [Supprimez les chargements partitionnés incomplets vers Amazon S3 à l'aide d'un AWS Kit SDK](#)
 - [Détectez le PPE dans les images avec Amazon Rekognition à l'aide d'un AWS Kit SDK](#)
 - [Détectez les entités dans le texte extrait d'une image à l'aide d'un AWS Kit SDK](#)
 - [Détecter les visages dans une image à l'aide d'un AWS Kit SDK](#)
 - [Détectez des objets dans des images avec Amazon Rekognition à l'aide d'un AWS Kit SDK](#)
 - [Détectez les personnes et les objets dans une vidéo avec Amazon Rekognition à l'aide d'un AWS Kit SDK](#)
 - [Téléchargement des « répertoires » S3 à partir d'un compartiment Amazon Simple Storage Service \(Amazon S3\)](#)
 - [Téléchargez tous les objets d'un compartiment Amazon Simple Storage Service \(Amazon S3\) dans un répertoire local.](#)
 - [Téléchargez un flux de taille inconnue à partir d'un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
 - [Obtenez un objet Amazon S3 à partir d'un point d' Multi-Region accès à l'aide d'un AWS Kit SDK](#)
 - [Obtenez un objet depuis un compartiment Amazon S3 à l'aide d'un AWS SDK, spécification d'un en-tête If-Modified-Since](#)

- [Commencez à chiffrer les objets Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Commencez à utiliser les balises pour les objets Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Débuter avec les clusters de traitement des mégadonnées](#)
- [Commencer à gérer la configuration](#)
- [Commencer à extraire le texte d'un document](#)
- [Commencer à utiliser les magasins de fonctionnalités d'apprentissage automatique](#)
- [Débuter avec le stockage d'objets](#)
- [Commencer à utiliser l'analyse des requêtes](#)
- [Utilisez les fonctionnalités de verrouillage d'objets d'Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Effectuez des demandes conditionnelles Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Gérez les listes de contrôle d'accès \(ACL\) pour les compartiments Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Gérez les messages Amazon SQS volumineux à l'aide d'Amazon S3 avec un AWS Kit SDK](#)
- [Gérez des objets Amazon S3 versionnés par lots à l'aide d'une fonction Lambda à l'aide d'un AWS Kit SDK](#)
- [Analysez les URI Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Effectuez une copie en plusieurs parties d'un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Recevez et traitez les notifications d'événements Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Enregistrez les informations EXIF et autres informations d'image à l'aide d'un AWS Kit SDK](#)
- [Envoyez des notifications d'événements S3 à Amazon à EventBridge l'aide d'un AWS Kit SDK](#)
- [Suivez le chargement ou le téléchargement d'un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Transformation des données pour votre application avec S3 Object Lambda](#)
- [Exemples d'approches pour les tests unitaires et d'intégration avec un AWS Kit SDK](#)
- [Charger récursivement un répertoire local dans un compartiment Amazon Simple Storage Service \(Amazon S3\)](#)
- [Chargez ou téléchargez des fichiers volumineux vers et depuis Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Chargez un flux de taille inconnue vers un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Utilisez des checksums pour travailler avec un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)

- [Utilisez les fonctionnalités d'intégrité des objets d'Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Travaillez avec des objets versionnés Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Exemples sans serveur pour Amazon S3](#)
 - [Invoquer une fonction lambda à partir d'un déclencheur Amazon S3](#)
- [Exemples de code pour Amazon S3 Control à l'aide de AWS Kits SDK](#)
- [Exemples de base pour l'utilisation d'Amazon S3 Control AWS Kits SDK](#)
 - [Bonjour Amazon S3 Control](#)
 - [Découvrez les principes de base d'Amazon S3 Control grâce à AWS Kit SDK](#)
 - [Actions pour Amazon S3 Control à l'aide de AWS Kits SDK](#)
 - [À utiliser CreateJobavec un AWS SDK ou CLI](#)
 - [À utiliser DeleteJobTaggingavec un AWS Kit SDK](#)
 - [À utiliser DescribeJobavec un AWS SDK ou CLI](#)
 - [À utiliser GetJobTaggingavec un AWS Kit SDK](#)
 - [À utiliser PutJobTaggingavec un AWS Kit SDK](#)
 - [À utiliser UpdateJobPriorityavec un AWS SDK ou CLI](#)
 - [À utiliser UpdateJobStatusavec un AWS SDK ou CLI](#)
- [Exemples de code pour les compartiments de répertoire S3 utilisant AWS Kits SDK](#)
- [Exemples de base pour les compartiments de répertoire S3 utilisant AWS Kits SDK](#)
 - [Bonjour compartiments de répertoires Amazon S3](#)
 - [Apprenez les bases des compartiments d'annuaire S3 avec un AWS Kit SDK](#)
 - [Actions pour les compartiments de répertoire S3 à l'aide de AWS Kits SDK](#)
 - [À utiliser AbortMultipartUploadavec un AWS Kit SDK](#)
 - [À utiliser CompleteMultipartUploadavec un AWS Kit SDK](#)
 - [À utiliser CopyObjectavec un AWS Kit SDK](#)
 - [À utiliser CreateBucketavec un AWS Kit SDK](#)
 - [À utiliser CreateMultipartUploadavec un AWS Kit SDK](#)
 - [À utiliser CreateSessionavec un AWS Kit SDK](#)
 - [À utiliser DeleteBucketavec un AWS Kit SDK](#)
 - [À utiliser DeleteBucketEncryptionavec un AWS Kit SDK](#)
 - [À utiliser DeleteBucketPolicyavec un AWS Kit SDK](#)

- [À utiliser DeleteObjectavec un AWS Kit SDK](#)
- [À utiliser DeleteObjectsavec un AWS Kit SDK](#)
- [À utiliser GetBucketEncryptionavec un AWS Kit SDK](#)
- [À utiliser GetBucketPolicyavec un AWS Kit SDK](#)
- [À utiliser GetObjectavec un AWS Kit SDK](#)
- [À utiliser GetObjectAttributesavec un AWS Kit SDK](#)
- [À utiliser HeadBucketavec un AWS Kit SDK](#)
- [À utiliser HeadObjectavec un AWS Kit SDK](#)
- [À utiliser ListDirectoryBucketsavec un AWS Kit SDK](#)
- [À utiliser ListMultipartUploadsavec un AWS Kit SDK](#)
- [Utilisez la ListObjectsV2 avec un AWS Kit SDK](#)
- [À utiliser ListPartsavec un AWS Kit SDK](#)
- [À utiliser PutBucketEncryptionavec un AWS Kit SDK](#)
- [À utiliser PutBucketPolicyavec un AWS Kit SDK](#)
- [À utiliser PutObjectavec un AWS Kit SDK](#)
- [À utiliser UploadPartavec un AWS Kit SDK](#)
- [À utiliser UploadPartCopyavec un AWS Kit SDK](#)
- [Scénarios pour les compartiments d'annuaire S3 utilisant AWS Kits SDK](#)
 - [Créez une URL présignée pour les compartiments d'annuaire Amazon S3 afin d'obtenir un objet à l'aide d'un AWS Kit SDK](#)

Exemples de code pour Amazon S3 utilisant AWS Kits SDK

Les exemples de code suivants montrent comment utiliser Amazon S3 avec un kit de développement AWS logiciel (SDK).

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Amazon S3

- [Exemples de base illustrant l'utilisation d'Amazon S3 AWS Kits SDK](#)
 - [Hello Amazon S3](#)
 - [Découvrez les bases d'Amazon S3 grâce à AWS Kit SDK](#)
 - [Actions pour Amazon S3 utilisant AWS Kits SDK](#)
 - [À utiliser AbortMultipartUploadavec un AWS SDK ou CLI](#)
 - [À utiliser CompleteMultipartUploadavec un AWS SDK ou CLI](#)
 - [À utiliser CopyObjectavec un AWS SDK ou CLI](#)
 - [À utiliser CreateBucketavec un AWS SDK ou CLI](#)
 - [À utiliser CreateMultiRegionAccessPointavec un AWS Kit SDK](#)
 - [À utiliser CreateMultipartUploadavec un AWS SDK ou CLI](#)
 - [À utiliser CreatePresignedPostavec un AWS Kit SDK](#)
 - [À utiliser DeleteBucketavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketAnalyticsConfiguration avec une CLI](#)
 - [À utiliser DeleteBucketCorsavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketEncryption avec une CLI](#)
 - [Utilisation de DeleteBucketInventoryConfiguration avec une CLI](#)
 - [À utiliser DeleteBucketLifecycleavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketMetricsConfiguration avec une CLI](#)
 - [À utiliser DeleteBucketPolicyavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketReplication avec une CLI](#)
 - [Utilisation de DeleteBucketTagging avec une CLI](#)
 - [À utiliser DeleteBucketWebsiteavec un AWS SDK ou CLI](#)
 - [À utiliser DeleteObjectavec un AWS SDK ou CLI](#)

- [Utilisation de DeleteObjectTagging avec une CLI](#)
- [À utiliser DeleteObjects avec un AWS SDK ou CLI](#)
- [Utilisation de DeletePublicAccessBlock avec une CLI](#)
- [Utilisation de GetBucketAccelerateConfiguration avec une CLI](#)
- [À utiliser GetBucketAcl avec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketAnalyticsConfiguration avec une CLI](#)
- [À utiliser GetBucketCors avec un AWS SDK ou CLI](#)
- [À utiliser GetBucketEncryption avec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketInventoryConfiguration avec une CLI](#)
- [À utiliser GetBucketLifecycleConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser GetBucketLocation avec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketLogging avec une CLI](#)
- [Utilisation de GetBucketMetricsConfiguration avec une CLI](#)
- [Utilisation de GetBucketNotification avec une CLI](#)
- [À utiliser GetBucketPolicy avec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketPolicyStatus avec une CLI](#)
- [À utiliser GetBucketReplication avec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketRequestPayment avec une CLI](#)
- [Utilisation de GetBucketTagging avec une CLI](#)
- [Utilisation de GetBucketVersioning avec une CLI](#)
- [À utiliser GetBucketWebsite avec un AWS SDK ou CLI](#)
- [À utiliser GetObject avec un AWS SDK ou CLI](#)
- [À utiliser GetObjectAcl avec un AWS SDK ou CLI](#)
- [À utiliser GetObjectAttributes avec un AWS SDK ou CLI](#)
- [À utiliser GetObjectLegalHold avec un AWS SDK ou CLI](#)
- [À utiliser GetObjectLockConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser GetObjectRetention avec un AWS SDK ou CLI](#)
- [Utilisation de GetObjectTagging avec une CLI](#)
- [Utilisation de GetPublicAccessBlock avec une CLI](#)
- [À utiliser HeadBucket avec un AWS SDK ou CLI](#)

- [À utiliser HeadObject avec un AWS SDK ou CLI](#)
- [Utilisation de ListBucketAnalyticsConfigurations avec une CLI](#)
- [Utilisation de ListBucketInventoryConfigurations avec une CLI](#)
- [À utiliser ListBuckets avec un AWS SDK ou CLI](#)
- [À utiliser ListMultipartUploads avec un AWS SDK ou CLI](#)
- [À utiliser ListObjectVersions avec un AWS SDK ou CLI](#)
- [Utilisation de ListObjects avec une CLI](#)
- [Utilisez la ListObjectsV2 avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketAccelerateConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketAcl avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketCors avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketEncryption avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketLifecycleConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketLogging avec un AWS SDK ou CLI](#)
- [Utilisation de PutBucketNotification avec une CLI](#)
- [À utiliser PutBucketNotificationConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketPolicy avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketReplication avec un AWS SDK ou CLI](#)
- [Utilisation de PutBucketRequestPayment avec une CLI](#)
- [Utilisation de PutBucketTagging avec une CLI](#)
- [À utiliser PutBucketVersioning avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketWebsite avec un AWS SDK ou CLI](#)
- [À utiliser PutObject avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectAcl avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectLegalHold avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectLockConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectRetention avec un AWS SDK ou CLI](#)
- [À utiliser RestoreObject avec un AWS SDK ou CLI](#)
- [À utiliser SelectObjectContent avec un AWS SDK ou CLI](#)
- [À utiliser UploadPart avec un AWS SDK ou CLI](#)

- [À utiliser UploadPartCopy avec un AWS SDK ou CLI](#)
- [Scénarios pour Amazon S3 utilisant AWS Kits SDK](#)
 - [Vérification de l'existence d'un compartiment](#)
 - [Vérifiez si un objet existe](#)
 - [Convertissez du texte en parole et de nouveau en texte à l'aide d'un AWS Kit SDK](#)
 - [Créez une URL présignée pour Amazon S3 à l'aide d'un AWS Kit SDK](#)
 - [Création d'une application de gestion des ressources photographiques permettant aux utilisateurs de gérer les photos à l'aide d'étiquettes](#)
 - [Une page Web répertoriant les objets Amazon S3 à l'aide d'un AWS Kit SDK](#)
 - [Créer une application Amazon Textract Explorer](#)
 - [Supprimez tous les objets d'un compartiment Amazon S3 donné à l'aide d'un AWS Kit SDK](#)
 - [Supprimez les chargements partitionnés incomplets vers Amazon S3 à l'aide d'un AWS Kit SDK](#)
 - [Détectez le PPE dans les images avec Amazon Rekognition à l'aide d'un AWS Kit SDK](#)
 - [Détectez les entités dans le texte extrait d'une image à l'aide d'un AWS Kit SDK](#)
 - [Détecter les visages dans une image à l'aide d'un AWS Kit SDK](#)
 - [Détectez des objets dans des images avec Amazon Rekognition à l'aide d'un AWS Kit SDK](#)
 - [Détectez les personnes et les objets dans une vidéo avec Amazon Rekognition à l'aide d'un AWS Kit SDK](#)
 - [Téléchargement des « répertoires » S3 à partir d'un compartiment Amazon Simple Storage Service \(Amazon S3\)](#)
 - [Téléchargez tous les objets d'un compartiment Amazon Simple Storage Service \(Amazon S3\) dans un répertoire local.](#)
 - [Téléchargez un flux de taille inconnue à partir d'un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
 - [Obtenez un objet Amazon S3 à partir d'un point d' Multi-Region accès à l'aide d'un AWS Kit SDK](#)
 - [Obtenez un objet depuis un compartiment Amazon S3 à l'aide d'un AWS SDK, spécification d'un en-tête If-Modified-Since](#)
 - [Commencez à chiffrer les objets Amazon S3 à l'aide d'un AWS Kit SDK](#)
 - [Commencez à utiliser les balises pour les objets Amazon S3 à l'aide d'un AWS Kit SDK](#)
 - [Débuter avec les clusters de traitement des mégadonnées](#)
 - [Commencer à gérer la configuration](#)
- [Commencer à extraire le texte d'un document](#)

- [Commencer à utiliser les magasins de fonctionnalités d'apprentissage automatique](#)
- [Débuter avec le stockage d'objets](#)
- [Commencer à utiliser l'analyse des requêtes](#)
- [Utilisez les fonctionnalités de verrouillage d'objets d'Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Effectuez des demandes conditionnelles Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Gérez les listes de contrôle d'accès \(ACL\) pour les compartiments Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Gérez les messages Amazon SQS volumineux à l'aide d'Amazon S3 avec un AWS Kit SDK](#)
- [Gérez des objets Amazon S3 versionnés par lots à l'aide d'une fonction Lambda à l'aide d'un AWS Kit SDK](#)
- [Analysez les URI Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Effectuez une copie en plusieurs parties d'un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Recevez et traitez les notifications d'événements Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Enregistrez les informations EXIF et autres informations d'image à l'aide d'un AWS Kit SDK](#)
- [Envoyez des notifications d'événements S3 à Amazon à EventBridge l'aide d'un AWS Kit SDK](#)
- [Suivez le chargement ou le téléchargement d'un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Transformation des données pour votre application avec S3 Object Lambda](#)
- [Exemples d'approches pour les tests unitaires et d'intégration avec un AWS Kit SDK](#)
- [Charger récursivement un répertoire local dans un compartiment Amazon Simple Storage Service \(Amazon S3\)](#)
- [Chargez ou téléchargez des fichiers volumineux vers et depuis Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Chargez un flux de taille inconnue vers un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Utilisez des checksums pour travailler avec un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Utilisez les fonctionnalités d'intégrité des objets d'Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Travaillez avec des objets versionnés Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Exemples sans serveur pour Amazon S3](#)
 - [Invoquer une fonction lambda à partir d'un déclencheur Amazon S3](#)

Exemples de base illustrant l'utilisation d'Amazon S3 AWS Kits SDK

Les exemples de code suivants illustrent comment utiliser les principes de base d'Amazon Simple Storage Service avec des kits AWS SDK.

Exemples

- [Hello Amazon S3](#)
- [Découvrez les bases d'Amazon S3 grâce à AWS Kit SDK](#)
- [Actions pour Amazon S3 utilisant AWS Kits SDK](#)
 - [À utiliser AbortMultipartUploadavec un AWS SDK ou CLI](#)
 - [À utiliser CompleteMultipartUploadavec un AWS SDK ou CLI](#)
 - [À utiliser CopyObjectavec un AWS SDK ou CLI](#)
 - [À utiliser CreateBucketavec un AWS SDK ou CLI](#)
 - [À utiliser CreateMultiRegionAccessPointavec un AWS Kit SDK](#)
 - [À utiliser CreateMultipartUploadavec un AWS SDK ou CLI](#)
 - [À utiliser CreatePresignedPostavec un AWS Kit SDK](#)
 - [À utiliser DeleteBucketavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketAnalyticsConfiguration avec une CLI](#)
 - [À utiliser DeleteBucketCorsavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketEncryption avec une CLI](#)
 - [Utilisation de DeleteBucketInventoryConfiguration avec une CLI](#)
 - [À utiliser DeleteBucketLifecycleavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketMetricsConfiguration avec une CLI](#)
 - [À utiliser DeleteBucketPolicyavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteBucketReplication avec une CLI](#)
 - [Utilisation de DeleteBucketTagging avec une CLI](#)
 - [À utiliser DeleteBucketWebsiteavec un AWS SDK ou CLI](#)
 - [À utiliser DeleteObjectavec un AWS SDK ou CLI](#)
 - [Utilisation de DeleteObjectTagging avec une CLI](#)
 - [À utiliser DeleteObjectswith un AWS SDK ou CLI](#)
 - [Utilisation de DeletePublicAccessBlock avec une CLI](#)
 - [Utilisation de GetBucketAccelerateConfiguration avec une CLI](#)

- [À utiliser GetBucketAcl avec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketAnalyticsConfiguration avec une CLI](#)
- [À utiliser GetBucketCors avec un AWS SDK ou CLI](#)
- [À utiliser GetBucketEncryption avec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketInventoryConfiguration avec une CLI](#)
- [À utiliser GetBucketLifecycleConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser GetBucketLocation avec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketLogging avec une CLI](#)
- [Utilisation de GetBucketMetricsConfiguration avec une CLI](#)
- [Utilisation de GetBucketNotification avec une CLI](#)
- [À utiliser GetBucketPolicy avec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketPolicyStatus avec une CLI](#)
- [À utiliser GetBucketReplication avec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketRequestPayment avec une CLI](#)
- [Utilisation de GetBucketTagging avec une CLI](#)
- [Utilisation de GetBucketVersioning avec une CLI](#)
- [À utiliser GetBucketWebsite avec un AWS SDK ou CLI](#)
- [À utiliser GetObject avec un AWS SDK ou CLI](#)
- [À utiliser GetObjectAcl avec un AWS SDK ou CLI](#)
- [À utiliser GetObjectAttributes avec un AWS SDK ou CLI](#)
- [À utiliser GetObjectLegalHold avec un AWS SDK ou CLI](#)
- [À utiliser GetObjectLockConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser GetObjectRetention avec un AWS SDK ou CLI](#)
- [Utilisation de GetObjectTagging avec une CLI](#)
- [Utilisation de GetPublicAccessBlock avec une CLI](#)
- [À utiliser HeadBucket avec un AWS SDK ou CLI](#)
- [À utiliser HeadObject avec un AWS SDK ou CLI](#)
- [Utilisation de ListBucketAnalyticsConfigurations avec une CLI](#)
- [Utilisation de ListBucketInventoryConfigurations avec une CLI](#)
- [À utiliser ListBuckets avec un AWS SDK ou CLI](#)

- [À utiliser ListMultipartUploads avec un AWS SDK ou CLI](#)
- [À utiliser ListObjectVersions avec un AWS SDK ou CLI](#)
- [Utilisation de ListObjects avec une CLI](#)
- [Utilisez la ListObjectsV2 avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketAccelerateConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketAcl avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketCors avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketEncryption avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketLifecycleConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketLogging avec un AWS SDK ou CLI](#)
- [Utilisation de PutBucketNotification avec une CLI](#)
- [À utiliser PutBucketNotificationConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketPolicy avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketReplication avec un AWS SDK ou CLI](#)
- [Utilisation de PutBucketRequestPayment avec une CLI](#)
- [Utilisation de PutBucketTagging avec une CLI](#)
- [À utiliser PutBucketVersioning avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketWebsite avec un AWS SDK ou CLI](#)
- [À utiliser PutObject avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectAcl avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectLegalHold avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectLockConfiguration avec un AWS SDK ou CLI](#)
- [À utiliser PutObjectRetention avec un AWS SDK ou CLI](#)
- [À utiliser RestoreObject avec un AWS SDK ou CLI](#)
- [À utiliser SelectObjectContent avec un AWS SDK ou CLI](#)
- [À utiliser UploadPart avec un AWS SDK ou CLI](#)
- [À utiliser UploadPartCopy avec un AWS SDK ou CLI](#)

Hello Amazon S3

Les exemples de code suivants montrent comment démarrer avec Amazon S3.

.NET

SDK pour .NET (v4)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Hello Amazon Simple Storage Service
/// (Amazon S3) example.
/// </summary>
public class HelloS3
{
    /// <summary>
    /// Main method to run the Hello S3 example.
    /// </summary>
    /// <param name="args">Command line arguments.</param>
    /// <returns>A Task object.</returns>
    public static async Task Main(string[] args)
    {
        var s3Client = new AmazonS3Client();

        try
        {
            Console.WriteLine("Hello Amazon S3! Let's list your buckets:");
            Console.WriteLine(new string('-', 80));

            // Use the built-in paginator to list buckets
            var request = new ListBucketsRequest();
            var paginator = s3Client.Paginators.ListBuckets(request);

            var buckets = new List<S3Bucket>();

            await foreach (var response in paginator.Responses)
            {
                buckets.AddRange(response.Buckets);
            }

            if (buckets.Any())
```

```
        {
            Console.WriteLine($"Found {buckets.Count} S3 buckets:");
            Console.WriteLine();

            foreach (var bucket in buckets)
            {
                Console.WriteLine($"- Bucket Name: {bucket.BucketName}");
                Console.WriteLine($"  Creation Date:
{bucket.CreationDate:yyyy-MM-dd HH:mm:ss UTC}");
                Console.WriteLine();
            }
        }
        else
        {
            Console.WriteLine("No S3 buckets found in your account.");
        }

        Console.WriteLine("Hello S3 completed successfully.");
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"S3 service error occurred: {ex.Message}");
    }
    catch (Exception ex)
    {
        Console.WriteLine($"Couldn't list S3 buckets. Here's why:
{ex.Message}");
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK pour .NET API.

C++

SDK pour C++

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Code pour le fichier CMakeLists.txt CMake.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
  libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
      "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
  endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
  # Copy relevant AWS SDK for C++ libraries into the current binary directory
  for running and debugging.
```

```

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
    may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

Code pour le fichier source hello_s3.cpp.

```

#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 * A "Hello S3" starter application which initializes an Amazon Simple Storage
 * Service (Amazon S3) client
 * and lists the Amazon S3 buckets in the selected region.
 *
 * main function
 *
 * Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
    }
}

```

```

// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

// You don't normally have to test that you are authenticated. But the
// S3 service permits anonymous requests, thus the s3Client will return "success"
// and 0 buckets even if you are unauthenticated, which can be confusing to a new
// user.

auto provider =
    Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-tag");
auto creds = provider->GetAWSCredentials();
if (creds.IsEmpty()) {
    std::cerr << "Failed authentication" << std::endl;
}

Aws::S3::S3Client s3Client(clientConfig);
auto outcome = s3Client.ListBuckets();

if (!outcome.IsSuccess()) {
    std::cerr << "Failed with error: " << outcome.GetError() <<
std::endl;
    result = 1;
} else {
    std::cout << "Found " << outcome.GetResult().GetBuckets().size()
        << " buckets\n";
    for (auto &bucket: outcome.GetResult().GetBuckets()) {
        std::cout << bucket.GetName() << std::endl;
    }
}

}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK pour C++ API.

Go

Kit SDK pour Go V2

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
package main

import (
    "context"
    "errors"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/smithy-go"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    s3Client := s3.NewFromConfig(sdkConfig)
    count := 10
    fmt.Printf("Let's list up to %v buckets for your account.\n", count)
    result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        var ae smithy.APIError
```



```

if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
    fmt.Println("You don't have permission to list buckets for this account.")
} else {
    fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
}
return
}
if len(result.Buckets) == 0 {
    fmt.Println("You don't have any buckets!")
} else {
    if count > len(result.Buckets) {
        count = len(result.Buckets)
    }
    for _, bucket := range result.Buckets[:count] {
        fmt.Printf("\t\t%v\n", *bucket.Name)
    }
}
}
}

```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.ListBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class HelloS3 {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listBuckets(s3);
    }

    /**
     * Lists all the S3 buckets associated with the provided AWS S3 client.
     *
     * @param s3 the S3Client instance used to interact with the AWS S3 service
     */
    public static void listBuckets(S3Client s3) {
        try {
            ListBucketsResponse response = s3.listBuckets();
            List<Bucket> bucketList = response.buckets();
            bucketList.forEach(bucket -> {
                System.out.println("Bucket Name: " + bucket.name());
            });

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import {
  paginateListBuckets,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the S3 buckets in your configured AWS account.
 */
export const helloS3 = async () => {
  // When no region or credentials are provided, the SDK will use the
  // region and credentials from the local AWS config.
  const client = new S3Client({});

  try {
    /**
     * @type { import("@aws-sdk/client-s3").Bucket[] }
     */
    const buckets = [];

    for await (const page of paginateListBuckets({ client }, {})) {
      buckets.push(...page.Buckets);
    }
    console.log("Buckets: ");
    console.log(buckets.map((bucket) => bucket.Name).join("\n"));
    return buckets;
  } catch (caught) {
    // ListBuckets does not throw any modeled errors. Any error caught
    // here will be something generic like `AccessDenied`.
    if (caught instanceof S3ServiceException) {
      console.error(`${caught.name}: ${caught.message}`);
    } else {

```

```
        // Something besides S3 failed.  
        throw caught;  
    }  
}  
};
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK pour JavaScript API.

PHP

Kit SDK pour PHP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
use Aws\S3\S3Client;  
  
$client = new S3Client(['region' => 'us-west-2']);  
$results = $client->listBuckets();  
var_dump($results);
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK pour PHP API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import boto3

def hello_s3():
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Simple Storage Service
    (Amazon S3) client and list the buckets in your account.
    This example uses the default settings specified in your shared credentials
    and config files.
    """

    # Create an S3 client.
    s3_client = boto3.client("s3")

    print("Hello, Amazon S3! Let's list your buckets:")

    # Create a paginator for the list_buckets operation.
    paginator = s3_client.get_paginator("list_buckets")

    # Use the paginator to get a list of all buckets.
    response_iterator = paginator.paginate(
        PaginationConfig={
            "PageSize": 50, # Adjust PageSize as needed.
            "StartingToken": None,
        }
    )

    # Iterate through the pages of the response.
    buckets_found = False
    for page in response_iterator:
        if "Buckets" in page and page["Buckets"]:
            buckets_found = True
            for bucket in page["Buckets"]:
                print(f"\t{bucket['Name']}")

    if not buckets_found:
        print("No buckets found!")

if __name__ == "__main__":
    hello_s3()
```

- Pour plus de détails sur l'API, consultez [ListBuckets](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end
  end

  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end
```

```
if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
  manager.list_buckets
end
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK pour Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// S3 Hello World Example using the AWS SDK for Rust.
///
/// This example lists the objects in a bucket, uploads an object to that bucket,
/// and then retrieves the object and prints some S3 information about the
/// object.
/// This shows a number of S3 features, including how to use built-in paginators
/// for large data sets.
///
/// # Arguments
///
/// * `client` - an S3 client configured appropriately for the environment.
/// * `bucket` - the bucket name that the object will be uploaded to. Must be
///   present in the region the `client` is configured to use.
/// * `filename` - a reference to a path that will be read and uploaded to S3.
/// * `key` - the string key that the object will be uploaded as inside the
///   bucket.
async fn list_bucket_and_upload_object(
  client: &aws_sdk_s3::Client,
  bucket: &str,
```

```

    filepath: &Path,
    key: &str,
) -> Result<(), S3ExampleError> {
    // List the buckets in this account
    let mut objects = client
        .list_objects_v2()
        .bucket(bucket)
        .into_paginator()
        .send();

    println!("key\tetag\tlast_modified\tstorage_class");
    while let Some(Ok(object)) = objects.next().await {
        for item in object.contents() {
            println!(
                "{}\t{}\t{}\t{}",
                item.key().unwrap_or_default(),
                item.e_tag().unwrap_or_default(),
                item.last_modified()
                    .map(|lm| format!("{lm}"))
                    .unwrap_or_default(),
                item.storage_class()
                    .map(|sc| format!("{sc}"))
                    .unwrap_or_default()
            );
        }
    }
}

// Prepare a ByteStream around the file, and upload the object using that
// ByteStream.
let body = aws_sdk_s3::primitives::ByteStream::from_path(filepath)
    .await
    .map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to create bytestream for {filepath:?} ({err:?})"
        ))
    })?;
let resp = client
    .put_object()
    .bucket(bucket)
    .key(key)
    .body(body)
    .send()
    .await?;

```



```
println!(
    "Upload success. Version: {:?}",
    resp.version_id()
        .expect("S3 Object upload missing version ID")
);

// Retrieve the just-uploaded object.
let resp = client.get_object().bucket(bucket).key(key).send().await?;
println!("etag: {}", resp.e_tag().unwrap_or("(missing)"));
println!("version: {}", resp.version_id().unwrap_or("(missing)"));

Ok(())
}
```

S3ExampleError services publics.

```
/// S3ExampleError provides a From<T: ProvideErrorMetadata> impl to extract
/// client-specific error details. This serves as a consistent backup to handling
/// specific service errors, depending on what is needed by the scenario.
/// It is used throughout the code examples for the AWS SDK for Rust.
#[derive(Debug)]
pub struct S3ExampleError(String);
impl S3ExampleError {
    pub fn new(value: impl Into<String>) -> Self {
        S3ExampleError(value.into())
    }

    pub fn add_message(self, message: impl Into<String>) -> Self {
        S3ExampleError(format!("{}: {}", message.into(), self.0))
    }
}

impl<T: aws_sdk_s3::error::ProvideErrorMetadata> From<T> for S3ExampleError {
    fn from(value: T) -> Self {
        S3ExampleError(format!(
            "{}: {}",
            value
                .code()
                .map(String::from)
                .unwrap_or("unknown code".into()),
            value
                .message()
        ))
    }
}
```

```
        .map(String::from)
        .unwrap_or("missing reason".into()),
    ))
}

impl std::error::Error for S3ExampleError {}

impl std::fmt::Display for S3ExampleError {
    fn fmt(&self, f: &mut std::fmt::Formatter<'_>) -> std::fmt::Result {
        write!(f, "{}", self.0)
    }
}
```

- Pour plus de détails sur l'API, voir [ListBuckets](#) la section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Découvrez les bases d'Amazon S3 grâce à AWS Kit SDK

Les exemples de code suivants montrent comment :

- créer un compartiment et y charger un fichier ;
- télécharger un objet à partir d'un compartiment ;
- copier un objet dans le sous-dossier d'un compartiment ;
- répertorier les objets d'un compartiment ;
- supprimer le compartiment et tous les objets qui y figurent.

.NET

SDK pour .NET (v4)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif illustrant les fonctionnalités d'Amazon S3.

```
public class S3_Basics
{
    public static bool IsInteractive = true;
    public static string BucketName = null!;
    public static string TempFilePath = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static ILogger<S3_Basics> _logger = null!;

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
            {
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddLogging(builder => builder.AddConsole());
            })
            .Build();

        _logger = LoggerFactory.Create(builder => builder.AddConsole())
            .CreateLogger<S3_Basics>();

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();

        var sepBar = new string('-', 45);

        Console.WriteLine(sepBar);
        Console.WriteLine("Amazon Simple Storage Service (Amazon S3) basic");
        Console.WriteLine("procedures. This application will:");
        Console.WriteLine("\n\t1. Create a bucket");
        Console.WriteLine("\n\t2. Upload an object to the new bucket");
```

```
        Console.WriteLine("\n\t3. Copy the uploaded object to a folder in the
bucket");
        Console.WriteLine("\n\t4. List the items in the new bucket");
        Console.WriteLine("\n\t5. Delete all the items in the bucket");
        Console.WriteLine("\n\t6. Delete the bucket");
        Console.WriteLine(sepBar);

        await RunScenario(_s3Wrapper, _logger);

        Console.WriteLine(sepBar);
        Console.WriteLine("The Amazon S3 scenario has successfully completed.");
        Console.WriteLine(sepBar);
    }

    /// <summary>
    /// Run the S3 Basics scenario with injected dependencies.
    /// </summary>
    /// <param name="s3Wrapper">The S3 wrapper instance.</param>
    /// <param name="scenarioLogger">The logger instance.</param>
    /// <returns>A Task object.</returns>
    public static async Task RunScenario(S3Wrapper s3Wrapper, ILogger<S3_Basics>
scenarioLogger)
    {
        string bucketName = BucketName;
        string filePath = TempFilePath;
        string keyName = string.Empty;

        var sepBar = new string('-', 45);

        try
        {
            // Create a bucket.
            Console.WriteLine($"{sepBar}");
            Console.WriteLine("\nCreate a new Amazon S3 bucket.\n");
            Console.WriteLine(sepBar);

            if (IsInteractive)
            {
                Console.Write("Please enter a name for the new bucket: ");
                bucketName = Console.ReadLine();
            }
            else
            {
                Console.WriteLine($"Using bucket name: {bucketName}");
            }
        }
    }
}
```

```
    }

    var success = await s3Wrapper.CreateBucketAsync(bucketName);
    if (success)
    {
        Console.WriteLine($"Successfully created bucket: {bucketName}.
\n");
    }
    else
    {
        Console.WriteLine($"Could not create bucket: {bucketName}.\n");
    }

    Console.WriteLine(sepBar);
    Console.WriteLine("Upload a file to the new bucket.");
    Console.WriteLine(sepBar);

    if (IsInteractive)
    {
        // Get the local path and filename for the file to upload.
        while (string.IsNullOrEmpty(filePath))
        {
            Console.Write("Please enter the path and filename of the file
to upload: ");
            filePath = Console.ReadLine();

            // Confirm that the file exists on the local computer.
            if (!File.Exists(filePath))
            {
                Console.WriteLine($"Couldn't find {filePath}. Try again.
\n");
                filePath = string.Empty;
            }
        }
    }
    else
    {
        // Use the public variable if set, otherwise create a temp file
        if (!string.IsNullOrEmpty(TempFilePath))
        {
            filePath = TempFilePath;
            Console.WriteLine($"Using provided test file: {filePath}");
        }
        else
```

```
        {
            // Create a temporary test file for non-interactive mode
            filePath = Path.GetTempFileName();
            var testContent = "This is a test file for S3 basics
scenario.\nGenerated on: " + DateTime.UtcNow.ToString("yyyy-MM-dd HH:mm:ss
UTC");

            await File.WriteAllTextAsync(filePath, testContent);
            Console.WriteLine($"Created temporary test file:
{filePath}");
        }
    }

    // Get the file name from the full path.
    keyName = Path.GetFileName(filePath);

    success = await s3Wrapper.UploadFileAsync(bucketName, keyName,
filePath);

    if (success)
    {
        Console.WriteLine($"Successfully uploaded {keyName} from
{filePath} to {bucketName}.\n");
    }
    else
    {
        Console.WriteLine($"Could not upload {keyName}.\n");
    }

    // Set up download path
    string downloadPath = string.Empty;

    if (IsInteractive)
    {
        // Now get a new location where we can save the file.
        while (string.IsNullOrEmpty(downloadPath))
        {
            // First get the path to which the file will be downloaded.
            Console.Write("Please enter the path where the file will be
downloaded: ");

            downloadPath = Console.ReadLine();

            // Confirm that the file doesn't already exist on the local
            computer.

            if (File.Exists($"{downloadPath}\\{keyName}"))
```

```
        {
            Console.WriteLine($"Sorry, the file already exists in
that location.\n");
            downloadPath = string.Empty;
        }
    }
else
{
    downloadPath = Path.GetTempPath();
    var downloadFile = Path.Combine(downloadPath, keyName);
    if (File.Exists(downloadFile))
    {
        File.Delete(downloadFile);
    }

    Console.WriteLine($"Using download path: {downloadPath}");
}

// Download an object from a bucket.
success = await s3Wrapper.DownloadObjectFromBucketAsync(bucketName,
keyName, downloadPath);

if (success)
{
    Console.WriteLine($"Successfully downloaded {keyName}.\n");
}
else
{
    Console.WriteLine($"Sorry, could not download {keyName}.\n");
}

// Copy the object to a different folder in the bucket.
string folderName = string.Empty;

if (IsInteractive)
{
    while (string.IsNullOrEmpty(folderName))
    {
        Console.Write("Please enter the name of the folder to copy
your object to: ");
        folderName = Console.ReadLine();
    }
}
```

```
        else
        {
            folderName = "test-folder";
            Console.WriteLine($"Using folder name: {folderName}");
        }

        await s3Wrapper.CopyObjectInBucketAsync(bucketName, keyName,
folderName);

        // List the objects in the bucket.
        await s3Wrapper.ListBucketContentsAsync(bucketName);

        // Delete the contents of the bucket.
        if (IsInteractive)
        {
            Console.WriteLine("Press <Enter> when you are ready to delete the
bucket contents.");
            _ = Console.ReadLine();
        }

        var deleteContentsSuccess = await
s3Wrapper.DeleteBucketContentsAsync(bucketName);
        if (deleteContentsSuccess)
        {
            Console.WriteLine($"Successfully deleted contents of
{bucketName}.\n");
        }
        else
        {
            Console.WriteLine($"Sorry, could not delete contents of
{bucketName}.\n");
        }

        if (IsInteractive)
        {
            // Deleting the bucket too quickly after separately deleting its
contents can
            // cause an error that the bucket isn't empty. To delete contents
and bucket in one
            // operation, use AmazonS3Util.DeleteS3BucketWithObjectsAsync
            Console.WriteLine("Press <Enter> when you are ready to delete the
bucket.");
            _ = Console.ReadLine();
        }
    }
```



```
        else
        {
            // Add a small delay for non-interactive mode to ensure objects
            are fully deleted.
            Console.WriteLine("Waiting a moment for objects to be fully
            deleted...");
            await Task.Delay(2000);
        }

        // Delete the bucket.
        var deleteSuccess = await s3Wrapper.DeleteBucketAsync(bucketName);
        if (deleteSuccess)
        {
            Console.WriteLine($"Successfully deleted {bucketName}.\n");
        }
        else
        {
            Console.WriteLine($"Sorry, could not delete {bucketName}.\n");
        }

        // Clean up temporary files in non-interactive mode
        if (!IsInteractive)
        {
            try
            {
                if (File.Exists(filePath))
                {
                    File.Delete(filePath);
                    Console.WriteLine("Cleaned up temporary test file.");
                }

                var downloadFile = Path.Combine(downloadPath, keyName);
                if (File.Exists(downloadFile))
                {
                    File.Delete(downloadFile);
                    Console.WriteLine("Cleaned up downloaded test file.");
                }
            }
            catch (Exception ex)
            {
                scenarioLogger.LogWarning(ex, "Failed to clean up temporary
            files.");
            }
        }
    }
```

```
    }
    catch (Exception ex)
    {
        scenarioLogger.LogError(ex, "An error occurred during the S3 scenario
execution.");

        // Clean up on error - delete bucket if it exists
        try
        {
            if (!string.IsNullOrEmpty(bucketName))
            {
                await s3Wrapper.DeleteBucketContentsAsync(bucketName);
                await s3Wrapper.DeleteBucketAsync(bucketName);
            }
        }
        catch (Exception cleanupEx)
        {
            scenarioLogger.LogError(cleanupEx, "Error during cleanup.");
        }

        // Clean up temporary files in non-interactive mode
        if (!IsInteractive)
        {
            try
            {
                if (!string.IsNullOrEmpty(filePath) && File.Exists(filePath))
                {
                    File.Delete(filePath);
                }
            }
            catch (Exception fileCleanupEx)
            {
                scenarioLogger.LogWarning(fileCleanupEx, "Failed to clean up
temporary files during error handling.");
            }
        }

        throw;
    }
}
```

Une classe d'encapsuleur pour les méthodes du kit SDK Amazon S3.

```
using Amazon.S3;
using Amazon.S3.Model;

namespace S3_Actions;

/// <summary>
/// This class contains all of the methods for working with Amazon Simple
/// Storage Service (Amazon S3) buckets.
/// </summary>
public class S3Wrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Initializes a new instance of the <see cref="S3Wrapper"/> class.
    /// </summary>
    /// <param name="amazonS3">An initialized Amazon S3 client object.</param>
    public S3Wrapper(IAmazonS3 amazonS3)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Shows how to create a new Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <returns>A boolean value representing the success or failure of
    /// the bucket creation process.</returns>
    public async Task<bool> CreateBucketAsync(string bucketName)
    {
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
            };

            var response = await _amazonS3.PutBucketAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
    }
}
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Shows how to upload a file from the local computer to an Amazon S3
    /// bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to which the object
    /// will be uploaded.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object
    /// on the local computer to upload.</param>
    /// <returns>A boolean value indicating the success or failure of the
    /// upload procedure.</returns>
    public async Task<bool> UploadFileAsync(
        string bucketName,
        string objectName,
        string filePath)
    {
        try
        {
            var request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = objectName,
                FilePath = filePath,
            };

            var response = await _amazonS3.PutObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
            return false;
        }
    }
}
```

```
/// <summary>
/// Shows how to download an object from an Amazon S3 bucket to the
/// local computer.
/// </summary>
/// <param name="bucketName">The name of the bucket where the object is
/// currently stored.</param>
/// <param name="objectName">The name of the object to download.</param>
/// <param name="filePath">The path, including filename, where the
/// downloaded object will be stored.</param>
/// <returns>A boolean value indicating the success or failure of the
/// download process.</returns>
public async Task<bool> DownloadObjectFromBucketAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    var request = new GetObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
    };

    using GetObjectResponse response = await
        _amazonS3.GetObjectAsync(request);

    try
    {
        // Save object to local file
        await response.WriteResponseStreamToFileAsync($"{filePath}\
        \{objectName}", true, CancellationToken.None);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error saving {objectName}: {ex.Message}");
        return false;
    }
}

/// <summary>
```

```

    /// Copies an object in an Amazon S3 bucket to a folder within the
    /// same bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object to copy is located.</param>
    /// <param name="objectName">The object to be copied.</param>
    /// <param name="folderName">The folder to which the object will
    /// be copied.</param>
    /// <returns>A boolean value that indicates the success or failure of
    /// the copy operation.</returns>
    public async Task<bool> CopyObjectInBucketAsync(
        string bucketName,
        string objectName,
        string folderName)
    {
        try
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Shows how to list the objects in an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket for which to list.
    /// <param name="printList">True to print out the list.
    /// <returns>The collection of objects.</returns>
    public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
        bool printList = true)

```

```
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
        };

        if (printList)
        {
            Console.WriteLine("-----");
            Console.WriteLine($"Listing the contents of {bucketName}:");
            Console.WriteLine("-----");
        }

        var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
        {
            BucketName = bucketName,
        });
        var s3objects = new List<S3Object>();
        await foreach (var response in listObjectsV2Paginator.Responses)
        {
            if (response.S3Objects != null)
            {
                s3objects.AddRange(response.S3Objects);
            }
        }

        if (printList)
        {
            Console.WriteLine($"Number of Objects: {s3objects.Count}");
            foreach (var entry in s3objects)
            {
                Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
            }
        }

        return s3objects;
    }
    catch (AmazonS3Exception ex)
    {

```

```

        Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' getting list of objects.");
        return null;
    }
}

/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>
public async Task<bool> DeleteBucketContentsAsync(string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
    try
    {
        // Delete all objects in the bucket.
        var deleteList = await ListBucketContentsAsync(bucketName, false);
        if (deleteList != null && deleteList.Any())
        {
            await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
            });
        }

        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting objects: {ex.Message}");
        return false;
    }
}

/// <summary>

```



```
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest { BucketName = bucketName, };

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting bucket: {ex.Message}");
        return false;
    }
}
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour .NET .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Bash

AWS CLI avec le script Bash

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# function s3_getting_started
#
# This function creates, copies, and deletes S3 buckets and objects.
#
# Returns:
#     0 - If successful.
#     1 - If an error occurred.
#####
function s3_getting_started() {
    {
        if [ "$BUCKET_OPERATIONS_SOURCED" != "True" ]; then
            cd bucket-lifecycle-operations || exit

            source ./bucket_operations.sh
            cd ..
        fi
    }

    echo_repeat "*" 88
    echo "Welcome to the Amazon S3 getting started demo."
    echo_repeat "*" 88
    echo "A unique bucket will be created by appending a Universally Unique
Identifier to a bucket name prefix."
    echo -n "Enter a prefix for the S3 bucket that will be used in this demo: "
    get_input
    bucket_name_prefix=$get_input_result
    local bucket_name
    bucket_name=$(generate_random_name "$bucket_name_prefix")

    local region_code
    region_code=$(aws configure get region)
```

```
if create_bucket -b "$bucket_name" -r "$region_code"; then
    echo "Created demo bucket named $bucket_name"
else
    errecho "The bucket failed to create. This demo will exit."
    return 1
fi

local file_name
while [ -z "$file_name" ]; do
    echo -n "Enter a file you want to upload to your bucket: "
    get_input
    file_name=$get_input_result

    if [ ! -f "$file_name" ]; then
        echo "Could not find file $file_name. Are you sure it exists?"
        file_name=""
    fi
done

local key
key="$(basename "$file_name")"

local result=0
if copy_file_to_bucket "$bucket_name" "$file_name" "$key"; then
    echo "Uploaded file $file_name into bucket $bucket_name with key $key."
else
    result=1
fi

local destination_file
destination_file="$file_name.download"
if yes_no_input "Would you like to download $key to the file $destination_file?
(y/n) "; then
    if download_object_from_bucket "$bucket_name" "$destination_file" "$key";
then
        echo "Downloaded $key in the bucket $bucket_name to the file
$destination_file."
    else
        result=1
    fi
fi
```

```
if yes_no_input "Would you like to copy $key a new object key in your bucket?
(y/n) "; then
    local to_key
    to_key="demo/$key"
    if copy_item_in_bucket "$bucket_name" "$key" "$to_key"; then
        echo "Copied $key in the bucket $bucket_name to the $to_key."
    else
        result=1
    fi
fi

local bucket_items
bucket_items=$(list_items_in_bucket "$bucket_name")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    result=1
fi

echo "Your bucket contains the following items."
echo -e "Name\t\tSize"
echo "$bucket_items"

if yes_no_input "Delete the bucket, $bucket_name, as well as the objects in it?
(y/n) "; then
    bucket_items=$(echo "$bucket_items" | cut -f 1)

    if delete_items_in_bucket "$bucket_name" "$bucket_items"; then
        echo "The following items were deleted from the bucket $bucket_name"
        echo "$bucket_items"
    else
        result=1
    fi

    if delete_bucket "$bucket_name"; then
        echo "Deleted the bucket $bucket_name"
    else
        result=1
    fi
fi

return $result
}
```

Fonctions Amazon S3 utilisées dans ce scénario.

```
#####
# function create-bucket
#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
#                        create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally
unique."
        echo "  [-r region_code]  The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
            ;;
        esac
    done
}
```

```

        return 0
        ;;
    \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
done

if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
fi

local bucket_config_arg
# A location constraint for "us-east-1" returns an error.
if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
fi

iecho "Parameters:\n"
iecho "    Bucket name:  $bucket_name"
iecho "    Region code:  $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
fi

```

```

}

#####
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.
#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

    response=$(aws s3api put-object \
        --bucket "$bucket_name" \
        --body "$source_file" \
        --key "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}

#####
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#     $1 - The name of the bucket to download the object from.
#     $2 - The path and file name to store the downloaded bucket.
#     $3 - The key (name) of the object in the bucket.
#

```

```

# Returns:
#      0 - If successful.
#      1 - If it fails.
#####
function download_object_from_bucket() {
    local bucket_name=$1
    local destination_file_name=$2
    local object_name=$3
    local response

    response=$(aws s3api get-object \
        --bucket "$bucket_name" \
        --key "$object_name" \
        "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}

#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#      $1 - The name of the bucket to copy the file from and to.
#      $2 - The key of the source file to copy.
#      $3 - The key of the destination file.
#
# Returns:
#      0 - If successful.
#      1 - If it fails.
#####
function copy_item_in_bucket() {
    local bucket_name=$1
    local source_key=$2
    local destination_key=$3
    local response

    response=$(aws s3api copy-object \
        --bucket "$bucket_name" \

```



```

--copy-source "$bucket_name/$source_key" \
--key "$destination_key")

# shellcheck disable=SC2181
if [[ $? -ne 0 ]]; then
    errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
    return 1
fi
}

#####
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     The list of files in text format.
#     And:
#     0 - If successful.
#     1 - If it fails.
#####
function list_items_in_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api list-objects \
        --bucket "$bucket_name" \
        --output text \
        --query 'Contents[].{Key: Key, Size: Size}')

    # shellcheck disable=SC2181
    if [[ ${?} -eq 0 ]]; then
        echo "$response"
    else
        errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
        return 1
    fi
}

```

```
#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - A list of keys in the bucket to delete.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":["
    for key in $keys; do
        delete_items="$delete_items{\"Key\": \"$key\"}, "
    done
    delete_items=${delete_items%?} # Remove the final comma.
    delete_items="$delete_items]"

    response=$(aws s3api delete-objects \
        --bucket "$bucket_name" \
        --delete "$delete_items")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
        return 1
    fi
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
```

```

# Parameters:
#     $1 - The name of the bucket.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api delete-bucket \
        --bucket "$bucket_name")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
        return 1
    fi
}

```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des commandes de l'AWS CLI .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

C++

SDK pour C++

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/utils/UUID.h>
#include <aws/core/utils/StringUtils.h>
#include <aws/core/utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
         * \param bucketName: The S3 bucket's name.
         * \param client: An S3 client.
         * \return bool: Function succeeded.
         */
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);

        //! Delete an object in an S3 bucket.
        /*!
```

```

        \param bucketName: The S3 bucket's name.
        \param key: The key for the object in the S3 bucket.
        \param client: An S3 client.
        \return bool: Function succeeded.
    */
    static bool
    deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
&key,
                           Aws::S3::S3Client &client);
    }
}

//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
    \param bucketNamePrefix: A prefix for a bucket name.
    \param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
    \param saveFilePath: Path for saving a downloaded S3 object.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &bucketNamePrefix,
      const Aws::String &uploadFilePath,
      const Aws::String &saveFilePath,
      const Aws::Client::ClientConfiguration
&clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: <bucketNamePrefix> + "-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = bucketNamePrefix +
        Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;
            createBucketConfiguration.WithLocationConstraint(

```

```

    Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
        clientConfig.region));
    request.WithCreateBucketConfiguration(createBucketConfiguration);
}

    Aws::S3::Model::CreateBucketOutcome outcome =
client.CreateBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        return false;
    } else {
        std::cout << "Created the bucket, '" << bucketName <<
            "', in the region, '" << clientConfig.region << "'." <<
std::endl;
    }
}

// 2. Upload a local file to the bucket.
Aws::String key = "key-for-test";
{
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    std::shared_ptr<Aws::FStream> input_data =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
            uploadFilePath,
            std::ios_base::in |
            std::ios_base::binary);

    if (!input_data->is_open()) {
        std::cerr << "Error: unable to open file, '" << uploadFilePath <<
            "'." << std::endl;
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    }

    request.SetBody(input_data);

```

```

    Aws::S3::Model::PutObjectOutcome outcome =
        client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
            outcome.GetError().GetMessage() << std::endl;
        AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    } else {
        std::cout << "Added the object with the key, '" << key
            << "', to the bucket, '"
            << bucketName << "'." << std::endl;
    }
}

// 3. Download the object to a local file.
{
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Downloaded the object with the key, '" << key
            << "', in the bucket, '"
            << bucketName << "'." << std::endl;

        Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
            GetBody();
        Aws::OFStream outStream(saveFilePath,
                                std::ios_base::out | std::ios_base::binary);
        if (!outStream.is_open()) {
            std::cout << "Error: unable to open file, '" << saveFilePath <<
                "'."
                << std::endl;

```

```

        } else {
            outputStream << ioStream.rdbuf();
            std::cout << "Wrote the downloaded object to the file '"
                << saveFilePath << "'." << std::endl;
        }
    }
}

// 4. Copy the object to a different "folder" in the bucket.
Aws::String copiedToKey = "test-folder/" + key;
{
    Aws::S3::Model::CopyObjectRequest request;
    request.WithBucket(bucketName)
        .WithKey(copiedToKey)
        .WithCopySource(bucketName + "/" + key);

    Aws::S3::Model::CopyObjectOutcome outcome =
        client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error: copyObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Copied the object with the key, '" << key
            << "', to the key, '" << copiedToKey
            << "', in the bucket, '" << bucketName << "'." << std::endl;
    }
}

// 5. List objects in the bucket.
{
    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken;
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }
        Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
            request);

        if (!outcome.IsSuccess()) {

```



```

        std::cerr << "Error: ListObjects: " <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    } else {
        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();
        allObjects.insert(allObjects.end(), objects.begin(),
objects.end());
        continuationToken = outcome.GetResult().GetContinuationToken();
    }
} while (!continuationToken.empty());

std::cout << allObjects.size() << " objects in the bucket, '" <<
bucketName
    << "':" << std::endl;

for (Aws::S3::Model::Object &object: allObjects) {
    std::cout << "    '" << object.GetKey() << "'" << std::endl;
}
}

// 6. Delete all objects in the bucket.
// All objects in the bucket must be deleted before deleting the bucket.
AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

// 7. Delete the bucket.
return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                       const Aws::String &key,
                                       Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {

```

```
        std::cout << "Deleted the object with the key, '" << key
                    << "', from the bucket, '"
                    << bucketName << "'.'" << std::endl;
    }

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client
&client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
                    err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'.'" << std::endl;
    }
    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour C++ .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Go

Kit SDK pour Go V2

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Définissez une structure qui enveloppe les actions de compartiment et d'objet utilisées par le scénario.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}
```

```
// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
    error) {
    var err error
    var output *s3.ListBucketsOutput
    var buckets []types.Bucket
    bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
        &s3.ListBucketsInput{})
    for bucketPaginator.HasMorePages() {
        output, err = bucketPaginator.NextPage(ctx)
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
                fmt.Println("You don't have permission to list buckets for this account.")
                err = apiErr
            } else {
                log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
            }
            break
        } else {
            buckets = append(buckets, output.Buckets...)
        }
    }
    return buckets, err
}

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
    (bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
    exists := true
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
            }
            err = nil
        }
    }
}
```

```

    default:
        log.Printf("Either you don't have access to bucket %v or another error
occurred. "+
            "Here's what happened: %v\n", bucketName, err)
    }
}
} else {
    log.Printf("Bucket %v exists and you already own it.", bucketName)
}

return exists, err
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    })
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", name)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", name)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
        }
    }
    return err
}

```

```
// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:     aws.String(objectKey),
            Body:    file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
                log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
                    fileName, bucketName, objectKey, err)
            }
        } else {
            err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                    aws.String(objectKey)}, time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
            }
        }
    }
    return err
}

// UploadLargeObject uses an upload manager to upload data to an object in a
// bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
```

```

func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
    string, objectKey string, largeObject []byte) error {
    largeBuffer := bytes.NewReader(largeObject)
    var partMiBs int64 = 10
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
        u.PartSize = partMiBs * 1024 * 1024
    })
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    largeBuffer,
    })
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
            log.Printf("Error while uploading object to %s. The object is too large.\n"+
                "The maximum size for a multipart upload is 5TB.", bucketName)
        } else {
            log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
                bucketName, objectKey, err)
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }

    return err
}

// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey

```

```

    if errors.As(err, &noKey) {
        log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
            objectKey, bucketName)
        err = noKey
    } else {
        log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
            objectKey, err)
    }
    return err
}
defer result.Body.Close()
file, err := os.Create(fileName)
if err != nil {
    log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
    return err
}
defer file.Close()
body, err := io.ReadAll(result.Body)
if err != nil {
    log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
        err)
}
_, err = file.Write(body)
return err
}

// DownloadLargeObject uses a download manager to download an object from a
// bucket.
// The download manager gets the data in parts and writes them to a buffer until
// all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
    string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10
    downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
    {
        d.PartSize = partMiBs * 1024 * 1024
    })
    buffer := manager.NewWriteAtBuffer([]byte{})
    _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    })

```



```

    })
    if err != nil {
        log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return buffer.Bytes(), err
}

// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(ctx context.Context, bucketName string,
    objectKey string, folderName string) error {
    objectDest := fmt.Sprintf("%v/%v", folderName, objectKey)
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(bucketName),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
        Key:         aws.String(objectDest),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
                active tier.\n",
                objectKey, bucketName)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                aws.String(objectDest)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectDest)
        }
    }
    return err
}

// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{

```

```

    Bucket:      aws.String(destinationBucket),
    CopySource:  aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
    Key:         aws.String(objectKey),
  })
  if err != nil {
    var notActive *types.ObjectNotInActiveTierError
    if errors.As(err, &notActive) {
      log.Printf("Couldn't copy object %s from %s because the object isn't in the
active tier.\n",
        objectKey, sourceBucket)
      err = notActive
    }
  } else {
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
      ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
aws.String(objectKey)}, time.Minute)
    if err != nil {
      log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
  }
  return err
}

// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
([]types.Object, error) {
  var err error
  var output *s3.ListObjectsV2Output
  input := &s3.ListObjectsV2Input{
    Bucket: aws.String(bucketName),
  }
  var objects []types.Object
  objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
  for objectPaginator.HasMorePages() {
    output, err = objectPaginator.NextPage(ctx)
    if err != nil {
      var noBucket *types.NoSuchBucket
      if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucketName)
        err = noBucket
      }
      break
    }
  }
}

```

```
    } else {
        objects = append(objects, output.Contents...)
    }
}
return objects, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(ctx context.Context, bucketName string,
    objectKeys []string) error {
    var objectIds []types.ObjectIdentifier
    for _, key := range objectKeys {
        objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
    }
    output, err := basics.S3Client.DeleteObjects(ctx, &s3.DeleteObjectsInput{
        Bucket: aws.String(bucketName),
        Delete: &types.Delete{Objects: objectIds, Quiet: aws.Bool(true)},
    })
    if err != nil || len(output.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucketName)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
        }
        } else if len(output.Errors) > 0 {
            for _, outErr := range output.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *output.Errors[0].Message)
        }
    } else {
        for _, delObjs := range output.Deleted {
            err = s3.NewObjectNotExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key: delObjs.Key},
                time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                    *delObjs.Key)
            } else {
                log.Printf("Deleted %s.\n", *delObjs.Key)
            }
        }
    }
}
```

```

    }
    }
}
return err
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is
// returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
}
return err
}

```

Exécutez un scénario interactif qui vous montre comment utiliser des compartiments et des objets S3.

```

import (
    "context"

```

```
"fmt"
"log"
"os"
"strings"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunGetStartedScenario is an interactive example that shows you how to use
// Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
// objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Download an object to a local file.
// 4. Copy an object to a different folder in the bucket.
// 5. List objects in the bucket.
// 6. Delete all objects in the bucket.
// 7. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the `..\..\demotools` folder of this repo.
func RunGetStartedScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := questioner.(*demotools.MockQuestioner)
            if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
                log.Println(r)
            }
        }
    }()
}
```

```
log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 getting started demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}

count := 10
log.Printf("Let's list up to %v buckets for your account:", count)
buckets, err := bucketBasics.ListBuckets(ctx)
if err != nil {
    panic(err)
}
if len(buckets) == 0 {
    log.Println("You don't have any buckets!")
} else {
    if count > len(buckets) {
        count = len(buckets)
    }
    for _, bucket := range buckets[:count] {
        log.Printf("\t%v\n", *bucket.Name)
    }
}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
    demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
```

```
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(ctx, bucketName, smallKey, smallFile)
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
demotools.NotEmpty{})
err = bucketBasics.DownloadFile(ctx, bucketName, smallKey, downloadFileName)
if err != nil {
    panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(ctx, bucketName, smallKey, folderName)
if err != nil {
    panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(ctx, bucketName)
if err != nil {
    panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {
    objKeys = append(objKeys, *object.Key)
    log.Printf("\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
"contents? (y/n)", "y") {
    log.Println("Deleting objects.")
}
```

```
err = bucketBasics.DeleteObjects(ctx, bucketName, objKeys)
if err != nil {
    panic(err)
}
log.Println("Deleting bucket.")
err = bucketBasics.DeleteBucket(ctx, bucketName)
if err != nil {
    panic(err)
}
log.Printf("Deleting downloaded file %v.\n", downloadFileName)
err = os.Remove(downloadFileName)
if err != nil {
    panic(err)
}
} else {
    log.Println("Okay. Don't forget to delete objects from your bucket to avoid charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour Go .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Un exemple de scénario.

```
import java.io.IOException;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 *
 * This Java code example performs the following tasks:
 *
 * 1. Creates an Amazon S3 bucket.
 * 2. Uploads an object to the bucket.
 * 3. Downloads the object to another local file.
 * 4. Uploads an object using multipart upload.
 * 5. List all objects located in the Amazon S3 bucket.
 * 6. Copies the object to another Amazon S3 bucket.
 * 7. Copy the object to another Amazon S3 bucket using multi copy.
 * 8. Deletes the object from the Amazon S3 bucket.
 * 9. Deletes the Amazon S3 bucket.
 */
```

```

public class S3Scenario {

    public static Scanner scanner = new Scanner(System.in);
    static S3Actions s3Actions = new S3Actions();
    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");
    private static final Logger logger =
    LoggerFactory.getLogger(S3Scenario.class);
    public static void main(String[] args) throws IOException {
        final String usage = ""
            Usage:
                <bucketName> <key> <objectPath> <savePath> <toBucket>

            Where:
                bucketName - The name of the  S3 bucket.
                key - The unique identifier for the object stored in the S3
bucket.
                objectPath - The full file path of the object within the S3
bucket (e.g., "documents/reports/annual_report.pdf").
                savePath - The local file path where the object will be
downloaded and saved (e.g., "C:/Users/username/Downloads/annual_report.pdf").
                toBucket - The name of the S3 bucket to which the object will be
copied.

            """;

        if (args.length != 5) {
            logger.info(usage);
            return;
        }

        String bucketName = args[0];
        String key = args[1];
        String objectPath = args[2];
        String savePath = args[3];
        String toBucket = args[4];

        logger.info(DASHES);
        logger.info("Welcome to the Amazon Simple Storage Service (S3) example
scenario.");
        logger.info("")
            Amazon S3 is a highly scalable and durable object storage
            service provided by Amazon Web Services (AWS). It is designed to
store and retrieve

```

any amount of data, from anywhere on the web, at any time.

The `S3AsyncClient` interface in the AWS SDK for Java 2.x provides a set of methods to programmatically interact with the Amazon S3 (Simple Storage Service) service. This allows developers to automate the management and manipulation of S3 buckets and objects as part of their application deployment pipelines. With S3, teams can focus on building and deploying their applications without having to worry about the underlying storage infrastructure required to host and manage large amounts of data.

This scenario walks you through how to perform key operations for this service.

Let's get started...

```
""");
waitForInputToContinue(scanner);
logger.info(DASHES);

try {
    // Run the methods that belong to this scenario.
    runScenario(bucketName, key, objectPath, savePath, toBucket);

} catch (Throwable rt) {
    Throwable cause = rt.getCause();
    if (cause instanceof S3Exception kmsEx) {
        logger.info("KMS error occurred: Error message: {}, Error code
{}", kmsEx.getMessage(), kmsEx.awsErrorDetails().errorCode());
    } else {
        logger.info("An unexpected error occurred: " + rt.getMessage());
    }
}

private static void runScenario(String bucketName, String key, String
objectPath, String savePath, String toBucket) throws Throwable {
    logger.info(DASHES);
    logger.info("1. Create an Amazon S3 bucket.");
    try {
        CompletableFuture<Void> future =
s3Actions.createBucketAsync(bucketName);
        future.join();
    }
```

```
        waitForInputToContinue(scanner);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code  
{})", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }

    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("2. Upload a local file to the Amazon S3 bucket.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<PutObjectResponse> future =
s3Actions.uploadLocalFileAsync(bucketName, key, objectPath);
        future.join();
        logger.info("File uploaded successfully to {}/{}", bucketName, key);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code  
{})", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("3. Download the object to another local file.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
s3Actions.getObjectBytesAsync(bucketName, key, savePath);
```

```
        future.join();
        logger.info("Successfully obtained bytes from S3 object and wrote to
file {}", savePath);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("4. Perform a multipart upload.");
    waitForInputToContinue(scanner);
    String multipartKey = "multiPartKey";
    try {
        // Call the multipartUpload method
        CompletableFuture<Void> future =
s3Actions.multipartUpload(bucketName, multipartKey);
        future.join();
        logger.info("Multipart upload completed successfully for bucket '{} '
and key '{}'", bucketName, multipartKey);

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("5. List all objects located in the Amazon S3 bucket.");
```

```
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.listAllObjectsAsync(bucketName);
            future.join();
            logger.info("Object listing completed successfully.");

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("6. Copy the object to another Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<String> future =
s3Actions.copyBucketObjectAsync(bucketName, key, toBucket);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
```

```
        logger.info("7. Copy the object to another Amazon S3 bucket using multi
copy.");
        waitForInputToContinue(scanner);

        try {
            CompletableFuture<String> future =
s3Actions.performMultiCopy(toBucket, bucketName, key);
            String result = future.join();
            logger.info("Copy operation result: {}", result);

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("KMS error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
        }
        waitForInputToContinue(scanner);
        logger.info(DASHES);

        logger.info(DASHES);
        logger.info("8. Delete objects from the Amazon S3 bucket.");
        waitForInputToContinue(scanner);
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, key);
            future.join();

        } catch (RuntimeException rt) {
            Throwable cause = rt.getCause();
            if (cause instanceof S3Exception s3Ex) {
                logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
            } else {
                logger.info("An unexpected error occurred: " + rt.getMessage());
            }
            throw cause;
        }
        try {
            CompletableFuture<Void> future =
s3Actions.deleteObjectFromBucketAsync(bucketName, "multiPartKey");
```

```
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("9. Delete the Amazon S3 bucket.");
    waitForInputToContinue(scanner);
    try {
        CompletableFuture<Void> future =
s3Actions.deleteBucketAsync(bucketName);
        future.join();

    } catch (RuntimeException rt) {
        Throwable cause = rt.getCause();
        if (cause instanceof S3Exception s3Ex) {
            logger.info("S3 error occurred: Error message: {}, Error code
{}", s3Ex.getMessage(), s3Ex.awsErrorDetails().errorCode());
        } else {
            logger.info("An unexpected error occurred: " + rt.getMessage());
        }
        throw cause;
    }
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    logger.info(DASHES);
    logger.info("You successfully completed the Amazon S3 scenario.");
    logger.info(DASHES);
}

private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        logger.info("");
    }
}
```



```

        logger.info("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            logger.info("Continuing with the program...");
            logger.info("");
            break;
        } else {
            // Handle invalid input.
            logger.info("Invalid input. Please try again.");
        }
    }
}

```

Une classe d'encapsuleur qui contient les opérations.

```

public class S3Actions {

    private static final Logger logger =
        LoggerFactory.getLogger(S3Actions.class);
    private static S3AsyncClient s3AsyncClient;

    public static S3AsyncClient getAsyncClient() {
        if (s3AsyncClient == null) {
            /*
             * The `NettyNioAsyncHttpClient` class is part of the AWS SDK for Java,
             * version 2,
             * and it is designed to provide a high-performance, asynchronous HTTP
             * client for interacting with AWS services.
             * It uses the Netty framework to handle the underlying network
             * communication and the Java NIO API to
             * provide a non-blocking, event-driven approach to HTTP requests and
             * responses.
             */

            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
                .maxConcurrency(50) // Adjust as needed.
                .connectionTimeout(Duration.ofSeconds(60)) // Set the connection
                timeout.
                .readTimeout(Duration.ofSeconds(60)) // Set the read timeout.
                .writeTimeout(Duration.ofSeconds(60)) // Set the write timeout.

```

```

        .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2)) // Set the overall API
call timeout.
        .apiCallAttemptTimeout(Duration.ofSeconds(90)) // Set the
individual call attempt timeout.
        .retryStrategy(RetryMode.STANDARD)
        .build();

        s3AsyncClient = S3AsyncClient.builder()
        .region(Region.US_EAST_1)
        .httpClient(httpClient)
        .overrideConfiguration(overrideConfig)
        .build();
    }
    return s3AsyncClient;
}

/**
 * Creates an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes when the bucket is
created and ready
 * @throws RuntimeException if there is a failure while creating the bucket
 */
public CompletableFuture<Void> createBucketAsync(String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    CompletableFuture<CreateBucketResponse> response =
getAsyncClient().createBucket(bucketRequest);
    return response.thenCompose(resp -> {
        S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<WaiterResponse<HeadBucketResponse>>
waiterResponseFuture =

```

```

        s3Waiter.waitForBucketExists(bucketRequestWait);
        return waiterResponseFuture.thenAccept(waiterResponse -> {
            waiterResponse.matched().response().ifPresent(headBucketResponse
-> {
                logger.info(bucketName + " is ready");
            });
        });
    }).whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to create bucket", ex);
        }
    });
}

/**
 * Uploads a local file to an AWS S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (object name) to use for the uploaded file
 * @param objectPath  the local file path of the file to be uploaded
 * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
 */
public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
    return response.whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to upload file", ex);
        }
    });
}

/**

```

```

    * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
    and writes them to a local file.
    *
    * @param bucketName the name of the S3 bucket containing the object
    * @param keyName    the key (or name) of the S3 object to retrieve
    * @param path        the local file path where the object's bytes will be
    written
    * @return a {@link CompletableFuture} that completes when the object bytes
    have been written to the local file
    */
    public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
    keyName, String path) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        CompletableFuture<ResponseBytes<GetObjectResponse>> response =
        getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
        return response.thenAccept(objectBytes -> {
            try {
                byte[] data = objectBytes.asByteArray();
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
ex);
            }
        });
    }

    /**
    * Asynchronously lists all objects in the specified S3 bucket.
    *
    * @param bucketName the name of the S3 bucket to list objects for
    * @return a {@link CompletableFuture} that completes when all objects have
    been listed
    */

```

```

    public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
        ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
        return paginator.subscribe(response -> {
            response.contents().forEach(s3Object -> {
                logger.info("Object key: " + s3Object.key());
            });
        }).thenRun(() -> {
            logger.info("Successfully listed all objects in the bucket: " +
bucketName);
        }).exceptionally(ex -> {
            throw new RuntimeException("Failed to list objects", ex);
        });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param fromBucket the name of the source S3 bucket
     * @param objectKey the key (name) of the object to be copied
     * @param toBucket the name of the destination S3 bucket
     * @return a {@link CompletableFuture} that completes with the copy result as
a {@link String}
     * @throws RuntimeException if the URL could not be encoded or an S3
exception occurred during the copy
     */
    public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(fromBucket)
            .sourceKey(objectKey)
            .destinationBucket(toBucket)
            .destinationKey(objectKey)
            .build();

        CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
        response.whenComplete((copyRes, ex) -> {

```

```

        if (copyRes != null) {
            logger.info("The " + objectKey + " was copied to " + toBucket);
        } else {
            throw new RuntimeException("An S3 exception occurred during
copy", ex);
        }
    });

    return response.thenApply(CopyObjectResponse::copyObjectResult)
        .thenApply(Object::toString);
}

/**
 * Performs a multipart upload to an Amazon S3 bucket.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (name) of the file to be uploaded
 * @return a {@link CompletableFuture} that completes when the multipart
upload is successful
 */
public CompletableFuture<Void> multipartUpload(String bucketName, String key)
{
    int mB = 1024 * 1024;

    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    return
getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
        .thenCompose(createResponse -> {
            String uploadId = createResponse.uploadId();
            System.out.println("Upload ID: " + uploadId);

            // Upload part 1.
            UploadPartRequest uploadPartRequest1 =
UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(1)
                .contentLength((long) (5 * mB)) // Specify the content length

```

```
        .build());

        CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(1)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Upload part 2.
        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .partNumber(2)
        .contentLength((long) (3 * mB))
        .build();

        CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(2)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Combine the results of both parts.
        return CompletableFuture.allOf(part1Future, part2Future)
        .thenCompose(v -> {
            CompletedPart part1 = part1Future.join();
            CompletedPart part2 = part2Future.join();

            CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
                .build();

            CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
                .bucket(bucketName)
```

```

        .key(key)
        .uploadId(uploadId)
        .multipartUpload(completedMultipartUpload)
        .build();

        // Complete the multipart upload
        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
    });
})
    .thenAccept(response -> System.out.println("Multipart upload
completed successfully"))
    .exceptionally(ex -> {
        System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
        throw new RuntimeException(ex);
    });
}

/**
 * Deletes an object from an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the key (file name) of the object to be deleted
 * @return a {@link CompletableFuture} that completes when the object has
been deleted
 */
public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(key + " was deleted");
        } else {
            throw new RuntimeException("An S3 exception occurred during
delete", ex);
        }
    })
}

```



```
    });

    return response.thenApply(r -> null);
}

/**
 * Deletes an S3 bucket asynchronously.
 *
 * @param bucket the name of the bucket to be deleted
 * @return a {@link CompletableFuture} that completes when the bucket
deletion is successful, or throws a {@link RuntimeException}
 * if an error occurs during the deletion process
 */
public CompletableFuture<Void> deleteBucketAsync(String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(bucket + " was deleted.");
        } else {
            throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
        }
    });
    return response.thenApply(r -> null);
}

public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(toBucket)
        .key(key)
        .build();

    getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
        .thenApply(createMultipartUploadResponse -> {
            String uploadId = createMultipartUploadResponse.uploadId();
            System.out.println("Upload ID: " + uploadId);
        });
}
```

```

        UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
    .sourceBucket(bucketName)
    .destinationBucket(toBucket)
    .sourceKey(key)
    .destinationKey(key)
    .uploadId(uploadId) // Use the valid uploadId.
    .partNumber(1) // Ensure the part number is correct.
    .copySourceRange("bytes=0-1023") // Adjust range as needed
    .build();

        return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
    })
    .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
    .whenComplete((uploadPartCopyResponse, exception) -> {
        if (exception != null) {
            // Handle any exceptions.
            logger.error("Error during upload part copy: " +
exception.getMessage());
        } else {
            // Successfully completed the upload part copy.
            System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
        }
    });
    return null;
}

private static ByteBuffer getRandomByteBuffer(int size) {
    ByteBuffer buffer = ByteBuffer.allocate(size);
    for (int i = 0; i < size; i++) {
        buffer.put((byte) (Math.random() * 256));
    }
    buffer.flip();
    return buffer;
}
}

```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .

- [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Tout d'abord, importez tous les modules nécessaires.

```
// Used to check if currently running file is this file.
import { fileURLToPath } from "node:url";
import { readdirSync, readFileSync, writeFileSync } from "node:fs";

// Local helper utils.
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { wrapText } from "@aws-doc-sdk-examples/lib/utils/util-string.js";

import {
  S3Client,
  CreateBucketCommand,
  PutObjectCommand,
  ListObjectsCommand,
  CopyObjectCommand,
  GetObjectCommand,
  DeleteObjectsCommand,
  DeleteBucketCommand,
} from "@aws-sdk/client-s3";
```

Les importations précédentes font référence à certains utilitaires d'annotations. Ces utilitaires sont locaux au GitHub référentiel lié au début de cette section. À titre de référence, consultez les implémentations suivantes de ces utilitaires.

```
export const dirnameFromMetaUrl = (metaUrl) =>
  fileURLToPath(new URL(".", metaUrl));

import { select, input, confirm, checkbox, password } from "@inquirer/prompts";

export class Prompter {
  /**
   * @param {{ message: string, choices: { name: string, value: string }[] }}
   options
   */
  select(options) {
    return select(options);
  }

  /**
   * @param {{ message: string }} options
   */
  input(options) {
    return input(options);
  }

  /**
   * @param {{ message: string }} options
   */
  password(options) {
    return password({ ...options, mask: true });
  }

  /**
   * @param {string} prompt
   */
  checkContinue = async (prompt = "") => {
    const prefix = prompt && `${prompt} `;
    const ok = await this.confirm({
      message: `${prefix}Continue?`,
    });
    if (!ok) throw new Error("Exiting...");
  };
}
```

```

/**
 * @param {{ message: string }} options
 */
confirm(options) {
  return confirm(options);
}

/**
 * @param {{ message: string, choices: { name: string, value: string }[] }}
options
 */
checkbox(options) {
  return checkbox(options);
}
}

export const wrapText = (text, char = "=") => {
  const rule = char.repeat(80);
  return `${rule}\n  ${text}\n${rule}\n`;
};

```

Les objets sont stockés dans des « compartiments ». Définissons une fonction pour créer un nouveau compartiment.

```

export const createBucket = async () => {
  const bucketName = await prompter.input({
    message: "Enter a bucket name. Bucket names must be globally unique:",
  });
  const command = new CreateBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log("Bucket created successfully.\n");
  return bucketName;
};

```

Les compartiments contiennent des « objets ». Cette fonction charge le contenu d'un répertoire dans votre compartiment sous forme d'objets.

```

export const uploadFilesToBucket = async ({ bucketName, folderPath }) => {
  console.log(`Uploading files from ${folderPath}\n`);
  const keys = readdirSync(folderPath);
  const files = keys.map((key) => {

```

```
const filePath = `${folderPath}/${key}`;  
const fileContent = readFileSync(filePath);  
return {  
  Key: key,  
  Body: fileContent,  
};  
});  
  
for (const file of files) {  
  await s3Client.send(  
    new PutObjectCommand({  
      Bucket: bucketName,  
      Body: file.Body,  
      Key: file.Key,  
    }),  
  );  
  console.log(`${file.Key} uploaded successfully.`);  
}  
};
```

Après avoir chargé des objets, vérifiez qu'ils ont été chargés correctement. Tu peux l'utiliser `ListObjects` pour ça. Vous utiliserez la propriété « Key » (Clé), mais la réponse contient également d'autres propriétés utiles.

```
export const listFilesInBucket = async ({ bucketName }) => {  
  const command = new ListObjectsCommand({ Bucket: bucketName });  
  const { Contents } = await s3Client.send(command);  
  const contentsList = Contents.map((c) => ` • ${c.Key}`).join("\n");  
  console.log("\nHere's a list of files in the bucket:");  
  console.log(`${contentsList}\n`);  
};
```

Il se peut que vous souhaitiez copier un objet d'un compartiment à un autre. Utilisez la `CopyObject` commande pour cela.

```
export const copyFileFromBucket = async ({ destinationBucket }) => {  
  const proceed = await prompter.confirm({  
    message: "Would you like to copy an object from another bucket?",  
  });  
};
```

```
if (!proceed) {
  return;
}
const copy = async () => {
  try {
    const sourceBucket = await prompter.input({
      message: "Enter source bucket name:",
    });
    const sourceKey = await prompter.input({
      message: "Enter source key:",
    });
    const destinationKey = await prompter.input({
      message: "Enter destination key:",
    });

    const command = new CopyObjectCommand({
      Bucket: destinationBucket,
      CopySource: `${sourceBucket}/${sourceKey}`,
      Key: destinationKey,
    });
    await s3Client.send(command);
    await copyFileFromBucket({ destinationBucket });
  } catch (err) {
    console.error("Copy error.");
    console.error(err);
    const retryAnswer = await prompter.confirm({ message: "Try again?" });
    if (retryAnswer) {
      await copy();
    }
  }
};
await copy();
};
```

Il n'existe aucune méthode de kit SDK pour obtenir plusieurs objets d'un compartiment. Vous allez plutôt créer une liste d'objets à charger et sur laquelle itérer.

```
export const downloadFilesFromBucket = async ({ bucketName }) => {
  const { Contents } = await s3Client.send(
    new ListObjectsCommand({ Bucket: bucketName }),
  );
  const path = await prompter.input({
```

```

    message: "Enter destination path for files:",
  });

  for (const content of Contents) {
    const obj = await s3Client.send(
      new GetObjectCommand({ Bucket: bucketName, Key: content.Key }),
    );
    writeFileSync(
      `${path}/${content.Key}`,
      await obj.Body.transformToByteArray(),
    );
  }
  console.log("Files downloaded successfully.\n");
};

```

Il est temps de nettoyer vos ressources. Un compartiment doit être vide avant de pouvoir être supprimé. Ces deux fonctions vident et suppriment le compartiment.

```

export const emptyBucket = async ({ bucketName }) => {
  const listObjectsCommand = new ListObjectsCommand({ Bucket: bucketName });
  const { Contents } = await s3Client.send(listObjectsCommand);
  const keys = Contents.map((c) => c.Key);

  const deleteObjectsCommand = new DeleteObjectsCommand({
    Bucket: bucketName,
    Delete: { Objects: keys.map((key) => ({ Key: key })) },
  });
  await s3Client.send(deleteObjectsCommand);
  console.log(`${bucketName} emptied successfully.\n`);
};

export const deleteBucket = async ({ bucketName }) => {
  const command = new DeleteBucketCommand({ Bucket: bucketName });
  await s3Client.send(command);
  console.log(`${bucketName} deleted successfully.\n`);
};

```

La fonction « main » regroupe tout. Si vous exécutez ce fichier directement, la fonction « main » sera appelée.

```
const main = async () => {
```



```
const OBJECT_DIRECTORY = `${dirnameFromMetaUrl(
  import.meta.url,
)}../../../../resources/sample_files/.sample_media`;

try {
  console.log(wrapText("Welcome to the Amazon S3 getting started example.));
  console.log("Let's create a bucket.");
  const bucketName = await createBucket();
  await prompter.confirm({ message: continueMessage });

  console.log(wrapText("File upload.));
  console.log(
    "I have some default files ready to go. You can edit the source code to
provide your own.",
  );
  await uploadFilesToBucket({
    bucketName,
    folderPath: OBJECT_DIRECTORY,
  });

  await listFilesInBucket({ bucketName });
  await prompter.confirm({ message: continueMessage });

  console.log(wrapText("Copy files.));
  await copyFileFromBucket({ destinationBucket: bucketName });
  await listFilesInBucket({ bucketName });
  await prompter.confirm({ message: continueMessage });

  console.log(wrapText("Download files.));
  await downloadFilesFromBucket({ bucketName });

  console.log(wrapText("Clean up.));
  await emptyBucket({ bucketName });
  await deleteBucket({ bucketName });
} catch (err) {
  console.error(err);
}
};
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour JavaScript .
 - [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for
example, C:/AWS/book2.pdf).
        """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

    val bucketName = args[0]
```

```
val key = args[1]
val objectPath = args[2]
val savePath = args[3]
val toBucket = args[4]

// Create an Amazon S3 bucket.
createBucket(bucketName)

// Update a local file to the Amazon S3 bucket.
putObject(bucketName, key, objectPath)

// Download the object to another local file.
getObjectFromMrap(bucketName, key, savePath)

// List all objects located in the Amazon S3 bucket.
listBucketObs(bucketName)

// Copy the object to another Amazon S3 bucket
copyBucketOb(bucketName, key, toBucket)

// Delete the object from the Amazon S3 bucket.
deleteBucketObs(bucketName, key)

// Delete the Amazon S3 bucket.
deleteBucket(bucketName)
println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}

suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
```

```
) {  
    val metadataVal = mutableMapOf<String, String>()  
    metadataVal["myVal"] = "test"  
  
    val request =  
        PutObjectRequest {  
            bucket = bucketName  
            key = objectKey  
            metadata = metadataVal  
            this.body = Paths.get(objectPath).asByteStream()  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        val response = s3.putObject(request)  
        println("Tag information is ${response.eTag}")  
    }  
}  
  
suspend fun getObjectFromMrap(  
    bucketName: String,  
    keyName: String,  
    path: String,  
) {  
    val request =  
        GetObjectRequest {  
            key = keyName  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.getObject(request) { resp ->  
            val myFile = File(path)  
            resp.body?.writeToFile(myFile)  
            println("Successfully read $keyName from $bucketName")  
        }  
    }  
}  
  
suspend fun listBucketObs(bucketName: String) {  
    val request =  
        ListObjectsRequest {  
            bucket = bucketName  
        }  
}
```

```
S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->

    val response = s3.listObjects(request)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
```

```
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour Kotlin.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

PHP

Kit SDK pour PHP

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
echo("\n");
echo("-----\n");
print("Welcome to the Amazon S3 getting started demo using PHP!\n");
echo("-----\n");

$region = 'us-west-2';

$this->s3client = new S3Client([
    'region' => $region,
]);
/* Inline declaration example
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
*/

$this->bucketName = "amzn-s3-demo-bucket-" . uniqid();

try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $this->bucketName \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}

$fileName = __DIR__ . "/local-file-" . uniqid();
try {
    $this->s3client->putObject([
        'Bucket' => $this->bucketName,
```

```
        'Key' => $fileName,
        'SourceFile' => __DIR__ . '/testfile.txt'
    ]);
    echo "Uploaded $fileName to $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to upload $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with file upload before continuing.");
}

try {
    $file = $this->s3client->getObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
    ]);
    echo "Copied $fileName to $folder/$fileName-copy.\n";
} catch (Exception $exception) {
    echo "Failed to copy $fileName with error: " . $exception-
>getMessage();
    exit("Please fix error with object copying before continuing.");
}

try {
    $contents = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    echo "The contents of your bucket are: \n";
    foreach ($contents['Contents'] as $content) {
```



```
        echo $content['Key'] . "\n";
    }
} catch (Exception $exception) {
    echo "Failed to list objects in $this->bucketName with error: " .
$exception->getMessage();
    exit("Please fix error with listing objects before continuing.");
}

try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $this->s3client->deleteObjects([
        'Bucket' => $this->bucketName,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    if (isset($check['Contents']) && count($check['Contents']) > 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}

try {
    $this->s3client->deleteBucket([
        'Bucket' => $this->bucketName,
    ]);
    echo "Deleted bucket $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $this->bucketName with error: " . $exception-
>getMessage();
    exit("Please fix error with bucket deletion before continuing.");
}
```

```
echo "Successfully ran the Amazon S3 with PHP demo.\n";
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour PHP .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import io
import os
import uuid

import boto3
from boto3.s3.transfer import S3UploadFailedError
from botocore.exceptions import ClientError

def do_scenario(s3_resource):
    print("-" * 88)
    print("Welcome to the Amazon S3 getting started demo!")
    print("-" * 88)
```

```
bucket_name = f"amzn-s3-demo-bucket-{uuid.uuid4()}"
bucket = s3_resource.Bucket(bucket_name)
try:
    bucket.create(
        CreateBucketConfiguration={
            "LocationConstraint": s3_resource.meta.client.meta.region_name
        }
    )
    print(f"Created demo bucket named {bucket.name}.")
except ClientError as err:
    print(f"Tried and failed to create demo bucket {bucket_name}.")
    print(f"\t{err.response['Error']['Code']}: {err.response['Error']
['Message']}")
    print(f"\nCan't continue the demo without a bucket!")
    return

file_name = None
while file_name is None:
    file_name = input("\nEnter a file you want to upload to your bucket: ")
    if not os.path.exists(file_name):
        print(f"Couldn't find file {file_name}. Are you sure it exists?")
        file_name = None

obj = bucket.Object(os.path.basename(file_name))
try:
    obj.upload_file(file_name)
    print(
        f"Uploaded file {file_name} into bucket {bucket.name} with key
{obj.key}."
    )
except S3UploadFailedError as err:
    print(f"Couldn't upload file {file_name} to {bucket.name}.")
    print(f"\t{err}")

answer = input(f"\nDo you want to download {obj.key} into memory (y/n)? ")
if answer.lower() == "y":
    data = io.BytesIO()
    try:
        obj.download_fileobj(data)
        data.seek(0)
        print(f"Got your object. Here are the first 20 bytes:\n")
        print(f"\t{data.read(20)}")
    except ClientError as err:
```

```

        print(f"Couldn't download {obj.key}.")
        print(
            f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
        )

    answer = input(
        f"\nDo you want to copy {obj.key} to a subfolder in your bucket (y/n)? "
    )
    if answer.lower() == "y":
        dest_obj = bucket.Object(f"demo-folder/{obj.key}")
        try:
            dest_obj.copy({"Bucket": bucket.name, "Key": obj.key})
            print(f"Copied {obj.key} to {dest_obj.key}.")
        except ClientError as err:
            print(f"Couldn't copy {obj.key} to {dest_obj.key}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

    print("\nYour bucket contains the following objects:")
    try:
        for o in bucket.objects.all():
            print(f"\t{o.key}")
    except ClientError as err:
        print(f"Couldn't list the objects in bucket {bucket.name}.")
        print(f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}")

    answer = input(
        "\nDo you want to delete all of the objects as well as the bucket (y/n)?
"
    )
    if answer.lower() == "y":
        try:
            bucket.objects.delete()
            bucket.delete()
            print(f"Emptied and deleted bucket {bucket.name}.\n")
        except ClientError as err:
            print(f"Couldn't empty and delete bucket {bucket.name}.")
            print(
                f"\t{err.response['Error']['Code']}:{err.response['Error']
['Message']}"
            )

```

```
)

print("Thanks for watching!")
print("-" * 88)

if __name__ == "__main__":
    do_scenario(boto3.resource("s3"))
```

- Pour plus d'informations sur l'API, consultez les rubriques suivantes dans AWS SDK for Python (Boto3) API Reference.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Ruby

Kit SDK pour Ruby

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
require 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
```

```

    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: 'us-east-1' # NOTE: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts('Tried and failed to create demo bucket.')
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end

  # Requests a file name from the user.
  #
  # @return The name of the file.
  def create_file
    File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
  end

  # Uploads a file to an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
  destination
  # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
  file.
  def upload_file(bucket)
    File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
    s3_object = bucket.object(File.basename('demo.txt'))
    s3_object.upload_file('demo.txt')
    puts("Uploaded file demo.txt into bucket #{bucket.name} with key
  #{s3_object.key}.")
  rescue Aws::Errors::ServiceError => e

```

```
puts("Couldn't upload file demo.txt to #{bucket.name}.")
puts("\t#{e.code}: #{e.message}")
raise
else
  s3_object
end

# Downloads an Amazon S3 object to a file.
#
# @param s3_object [Aws::S3::Object] The object to download.
def download_file(s3_object)
  puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
  answer = gets.chomp.downcase
  if answer == 'y'
    puts('Enter a name for the downloaded file: ')
    file_name = gets.chomp
    s3_object.download_file(file_name)
    puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't download #{s3_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Copies an Amazon S3 object to a subfolder within the same bucket.
#
# @param source_object [Aws::S3::Object] The source object to copy.
# @return [Aws::S3::Object, nil] The destination object.
def copy_object(source_object)
  dest_object = nil
  puts("\nDo you want to copy #{source_object.key} to a subfolder in your
bucket (y/n)? ")
  answer = gets.chomp.downcase
  if answer == 'y'
    dest_object = source_object.bucket.object("demo-folder/
#{source_object.key}")
    dest_object.copy_from(source_object)
    puts("Copied #{source_object.key} to #{dest_object.key}.")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't copy #{source_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

```
else
  dest_object
end

# Lists the objects in an Amazon S3 bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to query.
def list_objects(bucket)
  puts("\nYour bucket contains the following objects:")
  bucket.objects.each do |obj|
    puts("\t#{obj.key}")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list the objects in bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
```



```
s3_object = scenario.upload_file(bucket)
scenario.download_file(s3_object)
scenario.copy_object(s3_object)
scenario.list_objects(bucket)
scenario.delete_bucket(bucket)

puts('Thanks for watching!')
puts('-' * 88)
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo!')
end

run_scenario(ScenarioGettingStarted.new(Aws::S3::Resource.new)) if $PROGRAM_NAME
== __FILE__
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour Ruby .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Rust

SDK pour Rust

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Code pour la caisse binaire qui exécute le scénario.

```
#![allow(clippy::result_large_err)]

//! Purpose
//! Shows how to use the AWS SDK for Rust to get started using
//! Amazon Simple Storage Service (Amazon S3). Create a bucket, move objects
    into and out of it,
//! and delete all resources at the end of the demo.
//!
//! This example follows the steps in "Getting started with Amazon S3" in the
    Amazon S3
//! user guide.
//! - https://docs.aws.amazon.com/AmazonS3/latest/userguide/GetStartedWithS3.html

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::{config::Region, Client};
use s3_code_examples::error::S3ExampleError;
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    let shared_config =
aws_config::from_env().region(region_provider).load().await;
    let client = Client::new(&shared_config);
    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let file_name = "s3/testfile.txt".to_string();
    let key = "test file key name".to_string();
    let target_key = "target_key".to_string();

    if let Err(e) = run_s3_operations(region, client, bucket_name, file_name,
key, target_key).await
    {
        eprintln!("{:?}", e);
    };

    Ok(())
}

async fn run_s3_operations(
    region: Region,
    client: Client,
```

```

    bucket_name: String,
    file_name: String,
    key: String,
    target_key: String,
) -> Result<(), S3ExampleError> {
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;
    let run_example: Result<(), S3ExampleError> = (async {
        s3_code_examples::upload_object(&client, &bucket_name, &file_name,
&key).await?;
        let _object = s3_code_examples::download_object(&client, &bucket_name,
&key).await;
        s3_code_examples::copy_object(&client, &bucket_name, &bucket_name, &key,
&target_key)
            .await?;
        s3_code_examples::list_objects(&client, &bucket_name).await?;
        s3_code_examples::clear_bucket(&client, &bucket_name).await?;
        Ok(())
    })
    .await;
    if let Err(err) = run_example {
        eprintln!("Failed to complete getting-started example: {err:?}");
    }
    s3_code_examples::delete_bucket(&client, &bucket_name).await?;

    Ok(())
}

```

Actions courantes utilisées par le scénario.

```

pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
S3ExampleError> {
    let constraint =
aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()

```

```

        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
    task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}

pub async fn download_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::get_object::GetObjectOutput, S3ExampleError> {

```

```

        client
            .get_object()
            .bucket(bucket_name)
            .key(key)
            .send()
            .await
            .map_err(S3ExampleError::from)
    }

    /// Copy an object from one bucket to another.
    pub async fn copy_object(
        client: &aws_sdk_s3::Client,
        source_bucket: &str,
        destination_bucket: &str,
        source_object: &str,
        destination_object: &str,
    ) -> Result<(), S3ExampleError> {
        let source_key = format!("{source_bucket}/{source_object}");
        let response = client
            .copy_object()
            .copy_source(&source_key)
            .bucket(destination_bucket)
            .key(destination_object)
            .send()
            .await?;

        println!(
            "Copied from {source_key} to {destination_bucket}/{destination_object}
            with etag {}",
            response
                .copy_object_result
                .unwrap_or_else(||
                    aws_sdk_s3::types::CopyObjectResult::builder().build())
                .e_tag()
                .unwrap_or("missing")
        );
        Ok(())
    }

    pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
    Result<(), S3ExampleError> {
        let mut response = client
            .list_objects_v2()
            .bucket(bucket.to_owned())

```

```

        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

while let Some(result) = response.next().await {
    match result {
        Ok(output) => {
            for object in output.contents() {
                println!(" - {}", object.key().unwrap_or("Unknown"));
            }
        }
        Err(err) => {
            eprintln!("{err:?}")
        }
    }
}

Ok(())
}

/// Given a bucket, remove all objects in the bucket, and then ensure no objects
/// remain in the bucket.
pub async fn clear_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<Vec<String>, S3ExampleError> {
    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    // delete_objects no longer needs to be mutable.
    let objects_to_delete: Vec<String> = objects
        .contents()
        .iter()
        .filter_map(|obj| obj.key())
        .map(String::from)
        .collect();

    if objects_to_delete.is_empty() {
        return Ok(vec![]);
    }

    let return_keys = objects_to_delete.clone();

    delete_objects(client, bucket_name, objects_to_delete).await?;

```

```

    let objects = client.list_objects_v2().bucket(bucket_name).send().await?;

    eprintln!("{objects:?}");

    match objects.key_count {
        Some(0) => Ok(return_keys),
        _ => Err(S3ExampleError::new(
            "There were still objects left in the bucket.",
        )),
    }
}

pub async fn delete_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
) -> Result<(), S3ExampleError> {
    let resp = client.delete_bucket().bucket(bucket_name).send().await;
    match resp {
        Ok(_) => Ok(()),
        Err(err) => {
            if err
                .as_service_error()
                .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
                == Some("NoSuchBucket")
            {
                Ok(())
            } else {
                Err(S3ExampleError::from(err))
            }
        }
    }
}
}

```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour Rust.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)

- [ListObjectsV2](#)
- [PutObject](#)

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    " determine our region from our session
    DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
    DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
    " When in the us-east-1 region, you must not specify a constraint
    " In all other regions, specify the region as the constraint
    IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
    ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
    ENDIF.

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
CATCH /aws1/cx_s3_bucketalrdyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
CATCH /aws1/cx_s3_bktalrdyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.
```



```

"Upload an object to an S3 bucket."
TRY.
    "Get contents of file from application server."
    DATA lv_file_content TYPE xstring.
    OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
    READ DATASET iv_key INTO lv_file_content.
    CLOSE DATASET iv_key.

    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_file_content ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Get an object from a bucket. "
TRY.
    DATA(lo_result) = lo_s3->getobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key ).
    DATA(lv_object_data) = lo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" Copy an object to a subfolder in a bucket. "
TRY.
    lo_s3->copyobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
        iv_copysource = |{ iv_bucket_name }/{ iv_key }| ).
    MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" List objects in the bucket. "

```

```

TRY.
    DATA(lo_list) = lo_s3->listobjects(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

DATA text TYPE string VALUE 'Object List - '.
DATA lv_object_key TYPE /aws1/s3_objectkey.
LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
    lv_object_key = lo_object->get_key( ).
    CONCATENATE lv_object_key ', ' INTO text.
ENDLOOP.
MESSAGE text TYPE 'I'.

" Delete the objects in a bucket. "
TRY.
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key ).
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }| ).
    MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Delete the bucket. "
TRY.
    lo_s3->deletebucket(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour SAP ABAP.
- [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Swift

Kit SDK pour Swift

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import AWSS3

import Foundation
import AWSS3
import Smithy
import ClientRuntime

/// A class containing all the code that interacts with the AWS SDK for Swift.
public class ServiceHandler {
    let configuration: S3Client.S3ClientConfiguration
    let client: S3Client

    enum HandlerError: Error {
        case getObjectBody(String)
        case readGetObjectBody(String)
        case missingContents(String)
    }

    /// Initialize and return a new ``ServiceHandler`` object, which is used to
    drive the AWS calls
    /// used for the example.
    ///
```

```
/// - Returns: A new ``ServiceHandler`` object, ready to be called to
///           execute AWS operations.
public init() async throws {
    do {
        configuration = try await S3Client.S3ClientConfiguration()
        // configuration.region = "us-east-2" // Uncomment this to set the
region programmatically.
        client = S3Client(config: configuration)
    }
    catch {
        print("ERROR: ", dump(error, name: "Initializing S3 client"))
        throw error
    }
}

/// Create a new user given the specified name.
///
/// - Parameters:
///   - name: Name of the bucket to create.
/// Throws an exception if an error occurs.
public func createBucket(name: String) async throws {
    var input = CreateBucketInput(
        bucket: name

        // For regions other than "us-east-1", you must set the
locationConstraint in the createBucketConfiguration.
        // For more information, see LocationConstraint in the S3 API guide.
        // https://docs.aws.amazon.com/AmazonS3/latest/API/
API_CreateBucket.html#API_CreateBucket_RequestBody
        if let region = configuration.region {
            if region != "us-east-1" {
                input.createBucketConfiguration =
S3ClientTypes.CreateBucketConfiguration(locationConstraint:
S3ClientTypes.BucketLocationConstraint(rawValue: region))
            }
        }

        do {
            _ = try await client.createBucket(input: input)
        }
        catch let error as BucketAlreadyOwnedByYou {
```

```
        print("The bucket '\(name)' already exists and is owned by you. You
may wish to ignore this exception.")
        throw error
    }
    catch {
        print("ERROR: ", dump(error, name: "Creating a bucket"))
        throw error
    }
}

/// Delete a bucket.
/// - Parameter name: Name of the bucket to delete.
public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    do {
        _ = try await client.deleteBucket(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a bucket"))
        throw error
    }
}

/// Upload a file from local storage to the bucket.
/// - Parameters:
///   - bucket: Name of the bucket to upload the file to.
///   - key: Name of the file to create.
///   - file: Path name of the file to upload.
public func uploadFile(bucket: String, key: String, file: String) async
throws {
    let fileUrl = URL(fileURLWithPath: file)
    do {
        let fileData = try Data(contentsOf: fileUrl)
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        _ = try await client.putObject(input: input)
```

```
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object. "))
        throw error
    }
}

/// Create a file in the specified bucket with the given name. The new
/// file's contents are uploaded from a `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket to create a file in.
///   - key: Name of the file to create.
///   - data: A `Data` object to write into the new file.
public func createFile(bucket: String, key: String, withData data: Data)
async throws {
    let dataStream = ByteStream.data(data)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object. "))
        throw error
    }
}

/// Download the named file to the given directory on the local device.
///
/// - Parameters:
///   - bucket: Name of the bucket that contains the file to be copied.
///   - key: The name of the file to copy from the bucket.
///   - to: The path of the directory on the local device where you want to
///         download the file.
public func downloadFile(bucket: String, key: String, to: String) async
throws {
    let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)
```

```
        let input = GetObjectInput(
            bucket: bucket,
            key: key
        )
        do {
            let output = try await client.getObject(input: input)

            guard let body = output.body else {
                throw HandlerError.getObjectBody("GetObjectInput missing body.")
            }

            guard let data = try await body.readData() else {
                throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
            }

            try data.write(to: fileUrl)
        }
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file. "))
            throw error
        }
    }
}

/// Read the specified file from the given S3 bucket into a Swift
/// `Data` object.
///
/// - Parameters:
///   - bucket: Name of the bucket containing the file to read.
///   - key: Name of the file within the bucket to read.
///
/// - Returns: A `Data` object containing the complete file data.
public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }
    }
```

```

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        return data
    }
    catch {
        print("ERROR: ", dump(error, name: "Reading a file. "))
        throw error
    }
}

/// Copy a file from one bucket to another.
///
/// - Parameters:
///   - sourceBucket: Name of the bucket containing the source file.
///   - name: Name of the source file.
///   - destBucket: Name of the bucket to copy the file into.
public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
    let srcUrl = ("\"(sourceBucket)/
\"(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

    let input = CopyObjectInput(
        bucket: destBucket,
        copySource: srcUrl,
        key: name
    )
    do {
        _ = try await client.copyObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Copying an object. "))
        throw error
    }
}

/// Deletes the specified file from Amazon S3.
///
/// - Parameters:
///   - bucket: Name of the bucket containing the file to delete.
///   - key: Name of the file to delete.

```



```
///
public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a file. "))
        throw error
    }
}

/// Returns an array of strings, each naming one file in the
/// specified bucket.
///
/// - Parameter bucket: Name of the bucket to get a file listing for.
/// - Returns: An array of `String` objects, each giving the name of
///            one file contained in the bucket.
public func listBucketFiles(bucket: String) async throws -> [String] {
    do {
        let input = ListObjectsV2Input(
            bucket: bucket
        )

        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'continuationToken' in
        "ListObjectsV2Output".
        let output = client.listObjectsV2Paginated(input: input)
        var names: [String] = []

        for try await page in output {
            guard let objList = page.contents else {
                print("ERROR: listObjectsV2Paginated returned nil contents.")
                continue
            }

            for obj in objList {
                if let objName = obj.key {
                    names.append(objName)
                }
            }
        }
    }
}
```

```

        }
    }

    return names
}
catch {
    print("ERROR: ", dump(error, name: "Listing objects. "))
    throw error
}
}
}

```

```

import AWSS3

import Foundation
import ServiceHandler
import ArgumentParser

/// The command-line arguments and options available for this
/// example command.
struct ExampleCommand: ParsableCommand {
    @Argument(help: "Name of the S3 bucket to create")
    var bucketName: String

    @Argument(help: "Pathname of the file to upload to the S3 bucket")
    var uploadSource: String

    @Argument(help: "The name (key) to give the file in the S3 bucket")
    var objName: String

    @Argument(help: "S3 bucket to copy the object to")
    var destBucket: String

    @Argument(help: "Directory where you want to download the file from the S3
bucket")
    var downloadDir: String

    static var configuration = CommandConfiguration(
        commandName: "s3-basics",
        abstract: "Demonstrates a series of basic AWS S3 functions.",
        discussion: ""
    )
}

```

```
    Performs the following Amazon S3 commands:

    * `CreateBucket`
    * `PutObject`
    * `GetObject`
    * `CopyObject`
    * `ListObjects`
    * `DeleteObjects`
    * `DeleteBucket`
    ""
)

/// Called by ``main()`` to do the actual running of the AWS
/// example.
func runAsync() async throws {
    let serviceHandler = try await ServiceHandler()

    // 1. Create the bucket.
    print("Creating the bucket \(bucketName)...")
    try await serviceHandler.createBucket(name: bucketName)

    // 2. Upload a file to the bucket.
    print("Uploading the file \(uploadSource)...")
    try await serviceHandler.uploadFile(bucket: bucketName, key: objName,
file: uploadSource)

    // 3. Download the file.
    print("Downloading the file \(objName) to \(downloadDir)...")
    try await serviceHandler.downloadFile(bucket: bucketName, key: objName,
to: downloadDir)

    // 4. Copy the file to another bucket.
    print("Copying the file to the bucket \(destBucket)...")
    try await serviceHandler.copyFile(from: bucketName, name: objName, to:
destBucket)

    // 5. List the contents of the bucket.

    print("Getting a list of the files in the bucket \(bucketName)")
    let fileList = try await serviceHandler.listBucketFiles(bucket:
bucketName)
    let numFiles = fileList.count
    if numFiles != 0 {
```

```
        print("\(numFiles) file\((numFiles > 1) ? "s" : "") in bucket
        \(bucketName):")
        for name in fileList {
            print("  \(name)")
        }
    } else {
        print("No files found in bucket \(bucketName)")
    }

    // 6. Delete the objects from the bucket.

    print("Deleting the file \(objName) from the bucket \(bucketName)...")
    try await serviceHandler.deleteFile(bucket: bucketName, key: objName)
    print("Deleting the file \(objName) from the bucket \(destBucket)...")
    try await serviceHandler.deleteFile(bucket: destBucket, key: objName)

    // 7. Delete the bucket.
    print("Deleting the bucket \(bucketName)...")
    try await serviceHandler.deleteBucket(name: bucketName)

    print("Done.")
}
}

//
// Main program entry point.
//
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la référence de l'API du kit SDK AWS pour Swift.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Actions pour Amazon S3 utilisant AWS Kits SDK

Les exemples de code suivants montrent comment effectuer des actions Amazon S3 individuelles avec des AWS SDK. Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions pour configurer et exécuter le code.

Ces extraits appellent l'API Amazon S3 et sont des extraits de code de programmes plus volumineux qui doivent être exécutés en contexte. Vous pouvez voir les actions dans leur contexte dans [Scénarios pour Amazon S3 utilisant AWS Kits SDK](#).

Les exemples suivants incluent uniquement les actions les plus couramment utilisées. Pour une description complète, consultez le document [Référence de l'API Amazon Simple Storage Service](#).

Exemples

- [À utiliser AbortMultipartUploadavec un AWS SDK ou CLI](#)
- [À utiliser CompleteMultipartUploadavec un AWS SDK ou CLI](#)
- [À utiliser CopyObjectavec un AWS SDK ou CLI](#)
- [À utiliser CreateBucketavec un AWS SDK ou CLI](#)
- [À utiliser CreateMultiRegionAccessPointavec un AWS Kit SDK](#)
- [À utiliser CreateMultipartUploadavec un AWS SDK ou CLI](#)
- [À utiliser CreatePresignedPostavec un AWS Kit SDK](#)

- [À utiliser DeleteBucketavec un AWS SDK ou CLI](#)
- [Utilisation de DeleteBucketAnalyticsConfiguration avec une CLI](#)
- [À utiliser DeleteBucketCorsavec un AWS SDK ou CLI](#)
- [Utilisation de DeleteBucketEncryption avec une CLI](#)
- [Utilisation de DeleteBucketInventoryConfiguration avec une CLI](#)
- [À utiliser DeleteBucketLifecycleavec un AWS SDK ou CLI](#)
- [Utilisation de DeleteBucketMetricsConfiguration avec une CLI](#)
- [À utiliser DeleteBucketPolicyavec un AWS SDK ou CLI](#)
- [Utilisation de DeleteBucketReplication avec une CLI](#)
- [Utilisation de DeleteBucketTagging avec une CLI](#)
- [À utiliser DeleteBucketWebsiteavec un AWS SDK ou CLI](#)
- [À utiliser DeleteObjectavec un AWS SDK ou CLI](#)
- [Utilisation de DeleteObjectTagging avec une CLI](#)
- [À utiliser DeleteObjectswith un AWS SDK ou CLI](#)
- [Utilisation de DeletePublicAccessBlock avec une CLI](#)
- [Utilisation de GetBucketAccelerateConfiguration avec une CLI](#)
- [À utiliser GetBucketAclavec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketAnalyticsConfiguration avec une CLI](#)
- [À utiliser GetBucketCorsavec un AWS SDK ou CLI](#)
- [À utiliser GetBucketEncryptionavec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketInventoryConfiguration avec une CLI](#)
- [À utiliser GetBucketLifecycleConfigurationavec un AWS SDK ou CLI](#)
- [À utiliser GetBucketLocationavec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketLogging avec une CLI](#)
- [Utilisation de GetBucketMetricsConfiguration avec une CLI](#)
- [Utilisation de GetBucketNotification avec une CLI](#)
- [À utiliser GetBucketPolicyavec un AWS SDK ou CLI](#)
- [Utilisation de GetBucketPolicyStatus avec une CLI](#)
- [À utiliser GetBucketReplicationavec un AWS SDK ou CLI](#)

- [Utilisation de GetBucketRequestPayment avec une CLI](#)
- [Utilisation de GetBucketTagging avec une CLI](#)
- [Utilisation de GetBucketVersioning avec une CLI](#)
- [À utiliser GetBucketWebsiteavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectAclavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectAttributesavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectLegalHoldavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectLockConfigurationavec un AWS SDK ou CLI](#)
- [À utiliser GetObjectRetentionavec un AWS SDK ou CLI](#)
- [Utilisation de GetObjectTagging avec une CLI](#)
- [Utilisation de GetPublicAccessBlock avec une CLI](#)
- [À utiliser HeadBucketavec un AWS SDK ou CLI](#)
- [À utiliser HeadObjectavec un AWS SDK ou CLI](#)
- [Utilisation de ListBucketAnalyticsConfigurations avec une CLI](#)
- [Utilisation de ListBucketInventoryConfigurations avec une CLI](#)
- [À utiliser ListBucketsavec un AWS SDK ou CLI](#)
- [À utiliser ListMultipartUploadsavec un AWS SDK ou CLI](#)
- [À utiliser ListObjectVersionsavec un AWS SDK ou CLI](#)
- [Utilisation de ListObjects avec une CLI](#)
- [Utilisez la ListObjectsV2 avec un AWS SDK ou CLI](#)
- [À utiliser PutBucketAccelerateConfigurationavec un AWS SDK ou CLI](#)
- [À utiliser PutBucketAclavec un AWS SDK ou CLI](#)
- [À utiliser PutBucketCorsavec un AWS SDK ou CLI](#)
- [À utiliser PutBucketEncryptionavec un AWS SDK ou CLI](#)
- [À utiliser PutBucketLifecycleConfigurationavec un AWS SDK ou CLI](#)
- [À utiliser PutBucketLoggingavec un AWS SDK ou CLI](#)
- [Utilisation de PutBucketNotification avec une CLI](#)
- [À utiliser PutBucketNotificationConfigurationavec un AWS SDK ou CLI](#)
- [À utiliser PutBucketPolicyavec un AWS SDK ou CLI](#)

- [À utiliser PutBucketReplicationavec un AWS SDK ou CLI](#)
- [Utilisation de PutBucketRequestPayment avec une CLI](#)
- [Utilisation de PutBucketTagging avec une CLI](#)
- [À utiliser PutBucketVersioningavec un AWS SDK ou CLI](#)
- [À utiliser PutBucketWebsiteavec un AWS SDK ou CLI](#)
- [À utiliser PutObjectavec un AWS SDK ou CLI](#)
- [À utiliser PutObjectAclavec un AWS SDK ou CLI](#)
- [À utiliser PutObjectLegalHoldavec un AWS SDK ou CLI](#)
- [À utiliser PutObjectLockConfigurationavec un AWS SDK ou CLI](#)
- [À utiliser PutObjectRetentionavec un AWS SDK ou CLI](#)
- [À utiliser RestoreObjectavec un AWS SDK ou CLI](#)
- [À utiliser SelectObjectContentavec un AWS SDK ou CLI](#)
- [À utiliser UploadPartavec un AWS SDK ou CLI](#)
- [À utiliser UploadPartCopyavec un AWS SDK ou CLI](#)

À utiliser **AbortMultipartUpload**avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser AbortMultipartUpload.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action dans son contexte dans les exemples de code suivants :

- [Suppression des chargements partitionnés incomplets](#)
- [Utilisation de l'intégrité des objets Amazon S3](#)

C++

SDK pour C++

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```


    //! Abort a multipart upload to an S3 bucket.
    /*!
        \param bucket: The name of the S3 bucket where the object will be uploaded.
        \param key: The unique identifier (key) for the object within the S3 bucket.
        \param uploadID: An upload ID string.
        \param client: The S3 client instance used to perform the upload operation.
        \return bool: Function succeeded.
    */

    bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                           const Aws::String &key,
                                           const Aws::String &uploadID,
                                           const Aws::S3::S3Client &client) {
        Aws::S3::Model::AbortMultipartUploadRequest request;
        request.SetBucket(bucket);
        request.SetKey(key);
        request.SetUploadId(uploadID);

        Aws::S3::Model::AbortMultipartUploadOutcome outcome =
            client.AbortMultipartUpload(request);

        if (outcome.IsSuccess()) {
            std::cout << "Multipart upload aborted." << std::endl;
        } else {
            std::cerr << "Error aborting multipart upload: " <<
            outcome.GetError().GetMessage() << std::endl;
        }

        return outcome.IsSuccess();
    }


```

- Pour plus de détails sur l'API, reportez-vous [AbortMultipartUpload](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour annuler le chargement partitionné spécifié

La commande `abort-multipart-upload` suivante interrompt le chargement partitionné de la clé `multipart/01` dans le compartiment `amzn-s3-demo-bucket`.

```
aws s3api abort-multipart-upload \
  --bucket amzn-s3-demo-bucket \
  --key multipart/01 \
  --upload-
id dfRtDYU0WMCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P
```

L'ID de chargement requis par cette commande est généré par `create-multipart-upload` et peut également être extrait avec `list-multipart-uploads`.

- Pour plus de détails sur l'API, reportez-vous [AbortMultipartUpload](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
```

```
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class AbortMultipartUploadExamples {
    static final S3Client s3Client = S3Client.create();
    static final String classPathFilePath = "/multipartUploadFiles/s3-
userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    private static final Logger logger =
        LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
```

```

        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.
        String key = UUID.randomUUID().toString();

        createBucket(bucketName);
        try {
            initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread");
            abortIncompleteMultipartUploadsFromList(bucketName);
            abortMultipartUploadUsingUploadId(bucketName, key);
            abortMultipartUploadsUsingLifecycleConfig(bucketName);
        } catch (S3Exception e) {
            logger.error(e.getMessage());
        } finally {
            deleteResources(bucketName, key);
        }
    }

    /**
     * Aborts all incomplete multipart uploads from the specified S3 bucket.
     *
     * @param bucketName the name of the S3 bucket
     */
    public static void abortIncompleteMultipartUploadsFromList(String bucketName)
    {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
        ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        ListMultipartUploadsResponse response =
        s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
            s3Client.abortMultipartUpload(abortMultipartUploadRequest);

```

```

        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}

/**
 * Aborts incomplete multipart uploads older than the specified point in
time.
 *
 * @param bucketName the name of the S3 bucket
 * @param pointInTime the cutoff time; uploads initiated before this are
aborted
 */
static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
        if (upload.initiated().isBefore(pointInTime)) {
            abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {

```

```

        logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
    }
}

/**
 * Aborts a multipart upload using the upload ID.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 */
static void abortMultipartUploadUsingUploadId(String bucketName, String key)
{
    String uploadId = startUploadReturningUploadId(bucketName, key);
    AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
    }
}

/**
 * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
 *
 * @param bucketName the name of the S3 bucket
 */
static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
    Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());
}

```

```

        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }

    /*****
    Multipart upload methods
    *****/

    static void initiateAndInterruptMultiPartUpload(String bucketName, String
key, String threadName) {
        Runnable upload = () -> {
            try {
                doMultipartUpload(bucketName, key);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        };
        Thread uploadThread = new Thread(upload, threadName);
        uploadThread.start();
        try {
            Thread.sleep(Duration.ofSeconds(1).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        uploadThread.interrupt();
    }

    static Instant initiateAndInterruptTwoUploads(String bucketName, String key)
{
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread1");
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
    }

```

```
    }
    Instant secondUploadInstant = Instant.now();
    initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread2");
    return secondUploadInstant;
}

static void doMultipartUpload(String bucketName, String key) {
    String uploadId = step1CreateMultipartUpload(bucketName, key);
    List<CompletedPart> completedParts = step2UploadParts(bucketName, key,
uploadId);
    step3CompleteMultipartUpload(bucketName, key, uploadId, completedParts);
}

static String step1CreateMultipartUpload(String bucketName, String key) {
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    return createMultipartUploadResponse.uploadId();
}

static List<CompletedPart> step2UploadParts(String bucketName, String key,
String uploadId) {
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip();
            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
```



```

        requestBody.fromByteBuffer(bb));

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNumber)
            .eTag(partResponse.eTag())
            .build();
        completedParts.add(part);
        logger.info("Part {} upload", partNumber);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException | S3Exception e) {
    logger.error(e.getMessage());
    return null;
}
return completedParts;
}

static void step3CompleteMultipartUpload(String bucketName, String key,
String uploadId, List<CompletedPart> completedParts) {
    s3Client.completeMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)

        .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}

static String startUploadReturningUploadId(String bucketName, String key) {
    String uploadId = step1CreateMultipartUpload(bucketName, key);
    doMultipartUploadWithUploadId(bucketName, key, uploadId);
    return uploadId;
}

static void doMultipartUploadWithUploadId(String bucketName, String key,
String uploadId) {
    new Thread(() -> {
        try {
            List<CompletedPart> completedParts = step2UploadParts(bucketName,
key, uploadId);
            step3CompleteMultipartUpload(bucketName, key, uploadId,
completedParts);
        }
    })
    .start();
}

```

```

        } catch (SdkException e) {
            logger.error(e.getMessage());
        }
    }, "upload thread").start();
    try {
        Thread.sleep(Duration.ofSeconds(2L).toMillis());
    } catch (InterruptedException e) {
        logger.error(e.getMessage());
        System.exit(1);
    }
}

/*****
Resource handling methods
*****/

static void createBucket(String bucketName) {
    logger.info("Creating bucket: [{}]", bucketName);
    s3Client.createBucket(b -> b.bucket(bucketName));
    try (S3Waiter s3Waiter = s3Client.waiter()) {
        s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
    }
    logger.info("Bucket created.");
}

static void deleteResources(String bucketName, String key) {
    logger.info("Deleting resources ...");
    s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
    s3Client.deleteBucket(b -> b.bucket(bucketName));
    try (S3Waiter s3Waiter = s3Client.waiter()) {
        s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
    }
    logger.info("Resources deleted.");
}

private static String getAccountId() {
    try (StsClient stsClient = StsClient.create()) {
        return stsClient.getCallerIdentity().account();
    }
}

static String getFullFilePath(String filePath) {
    URL uploadDirectoryURL =
PerformMultiPartUpload.class.getResource(filePath);

```

```
String fullFilePath;
try {
    fullFilePath =
Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
} catch (URISyntaxException e) {
    throw new RuntimeException(e);
}
return fullFilePath;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [AbortMultipartUpload](#) à la section Référence des AWS SDK for Java 2.x API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande annule les chargements partitionnés créés il y a moins de cinq jours.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Exemple 2 : cette commande annule les chargements partitionnés créés avant le 2 janvier 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate
"Thursday, January 02, 2014"
```

Exemple 3 : cette commande annule les chargements partitionnés créés avant le 2 janvier 2014, à 10 h 45 m 37 s.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate
"2014/01/02 10:45:37"
```

- Pour plus de détails sur l'API, reportez-vous [AbortMultipartUpload](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande annule les chargements partitionnés créés il y a moins de cinq jours.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Exemple 2 : cette commande annule les chargements partitionnés créés avant le 2 janvier 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

Exemple 3 : cette commande annule les chargements partitionnés créés avant le 2 janvier 2014, à 10 h 45 m 37 s.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- Pour plus de détails sur l'API, reportez-vous [AbortMultipartUpload](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **CompleteMultipartUpload** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser CompleteMultipartUpload.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action dans son contexte dans les exemples de code suivants :

- [Effectuer une copie en plusieurs parties](#)
- [Utiliser les totaux de contrôle](#)
- [Utilisation de l'intégrité des objets Amazon S3](#)

C++

SDK pour C++

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

    const Aws::String &key,

    const Aws::String &uploadID,

    const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

    const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
        client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
```

```
std::cerr << "Error completing multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
}
return outcome;
}
```

- Pour plus de détails sur l'API, reportez-vous [CompleteMultipartUpload](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante effectue un chargement partitionné de la clé `multipart/01` dans le compartiment `amzn-s3-demo-bucket` :

```
aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P
```

L'ID de chargement requis par cette commande est généré par `create-multipart-upload` et peut également être extrait avec `list-multipart-uploads`.

L'option de chargement partitionné de la commande ci-dessus utilise une structure JSON qui décrit les parties du chargement partitionné qui doivent être réassemblées dans le fichier complet. Dans cet exemple, le préfixe `file://` est utilisé pour charger la structure JSON à partir d'un fichier du dossier local nommé `mpustruct`.

`mpustruct` :

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    }
  ]
}
```

```
    },  
    {  
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",  
      "PartNumber": 3  
    }  
  ]  
}
```

La valeur ETag de chaque partie chargée est générée chaque fois que vous chargez une partie à l'aide de la commande `upload-part`. Elle peut également être récupérée en appelant `list-parts` ou calculée en utilisant la somme de contrôle MD5 de chaque partie.

Sortie :

```
{  
  "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",  
  "Bucket": "amzn-s3-demo-bucket",  
  "Location": "https://amzn-s3-demo-bucket.s3.amazonaws.com/multipart%2F01",  
  "Key": "multipart/01"  
}
```

- Pour plus de détails sur l'API, reportez-vous [CompleteMultipartUpload](#) à la section Référence des AWS CLI commandes.

Rust

SDK pour Rust

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>  
let completed_multipart_upload: CompletedMultipartUpload =  
CompletedMultipartUpload::builder()  
    .set_parts(Some(upload_parts))  
    .build();
```

```
let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;
```

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
```



```
        .upload_part()  
        .key(&key)  
        .bucket(&bucket_name)  
        .upload_id(upload_id)  
        .body(stream)  
        .part_number(part_number)  
        .send()  
        .await?;  
  
    upload_parts.push(  
        CompletedPart::builder()  
            .e_tag(upload_part_res.e_tag.unwrap_or_default())  
            .part_number(part_number)  
            .build(),  
    );  
}
```

- Pour plus de détails sur l'API, voir [CompleteMultipartUpload](#) la section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **CopyObject** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser CopyObject.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Principes de base](#)
- [Démarrer avec le chiffrement](#)
- [Débuter avec le stockage d'objets](#)
- [Réalisation de demandes conditionnelles](#)

.NET

SDK pour .NET (v4)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Copies an object in an Amazon S3 bucket to a folder within the
/// same bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket where the
/// object to copy is located.</param>
/// <param name="objectName">The object to be copied.</param>
/// <param name="folderName">The folder to which the object will
/// be copied.</param>
/// <returns>A boolean value that indicates the success or failure of
/// the copy operation.</returns>
public async Task<bool> CopyObjectInBucketAsync(
    string bucketName,
    string objectName,
    string folderName)
{
    try
    {
        var request = new CopyObjectRequest
        {
            SourceBucket = bucketName,
            SourceKey = objectName,
            DestinationBucket = bucketName,
            DestinationKey = $"{folderName}\\{objectName}",
        };
        var response = await _amazonS3.CopyObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error copying object: '{ex.Message}'");
    }
}
```

```
        return false;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS SDK pour .NET API.

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Copiez un objet à l'aide d'une demande conditionnelle.

```
/// <summary>
/// Copies an object from one Amazon S3 bucket to another with a conditional
request.
/// </summary>
/// <param name="sourceKey">The key of the source object to copy.</param>
/// <param name="destKey">The key of the destination object.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="destBucket">The destination bucket of the object.</param>
/// <param name="conditionType">The type of condition to apply, e.g.
'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional copy is successful, False otherwise.</
returns>
public async Task<bool> CopyObjectConditional(string sourceKey, string
destKey, string sourceBucket, string destBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
```

```
var copyObjectRequest = new CopyObjectRequest
{
    DestinationBucket = destBucket,
    DestinationKey = destKey,
    SourceBucket = sourceBucket,
    SourceKey = sourceKey
};

switch (conditionType)
{
    case S3ConditionType.IfMatch:
        copyObjectRequest.ETagToMatch = etagConditionalValue;
        break;
    case S3ConditionType.IfNoneMatch:
        copyObjectRequest.ETagToNotMatch = etagConditionalValue;
        break;
    case S3ConditionType.IfModifiedSince:
        copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
        break;
    case S3ConditionType.IfUnmodifiedSince:
        copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
        break;
    default:
        throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
}

await _amazonS3.CopyObjectAsync(copyObjectRequest);
_logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
return true;
}
catch (AmazonS3Exception e)
{
    if (e.ErrorCode == "PreconditionFailed")
    {
        _logger.LogError("Conditional copy failed: Precondition failed");
    }
    else if (e.ErrorCode == "304")
    {
        _logger.LogError("Conditional copy failed: Object not modified");
    }
}
```

```

        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS SDK pour .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function copy_item_in_bucket
#
# This function creates a copy of the specified file in the same bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file from and to.
#     $2 - The key of the source file to copy.
#     $3 - The key of the destination file.

```

```
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_item_in_bucket() {
    local bucket_name=$1
    local source_key=$2
    local destination_key=$3
    local response

    response=$(aws s3api copy-object \
        --bucket "$bucket_name" \
        --copy-source "$bucket_name/$source_key" \
        --key "$destination_key")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api copy-object operation failed.\n$response"
        return 1
    fi
}
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
    &fromBucket, const Aws::String &toBucket,
                        const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
```

```

    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
            .WithKey(objectKey)
            .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
                    err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;

        } else {
            std::cout << "Successfully copied " << objectKey << " from " <<
fromBucket <<
                    " to " << toBucket << "." << std::endl;
        }

        return outcome.IsSuccess();
    }

```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante copie un objet de bucket-1 dans bucket-2 :

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --
bucket bucket-2
```

Sortie :

```
{
  "CopyObjectResult": {
    "LastModified": "2015-11-10T01:07:25.000Z",
    "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""
  },

```

```
"VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifM1kA"
}
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "io"
    "log"
    "os"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
```



```
S3Client *s3.Client
}

// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(destinationBucket),
        CopySource:   aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
        Key:          aws.String(objectKey),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
                active tier.\n",
                objectKey, sourceBucket)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
                aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }
    return err
}
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Copiez un objet en utilisant un [S3Client](#).

```
/**
 * Asynchronously copies an object from one S3 bucket to another.
 *
 * @param fromBucket the name of the source S3 bucket
 * @param objectKey the key (name) of the object to be copied
 * @param toBucket the name of the destination S3 bucket
 * @return a {@link CompletableFuture} that completes with the copy result as
 * a {@link String}
 * @throws RuntimeException if the URL could not be encoded or an S3
 * exception occurred during the copy
 */
public CompletableFuture<String> copyBucketObjectAsync(String fromBucket,
String objectKey, String toBucket) {
    CopyObjectRequest copyReq = CopyObjectRequest.builder()
        .sourceBucket(fromBucket)
        .sourceKey(objectKey)
        .destinationBucket(toBucket)
        .destinationKey(objectKey)
        .build();

    CompletableFuture<CopyObjectResponse> response =
getAsyncClient().copyObject(copyReq);
    response.whenComplete((copyRes, ex) -> {
        if (copyRes != null) {
            logger.info("The " + objectKey + " was copied to " + toBucket);
        } else {
            throw new RuntimeException("An S3 exception occurred during
copy", ex);
        }
    });
}
```

```
        return response.thenApply(CopyObjectResponse::copyObjectResult)
                           .thenApply(Object::toString);
    }
```

Utilisez un [S3TransferManager](#) pour [copier un objet](#) d'un compartiment vers un autre. Consultez le [fichier complet](#) et le [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedCopy;
import software.amazon.awssdk.transfer.s3.model.Copy;
import software.amazon.awssdk.transfer.s3.model.CopyRequest;

import java.util.UUID;

    public String copyObject(S3TransferManager transferManager, String
    bucketName,
        String key, String destinationBucket, String destinationKey) {
        CopyObjectRequest copyObjectRequest = CopyObjectRequest.builder()
            .sourceBucket(bucketName)
            .sourceKey(key)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        CopyRequest copyRequest = CopyRequest.builder()
            .copyObjectRequest(copyObjectRequest)
            .build();

        Copy copy = transferManager.copy(copyRequest);

        CompletedCopy completedCopy = copy.completionFuture().join();
        return completedCopy.response().copyObjectResult().eTag();
    }
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Copiez l'objet.

```
import {
  S3Client,
  CopyObjectCommand,
  ObjectNotInActiveTierError,
  waitUntilObjectExists,
} from "@aws-sdk/client-s3";

/**
 * Copy an S3 object from one bucket to another.
 *
 * @param {{
 *   sourceBucket: string,
 *   sourceKey: string,
 *   destinationBucket: string,
 *   destinationKey: string }} config
 */
export const main = async ({
  sourceBucket,
  sourceKey,
  destinationBucket,
  destinationKey,
}) => {
  const client = new S3Client({});

  try {
    await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucket}/${sourceKey}`,
        Bucket: destinationBucket,
        Key: destinationKey,
      }),
    ),
  }
```

```

    );
    await waitUntilObjectExists(
        { client },
        { Bucket: destinationBucket, Key: destinationKey },
    );
    console.log(
        `Successfully copied ${sourceBucket}/${sourceKey} to ${destinationBucket}/${destinationKey}`,
    );
} catch (caught) {
    if (caught instanceof ObjectNotInActiveTierError) {
        console.error(
            `Could not copy ${sourceKey} from ${sourceBucket}. Object is not in the active tier.`,
        );
    } else {
        throw caught;
    }
}
};

```

Copiez l'objet à condition que son ETag ne corresponde pas à celle fournie.

```

import {
    CopyObjectCommand,
    NoSuchKey,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
    type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
    destinationBucketName: string, eTag: string }}
 */

```

```
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;
  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket
"${sourceBucketName}": ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
```

```
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Copiez l'objet à condition que son ETag ne corresponde pas à celle fournie.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";
```

```
// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string, eTag: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
  eTag,
}) => {
  const client = new S3Client({});
  const name = data.name;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: `${sourceBucketName}/${sourceKeyName}`,
        Bucket: destinationBucketName,
        Key: `${name}${sourceKeyName}`,
        CopySourceIfNoneMatch: eTag,
      }),
    );
    console.log("Successfully copied object to bucket.");
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while copying object "${sourceKeyName}" from
        "${sourceBucketName}". No such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Unable to copy object "${sourceKeyName}" to bucket
        "${sourceBucketName}": ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
}
```



```
    }  
  };  
  
  // Call function if run directly  
  import { parseArgs } from "node:util";  
  import {  
    isMain,  
    validateArgs,  
  } from "@aws-doc-sdk-examples/lib/utils/util-node.js";  
  
  const loadArgs = () => {  
    const options = {  
      sourceBucketName: {  
        type: "string",  
        required: true,  
      },  
      sourceKeyName: {  
        type: "string",  
        required: true,  
      },  
      destinationBucketName: {  
        type: "string",  
        required: true,  
      },  
      eTag: {  
        type: "string",  
        required: true,  
      },  
    };  
  };  
  const results = parseArgs({ options });  
  const { errors } = validateArgs({ options }, results);  
  return { errors, results };  
};  
  
if (isMain(import.meta.url)) {  
  const { errors, results } = loadArgs();  
  if (!errors) {  
    main(results.values);  
  } else {  
    console.error(errors.join("\n"));  
  }  
}
```

Copiez l'objet à condition qu'il ait été créé ou modifié dans un laps de temps donné.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);

  const name = data.name;
  const client = new S3Client({});
  const copySource = `${sourceBucketName}/${sourceKeyName}`;
  const copiedKey = name + sourceKeyName;

  try {
    const response = await client.send(
      new CopyObjectCommand({
        CopySource: copySource,
        Bucket: destinationBucketName,
        Key: copiedKey,
        CopySourceIfModifiedSince: date,
      }),
    );
    console.log("Successfully copied object to bucket.");
  }
```

```
    } catch (caught) {
      if (caught instanceof NoSuchKey) {
        console.error(
          `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`
        );
      } else if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while copying object from ${sourceBucketName}.
${caught.name}: ${caught.message}`
        );
      } else {
        throw caught;
      }
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};
```

```
if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Copiez l'objet à condition qu'il n'ait pas été créé ou modifié dans un laps de temps donné.

```
import {
  CopyObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

// Optionally edit the default key name of the copied object in
'object_name.json'
import data from "../scenarios/conditional-requests/object_name.json" assert {
  type: "json",
};

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ sourceBucketName: string, sourceKeyName: string,
  destinationBucketName: string }}
 */
export const main = async ({
  sourceBucketName,
  sourceKeyName,
  destinationBucketName,
}) => {
  const date = new Date();
  date.setDate(date.getDate() - 1);
  const client = new S3Client({});
  const name = data.name;
  const copiedKey = name + sourceKeyName;
  const copySource = `${sourceBucketName}/${sourceKeyName}`;
```

```
try {
  const response = await client.send(
    new CopyObjectCommand({
      CopySource: copySource,
      Bucket: destinationBucketName,
      Key: copiedKey,
      CopySourceIfUnmodifiedSince: date,
    }),
  );
  console.log("Successfully copied object to bucket.");
} catch (caught) {
  if (caught instanceof NoSuchKey) {
    console.error(
      `Error from S3 while copying object "${sourceKeyName}" from
"${sourceBucketName}". No such key exists.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while copying object from ${sourceBucketName}.
${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    sourceBucketName: {
      type: "string",
      required: true,
    },
    sourceKeyName: {
      type: "string",
      required: true,
    },
  },
```

```
    destinationBucketName: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS SDK pour JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
```

```
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- Pour plus de détails sur l'API, consultez [CopyObject](#) la section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Copie simple d'un objet.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $folder = "copied-folder";
    $this->s3client->copyObject([
        'Bucket' => $this->bucketName,
        'CopySource' => "$this->bucketName/$fileName",
        'Key' => "$folder/$fileName-copy",
```

```
    ]);  
    echo "Copied $fileName to $folder/$fileName-copy.\n";  
} catch (Exception $exception) {  
    echo "Failed to copy $fileName with error: " . $exception-  
>getMessage();  
    exit("Please fix error with object copying before continuing.");  
}
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS SDK pour PHP API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande copie l'objet « sample.txt » depuis le compartiment « test-files » vers le même compartiment, mais avec une nouvelle clé « sample-copy.txt ».

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey  
sample-copy.txt
```

Exemple 2 : cette commande copie l'objet « sample.txt » depuis le compartiment « test-files » vers le compartiment « backup-files » avec une clé « sample-copy.txt ».

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -  
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Exemple 3 : cette commande télécharge l'objet « sample.txt » depuis le compartiment « test-files » vers un fichier local nommé « local-sample.txt ».

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-  
sample.txt
```

Exemple 4 : télécharge l'objet unique dans le fichier spécifié. Le fichier téléchargé se trouve dans c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder  
c:\downloads
```


Exemple 5 : télécharge tous les objets qui correspondent au préfixe de clé spécifié dans le dossier local. La hiérarchie des clés relative sera préservée sous forme de sous-dossiers dans l'emplacement de téléchargement global.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:\downloads
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande copie l'objet « sample.txt » depuis le compartiment « test-files » vers le même compartiment, mais avec une nouvelle clé « sample-copy.txt ».

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey sample-copy.txt
```

Exemple 2 : cette commande copie l'objet « sample.txt » depuis le compartiment « test-files » vers le compartiment « backup-files » avec une clé « sample-copy.txt ».

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Exemple 3 : cette commande télécharge l'objet « sample.txt » depuis le compartiment « test-files » vers un fichier local nommé « local-sample.txt ».

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-sample.txt
```

Exemple 4 : télécharge l'objet unique dans le fichier spécifié. Le fichier téléchargé se trouve dans c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder c:\downloads
```

Exemple 5 : télécharge tous les objets qui correspondent au préfixe de clé spécifié dans le dossier local. La hiérarchie des clés relative sera préservée sous forme de sous-dossiers dans l'emplacement de téléchargement global.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:\downloads
```

Exemple 6 : télécharge un seul objet dans un fichier local à l'aide du téléchargement parallèle en plusieurs parties pour améliorer les performances sur les fichiers volumineux.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-sample.txt -UseMultipartDownload
```

Exemple 7 : télécharge un objet volumineux en mode RANGE avec une taille de pièce personnalisée de 64 Mo. Le mode RANGE utilise des requêtes HTTP par plage d'octets et fonctionne pour tous les objets, quelle que soit la manière dont ils ont été téléchargés.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Exemple 8 : télécharge tous les objets correspondant au préfixe de clé dans un dossier local à l'aide du téléchargement en plusieurs parties avec des téléchargements de fichiers simultanés.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Exemple 9 : télécharge un seul objet à l'aide d'un téléchargement en plusieurs parties avec 20 connexions simultanées pour un débit maximal sur les réseaux à large bande passante.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Exemple 10 : télécharge tous les objets correspondant au préfixe de clé dans un dossier local à l'aide du téléchargement en plusieurs parties, en continuant à corriger les défaillances individuelles des fichiers au lieu d'abandonner.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def copy(self, dest_object):
        """
        Copies the object to another bucket.

        :param dest_object: The destination object initialized with a bucket and
        key.
                           This is a Boto3 Object resource.
        """
        try:
            dest_object.copy_from(
                CopySource={"Bucket": self.object.bucket_name, "Key":
                self.object.key}
            )
```

```

        dest_object.wait_until_exists()
        logger.info(
            "Copied object from %s:%s to %s:%s.",
            self.object.bucket_name,
            self.object.key,
            dest_object.bucket_name,
            dest_object.key,
        )
    except ClientError:
        logger.exception(
            "Couldn't copy object from %s/%s to %s/%s.",
            self.object.bucket_name,
            self.object.key,
            dest_object.bucket_name,
            dest_object.key,
        )
        raise

```

Copiez un objet à l'aide d'une demande conditionnelle.

```

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def copy_object_conditional(
        self,
        source_key: str,
        dest_key: str,
        source_bucket: str,
        dest_bucket: str,

```

```

        condition_type: str,
        condition_value: str,
    ):
        """
        Copies an object from one Amazon S3 bucket to another with a conditional
        request.

        :param source_key: The key of the source object to copy.
        :param dest_key: The key of the destination object.
        :param source_bucket: The source bucket of the object.
        :param dest_bucket: The destination bucket of the object.
        :param condition_type: The type of condition to apply, e.g.
            'CopySourceIfMatch', 'CopySourceIfNoneMatch',
            'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            self.s3.copy_object(
                Bucket=dest_bucket,
                Key=dest_key,
                CopySource={"Bucket": source_bucket, "Key": source_key},
                **{condition_type: condition_value},
            )
            print(
                f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."
            )
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional copy failed: Precondition failed")
            elif error_code == "304": # Not modified error code.
                print("\tConditional copy failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise

```

- Pour plus de détails sur l'API, consultez [CopyObject](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Copiez un objet.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for
  #                                     copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
  # target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  # nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
    why: #{e.message}"
  end
end

# Example usage:
```

```
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
  #{target_object.bucket_name}:#{target_object.key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Copier un objet et ajouter le chiffrement côté serveur à l'objet de destination.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
  # target key, and encrypt it.
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  nil.
  def copy_object(target_bucket, target_object_key, encryption)
```

```
@source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
server_side_encryption: encryption)
  target_bucket.object(target_object_key)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
why: #{e.message}"
end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
  target_object = wrapper.copy_object(target_bucket, target_key,
target_encryption)
  return unless target_object

  puts "Copied #{source_key} from #{source_bucket_name} to
#{target_object.bucket_name}:#{target_object.key} and "\
    "encrypted the target with #{target_object.server_side_encryption}
encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS SDK pour Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// Copy an object from one bucket to another.
pub async fn copy_object(
    client: &aws_sdk_s3::Client,
    source_bucket: &str,
    destination_bucket: &str,
    source_object: &str,
    destination_object: &str,
) -> Result<(), S3ExampleError> {
    let source_key = format!("{source_bucket}/{source_object}");
    let response = client
        .copy_object()
        .copy_source(&source_key)
        .bucket(destination_bucket)
        .key(destination_object)
        .send()
        .await?;

    println!(
        "Copied from {source_key} to {destination_bucket}/{destination_object}
with etag {}",
        response
            .copy_object_result
            .unwrap_or_else(||
aws_sdk_s3::types::CopyObjectResult::builder().build())
            .e_tag()
            .unwrap_or("missing")
    );
    Ok(())
}
```

- Pour plus de détails sur l'API, voir [CopyObject](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
  lo_s3->copyobject(  
    iv_bucket = iv_dest_bucket  
    iv_key = iv_dest_object  
    iv_copysource = |{ iv_src_bucket }/{ iv_src_object }| ).  
  MESSAGE 'Object copied to another bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
  MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Swift

Kit SDK pour Swift

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import AWSS3

    public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
        let srcUrl = ("\"(sourceBucket)/
\"(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

        let input = CopyObjectInput(
            bucket: destBucket,
            copySource: srcUrl,
            key: name
        )
        do {
            _ = try await client.copyObject(input: input)
        }
        catch {
            print("ERROR: ", dump(error, name: "Copying an object. "))
            throw error
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section AWS SDK pour la référence de l'API Swift.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **CreateBucket** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser CreateBucket.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Principes de base](#)
- [Commencer à gérer la configuration](#)
- [Commencer à utiliser les magasins de fonctionnalités d'apprentissage automatique](#)

- [Débuter avec le stockage d'objets](#)
- [Gestion de messages volumineux à l'aide de S3](#)
- [Utiliser les objets soumis au contrôle de version](#)

.NET

SDK pour .NET (v4)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Shows how to create a new Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>A boolean value representing the success or failure of
/// the bucket creation process.</returns>
public async Task<bool> CreateBucketAsync(string bucketName)
{
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
        };

        var response = await _amazonS3.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS SDK pour .NET API.

SDK pour .NET

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez un compartiment avec le verrouillage d'objet activé.

```
/// <summary>
/// Create a new Amazon S3 bucket with object lock actions.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
{
    Console.WriteLine($"\\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
            ObjectLockEnabledForBucket = enableObjectLock,
        };

        var response = await _amazonS3.PutBucketAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
    }
}
```

```
        return false;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS SDK pour .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#####
function iecho() {
    if [[ $VERBOSE == true ]]; then
        echo "$@"
    fi
}

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function create-bucket
```

```

#
# This function creates the specified bucket in the specified AWS Region, unless
# it already exists.
#
# Parameters:
#     -b bucket_name  -- The name of the bucket to create.
#     -r region_code  -- The code for an AWS Region in which to
#                        create the bucket.
#
# Returns:
#     The URL of the bucket that was created.
# And:
#     0 - If successful.
#     1 - If it fails.
#####
function create_bucket() {
    local bucket_name region_code response
    local option OPTARG # Required to use getopt command in a function.

    # bashsupport disable=BP5008
    function usage() {
        echo "function create_bucket"
        echo "Creates an Amazon S3 bucket. You must supply a bucket name:"
        echo "  -b bucket_name    The name of the bucket. It must be globally
unique."
        echo "  [-r region_code]    The code for an AWS Region in which the bucket is
created."
        echo ""
    }

    # Retrieve the calling parameters.
    while getopt "b:r:h" option; do
        case "${option}" in
            b) bucket_name="${OPTARG}" ;;
            r) region_code="${OPTARG}" ;;
            h)
                usage
                return 0
                ;;
            \?)
                echo "Invalid parameter"
                usage
                return 1
                ;;
        esac
    done
}

```

```

    esac
done

if [[ -z "$bucket_name" ]]; then
    errecho "ERROR: You must provide a bucket name with the -b parameter."
    usage
    return 1
fi

local bucket_config_arg
# A location constraint for "us-east-1" returns an error.
if [[ -n "$region_code" ]] && [[ "$region_code" != "us-east-1" ]]; then
    bucket_config_arg="--create-bucket-configuration LocationConstraint=
$region_code"
fi

iecho "Parameters:\n"
iecho "    Bucket name:  $bucket_name"
iecho "    Region code:  $region_code"
iecho ""

# If the bucket already exists, we don't want to try to create it.
if (bucket_exists "$bucket_name"); then
    errecho "ERROR: A bucket with that name already exists. Try again."
    return 1
fi

# shellcheck disable=SC2086
response=$(aws s3api create-bucket \
    --bucket "$bucket_name" \
    $bucket_config_arg)

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports create-bucket operation failed.\n$response"
    return 1
fi
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(
            Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
            std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
            " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Exemple 1 : pour créer un compartiment

L'exemple `create-bucket` suivant crée un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region us-east-1
```

Sortie :

```
{  
  "Location": "/amzn-s3-demo-bucket"  
}
```

Pour plus d'informations, consultez [Création d'un compartiment](#) dans le Guide de l'utilisateur Amazon S3.

Exemple 2 : pour créer un compartiment avec un propriétaire appliqué

L'exemple `create-bucket` suivant crée un compartiment nommé `amzn-s3-demo-bucket` qui utilise le paramètre de propriétaire du compartiment appliqué pour la propriété de l'objet S3.

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region us-east-1 \  
  --object-ownership BucketOwnerEnforced
```

Sortie :

```
{  
  "Location": "/amzn-s3-demo-bucket"  
}
```

Pour plus d'informations, veuillez consulter la rubrique [Contrôle de la propriété des objets et désactivation des listes ACL](#) dans le Guide de l'utilisateur Amazon S3.

Exemple 3 : pour créer un compartiment en dehors de la région us-east-1

L'exemple `create-bucket` suivant crée un compartiment nommé `amzn-s3-demo-bucket` dans la région `eu-west-1`. Les régions autres que `us-east-1` exigent la bonne `LocationConstraint` pour créer le compartiment dans la région souhaitée.

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region eu-west-1 \  
  --create-bucket-configuration LocationConstraint=eu-west-1
```

Sortie :

```
{  
  "Location": "http://amzn-s3-demo-bucket.s3.amazonaws.com/"  
}
```

Pour plus d'informations, consultez [Création d'un compartiment](#) dans le Guide de l'utilisateur Amazon S3.

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez un compartiment avec la configuration par défaut.

```
import (  
  "bytes"
```

```
"context"
"errors"
"fmt"
"io"
"log"
"os"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
    string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    })
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", name)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", name)
        }
    }
}
```

```
    err = exists
}
} else {
    err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
    }
}
return err
}
```

Créez un compartiment avec verrouillage d'objets et attendez qu'il existe.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking
// enabled
// and waits for the bucket to exist before returning.
```

```
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    }

    if enableObjectLock {
        input.ObjectLockEnabledForBucket = aws.Bool(true)
    }

    _, err := actor.S3Client.CreateBucket(ctx, input)
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", bucket)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", bucket)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
        }
    }

    return bucket, err
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez un compartiment.

```
/**
 * Creates an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes when the bucket is
 *         created and ready
 * @throws RuntimeException if there is a failure while creating the bucket
 */
public CompletableFuture<Void> createBucketAsync(String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    CompletableFuture<CreateBucketResponse> response =
        getAsyncClient().createBucket(bucketRequest);
    return response.thenCompose(resp -> {
        S3AsyncWaiter s3Waiter = getAsyncClient().waiter();
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        CompletableFuture<WaiterResponse<HeadBucketResponse>>
        waiterResponseFuture =
            s3Waiter.waitUntilBucketExists(bucketRequestWait);
        return waiterResponseFuture.thenAccept(waiterResponse -> {
            waiterResponse.matched().response().ifPresent(headBucketResponse
        -> {
            logger.info(bucketName + " is ready");
        });
    }));
}
```

```
    }).whenComplete((resp, ex) -> {  
        if (ex != null) {  
            throw new RuntimeException("Failed to create bucket", ex);  
        }  
    });  
}
```

Créez un compartiment avec le verrouillage d'objet activé.

```
// Create a new Amazon S3 bucket with object lock options.  
public void createBucketWithLockOptions(boolean enableObjectLock, String  
bucketName) {  
    S3Waiter s3Waiter = getClient().waiter();  
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()  
        .bucket(bucketName)  
        .objectLockEnabledForBucket(enableObjectLock)  
        .build();  
  
    getClient().createBucket(bucketRequest);  
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()  
        .bucket(bucketName)  
        .build();  
  
    // Wait until the bucket is created and print out the response.  
    s3Waiter.waitUntilBucketExists(bucketRequestWait);  
    System.out.println(bucketName + " is ready");  
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créer le compartiment.

```
import {
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  CreateBucketCommand,
  S3Client,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * Create an Amazon S3 bucket.
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Location } = await client.send(
      new CreateBucketCommand({
        // The name of the bucket. Bucket names are unique and have several other
        // constraints.
        // See https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucketnamingrules.html
        Bucket: bucketName,
      }),
    );
    await waitUntilBucketExists({ client }, { Bucket: bucketName });
    console.log(`Bucket created with location ${Location}`);
  } catch (caught) {
    if (caught instanceof BucketAlreadyExists) {
      console.error(
        `The bucket "${bucketName}" already exists in another AWS account. Bucket names must be globally unique.`,
      );
    }
    // WARNING: If you try to create a bucket in the North Virginia region,
    // and you already own a bucket in that region with the same name, this
    // error will not be thrown. Instead, the call will return successfully
    // and the ACL on that bucket will be reset.
    else if (caught instanceof BucketAlreadyOwnedByYou) {
      console.error(
        `The bucket "${bucketName}" already exists in this AWS account.`,
      );
    }
  }
}
```

```
    } else {  
        throw caught;  
    }  
}  
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS SDK pour JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun createNewBucket(bucketName: String) {  
    val request =  
        CreateBucketRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.createBucket(request)  
        println("$bucketName is ready")  
    }  
}
```

- Pour plus de détails sur l'API, consultez [CreateBucket](#) la section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez un compartiment.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $this->s3client->createBucket([
        'Bucket' => $this->bucketName,
        'CreateBucketConfiguration' => ['LocationConstraint' => $region],
    ]);
    echo "Created bucket named: $this->bucketName \n";
} catch (Exception $exception) {
    echo "Failed to create bucket $this->bucketName with error: " .
    $exception->getMessage();
    exit("Please fix error with bucket creation before continuing.");
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS SDK pour PHP API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créer un compartiment avec les paramètres par défaut.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def create(self, region_override=None):
        """
        Create an Amazon S3 bucket in the default Region for the account or in
        the
        specified Region.

        :param region_override: The Region in which to create the bucket. If this
        is
                                not specified, the Region configured in your
        shared
                                credentials is used.
        """
        if region_override is not None:
            region = region_override
        else:
            region = self.bucket.meta.client.meta.region_name
        try:
            self.bucket.create(CreateBucketConfiguration={"LocationConstraint":
            region})

            self.bucket.wait_until_exists()
            logger.info("Created bucket '%s' in region=%s", self.bucket.name,
            region)
        except ClientError as error:
            logger.exception(
                "Couldn't create bucket named '%s' in region=%s.",
                self.bucket.name,
                region,
            )
            raise error

```

Créer un compartiment soumis à la gestion des versions avec une configuration de cycle de vie.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
    lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                    configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
```

```

except ClientError:
    logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
    raise

try:
    expiration = 7
    bucket.LifecycleConfiguration().put(
        LifecycleConfiguration={
            "Rules": [
                {
                    "Status": "Enabled",
                    "Prefix": prefix,
                    "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                }
            ]
        }
    )
    logger.info(
        "Configured lifecycle to expire noncurrent versions after %s days "
        "on bucket %s.",
        expiration,
        bucket.name,
    )
except ClientError as error:
    logger.warning(
        "Couldn't configure lifecycle on bucket %s because %s. "
        "Continuing anyway.",
        bucket.name,
        error,
    )

return bucket

```

- Pour plus de détails sur l'API, consultez [CreateBucket](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
  # This is a client-side object until
  #                               create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
    else

```

```

        @bucket.client.get_bucket_location(bucket:
@bucket.name).location_constraint
    end
    rescue Aws::Errors::ServiceError => e
        "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
    end
end

# Example usage:
def run_demo
    region = "us-west-2"
    wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{@Random.uuid}"))
    return unless wrapper.create?(region)

    puts "Created bucket #{@wrapper.bucket.name}."
    puts "Your bucket's region is: #{@wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS SDK pour Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

pub async fn create_bucket(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    region: &aws_config::Region,
) -> Result<Option<aws_sdk_s3::operation::create_bucket::CreateBucketOutput>,
    S3ExampleError> {

```



```

    let constraint =
aws_sdk_s3::types::BucketLocationConstraint::from(region.to_string().as_str());
    let cfg = aws_sdk_s3::types::CreateBucketConfiguration::builder()
        .location_constraint(constraint)
        .build();
    let create = client
        .create_bucket()
        .create_bucket_configuration(cfg)
        .bucket(bucket_name)
        .send()
        .await;

    // BucketAlreadyExists and BucketAlreadyOwnedByYou are not problems for this
    task.
    create.map(Some).or_else(|err| {
        if err
            .as_service_error()
            .map(|se| se.is_bucket_already_exists() ||
se.is_bucket_already_owned_by_you())
            == Some(true)
        {
            Ok(None)
        } else {
            Err(S3ExampleError::from(err))
        }
    })
}

```

- Pour plus de détails sur l'API, voir [CreateBucket](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    " determine our region from our session
    DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
    DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
    " When in the us-east-1 region, you must not specify a constraint
    " In all other regions, specify the region as the constraint
    IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
    ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
    ENDIF.

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
    CATCH /aws1/cx_s3_bucketalrddyexists.
        MESSAGE 'Bucket name already exists.' TYPE 'E'.
    CATCH /aws1/cx_s3_bktalrddyownedbyyou.
        MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
    ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Swift

Kit SDK pour Swift

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

import AWSS3

public func createBucket(name: String) async throws {
    var input = CreateBucketInput(

```

```

        bucket: name
    )

    // For regions other than "us-east-1", you must set the
    locationConstraint in the createBucketConfiguration.
    // For more information, see LocationConstraint in the S3 API guide.
    // https://docs.aws.amazon.com/AmazonS3/latest/API/
    API_CreateBucket.html#API_CreateBucket_RequestBody
    if let region = configuration.region {
        if region != "us-east-1" {
            input.createBucketConfiguration =
S3ClientTypes.CreateBucketConfiguration(locationConstraint:
S3ClientTypes.BucketLocationConstraint(rawValue: region))
        }
    }

    do {
        _ = try await client.createBucket(input: input)
    }
    catch let error as BucketAlreadyOwnedByYou {
        print("The bucket '\(name)' already exists and is owned by you. You
may wish to ignore this exception.")
        throw error
    }
    catch {
        print("ERROR: ", dump(error, name: "Creating a bucket"))
        throw error
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section AWS SDK pour la référence de l'API Swift.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **CreateMultiRegionAccessPoint** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser **CreateMultiRegionAccessPoint**.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Configurez le client de contrôle S3 pour envoyer une demande à la région us-west-2.

```
suspend fun createS3ControlClient(): S3ControlClient {  
    // Configure your S3ControlClient to send requests to US West  
    (Oregon).  
    val s3Control = S3ControlClient.fromEnvironment {  
        region = "us-west-2"  
    }  
    return s3Control  
}
```

Créez le point Multi-Region d'accès.

```
suspend fun createMrap(  
    s3Control: S3ControlClient,  
    accountIdParam: String,  
    bucketName1: String,  
    bucketName2: String,  
    mrapName: String,  
): String {  
    println("Creating MRAP ...")  
    val createMrapResponse: CreateMultiRegionAccessPointResponse =  
        s3Control.createMultiRegionAccessPoint {  
            accountId = accountIdParam  
            clientToken = UUID.randomUUID().toString()  
            details {  
                name = mrapName  
                regions = listOf(  
                    Region {  
                        bucket = bucketName1  
                    },  
                ),  
            }  
        }  
    return createMrapResponse.accessPointArn  
}
```

```

        Region {
            bucket = bucketName2
        },
    )
}
}

val requestToken: String? = createMrapResponse.requestTokenArn

// Use the request token to check for the status of the
CreateMultiRegionAccessPoint operation.
if (requestToken != null) {
    waitForSucceededStatus(s3Control, requestToken, accountIdParam)
    println("MRAP created")
}

val getMrapResponse =
    s3Control.getMultiRegionAccessPoint(
        input = GetMultiRegionAccessPointRequest {
            accountId = accountIdParam
            name = mrapName
        },
    )
val mrapAlias = getMrapResponse.accessPoint?.alias
return "arn:aws:s3:::$accountIdParam:accesspoint/$mrapAlias"
}

```

Attendez que le point Multi-Region d'accès soit disponible.

```

suspend fun waitForSucceededStatus(
    s3Control: S3ControlClient,
    requestToken: String,
    accountIdParam: String,
    timeBetweenChecks: Duration = 1.minutes,
) {
    var describeResponse: DescribeMultiRegionAccessPointOperationResponse
    describeResponse = s3Control.describeMultiRegionAccessPointOperation(
        input = DescribeMultiRegionAccessPointOperationRequest {
            accountId = accountIdParam
            requestTokenArn = requestToken
        },
    )
}

```

```
        var status: String? = describeResponse.asyncOperation?.requestStatus
        while (status != "SUCCEEDED") {
            delay(timeBetweenChecks)
            describeResponse =
s3Control.describeMultiRegionAccessPointOperation(
                input = DescribeMultiRegionAccessPointOperationRequest {
                    accountId = accountIdParam
                    requestTokenArn = requestToken
                },
            )
            status = describeResponse.asyncOperation?.requestStatus
            println(status)
        }
    }
```

- Pour en savoir plus, consultez [Guide du développeur d'AWS SDK pour Kotlin](#).
- Pour plus de détails sur l'API, consultez [CreateMultiRegionAccessPoint](#) la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **CreateMultipartUpload** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser CreateMultipartUpload.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action dans son contexte dans les exemples de code suivants :

- [Effectuer une copie en plusieurs parties](#)
- [Utiliser les totaux de contrôle](#)
- [Utilisation de l'intégrité des objets Amazon S3](#)

C++

SDK pour C++

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
    &key,
                                Aws::S3::Model::ChecksumAlgorithm
checksumAlgorithm,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
        client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
}
```

```
    return uploadID;
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateMultipartUpload](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante crée un chargement partitionné dans le compartiment `amzn-s3-demo-bucket` avec la clé `multipart/01` :

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket --key  
'multipart/01'
```

Sortie :

```
{
  "Bucket": "amzn-s3-demo-bucket",
  "UploadId":
  "dfRtDYU0WWCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3
  "Key": "multipart/01"
}
```

Le fichier finalisé est nommé `01` dans un dossier nommé `multipart` dans le compartiment `amzn-s3-demo-bucket`. Enregistrez l'ID de chargement, la clé et le nom du compartiment à utiliser avec la commande `upload-part`.

- Pour plus de détails sur l'API, reportez-vous [CreateMultipartUpload](#) à la section Référence des AWS CLI commandes.

Rust

SDK pour Rust

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
```

```

        let upload_part_res = client
            .upload_part()
            .key(&key)
            .bucket(&bucket_name)
            .upload_id(upload_id)
            .body(stream)
            .part_number(part_number)
            .send()
            .await?;

        upload_parts.push(
            CompletedPart::builder()
                .e_tag(upload_part_res.e_tag.unwrap_or_default())
                .part_number(part_number)
                .build(),
        );
    }

```

```

// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
    CompletedMultipartUpload::builder()
        .set_parts(Some(upload_parts))
        .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;

```

- Pour plus de détails sur l'API, voir [CreateMultipartUpload](#) la section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **CreatePresignedPost** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `CreatePresignedPost`.

.NET

SDK pour .NET (v4)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez une URL POST présignée.

```
/// <summary>
/// Create a presigned POST URL with conditions.
/// </summary>
/// <param name="s3Client">The Amazon S3 client.</param>
/// <param name="bucketName">The name of the bucket.</param>
/// <param name="objectKey">The object key (path) where the uploaded file
will be stored.</param>
/// <param name="expires">When the presigned URL expires.</param>
/// <param name="fields">Dictionary of fields to add to the form.</param>
/// <param name="conditions">List of conditions to apply.</param>
/// <returns>A CreatePresignedPostResponse object with URL and form fields.</
returns>
public async Task<CreatePresignedPostResponse> CreatePresignedPostAsync(
    IAmazonS3 s3Client,
    string bucketName,
    string objectKey,
    DateTime expires,
    Dictionary<string, string>? fields = null,
    List<S3PostCondition>? conditions = null)
{
    var request = new CreatePresignedPostRequest
    {
        BucketName = bucketName,
        Key = objectKey,
        Expires = expires
    };
};
```

```
// Add custom fields if provided
if (fields != null)
{
    foreach (var field in fields)
    {
        request.Fields.Add(field.Key, field.Value);
    }
}

// Add conditions if provided
if (conditions != null)
{
    foreach (var condition in conditions)
    {
        request.Conditions.Add(condition);
    }
}

return await s3Client.CreatePresignedPostAsync(request);
}
```

- Pour plus de détails sur l'API, reportez-vous [CreatePresignedPost](#) à la section Référence des AWS SDK pour .NET API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteBucket** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser DeleteBucket.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Principes de base](#)
- [Commencer à gérer la configuration](#)
- [Commencer à utiliser les magasins de fonctionnalités d'apprentissage automatique](#)
- [Débuter avec le stockage d'objets](#)

.NET

SDK pour .NET (v4)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Shows how to delete an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the Amazon S3 bucket to delete.</
param>
/// <returns>A boolean value that represents the success or failure of
/// the delete operation.</returns>
public async Task<bool> DeleteBucketAsync(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest { BucketName = bucketName, };

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting bucket: {ex.Message}");
        return false;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des AWS SDK pour .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api delete-bucket \
        --bucket "$bucket_name")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-bucket failed.\n$response"
        return 1
    fi
}
```

```
    fi  
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,  
                               const Aws::S3::S3ClientConfiguration &clientConfig)  
{  
  
    Aws::S3::S3Client client(clientConfig);  
  
    Aws::S3::Model::DeleteBucketRequest request;  
    request.SetBucket(bucketName);  
  
    Aws::S3::Model::DeleteBucketOutcome outcome =  
        client.DeleteBucket(request);  
  
    if (!outcome.IsSuccess()) {  
        const Aws::S3::S3Error &err = outcome.GetError();  
        std::cerr << "Error: deleteBucket: " <<  
                    err.GetExceptionName() << ": " << err.GetMessage() <<  
std::endl;  
    } else {  
        std::cout << "The bucket was deleted" << std::endl;  
    }  
  
    return outcome.IsSuccess();  
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante supprime un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket --region us-east-1
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"
```



```
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is
// returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
    return err
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Deletes an S3 bucket asynchronously.
 *
 * @param bucket the name of the bucket to be deleted
 * @return a {@link CompletableFuture} that completes when the bucket
 * deletion is successful, or throws a {@link RuntimeException}
 * if an error occurs during the deletion process
 */
public CompletableFuture<Void> deleteBucketAsync(String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(bucket + " was deleted.");
        } else {
            throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
        }
    });
    return response.thenApply(r -> null);
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez le compartiment.

```
import {
  DeleteBucketCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Delete an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new DeleteBucketCommand({
    Bucket: bucketName,
  });

  try {
    await client.send(command);
    console.log("Bucket was deleted.");
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting bucket. The bucket doesn't exist.`
      );
    }
  }
}
```

```
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while deleting the bucket. ${caught.name}:
            ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des AWS SDK pour JavaScript API.

PHP

Kit SDK pour PHP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez un compartiment vide.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $this->s3client->deleteBucket([
        'Bucket' => $this->bucketName,
    ]);
    echo "Deleted bucket $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $this->bucketName with error: " . $exception-
    >getMessage();
    exit("Please fix error with bucket deletion before continuing.");
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des AWS SDK pour PHP API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande supprime tous les objets et toutes les versions d'objets du compartiment « test-files », puis supprime le compartiment. La commande vous invite à confirmer avant de poursuivre. Ajoutez le commutateur -Force pour supprimer la confirmation. Notez que les compartiments qui ne sont pas vides ne peuvent pas être supprimés.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande supprime tous les objets et toutes les versions d'objets du compartiment « test-files », puis supprime le compartiment. La commande vous invite à confirmer avant de poursuivre. Ajoutez le commutateur -Force pour supprimer la confirmation. Notez que les compartiments qui ne sont pas vides ne peuvent pas être supprimés.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete(self):
        """
        Delete the bucket. The bucket must be empty or an error is raised.
        """
        try:
            self.bucket.delete()
            self.bucket.wait_until_not_exists()
            logger.info("Bucket %s successfully deleted.", self.bucket.name)
        except ClientError:
            logger.exception("Couldn't delete bucket %s.", self.bucket.name)
            raise
```

- Pour plus de détails sur l'API, consultez [DeleteBucket](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
```

```
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
end
answer = gets.chomp.downcase
if answer == 'y'
  bucket.objects.batch_delete!
  bucket.delete
  puts("Emptied and deleted bucket #{bucket.name}.\n")
end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des AWS SDK pour Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
pub async fn delete_bucket(
  client: &aws_sdk_s3::Client,
  bucket_name: &str,
) -> Result<(), S3ExampleError> {
  let resp = client.delete_bucket().bucket(bucket_name).send().await;
  match resp {
    Ok(_) => Ok(()),
    Err(err) => {
      if err
        .as_service_error()
```

```

        .and_then(aws_sdk_s3::error::ProvideErrorMetadata::code)
        == Some("NoSuchBucket")
    {
        Ok(())
    } else {
        Err(S3ExampleError::from(err))
    }
}
}
}
}

```

- Pour plus de détails sur l'API, voir [DeleteBucket](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

TRY.

```

lo_s3->deletebucket(
    iv_bucket = iv_bucket_name ).
MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Swift

Kit SDK pour Swift

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import AWSS3

public func deleteBucket(name: String) async throws {
    let input = DeleteBucketInput(
        bucket: name
    )
    do {
        _ = try await client.deleteBucket(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a bucket"))
        throw error
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section AWS SDK pour la référence de l'API Swift.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **DeleteBucketAnalyticsConfiguration** avec une CLI

Les exemples de code suivants illustrent comment utiliser `DeleteBucketAnalyticsConfiguration`.

CLI

AWS CLI

Pour supprimer une configuration d'analytique d'un compartiment

L'exemple `delete-bucket-analytics-configuration` suivant supprime la configuration d'analyse pour le compartiment et l'ID spécifiés.

```
aws s3api delete-bucket-analytics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 1
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketAnalyticsConfiguration](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : ma commande supprime le filtre d'analyse nommé « testfilter » dans le compartiment S3 donné.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketAnalyticsConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : ma commande supprime le filtre d'analyse nommé « testfilter » dans le compartiment S3 donné.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AnalyticsId 'testfilter'
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketAnalyticsConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteBucketCors** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser DeleteBucketCors.

.NET

SDK pour .NET

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Deletes a CORS configuration from an Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used
/// to delete the CORS configuration from the bucket.</param>
private static async Task DeleteCORSConfigurationAsync(AmazonS3Client
client)
{
    DeleteCORSConfigurationRequest request = new
DeleteCORSConfigurationRequest()
    {
        BucketName = BucketName,
    };
    await client.DeleteCORSConfigurationAsync(request);
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketCors](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

La commande suivante supprime une configuration de partage de Cross-Origin ressources d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api delete-bucket-cors --bucket amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketCors](#) à la section Référence des AWS CLI commandes.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_cors(self):
        """
        Delete the CORS rules from the bucket.

        :param bucket_name: The name of the bucket to update.
        """
```

```
try:
    self.bucket.Cors().delete()
    logger.info("Deleted CORS from bucket '%s'.", self.bucket.name)
except ClientError:
    logger.exception("Couldn't delete CORS from bucket '%s'.",
self.bucket.name)
    raise
```

- Pour plus de détails sur l'API, consultez [DeleteBucketCors](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
  def delete_cors
    @bucket_cors.delete
    true
  end
end
```

```
rescue Aws::Errors::ServiceError => e
  puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
#{e.message}"
  false
end

end
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketCors](#) à la section Référence des AWS SDK pour Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.
  lo_s3->deletebucketcors(
    iv_bucket = iv_bucket_name ).
  MESSAGE 'Bucket CORS configuration deleted.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketCors](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **DeleteBucketEncryption** avec une CLI

Les exemples de code suivants illustrent comment utiliser DeleteBucketEncryption.

CLI

AWS CLI

Pour supprimer la configuration du chiffrement côté serveur d'un compartiment

L'exemple delete-bucket-encryption suivant supprime la configuration du chiffrement côté serveur du compartiment spécifié.

```
aws s3api delete-bucket-encryption \  
  --bucket amzn-s3-demo-bucket
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketEncryption](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cela désactive le chiffrement activé pour le compartiment S3 fourni.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on  
target "s3casetestbucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketEncryption](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cela désactive le chiffrement activé pour le compartiment S3 fourni.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketEncryption (DeleteBucketEncryption)" on
target "s3casetestbucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketEncryption](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **DeleteBucketInventoryConfiguration** avec une CLI

Les exemples de code suivants illustrent comment utiliser DeleteBucketInventoryConfiguration.

CLI

AWS CLI

Pour supprimer la configuration d'inventaire d'un compartiment

L'exemple delete-bucket-inventory-configuration suivant supprime la configuration d'inventaire possédant l'ID 1 du compartiment spécifié.

```
aws s3api delete-bucket-inventory-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 1
```


Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketInventoryConfiguration](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : Cette commande supprime l'inventaire nommé « test InventoryName » correspondant au compartiment S3 donné.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -  
InventoryId 'testInventoryName'
```

Sortie :

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketInventoryConfiguration  
(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketInventoryConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : Cette commande supprime l'inventaire nommé « test InventoryName » correspondant au compartiment S3 donné.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -  
InventoryId 'testInventoryName'
```

Sortie :

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3BucketInventoryConfiguration  
(DeleteBucketInventoryConfiguration)" on target "amzn-s3-demo-bucket".
```

```
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketInventoryConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteBucketLifecycle** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser DeleteBucketLifecycle.

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// This method removes the Lifecycle configuration from the named
/// S3 bucket.
/// </summary>
/// <param name="client">The S3 client object used to call
/// the RemoveLifecycleConfigAsync method.</param>
/// <param name="bucketName">A string representing the name of the
/// S3 bucket from which the configuration will be removed.</param>
public static async Task RemoveLifecycleConfigAsync(IAmazonS3 client,
string bucketName)
{
    var request = new DeleteLifecycleConfigurationRequest()
    {
        BucketName = bucketName,
    };
    await client.DeleteLifecycleConfigurationAsync(request);
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketLifecycle](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

La commande suivante supprime une configuration de cycle de vie d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api delete-bucket-lifecycle --bucket amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketLifecycle](#) à la section Référence des AWS CLI commandes.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
def delete_lifecycle_configuration(self):
    """
    Remove the lifecycle configuration from the specified bucket.
    """
    try:
        self.bucket.LifecycleConfiguration().delete()
        logger.info(
            "Deleted lifecycle configuration for bucket '%s'.",
            self.bucket.name
        )
    except ClientError:
        logger.exception(
            "Couldn't delete lifecycle configuration for bucket '%s'.",
            self.bucket.name,
        )
        raise
```

- Pour plus de détails sur l'API, consultez [DeleteBucketLifecycle](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.
    lo_s3->deletebucketlifecycle(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket lifecycle configuration deleted.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketLifecycle](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **DeleteBucketMetricsConfiguration** avec une CLI

Les exemples de code suivants illustrent comment utiliser DeleteBucketMetricsConfiguration.

CLI

AWS CLI

Pour supprimer une configuration des métriques d'un compartiment

L'exemple delete-bucket-metrics-configuration suivant supprime la configuration des métriques pour le compartiment et l'ID spécifiés.

```
aws s3api delete-bucket-metrics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 123
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketMetricsConfiguration](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : la commande supprime le filtre de métriques nommé « testmetrics » dans le compartiment S3 donné.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketMetricsConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : la commande supprime le filtre de métriques nommé « testmetrics » dans le compartiment S3 donné.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketMetricsConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteBucketPolicy** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser DeleteBucketPolicy.

C++

Kit de développement logiciel (SDK) for C++

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,  
                                     const Aws::S3::S3ClientConfiguration  
&clientConfig) {  
    Aws::S3::S3Client client(clientConfig);  
  
    Aws::S3::Model::DeleteBucketPolicyRequest request;  
    request.SetBucket(bucketName);
```

```
Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
client.DeleteBucketPolicy(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: deleteBucketPolicy: " <<
        err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
} else {
    std::cout << "Policy was deleted from the bucket." << std::endl;
}

return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketPolicy](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante supprime la stratégie d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api delete-bucket-policy --bucket amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketPolicy](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteBucketPolicy {
    public static void main(String[] args) {

        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the policy from
(for example, bucket1)."";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting policy from bucket: \"%s\"\n\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteS3BucketPolicy(s3, bucketName);
        s3.close();
    }
}
```



```
/**
 * Deletes the S3 bucket policy for the specified bucket.
 *
 * @param s3 the {@link S3Client} instance to use for the operation
 * @param bucketName the name of the S3 bucket for which the policy should be
deleted
 *
 * @throws S3Exception if there is an error deleting the bucket policy
 */
public static void deleteS3BucketPolicy(S3Client s3, String bucketName) {
    DeleteBucketPolicyRequest delReq = DeleteBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        s3.deleteBucketPolicy(delReq);
        System.out.println("Done!");
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketPolicy](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez la politique du compartiment.

```
import {
```

```
DeleteBucketPolicyCommand,
S3Client,
S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the policy from an Amazon S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Bucket policy deleted from "${bucketName}".`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}. The bucket
        doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting policy from ${bucketName}. ${caught.name}:
        ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [DeleteBucketPolicy](#) à la section Référence des AWS SDK pour JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {  
    val request =  
        DeleteBucketPolicyRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteBucketPolicy(request)  
        println("Done!")  
    }  
}
```

- Pour plus de détails sur l'API, consultez [DeleteBucketPolicy](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell V4

Exemple 1 : la commande supprime la stratégie de compartiment associée au compartiment S3 donné.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketPolicy](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : la commande supprime la stratégie de compartiment associée au compartiment S3 donné.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketPolicy](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def delete_policy(self):
        """
        Delete the security policy from the bucket.
        """
        try:
            self.bucket.Policy().delete()
            logger.info("Deleted policy for bucket '%s'.", self.bucket.name)
        except ClientError:
```

```
logger.exception(  
    "Couldn't delete policy for bucket '%s'." % self.bucket.name  
)  
raise
```

- Pour plus de détails sur l'API, consultez [DeleteBucketPolicy](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
# Wraps an Amazon S3 bucket policy.  
class BucketPolicyWrapper  
  attr_reader :bucket_policy  
  
  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object  
  # configured with an existing bucket.  
  def initialize(bucket_policy)  
    @bucket_policy = bucket_policy  
  end  
  
  def delete_policy  
    @bucket_policy.delete  
    true  
  rescue Aws::Errors::ServiceError => e  
    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's  
    why: #{e.message}"  
    false  
  end  
  
end
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketPolicy](#) à la section Référence des AWS SDK pour Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
    lo_s3->deletebucketpolicy(  
        iv_bucket = iv_bucket_name ).  
    MESSAGE 'Bucket policy deleted.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketPolicy](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **DeleteBucketReplication** avec une CLI

Les exemples de code suivants illustrent comment utiliser DeleteBucketReplication.

CLI

AWS CLI

La commande suivante supprime une configuration de réplication d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api delete-bucket-replication --bucket amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketReplication](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : supprime la configuration de réplication associée au compartiment nommé « amzn-s3-demo-bucket ». Notez que cette opération nécessite une autorisation pour l'DeleteReplicationConfiguration action s3 :. Vous devrez confirmer avant que l'opération ne se poursuive. Pour supprimer la confirmation, utilisez le commutateur -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketReplication](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : supprime la configuration de réplication associée au compartiment nommé « amzn-s3-demo-bucket ». Notez que cette opération nécessite une autorisation pour l'DeleteReplicationConfiguration action s3 :. Vous devrez confirmer avant que l'opération ne se poursuive. Pour supprimer la confirmation, utilisez le commutateur -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketReplication](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **DeleteBucketTagging** avec une CLI

Les exemples de code suivants illustrent comment utiliser DeleteBucketTagging.

CLI

AWS CLI

La commande suivante supprime une configuration de balisage d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api delete-bucket-tagging --bucket amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketTagging](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande supprime toutes les balises associées au compartiment S3 donné.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketTagging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande supprime toutes les balises associées au compartiment S3 donné.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

Confirm

Are you sure you want to perform this action?

Performing the operation "Remove-S3BucketTagging (DeleteBucketTagging)" on target "amzn-s3-demo-bucket".

[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "Y"): Y

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketTagging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteBucketWebsite** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser DeleteBucketWebsite.

C++

Kit de développement logiciel (SDK) for C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
                                     &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
        client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
```

```
err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
} else {
    std::cout << "Website configuration was removed." << std::endl;
}

return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketWebsite](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante supprime la configuration de site Web d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api delete-bucket-website --bucket amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketWebsite](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:      <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to delete the website
configuration from.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Deleting website configuration for Amazon S3 bucket:
%s\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        deleteBucketWebsiteConfig(s3, bucketName);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Deletes the website configuration for an Amazon S3 bucket.
     *
     * @param s3 The {@link S3Client} instance used to interact with Amazon S3.
     */
}
```

```
    * @param bucketName The name of the S3 bucket for which the website
    configuration should be deleted.
    * @throws S3Exception If an error occurs while deleting the website
    configuration.
    */
    public static void deleteBucketWebsiteConfig(S3Client s3, String bucketName)
    {
        DeleteBucketWebsiteRequest delReq = DeleteBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .build();

        try {
            s3.deleteBucketWebsite(delReq);
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.out.println("Failed to delete website configuration!");
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketWebsite](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez la configuration du site Web du compartiment.

```
import {
    DeleteBucketWebsiteCommand,
    S3Client,
```

```
S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Remove the website configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    // The response code will be successful for both removed configurations and
    // configurations that did not exist in the first place.
    console.log(
      `The bucket "${bucketName}" is not longer configured as a website, or it
never was.`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}.
The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while removing website configuration from ${bucketName}.
${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketWebsite](#) à la section Référence des AWS SDK pour JavaScript API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande désactive la propriété d'hébergement statique du site Web du compartiment S3 donné.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketWebsite](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande désactive la propriété d'hébergement statique du site Web du compartiment S3 donné.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Confirm
Are you sure you want to perform this action?
Performing the operation "Remove-S3BucketWebsite (DeleteBucketWebsite)" on target
"amzn-s3-demo-bucket".
[Y] Yes  [A] Yes to All  [N] No  [L] No to All  [S] Suspend  [?] Help (default is
"Y"): Y
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketWebsite](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteObject** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser DeleteObject.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Utilisation de l'intégrité des objets Amazon S3](#)
- [Utiliser les objets soumis au contrôle de version](#)

.NET

SDK pour .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez un objet dans un compartiment S3 non soumis à la gestion des versions.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete an object from a non-versioned Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteObject
```

```
{
    /// <summary>
    /// The Main method initializes the necessary variables and then calls
    /// the DeleteObjectNonVersionedBucketAsync method to delete the object
    /// named by the keyName parameter.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string keyName = "testfile.txt";

        // If the Amazon S3 bucket is located in an AWS Region other than the
        // Region of the default account, define the AWS Region for the
        // Amazon S3 bucket in your call to the AmazonS3Client constructor.
        // For example RegionEndpoint.USSouth1.
        IAmazonS3 client = new AmazonS3Client();
        await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
    }

    /// <summary>
    /// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
    /// desired object from the named bucket.
    /// </summary>
    /// <param name="client">An initialized Amazon S3 client used to delete
    /// an object from an Amazon S3 bucket.</param>
    /// <param name="bucketName">The name of the bucket from which the
    /// object will be deleted.</param>
    /// <param name="keyName">The name of the object to delete.</param>
    public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
    {
        try
        {
            var deleteObjectRequest = new DeleteObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
            };

            Console.WriteLine($"Deleting object: {keyName}");
            await client.DeleteObjectAsync(deleteObjectRequest);
            Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
        }
    }
}
```



```
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' when deleting an object.");
    }
}
}
```

Supprimez un objet dans un compartiment S3 soumis à la gestion des versions.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example creates an object in an Amazon Simple Storage Service
/// (Amazon S3) bucket and then deletes the object version that was
/// created.
/// </summary>
public class DeleteObjectVersion
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "versioned-object.txt";

        // If the AWS Region of the default user is different from the AWS
        // Region of the Amazon S3 bucket, pass the AWS Region of the
        // bucket region to the Amazon S3 client object's constructor.
        // Define it like this:
        //      RegionEndpoint bucketRegion = RegionEndpoint.USWest2;
        IAmazonS3 client = new AmazonS3Client();

        await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);
    }

    /// <summary>
    /// This method creates and then deletes a versioned object.
    /// </summary>
```

```

    /// <param name="client">The initialized Amazon S3 client object used to
    /// create and delete the object.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object will be created and deleted.</param>
    /// <param name="keyName">The key name of the object to create.</param>
    public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
client, string bucketName, string keyName)
    {
        try
        {
            // Add a sample object.
            string versionID = await PutAnObject(client, bucketName,
keyName);

            // Delete the object by specifying an object key and a version
ID.

            DeleteObjectRequest request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = keyName,
                VersionId = versionID,
            };

            Console.WriteLine("Deleting an object");
            await client.DeleteObjectAsync(request);
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }

    /// <summary>
    /// This method is used to create the temporary Amazon S3 object.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 object which will be
used
    /// to create the temporary Amazon S3 object.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
object
    /// will be created.</param>
    /// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
    /// <returns>The Version ID of the created object.</returns>

```

```

    public static async Task<string> PutAnObject(IAmazonS3 client, string
bucketName, string objectKey)
    {
        PutObjectRequest request = new PutObjectRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            ContentBody = "This is the content body!",
        };

        PutObjectResponse response = await client.PutObjectAsync(request);
        return response.VersionId;
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section Référence des AWS SDK pour .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_item_in_bucket

```

```
#
# This function deletes the specified file from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - The key (file name) in the bucket to delete.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_item_in_bucket() {
    local bucket_name=$1
    local key=$2
    local response

    response=$(aws s3api delete-object \
        --bucket "$bucket_name" \
        --key "$key")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
        return 1
    fi
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
        .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante supprime un objet nommé `test.txt` d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

Si la gestion des versions de compartiment est activée, la sortie contient l'ID de la version du marqueur de suppression :

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tND1WlGq",
  "DeleteMarker": true
}
```

Pour plus d'informations sur la suppression d'objets, consultez [Suppression d'objets](#) dans le Guide du développeur Amazon S3.

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}
```

```
}

// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "AccessDenied":
                log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
                err = nil
            case "InvalidArgument":
                if bypassGovernance {
                    log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
                    err = nil
                }
            }
        }
    } else {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
            time.Minute)
        if err != nil {
```

```
    log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.\n", key, bucket)\n    } else {\n        deleted = true\n    }\n}\nreturn deleted, err\n}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**\n * Deletes an object from an S3 bucket asynchronously.\n *\n * @param bucketName the name of the S3 bucket\n * @param key         the key (file name) of the object to be deleted\n * @return a {@link CompletableFuture} that completes when the object has\n *         been deleted\n */\npublic CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,\nString key) {\n    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()\n        .bucket(bucketName)\n        .key(key)\n        .build();
```



```
CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
response.whenComplete((deleteRes, ex) -> {
    if (deleteRes != null) {
        logger.info(key + " was deleted");
    } else {
        throw new RuntimeException("An S3 exception occurred during
delete", ex);
    }
});

return response.thenApply(r -> null);
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez un objet.

```
import {
    DeleteObjectCommand,
    S3Client,
    S3ServiceException,
    waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete one object from an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
```

```
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new DeleteObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    await waitUntilObjectNotExists(
      { client },
      { Bucket: bucketName, Key: key },
    );
    // A successful delete, or a delete for a non-existent object, both return
    // a 204 response code.
    console.log(
      `The object "${key}" from bucket "${bucketName}" was deleted, or it didn't
      exist.` ,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}. The bucket
        doesn't exist.` ,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while deleting object from ${bucketName}.  ${caught.name}:
        ${caught.message}` ,
      );
    } else {
      throw caught;
    }
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section Référence des AWS SDK pour JavaScript API.

PHP

Kit SDK pour PHP

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
public function deleteObject(string $bucketName, string $fileName, array
$args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
    $args);
    try {
        $this->client->deleteObject($parameters);
        if ($this->verbose) {
            echo "Deleted the object named: $fileName from $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section Référence des AWS SDK pour PHP API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez un objet.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                               that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def delete(self):
        """
        Deletes the object.
        """
        try:
            self.object.delete()
            self.object.wait_until_not_exists()
            logger.info(
                "Deleted object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
        except ClientError:
            logger.exception(
                "Couldn't delete object '%s' from bucket '%s'.",
                self.object.key,
                self.object.bucket_name,
            )
```

```
raise
```

Faire revenir un objet à une version antérieure en supprimant les versions ultérieures de l'objet.

```
def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are
    # at the end of the list even when they are interspersed in time.
    versions = sorted(
        bucket.object_versions.filter(Prefix=object_key),
        key=attrgetter("last_modified"),
        reverse=True,
    )

    logger.debug(
        "Got versions:\n%s",
        "\n".join(
            [
                f"\t{version.version_id}, last modified {version.last_modified}"
                for version in versions
            ]
        ),
    )

    if version_id in [ver.version_id for ver in versions]:
        print(f"Rolling back to version {version_id}")
        for version in versions:
            if version.version_id != version_id:
                version.delete()
                print(f"Deleted version {version.version_id}")
```

```

        else:
            break

        print(f"Active version is now {bucket.Object(object_key).version_id}")
    else:
        raise KeyError(
            f"{version_id} was not found in the list of versions for "
            f"{object_key}."
        )

```

Faire revenir un objet supprimé en supprimant le marqueur de suppression actif de l'objet.

```

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest
    version
    and the object then presents as not deleted.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to revive.
    """
    # Get the latest version for the object.
    response = s3.meta.client.list_object_versions(
        Bucket=bucket.name, Prefix=object_key, MaxKeys=1
    )

    if "DeleteMarkers" in response:
        latest_version = response["DeleteMarkers"][0]
        if latest_version["IsLatest"]:
            logger.info(
                "Object %s was indeed deleted on %s. Let's revive it.",
                object_key,
                latest_version["LastModified"],
            )

```

```

    )
    obj = bucket.Object(object_key)
    obj.Version(latest_version["VersionId"]).delete()
    logger.info(
        "Revived %s, active version is now %s with body '%s'",
        object_key,
        obj.version_id,
        obj.get()["Body"].read(),
    )
else:
    logger.warning(
        "Delete marker is not the latest version for %s!", object_key
    )
elif "Versions" in response:
    logger.warning("Got an active version for %s, nothing to do.",
object_key)
else:
    logger.error("Couldn't get any version info for %s.", object_key)

```

Créez un gestionnaire Lambda qui supprime un marqueur de suppression d'un objet S3. Ce gestionnaire peut être utilisé pour nettoyer efficacement les marqueurs de suppression superflus dans un compartiment soumis au contrôle de version.

```

import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")

def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    """

```

```

:param context: Context about the event.
:return: A result structure that Amazon S3 uses to interpret the result of
the
        operation. When the result code is TemporaryFailure, S3 retries the
        operation.
"""
# Parse job parameters from Amazon S3 batch operations
invocation_id = event["invocationId"]
invocation_schema_version = event["invocationSchemaVersion"]

results = []
result_code = None
result_string = None

task = event["tasks"][0]
task_id = task["taskId"]

try:
    obj_key = parse.unquote_plus(task["s3Key"], encoding="utf-8")
    obj_version_id = task["s3VersionId"]
    bucket_name = task["s3BucketArn"].split(":")[-1]

    logger.info(
obj_key
        "Got task: remove delete marker %s from object %s.", obj_version_id,
    )

    try:
        # If this call does not raise an error, the object version is not a
delete
        # marker and should not be deleted.
        response = s3.head_object(
            Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
        )
        result_code = "PermanentFailure"
        result_string = (
marker."
            f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
        )

        logger.debug(response)
        logger.warning(result_string)
    except ClientError as error:

```



```

        delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
    "x-amz-delete-marker", "false"
)
    if delete_marker == "true":
        logger.info(
            "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
        )
        try:
            s3.delete_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "Succeeded"
            result_string = (
                f"Successfully removed delete marker "
                f"{obj_version_id} from object {obj_key}."
            )
            logger.info(result_string)
        except ClientError as error:
            # Mark request timeout as a temporary failure so it will be
retried.

            if error.response["Error"]["Code"] == "RequestTimeout":
                result_code = "TemporaryFailure"
                result_string = (
                    f"Attempt to remove delete marker from "
                    f"object {obj_key} timed out."
                )
                logger.info(result_string)
            else:
                raise
        else:
            raise ValueError(
                f"The x-amz-delete-marker header is either not "
                f"present or is not 'true'."
            )
    except Exception as error:
        # Mark all other exceptions as permanent failures.
        result_code = "PermanentFailure"
        result_string = str(error)
        logger.exception(error)
    finally:
        results.append(
            {

```

```
        "taskId": task_id,  
        "resultCode": result_code,  
        "resultString": result_string,  
    }  
)  
return {  
    "invocationSchemaVersion": invocation_schema_version,  
    "treatMissingKeysAs": "PermanentFailure",  
    "invocationId": invocation_id,  
    "results": results,  
}
```

- Pour plus de détails sur l'API, consultez [DeleteObject](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// Delete an object from a bucket.  
pub async fn remove_object(  
    client: &aws_sdk_s3::Client,  
    bucket: &str,  
    key: &str,  
) -> Result<(), S3ExampleError> {  
    client  
        .delete_object()  
        .bucket(bucket)  
        .key(key)  
        .send()  
        .await?;
```

```
// There are no modeled errors to handle when deleting an object.  
  
Ok()  
}
```

- Pour plus de détails sur l'API, voir [DeleteObject](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
    lo_s3->deleteobject(  
        iv_bucket = iv_bucket_name  
        iv_key = iv_object_key ).  
    MESSAGE 'Object deleted from S3 bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Swift

Kit SDK pour Swift

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import AWSS3

public func deleteFile(bucket: String, key: String) async throws {
    let input = DeleteObjectInput(
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.deleteObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Deleting a file. "))
        throw error
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section AWS SDK pour la référence de l'API Swift.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **DeleteObjectTagging** avec une CLI

Les exemples de code suivants illustrent comment utiliser DeleteObjectTagging.

CLI

AWS CLI

Pour supprimer les balises d'un objet

L'exemple `delete-object-tagging` suivant supprime la balise de la clé spécifiée de l'objet `doc1.rtf`.

```
aws s3api delete-object-tagging \  
    --bucket amzn-s3-demo-bucket \  
    --key doc1.rtf
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [DeleteObjectTagging](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande supprime toutes les balises associées à l'objet avec la clé « `testfile.txt` » dans le compartiment S3 donné.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

Sortie :

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target  
"testfile.txt".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y  
testfile.txt
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjectTagging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande supprime toutes les balises associées à l'objet avec la clé « testfile.txt » dans le compartiment S3 donné.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

Sortie :

```
Confirm  
Are you sure you want to perform this action?  
Performing the operation "Remove-S3ObjectTagSet (DeleteObjectTagging)" on target  
"testfile.txt".  
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is  
"Y"): Y  
testfile.txt
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjectTagging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteObjects** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser DeleteObjects.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Principes de base](#)
- [Suppression de tous les objets dans un compartiment](#)
- [Débuter avec le stockage d'objets](#)

.NET

SDK pour .NET (v4)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>
public async Task<bool> DeleteBucketContentsAsync(string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
    try
    {
        // Delete all objects in the bucket.
        var deleteList = await ListBucketContentsAsync(bucketName, false);
        if (deleteList != null && deleteList.Any())
        {
            await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
            });
        }

        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error deleting objects: {ex.Message}");
        return false;
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS SDK pour .NET API.

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez plusieurs objets dans un compartiment S3 non soumis à la gestion des versions.

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete multiple objects from an Amazon Simple
/// Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    /// <summary>
    /// The Main method initializes the Amazon S3 client and the name of
    /// the bucket and then passes those values to MultiObjectDeleteAsync.
    /// </summary>
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";

        // If the Amazon S3 bucket from which you wish to delete objects is
        not
        // located in the same AWS Region as the default user, define the
        // AWS Region for the Amazon S3 bucket as a parameter to the client
        // constructor.
        IAmazonS3 s3Client = new AmazonS3Client();
```



```

        await MultiObjectDeleteAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method uses the passed Amazon S3 client to first create and then
    /// delete three files from the named bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// Amazon S3 methods.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where
objects
    /// will be created and then deleted.</param>
    public static async Task MultiObjectDeleteAsync(IAmazonS3 client, string
bucketName)
    {
        // Create three sample objects which we will then delete.
        var keysAndVersions = await PutObjectsAsync(client, 3, bucketName);

        // Now perform the multi-object delete, passing the key names and
        // version IDs. Since we are working with a non-versioned bucket,
        // the object keys collection includes null version IDs.
        DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest
        {
            BucketName = bucketName,
            Objects = keysAndVersions,
        };

        // You can add a specific object key to the delete request using the
        // AddKey method of the multiObjectDeleteRequest.
        try
        {
            DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
        }
        catch (DeleteObjectsException e)
        {
            PrintDeletionErrorStatus(e);
        }
    }
}

```

```

    /// <summary>
    /// Prints the list of errors raised by the call to DeleteObjectsAsync.
    /// </summary>
    /// <param name="ex">A collection of exceptions returned by the call to
    /// DeleteObjectsAsync.</param>
    public static void PrintDeletionErrorStatus(DeleteObjectsException ex)
    {
        DeleteObjectsResponse errorResponse = ex.Response;
        Console.WriteLine("x {0}", errorResponse.DeletedObjects.Count);

        Console.WriteLine($"Successfully deleted
{errorResponse.DeletedObjects.Count}.");
        Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");

        Console.WriteLine("Printing error data...");
        foreach (DeleteError deleteError in errorResponse.DeleteErrors)
        {
            Console.WriteLine($"Object Key:
{deleteError.Key}\t{deleteError.Code}\t{deleteError.Message}");
        }
    }

    /// <summary>
    /// This method creates simple text file objects that can be used in
    /// the delete method.
    /// </summary>
    /// <param name="client">The Amazon S3 client used to call
    PutObjectAsync.</param>
    /// <param name="number">The number of objects to create.</param>
    /// <param name="bucketName">The name of the bucket where the objects
    /// will be created.</param>
    /// <returns>A list of keys (object keys) and versions that the calling
    /// method will use to delete the newly created files.</returns>
    public static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
    client, int number, string bucketName)
    {
        List<KeyVersion> keys = new List<KeyVersion>();
        for (int i = 0; i < number; i++)
        {
            string key = "ExampleObject-" + new System.Random().Next();
            PutObjectRequest request = new PutObjectRequest
            {

```

```

        BucketName = bucketName,
        Key = key,
        ContentBody = "This is the content body!",
    };

    PutObjectResponse response = await
client.PutObjectAsync(request);

    // For non-versioned bucket operations, we only need the
    // object key.
    KeyVersion keyVersion = new KeyVersion
    {
        Key = key,
    };
    keys.Add(keyVersion);
}

return keys;
}
}

```

Supprimez plusieurs objets dans un compartiment S3 soumis à la gestion des versions.

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete objects in a version-enabled Amazon
/// Simple StorageService (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region for your Amazon S3 bucket is different from
        // the AWS Region of the default user, define the AWS Region for
    }
}

```

```

        // the Amazon S3 bucket and pass it to the client constructor
        // like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
        IAmazonS3 s3Client;

        s3Client = new AmazonS3Client();
        await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
    }

    /// <summary>
    /// This method removes multiple versions and objects from a
    /// version-enabled Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
    {
        // Delete objects (specifying object version in the request).
        await DeleteObjectVersionsAsync(client, bucketName);

        // Delete objects (without specifying object version in the request).
        var deletedObjects = await DeleteObjectsAsync(client, bucketName);

        // Additional exercise - remove the delete markers Amazon S3 returned
from
        // the preceding response. This results in the objects reappearing
        // in the bucket (you can verify the appearance/disappearance of
        // objects in the console).
        await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
    }

    /// <summary>
    /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
    /// them again. The method returns a list of the Amazon S3 objects
deleted.
    /// </summary>

```

```

call    /// <param name="client">The initialized Amazon S3 client object used to
        /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
        /// <param name="bucketName">The name of the bucket where the objects
        /// will be created and then deleted.</param>
        /// <returns>A list of DeletedObjects.</returns>
        public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

            // Delete objects using only keys. Amazon S3 creates a delete marker
and        // returns its version ID in the response.
            List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
            return deletedObjects;
        }

        /// <summary>
        /// This method creates several temporary objects and then deletes them.
        /// </summary>
        /// <param name="client">The S3 client.</param>
        /// <param name="bucketName">Name of the bucket.</param>
        /// <returns>Async task.</returns>
        public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
        {
            // Upload the sample objects.
            var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

            // Delete the specific object versions.
            await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
        }

        /// <summary>
        /// Displays the list of information about deleted files to the console.
        /// </summary>
        /// <param name="e">Error information from the delete process.</param>
        private static void DisplayDeletionErrors(DeleteObjectsException e)
        {
            var errorResponse = e.Response;

```

```

        Console.WriteLine($"No. of objects successfully deleted =
{errorResponse.DeletedObjects.Count}");
        Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");
        Console.WriteLine("Printing error data...");
        foreach (var deleteError in errorResponse.DeleteErrors)
        {
            Console.WriteLine($"Object Key:
{deleteError.Key}\\t{deleteError.Code}\\t{deleteError.Message}");
        }
    }

    /// <summary>
    /// Delete multiple objects from a version-enabled bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    /// <param name="keys">A list of key names for the objects to delete.</
param>
    private static async Task VersionedDeleteAsync(IAmazonS3 client, string
bucketName, List<KeyVersion> keys)
    {
        var multiObjectDeleteRequest = new DeleteObjectsRequest
        {
            BucketName = bucketName,
            Objects = keys, // This includes the object keys and specific
version IDs.
        };

        try
        {
            Console.WriteLine("Executing VersionedDelete...");
            DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted all the
{response.DeletedObjects.Count} items");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

```

```

    }
}

/// <summary>
/// Deletes multiple objects from a non-versioned Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
/// RemoveDeleteMarkersAsync.</param>
/// <param name="bucketName">The name of the bucket from which to delete
/// objects.</param>
/// <param name="keys">A list of key names for the objects to delete.</
param>
/// <returns>A list of the deleted objects.</returns>
private static async Task<List<DeletedObject>>
NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
keys)
{
    // Create a request that includes only the object key names.
    DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest();
    multiObjectDeleteRequest.BucketName = bucketName;

    foreach (var key in keys)
    {
        multiObjectDeleteRequest.AddKey(key.Key);
    }

    // Execute DeleteObjectsAsync.
    // The DeleteObjectsAsync method adds a delete marker for each
    // object deleted. You can verify that the objects were removed
    // using the Amazon S3 console.
    DeleteObjectsResponse response;
    try
    {
        Console.WriteLine("Executing NonVersionedDelete...");
        response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
        Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
    }
    catch (DeleteObjectsException ex)
    {

```

```

        DisplayDeletionErrors(ex);
        throw; // Some deletions failed. Investigate before continuing.
    }

    // This response contains the DeletedObjects list which we use to
delete the delete markers.
    return response.DeletedObjects;
}

/// <summary>
/// Deletes the markers left after deleting the temporary objects.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
/// RemoveDeleteMarkersAsync.</param>
/// <param name="bucketName">The name of the bucket from which to delete
/// objects.</param>
/// <param name="deletedObjects">A list of the objects that were
deleted.</param>
private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
{
    var keyVersionList = new List<KeyVersion>();

    foreach (var deletedObject in deletedObjects)
    {
        KeyVersion keyVersion = new KeyVersion
        {
            Key = deletedObject.Key,
            VersionId = deletedObject.DeleteMarkerVersionId,
        };
        keyVersionList.Add(keyVersion);
    }

    // Create another request to delete the delete markers.
    var multiObjectDeleteRequest = new DeleteObjectsRequest
    {
        BucketName = bucketName,
        Objects = keyVersionList,
    };

    // Now, delete the delete marker to bring your objects back to the
bucket.

```



```

        try
        {
            Console.WriteLine("Removing the delete markers .....");
            var deleteObjectResponse = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted the
{deleteObjectResponse.DeletedObjects.Count} delete markers");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

    /// <summary>
    /// Create temporary Amazon S3 objects to show how object deletion works
in an
    /// Amazon S3 bucket with versioning enabled.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// PutObjectAsync to create temporary objects for the example.</param>
    /// <param name="bucketName">A string representing the name of the S3
    /// bucket where we will create the temporary objects.</param>
    /// <param name="number">The number of temporary objects to create.</
param>
    /// <returns>A list of the KeyVersion objects.</returns>
    private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, string bucketName, int number)
    {
        var keys = new List<KeyVersion>();

        for (var i = 0; i < number; i++)
        {
            string key = "ObjectToDelete-" + new System.Random().Next();
            PutObjectRequest request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = key,
                ContentBody = "This is the content body!",
            };

            var response = await client.PutObjectAsync(request);
            KeyVersion keyVersion = new KeyVersion

```

```

        {
            Key = key,
            VersionId = response.VersionId,
        };

        keys.Add(keyVersion);
    }

    return keys;
}
}

```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS SDK pour .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_items_in_bucket
#
# This function deletes the specified list of keys from the specified bucket.
#

```

```

# Parameters:
#     $1 - The name of the bucket.
#     $2 - A list of keys in the bucket to delete.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

    # Create the JSON for the items to delete.
    local delete_items
    delete_items="{\"Objects\":["
    for key in $keys; do
        delete_items="$delete_items{\"Key\": \"$key\"},"
    done
    delete_items=${delete_items%?} # Remove the final comma.
    delete_items="$delete_items]"

    response=$(aws s3api delete-objects \
        --bucket "$bucket_name" \
        --delete "$delete_items")

    # shellcheck disable=SC2181
    if [[ $? -ne 0 ]]; then
        errecho "ERROR: AWS reports s3api delete-object operation failed.\n
$response"
        return 1
    fi
}

```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                                const Aws::String &fromBucket,
                                const Aws::S3::S3ClientConfiguration
                                &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {

deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
        client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }
    }
}
```

```
        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante supprime un objet nommé d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

`delete.json` est un document JSON du répertoire en cours qui spécifie l'objet à supprimer :

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

Sortie :

```
{
  "Deleted": [
    {
      "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUP01koHwzU",
      "Key": "test1.txt",
      "DeleteMarker": true
    }
  ]
}
```

```
]
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// DeleteObjects deletes a list of objects from a bucket.
```

```

func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &types.Delete{
            Objects: objects,
            Quiet:   aws.Bool(true),
        },
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
    if err != nil || len(delOut.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucket)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
        } else if len(delOut.Errors) > 0 {
            for _, outErr := range delOut.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *delOut.Errors[0].Message)
        }
    } else {
        for _, delObj := range delOut.Deleted {
            err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObj.Key},
                time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to be deleted.\n",
*delObj.Key)
            } else {
                log.Printf("Deleted %s.\n", *delObj.Key)
            }
        }
    }
}

```

```
    return err
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class DeleteMultiObjects {
```



```
public static void main(String[] args) {
    final String usage = ""

        Usage:    <bucketName>

        Where:
            bucketName - the Amazon S3 bucket name.
        "";

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    deleteBucketObjects(s3, bucketName);
    s3.close();
}

/**
 * Deletes multiple objects from an Amazon S3 bucket.
 *
 * @param s3 An Amazon S3 client object.
 * @param bucketName The name of the Amazon S3 bucket to delete objects from.
 */
public static void deleteBucketObjects(S3Client s3, String bucketName) {
    // Upload three sample objects to the specified Amazon S3 bucket.
    ArrayList<ObjectIdentifier> keys = new ArrayList<>();
    PutObjectRequest putObj;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putObj = PutObjectRequest.builder()
            .bucket(bucketName)
```

```
        .key(keyName)
        .build();

        s3.putObject(putOb, RequestBody.fromString(keyName));
        keys.add(objectId);
    }

    System.out.println(keys.size() + " objects successfully created.");

    // Delete multiple objects in one request.
    Delete del = Delete.builder()
        .objects(keys)
        .build();

    try {
        DeleteObjectsRequest multiObjectDeleteRequest =
DeleteObjectsRequest.builder()
        .bucket(bucketName)
        .delete(del)
        .build();

        s3.deleteObjects(multiObjectDeleteRequest);
        System.out.println("Multiple objects are deleted!");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez plusieurs objets.

```
import {
  DeleteObjectsCommand,
  S3Client,
  S3ServiceException,
  waitUntilObjectNotExists,
} from "@aws-sdk/client-s3";

/**
 * Delete multiple objects from an S3 bucket.
 * @param {{ bucketName: string, keys: string[] }}
 */
export const main = async ({ bucketName, keys }) => {
  const client = new S3Client({});

  try {
    const { Deleted } = await client.send(
      new DeleteObjectsCommand({
        Bucket: bucketName,
        Delete: {
          Objects: keys.map((k) => ({ Key: k })),
        },
      }),
    );
    for (const key in keys) {
      await waitUntilObjectNotExists(
        { client },
        { Bucket: bucketName, Key: key },
      );
    }
    console.log(
```

```
        `Successfully deleted ${Deleted.length} objects from S3 bucket. Deleted
objects:`,
    );
    console.log(Deleted.map((d) => ` • ${d.Key}`).join("\n"));
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while deleting objects from ${bucketName}. The bucket
doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while deleting objects from ${bucketName}.
${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS SDK pour JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun deleteBucketObjects(
    bucketName: String,
    objectName: String,
```

```
) {  
    val objectId =  
        ObjectIdentifier {  
            key = objectName  
        }  
  
    val delObj =  
        Delete {  
            objects = listOf(objectId)  
        }  
  
    val request =  
        DeleteObjectsRequest {  
            bucket = bucketName  
            delete = delObj  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteObjects(request)  
        println("$objectName was deleted from $bucketName")  
    }  
}
```

- Pour plus de détails sur l'API, consultez [DeleteObjects](#) la section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez un ensemble d'objets d'une liste de clés.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);
```

```
try {
    $objects = [];
    foreach ($contents['Contents'] as $content) {
        $objects[] = [
            'Key' => $content['Key'],
        ];
    }
    $this->s3client->deleteObjects([
        'Bucket' => $this->bucketName,
        'Delete' => [
            'Objects' => $objects,
        ],
    ]);
    $check = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    if (isset($check['Contents']) && count($check['Contents']) > 0) {
        throw new Exception("Bucket wasn't empty.");
    }
    echo "Deleted all objects and folders from $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to delete $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with object deletion before continuing.");
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS SDK pour PHP API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande supprime l'objet « sample.txt » du compartiment « test-files ». Vous devrez confirmer avant l'exécution de la commande ; pour supprimer l'invite, utilisez le commutateur -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemple 2 : cette commande supprime la version de l'objet « sample.txt » depuis le compartiment « test-files », en supposant que le compartiment a été configuré pour activer les versions d'objet.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId  
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Exemple 3 : cette commande supprime les objets « sample1.txt », « sample2.txt » et « sample3.txt » du compartiment « test-files » en une seule opération par lots. La réponse du service répertoriera toutes les clés traitées, quel que soit le statut de réussite ou d'erreur de la suppression. Pour obtenir uniquement les erreurs pour les clés qui n'ont pas pu être traitées par le service, ajoutez le ReportErrorsOnly paramètre - (ce paramètre peut également être spécifié avec l'alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

Exemple 4 : Cet exemple utilise une expression en ligne avec le KeyCollection paramètre - pour obtenir les clés des objets à supprimer. Get-S3Object renvoie une collection d'Amazon.S3.Model.S3Object instances dont chacune possède un membre clé de type chaîne identifiant l'objet.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Exemple 5 : Cet exemple permet d'obtenir tous les objets dont le préfixe clé est prefix/subprefix « » dans le compartiment et de les supprimer. Notez que les objets entrants sont traités un par un. Pour les collections volumineuses, pensez à transmettre la collection au paramètre - InputObject (alias -S3ObjectCollection) de l'applet de commande pour permettre à la suppression de se produire par lots avec un seul appel au service.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

Exemple 6 : Cet exemple dirige une collection d'Amazon.S3.Model.S3ObjectVersion instances représentant des marqueurs de suppression vers l'applet de commande pour suppression. Notez que les objets entrants sont traités un par un. Pour les collections volumineuses, pensez à transmettre la collection au paramètre - InputObject (alias -

S3ObjectCollection) de l'applet de commande pour permettre à la suppression de se produire par lots avec un seul appel au service.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Exemple 7 : Ce script montre comment supprimer par lots un ensemble d'objets (dans ce cas, des marqueurs de suppression) en créant un tableau d'objets à utiliser avec le KeyAndVersionCollection paramètre -.

```
$keyVersions = @()  
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where  
  {$_ .IsDeleteMarker -eq "True"}  
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =  
  $marker.VersionId } }  
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -  
Force
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande supprime l'objet « sample.txt » du compartiment « test-files ». Vous devrez confirmer avant l'exécution de la commande ; pour supprimer l'invite, utilisez le commutateur -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemple 2 : cette commande supprime la version de l'objet « sample.txt » depuis le compartiment « test-files », en supposant que le compartiment a été configuré pour activer les versions d'objet.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId  
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Exemple 3 : cette commande supprime les objets « sample1.txt », « sample2.txt » et « sample3.txt » du compartiment « test-files » en une seule opération par lots. La réponse du service répertoriera toutes les clés traitées, quel que soit le statut de réussite ou d'erreur de la suppression. Pour obtenir uniquement les erreurs pour les clés qui n'ont pas pu être traitées

par le service, ajoutez le ReportErrorsOnly paramètre - (ce paramètre peut également être spécifié avec l'alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

Exemple 4 : Cet exemple utilise une expression en ligne avec le KeyCollection paramètre - pour obtenir les clés des objets à supprimer. Get-S3Object renvoie une collection d'Amazon.S3.Model.S3Object instances dont chacune possède un membre clé de type chaîne identifiant l'objet.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Exemple 5 : Cet exemple permet d'obtenir tous les objets dont le préfixe clé est prefix/subprefix « » dans le compartiment et de les supprimer. Notez que les objets entrants sont traités un par un. Pour les collections volumineuses, pensez à transmettre la collection au paramètre - InputObject (alias -S3ObjectCollection) de l'applet de commande pour permettre à la suppression de se produire par lots avec un seul appel au service.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

Exemple 6 : Cet exemple dirige une collection d'Amazon.S3.Model.S3ObjectVersion instances représentant des marqueurs de suppression vers l'applet de commande pour suppression. Notez que les objets entrants sont traités un par un. Pour les collections volumineuses, pensez à transmettre la collection au paramètre - InputObject (alias -S3ObjectCollection) de l'applet de commande pour permettre à la suppression de se produire par lots avec un seul appel au service.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Exemple 7 : Ce script montre comment supprimer par lots un ensemble d'objets (dans ce cas, des marqueurs de suppression) en créant un tableau d'objets à utiliser avec le KeyAndVersionCollection paramètre -.

```
$keyVersions = @()
```

```
$markers = (Get-S3Version -BucketName $BucketName).Versions | Where
{$_ .IsDeleteMarker -eq "True"}
foreach ($marker in $markers) { $keyVersions += @{ Key = $marker.Key; VersionId =
$marker.VersionId } }
Remove-S3Object -BucketName $BucketName -KeyAndVersionCollection $keyVersions -
Force
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez un ensemble d'objets à l'aide d'une liste de clés d'objet.

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def delete_objects(bucket, object_keys):
        """
        Removes a list of objects from a bucket.
        This operation is done as a batch in a single request.
        """
```

```

:param bucket: The bucket that contains the objects. This is a Boto3
Bucket
                resource.
:param object_keys: The list of keys that identify the objects to remove.
:return: The response that contains data about which objects were deleted
        and any that could not be deleted.
"""
try:
    response = bucket.delete_objects(
        Delete={"Objects": [{"Key": key} for key in object_keys]}
    )
    if "Deleted" in response:
        logger.info(
            "Deleted objects '%s' from bucket '%s'.",
            [del_obj["Key"] for del_obj in response["Deleted"]],
            bucket.name,
        )
    if "Errors" in response:
        logger.warning(
            "Could not delete objects '%s' from bucket '%s'.",
            [
                f"{del_obj['Key']}: {del_obj['Code']}"
                for del_obj in response["Errors"]
            ],
            bucket.name,
        )
    except ClientError:
        logger.exception("Couldn't delete any objects from bucket %s.",
            bucket.name)
        raise
    else:
        return response

```

Supprimez tous les objets dans un compartiment.

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """

```

```

        :param s3_object: A Boto3 Object resource. This is a high-level resource
in Boto3
                                that wraps object actions in a class-like structure.
    """
    self.object = s3_object
    self.key = self.object.key

    @staticmethod
    def empty_bucket(bucket):
        """
        Remove all objects from a bucket.

        :param bucket: The bucket to empty. This is a Boto3 Bucket resource.
        """
        try:
            bucket.objects.delete()
            logger.info("Emptied bucket '%s'.", bucket.name)
        except ClientError:
            logger.exception("Couldn't empty bucket '%s'.", bucket.name)
            raise

```

Supprimez définitivement un objet soumis à la gestion des versions en supprimant toutes ses versions.

```

def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that contains the object.
    :param object_key: The object to delete.
    """
    try:
        bucket.object_versions.filter(Prefix=object_key).delete()
        logger.info("Permanently deleted all versions of object %s.", object_key)
    except ClientError:
        logger.exception("Couldn't delete all versions of %s.", object_key)
        raise

```

- Pour plus de détails sur l'API, consultez [DeleteObjects](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)?")
  answer = gets.chomp.downcase
  if answer == 'y'
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS SDK pour Ruby API.

Rust

SDK pour Rust

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

/// Delete the objects in a bucket.
pub async fn delete_objects(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    objects_to_delete: Vec<String>,
) -> Result<(), S3ExampleError> {
    // Push into a mut vector to use '?' early return errors while building
    object keys.
    let mut delete_object_ids: Vec<aws_sdk_s3::types::ObjectIdentifier> = vec![];
    for obj in objects_to_delete {
        let obj_id = aws_sdk_s3::types::ObjectIdentifier::builder()
            .key(obj)
            .build()
            .map_err(|err| {
                S3ExampleError::new(format!("Failed to build key for
delete_object: {err:?}"))
            })?;
        delete_object_ids.push(obj_id);
    }

    client
        .delete_objects()
        .bucket(bucket_name)
        .delete(
            aws_sdk_s3::types::Delete::builder()
                .set_objects(Some(delete_object_ids))
                .build()
                .map_err(|err| {
                    S3ExampleError::new(format!("Failed to build delete_object
input {err:?}"))
                })?,
        )
}

```

```
        .send()
        .await?;
    Ok(())
}
```

- Pour plus de détails sur l'API, voir [DeleteObjects](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.
    oo_result = lo_s3->deleteobjects(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name
    io_delete = NEW /aws1/cl_s3_delete( it_objects = it_object_keys ) ).
    MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Swift

Kit SDK pour Swift

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import AWSS3

public func deleteObjects(bucket: String, keys: [String]) async throws {
    let input = DeleteObjectsInput(
        bucket: bucket,
        delete: S3ClientTypes.Delete(
            objects: keys.map { S3ClientTypes.ObjectIdentifier(key: $0) },
            quiet: true
        )
    )

    do {
        _ = try await client.deleteObjects(input: input)
    } catch {
        print("ERROR: deleteObjects:", dump(error))
        throw error
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section AWS SDK pour la référence de l'API Swift.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **DeletePublicAccessBlock** avec une CLI

Les exemples de code suivants illustrent comment utiliser DeletePublicAccessBlock.

CLI

AWS CLI

Pour supprimer la configuration du blocage de l'accès public d'un compartiment

L'exemple `delete-public-access-block` suivant supprime la configuration du blocage de l'accès public du compartiment spécifié.

```
aws s3api delete-public-access-block \
  --bucket amzn-s3-demo-bucket
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [DeletePublicAccessBlock](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande désactive le paramètre de blocage de l'accès public pour le compartiment donné.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
'^BucketName'
```

Sortie :

```
amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [DeletePublicAccessBlock](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande désactive le paramètre de blocage de l'accès public pour le compartiment donné.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
'^BucketName'
```

Sortie :

```
amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [DeletePublicAccessBlock](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetBucketAccelerateConfiguration** avec une CLI

Les exemples de code suivants illustrent comment utiliser **GetBucketAccelerateConfiguration**.

CLI

AWS CLI

Pour extraire la configuration accélérée d'un compartiment

L'exemple `get-bucket-accelerate-configuration` suivant extrait la configuration accélérée du compartiment spécifié.

```
aws s3api get-bucket-accelerate-configuration \  
--bucket amzn-s3-demo-bucket
```

Sortie :

```
{  
  "Status": "Enabled"  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAccelerateConfiguration](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie la valeur Enabled, si les paramètres d'accélération du transfert sont activés pour le compartiment spécifié.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Value
-----
Enabled
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAccelerateConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie la valeur Enabled, si les paramètres d'accélération du transfert sont activés pour le compartiment spécifié.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Value
-----
Enabled
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAccelerateConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetBucketAc1** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser GetBucketAc1.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Vérification de l'existence d'un compartiment](#)
- [Gérer les listes de contrôle d'accès \(ACL\)](#)

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Get the access control list (ACL) for the new bucket.
/// </summary>
/// <param name="client">The initialized client object used to get the
/// access control list (ACL) of the bucket.</param>
/// <param name="newBucketName">The name of the newly created bucket.</
param>
/// <returns>An S3AccessControlList.</returns>
public static async Task<S3AccessControlList>
GetACLForBucketAsync(IAmazonS3 client, string newBucketName)
{
    // Retrieve bucket ACL to show that the ACL was properly applied to
    // the new bucket.
    GetACLResponse getACLResponse = await client.GetACLAsync(new
GetACLRequest
    {
        BucketName = newBucketName,
    });

    return getACLResponse.AccessControlList;
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAcl](#) à la section Référence des AWS SDK pour .NET API.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
        s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                    << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            std::cout << "For bucket " << bucketName << ": "
                      << std::endl << std::endl;
        }
    }
}
```

```

        if (grantee.TypeHasBeenSet()) {
            std::cout << "Type:          "
                        << getGranteeTypeString(grantee.GetType()) <<
std::endl;
        }

        if (grantee.DisplayNameHasBeenSet()) {
            std::cout << "Display name:  "
                        << grantee.GetDisplayName() << std::endl;
        }

        if (grantee.EmailAddressHasBeenSet()) {
            std::cout << "Email address: "
                        << grantee.GetEmailAddress() << std::endl;
        }

        if (grantee.IDHasBeenSet()) {
            std::cout << "ID:           "
                        << grantee.GetID() << std::endl;
        }

        if (grantee.URIHasBeenSet()) {
            std::cout << "URI:          "
                        << grantee.GetURI() << std::endl;
        }

        std::cout << "Permission:    " <<
                    getPermissionString(grant.GetPermission()) <<
                    std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
    string.
/*!
    \param type: Type enumeration.
    \return String: Human-readable string.
*/

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {

```

```

    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable
string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string.
 */

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                "objects in this bucket, and read/write this "
                "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }

    return "Permission unknown";
}

```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAcl](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante extrait la liste de contrôle d'accès d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAcl](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAclRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAclResponse;
import software.amazon.awssdk.services.s3.model.Grant;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */

public class GetAcl {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <bucketName> <objectKey>

        Where:
            bucketName - The Amazon S3 bucket to get the access control list
            (ACL) for.
            objectKey - The object to get the ACL for.\s
        "";
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        System.out.println("Retrieving ACL for object: " + objectKey);
        System.out.println("in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getBucketACL(s3, objectKey, bucketName);
        s3.close();
        System.out.println("Done!");
    }

    /**
     * Retrieves the Access Control List (ACL) for an object in an Amazon S3
     * bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param objectKey The key of the object for which the ACL is to be
     * retrieved.
     * @param bucketName The name of the bucket containing the object.
     * @return The ID of the grantee who has permission on the object, or an
     * empty string if an error occurs.
     */
    public static String getBucketACL(S3Client s3, String objectKey, String
    bucketName) {
        try {
            GetObjectAclRequest aclReq = GetObjectAclRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            GetObjectAclResponse aclRes = s3.getObjectAcl(aclReq);
            List<Grant> grants = aclRes.grants();
            String grantee = "";
            for (Grant grant : grants) {
```

```

        System.out.format("  %s: %s\n", grant.grantee().id(),
grant.permission());
        grantee = grant.grantee().id();
    }

    return grantee;
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

return "";
}
}

```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAcl](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez les autorisations ACL.

```

import {
    GetBucketAclCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Retrieves the Access Control List (ACL) for an S3 bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {

```

```
const client = new S3Client({});

try {
  const response = await client.send(
    new GetBucketAclCommand({
      Bucket: bucketName,
    }),
  );
  console.log(`ACL for bucket "${bucketName}":`);
  console.log(JSON.stringify(response, null, 2));
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while getting ACL for ${bucketName}. The bucket doesn't exist.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting ACL for ${bucketName}. ${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [GetBucketAcl](#) à la section Référence des AWS SDK pour JavaScript API.

PowerShell

Outils pour PowerShell V5

Exemple 1 : la commande obtient les détails du propriétaire de l'objet S3.

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

Sortie :

```

DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544

```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAcl](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_acl(self):
        """
        Get the ACL of the bucket.

        :return: The ACL of the bucket.
        """
        try:
            acl = self.bucket.Acl()

```

```

        logger.info(
            "Got ACL for bucket %s. Owner is %s.", self.bucket.name,
            acl.owner
        )
    except ClientError:
        logger.exception("Couldn't get ACL for bucket %s.", self.bucket.name)
        raise
    else:
        return acl

```

- Pour plus de détails sur l'API, consultez [GetBucketAcl](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_s3->getbucketacl(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved bucket ACL.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAcl](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetBucketAnalyticsConfiguration** avec une CLI

Les exemples de code suivants illustrent comment utiliser `GetBucketAnalyticsConfiguration`.

CLI

AWS CLI

Pour récupérer la configuration d'analytique d'un compartiment possédant un ID spécifique

L'exemple `get-bucket-analytics-configuration` suivant affiche la configuration d'analyse pour le compartiment et l'ID spécifiés.

```
aws s3api get-bucket-analytics-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 1
```

Sortie :

```
{
  "AnalyticsConfiguration": {
    "StorageClassAnalysis": {},
    "Id": "1"
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAnalyticsConfiguration](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie les détails du filtre d'analyse nommé « testfilter » dans le compartiment S3 donné.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAnalyticsConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie les détails du filtre d'analyse nommé « testfilter » dans le compartiment S3 donné.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAnalyticsConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetBucketCors** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetBucketCors`.

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>  
/// Retrieve the CORS configuration applied to the Amazon S3 bucket.  
/// </summary>  
/// <param name="client">The initialized Amazon S3 client object used  
/// to retrieve the CORS configuration.</param>
```



```
/// <returns>The created CORS configuration object.</returns>
private static async Task<CORSConfiguration>
RetrieveCORSConfigurationAsync(AmazonS3Client client)
{
    GetCORSConfigurationRequest request = new
GetCORSConfigurationRequest()
    {
        BucketName = BucketName,
    };
    var response = await client.GetCORSConfigurationAsync(request);
    var configuration = response.Configuration;
    PrintCORSRules(configuration);
    return configuration;
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketCors](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

La commande suivante permet de récupérer la configuration de partage Cross-Origin des ressources pour un bucket nommé `amzn-s3-demo-bucket` :

```
aws s3api get-bucket-cors --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "CORSRules": [
    {
      "AllowedHeaders": [
        "*"
      ],
      "ExposeHeaders": [
        "x-amz-server-side-encryption"
      ],
      "AllowedMethods": [
        "PUT",
```

```
        "POST",
        "DELETE"
    ],
    "MaxAgeSeconds": 3000,
    "AllowedOrigins": [
        "http://www.example.com"
    ]
},
{
    "AllowedHeaders": [
        "Authorization"
    ],
    "MaxAgeSeconds": 3000,
    "AllowedMethods": [
        "GET"
    ],
    "AllowedOrigins": [
        "*"
    ]
}
]
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketCors](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez la politique CORS pour le compartiment.

```
import {
    GetBucketCorsCommand,
    S3Client,
    S3ServiceException,
```

```
} from "@aws-sdk/client-s3";

/**
 * Log the Cross-Origin Resource Sharing (CORS) configuration information
 * set for the bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new GetBucketCorsCommand({
    Bucket: bucketName,
  });

  try {
    const { CORSRules } = await client.send(command);
    console.log(JSON.stringify(CORSRules));
    CORSRules.forEach((cr, i) => {
      console.log(
        `\\nCORSRule ${i + 1}`,
        `\\n${'"'}.repeat(10)}`,
        `\\nAllowedHeaders: ${cr.AllowedHeaders}`,
        `\\nAllowedMethods: ${cr.AllowedMethods}`,
        `\\nAllowedOrigins: ${cr.AllowedOrigins}`,
        `\\nExposeHeaders: ${cr.ExposeHeaders}`,
        `\\nMaxAgeSeconds: ${cr.MaxAgeSeconds}`,
      );
    });
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting bucket CORS rules for ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
}
```

```
}  
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [GetBucketCors](#) à la section Référence des AWS SDK pour JavaScript API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""  
  
    def __init__(self, bucket):  
        """  
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in  
Boto3  
        that wraps bucket actions in a class-like structure.  
        """  
        self.bucket = bucket  
        self.name = bucket.name  
  
    def get_cors(self):  
        """  
        Get the CORS rules for the bucket.  
  
        :return The CORS rules for the specified bucket.  
        """  
        try:  
            cors = self.bucket.Cors()  
            logger.info(  

```

```

        "Got CORS rules %s for bucket '%s'.", cors.cors_rules,
        self.bucket.name
    )
    except ClientError:
        logger.exception(("Couldn't get CORS for bucket %s.",
        self.bucket.name))
        raise
    else:
        return cors

```

- Pour plus de détails sur l'API, consultez [GetBucketCors](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
  # configuration for the bucket.

```

```

def cors
  @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
why: #{e.message}"
    nil
  end
end

end

```

- Pour plus de détails sur l'API, reportez-vous [GetBucketCors](#) à la section Référence des AWS SDK pour Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_s3->getbucketcors(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved bucket CORS configuration.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [GetBucketCors](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetBucketEncryption** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetBucketEncryption`.

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Get and print the encryption settings of a bucket.
/// </summary>
/// <param name="bucketName">Name of the bucket.</param>
/// <returns>Async task.</returns>
public static async Task GetEncryptionSettings(string bucketName)
{
    // Check and print the bucket encryption settings.
    Console.WriteLine($"Getting encryption settings for bucket
{bucketName}.");

    try
    {
        var settings =
            await _s3Client.GetBucketEncryptionAsync(
                new GetBucketEncryptionRequest() { BucketName =
bucketName });

        foreach (var encryptionSettings in
settings?.ServerSideEncryptionConfiguration?.ServerSideEncryptionRules!)
        {
            Console.WriteLine(
                $"Algorithm:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm}");
            Console.WriteLine(
                $"Key:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionKeyManagementServi
        }
    }
}
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine(ex.ErrorCode == "InvalidBucketName"
                ? $"Bucket {bucketName} was not found."
                : $"Unable to get bucket encryption for bucket {bucketName},
{ex.Message}");
        }
    }
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketEncryption](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour extraire la configuration du chiffrement côté serveur d'un compartiment

L'exemple `get-bucket-encryption` suivant extrait la configuration du chiffrement côté serveur du compartiment `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketEncryption](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie toutes les règles de chiffrement côté serveur associées au compartiment donné.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketEncryption](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie toutes les règles de chiffrement côté serveur associées au compartiment donné.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketEncryption](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetBucketInventoryConfiguration** avec une CLI

Les exemples de code suivants illustrent comment utiliser **GetBucketInventoryConfiguration**.

CLI

AWS CLI

Pour extraire la configuration d'inventaire d'un compartiment

L'exemple `get-bucket-inventory-configuration` suivant extrait la configuration d'inventaire du compartiment spécifié avec l'ID 1.

```
aws s3api get-bucket-inventory-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --inventory-id 1
```

--id 1

Sortie :

```
{
  "InventoryConfiguration": {
    "IsEnabled": true,
    "Destination": {
      "S3BucketDestination": {
        "Format": "ORC",
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
        "AccountId": "123456789012"
      }
    },
    "IncludedObjectVersions": "Current",
    "Id": "1",
    "Schedule": {
      "Frequency": "Weekly"
    }
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketInventoryConfiguration](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie les détails de l'inventaire nommé « testinventory » pour le compartiment S3 donné.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId 'testinventory'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketInventoryConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie les détails de l'inventaire nommé « testinventory » pour le compartiment S3 donné.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId  
'testinventory'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketInventoryConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetBucketLifecycleConfiguration** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetBucketLifecycleConfiguration`.

.NET

SDK pour .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>  
/// Returns a configuration object for the supplied bucket name.  
/// </summary>  
/// <param name="client">The S3 client object used to call  
/// the GetLifecycleConfigurationAsync method.</param>  
/// <param name="bucketName">The name of the S3 bucket for which a  
/// configuration will be created.</param>  
/// <returns>Returns a new LifecycleConfiguration object.</returns>  
public static async Task<LifecycleConfiguration>  
RetrieveLifecycleConfigAsync(IAmazonS3 client, string bucketName)  
{  
    var request = new GetLifecycleConfigurationRequest()  
    {  
        BucketName = bucketName,  
    };  
    var response = await client.GetLifecycleConfigurationAsync(request);
```

```
        var configuration = response.Configuration;
        return configuration;
    }
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketLifecycleConfiguration](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

La commande suivante extrait la configuration du cycle de vie d'un compartiment intitulé `amzn-s3-demo-bucket` :

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "Rules": [
    {
      "ID": "Move rotated logs to Glacier",
      "Prefix": "rotated/",
      "Status": "Enabled",
      "Transitions": [
        {
          "Date": "2015-11-10T00:00:00.000Z",
          "StorageClass": "GLACIER"
        }
      ]
    },
    {
      "Status": "Enabled",
      "Prefix": "",
      "NoncurrentVersionTransitions": [
        {
          "NoncurrentDays": 0,
          "StorageClass": "GLACIER"
        }
      ]
    }
  ],
}
```

```

        "ID": "Move old versions to Glacier"
    }
]
}

```

- Pour plus de détails sur l'API, reportez-vous [GetBucketLifecycleConfiguration](#) à la section Référence des AWS CLI commandes.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_lifecycle_configuration(self):
        """
        Get the lifecycle configuration of the bucket.

        :return: The lifecycle rules of the specified bucket.
        """
        try:
            config = self.bucket.LifecycleConfiguration()
            logger.info(
                "Got lifecycle rules %s for bucket '%s'.",

```

```

        config.rules,
        self.bucket.name,
    )
except:
    logger.exception(
        "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
    )
    raise
else:
    return config.rules

```

- Pour plus de détails sur l'API, consultez [GetBucketLifecycleConfiguration](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_s3->getbucketlifecycleconf(          " oo_result is
returned for testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved bucket lifecycle configuration.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [GetBucketLifecycleConfiguration](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetBucketLocation** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser GetBucketLocation.

CLI

AWS CLI

La commande suivante extrait la contrainte d'emplacement pour un compartiment nommé `amzn-s3-demo-bucket`, s'il en existe une :

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "LocationConstraint": "us-west-2"
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketLocation](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie la contrainte d'emplacement pour le compartiment « `amzn-s3-demo-bucket` », s'il en existe une.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Value
-----
ap-south-1
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketLocation](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie la contrainte d'emplacement pour le compartiment « amzn-s3-demo-bucket », s'il en existe une.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Value
-----
ap-south-1
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketLocation](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
```



```

        num_buckets += 1;
        if strict {
            let r = client
                .get_bucket_location()
                .bucket(bucket.name().unwrap_or_default())
                .send()
                .await?;

            if r.location_constraint() == Some(&region) {
                println!("{}", bucket.name().unwrap_or_default());
                in_region += 1;
            }
        } else {
            println!("{}", bucket.name().unwrap_or_default());
        }
    }

    println!();
    if strict {
        println!(
            "Found {} buckets in the {} region out of a total of {} buckets.",
            in_region, region, num_buckets
        );
    } else {
        println!("Found {} buckets in all regions.", num_buckets);
    }

    Ok(())
}

```

- Pour plus de détails sur l'API, voir [GetBucketLocation](#) la section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetBucketLogging** avec une CLI

Les exemples de code suivants illustrent comment utiliser `GetBucketLogging`.

CLI

AWS CLI

Pour extraire le statut de journalisation d'un compartiment

L'exemple `get-bucket-logging` suivant extrait le statut de journalisation du compartiment spécifié.

```
aws s3api get-bucket-logging \  
  --bucket amzn-s3-demo-bucket
```

Sortie :

```
{  
  "LoggingEnabled": {  
    "TargetPrefix": "",  
    "TargetBucket": "amzn-s3-demo-bucket-logs"  
  }  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketLogging](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie le statut de journalisation pour le compartiment spécifié.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- Pour plus de détails sur l'API, reportez-vous [GetBucketLogging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie le statut de journalisation pour le compartiment spécifié.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

TargetBucketName	Grants	TargetPrefix
testbucket1	{}	testprefix

- Pour plus de détails sur l'API, reportez-vous [GetBucketLogging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetBucketMetricsConfiguration** avec une CLI

Les exemples de code suivants illustrent comment utiliser GetBucketMetricsConfiguration.

CLI

AWS CLI

Pour extraire la configuration des métriques d'un compartiment avec un ID spécifique

L'exemple get-bucket-metrics-configuration suivant affiche la configuration des métriques pour le compartiment et l'ID spécifiés.

```
aws s3api get-bucket-metrics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 123
```

Sortie :

```
{  
  "MetricsConfiguration": {  
    "Filter": {
```

```
        "Prefix": "logs"
      },
      "Id": "123"
    }
  }
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketMetricsConfiguration](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie les détails du filtre de métriques nommé « testfilter » pour le compartiment S3 donné.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketMetricsConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie les détails du filtre de métriques nommé « testfilter » pour le compartiment S3 donné.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketMetricsConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetBucketNotification** avec une CLI

Les exemples de code suivants illustrent comment utiliser `GetBucketNotification`.

CLI

AWS CLI

La commande suivante extrait la notification de configuration d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api get-bucket-notification --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "TopicConfiguration": {
    "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
    "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNl",
    "Event": "s3:ObjectCreated:*",
    "Events": [
      "s3:ObjectCreated:*"
    ]
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketNotification](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cet exemple extrait la configuration des notifications pour le compartiment donné

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
TopicConfigurations
```

Sortie :

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketNotification](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cet exemple extrait la configuration des notifications pour le compartiment donné

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty TopicConfigurations
```

Sortie :

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketNotification](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetBucketPolicy** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetBucketPolicy`.

C++

Kit de développement logiciel (SDK) for C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                  const Aws::S3::S3ClientConfiguration
                                  &clientConfig) {
```

```

    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
        s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
                    << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n"
<<
                policy_stream.str() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicy](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante extrait la stratégie du compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "Policy": "{\"Version\":\"2008-10-17\",\"Statement\": [{\"Sid\":\"\",\"Effect\":\"Allow\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::amzn-s3-demo-bucket/*\"}, {\"Sid\":\"\",\"Effect\":\"Deny\",\"Principal\":\"*\",\"Action\":\"s3:GetObject\",\"Resource\":\"arn:aws:s3:::amzn-s3-demo-bucket/secret/*\"}]}"
}
```

Stratégie Get and put a bucket L'exemple suivant montre comment télécharger une stratégie de compartiment Amazon S3, modifier le fichier, puis utiliser `put-bucket-policy` pour appliquer la stratégie de compartiment modifiée. Pour télécharger la stratégie de compartiment dans un fichier, vous pouvez exécuter :

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output text > policy.json
```

Vous pouvez ensuite modifier le fichier `policy.json` si besoin. Enfin, vous pouvez réappliquer cette stratégie modifiée au compartiment S3 en exécutant :

le fichier `policy.json` si besoin. Enfin, vous pouvez réappliquer cette stratégie modifiée au compartiment S3 en exécutant :

le fichier si besoin. Enfin, vous pouvez réappliquer cette stratégie modifiée au compartiment S3 en exécutant :

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://policy.json
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicy](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class GetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>

            Where:
                bucketName - The Amazon S3 bucket to get the policy from.
            "";

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        System.out.format("Getting policy for bucket: \"%s\"\\n\\n", bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        String polText = getPolicy(s3, bucketName);
        System.out.println("Policy Text: " + polText);
        s3.close();
    }
}
```

```
}

/**
 * Retrieves the policy for the specified Amazon S3 bucket.
 *
 * @param s3 the {@link S3Client} instance to use for making the request
 * @param bucketName the name of the S3 bucket for which to retrieve the
policy
 * @return the policy text for the specified bucket, or an empty string if an
error occurs
 */
public static String getPolicy(S3Client s3, String bucketName) {
    String policyText;
    System.out.format("Getting policy for bucket: \"%s\"\n\n", bucketName);
    GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    try {
        GetBucketPolicyResponse policyRes = s3.getBucketPolicy(policyReq);
        policyText = policyRes.policy();
        return policyText;

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }

    return "";
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicy](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez la politique de compartiment.

```
import {
  GetBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Logs the policy for a specified bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const { Policy } = await client.send(
      new GetBucketPolicyCommand({
        Bucket: bucketName,
      }),
    );
    console.log(`Policy for "${bucketName}":\n${Policy}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while getting policy from ${bucketName}. The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
```

```
        `Error from S3 while getting policy from ${bucketName}.  ${caught.name}:  
        ${caught.message}` ,  
        );  
    } else {  
        throw caught;  
    }  
}  
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicy](#) à la section Référence des AWS SDK pour JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getPolicy(bucketName: String): String? {  
    println("Getting policy for bucket $bucketName")  
  
    val request =  
        GetBucketPolicyRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        val policyRes = s3.getBucketPolicy(request)  
        return policyRes.policy  
    }  
}
```

- Pour plus de détails sur l'API, consultez [GetBucketPolicy](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande génère la stratégie de compartiment associée au compartiment S3 donné.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicy](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande génère la stratégie de compartiment associée au compartiment S3 donné.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicy](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:  
    """Encapsulates S3 bucket actions."""
```

```
def __init__(self, bucket):
    """
    :param bucket: A Boto3 Bucket resource. This is a high-level resource in
    Boto3
                    that wraps bucket actions in a class-like structure.
    """
    self.bucket = bucket
    self.name = bucket.name

def get_policy(self):
    """
    Get the security policy of the bucket.

    :return: The security policy of the specified bucket, in JSON format.
    """
    try:
        policy = self.bucket.Policy()
        logger.info(
            "Got policy %s for bucket '%s'.", policy.policy, self.bucket.name
        )
    except ClientError:
        logger.exception("Couldn't get policy for bucket '%s'.",
            self.bucket.name)
        raise
    else:
        return json.loads(policy.policy)
```

- Pour plus de détails sur l'API, consultez [GetBucketPolicy](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  # configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
    #{e.message}"
    nil
  end
end

end
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicy](#) à la section Référence des AWS SDK pour Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.
    oo_result = lo_s3->getbucketpolicy(
testing purposes. " " oo_result is returned for
```

```
        iv_bucket = iv_bucket_name ).
    DATA(lv_policy) = oo_result->get_policy( ).
    MESSAGE 'Retrieved bucket policy.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicy](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetBucketPolicyStatus** avec une CLI

Les exemples de code suivants illustrent comment utiliser GetBucketPolicyStatus.

CLI

AWS CLI

Pour extraire le statut de la stratégie indiquant si un compartiment est public

L'exemple get-bucket-policy-status suivant extrait le statut de la stratégie du compartiment amzn-s3-demo-bucket.

```
aws s3api get-bucket-policy-status \
    --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "PolicyStatus": {
    "IsPublic": false
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicyStatus](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie le statut de la stratégie pour le compartiment S3 donné, indiquant si ce compartiment est public.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicyStatus](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie le statut de la stratégie pour le compartiment S3 donné, indiquant si ce compartiment est public.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicyStatus](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetBucketReplication** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser GetBucketReplication.

CLI

AWS CLI

La commande suivante extrait la configuration de réplication d'un compartiment nommé amzn-s3-demo-bucket :

```
aws s3api get-bucket-replication --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "ReplicationConfiguration": {
    "Rules": [
      {
        "Status": "Enabled",
        "Prefix": "",
        "Destination": {
          "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket-backup",
          "StorageClass": "STANDARD"
        },
        "ID": "ZmUwNzE4ZmQ4tMjVhOS00MTlkLOGI4NDkzZTIWJjNTUtYTA1"
      }
    ],
    "Role": "arn:aws:iam::123456789012:role/s3-replication-role"
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketReplication](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Retrieves the replication details for the specified S3 bucket.
 *
 * @param s3Client      the S3 client used to interact with the S3
service
 * @param sourceBucketName the name of the S3 bucket to retrieve the
replication details for
 *
 * @throws S3Exception if there is an error retrieving the replication
details
 */
```

```
public static void getReplicationDetails(S3Client s3Client, String
sourceBucketName) {
    GetBucketReplicationRequest getRequest =
    GetBucketReplicationRequest.builder()
        .bucket(sourceBucketName)
        .build();

    try {
        ReplicationConfiguration replicationConfig =
        s3Client.getBucketReplication(getRequest).replicationConfiguration();
        ReplicationRule rule = replicationConfig.rules().get(0);
        System.out.println("Retrieved destination bucket: " +
        rule.destination().bucket());
        System.out.println("Retrieved priority: " + rule.priority());
        System.out.println("Retrieved source-bucket replication rule status:
" + rule.status());

    } catch (S3Exception e) {
        System.err.println("Failed to retrieve replication details: " +
        e.awsErrorDetails().errorMessage());
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketReplication](#) à la section Référence des AWS SDK for Java 2.x API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : renvoie les informations de configuration de réplication définies sur le compartiment nommé « amzn-s3-demo-bucket ».

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketReplication](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : renvoie les informations de configuration de réplication définies sur le compartiment nommé « amzn-s3-demo-bucket ».

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketReplication](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetBucketRequestPayment** avec une CLI

Les exemples de code suivants illustrent comment utiliser GetBucketRequestPayment.

CLI

AWS CLI

Pour extraire la configuration Paiement par le demandeur d'un compartiment

L'exemple get-bucket-request-payment suivant extrait la configuration « Paiement par le demandeur » du compartiment spécifié.

```
aws s3api get-bucket-request-payment \  
  --bucket amzn-s3-demo-bucket
```

Sortie :

```
{  
  "Payer": "BucketOwner"  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketRequestPayment](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : renvoie la configuration de paiement des demandes pour le compartiment nommé « amzn-s3-demo-bucket ». Par défaut, le propriétaire du compartiment paie les téléchargements effectués depuis le compartiment.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketRequestPayment](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : renvoie la configuration de paiement des demandes pour le compartiment nommé « amzn-s3-demo-bucket ». Par défaut, le propriétaire du compartiment paie les téléchargements effectués depuis le compartiment.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketRequestPayment](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetBucketTagging** avec une CLI

Les exemples de code suivants illustrent comment utiliser `GetBucketTagging`.

CLI

AWS CLI

La commande suivante extrait la configuration de balisage d'un compartiment nommé amzn-s3-demo-bucket :

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "TagSet": [
    {
      "Value": "marketing",
      "Key": "organization"
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketTagging](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie toutes les balises associées au compartiment donné.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketTagging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie toutes les balises associées au compartiment donné.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketTagging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetBucketVersioning** avec une CLI

Les exemples de code suivants illustrent comment utiliser `GetBucketVersioning`.

CLI

AWS CLI

La commande suivante extrait la configuration de gestion des versions d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

Sortie :

```
{  
  "Status": "Enabled"  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketVersioning](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie le statut de la gestion des versions par rapport au compartiment donné.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketVersioning](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie le statut de la gestion des versions par rapport au compartiment donné.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketVersioning](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetBucketWebsite** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetBucketWebsite`.

.NET

SDK pour .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get the website configuration.
GetBucketWebsiteRequest getRequest = new
GetBucketWebsiteRequest()
{
    BucketName = bucketName,
};
GetBucketWebsiteResponse getResponse = await
client.GetBucketWebsiteAsync(getRequest);
Console.WriteLine($"Index document:
{getResponse.WebsiteConfiguration.IndexDocumentSuffix}");
Console.WriteLine($"Error document:
{getResponse.WebsiteConfiguration.ErrorDocument}");
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketWebsite](#) à la section Référence des AWS SDK pour .NET API.

C++

SDK pour C++

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::S3::S3ClientConfiguration
                                   &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
        s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                   << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult =
            outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"
                  << std::endl
                  << "Index page : "
                  << websiteResult.GetIndexDocument().GetSuffix()
                  << std::endl
                  << "Error page: "
                  << websiteResult.GetErrorDocument().GetKey()
                  << std::endl;
    }
}
```

```
    return outcome.IsSuccess();  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketWebsite](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante extrait la configuration de site web statique d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

Sortie :

```
{  
  "IndexDocument": {  
    "Suffix": "index.html"  
  },  
  "ErrorDocument": {  
    "Key": "error.html"  
  }  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketWebsite](#) à la section Référence des AWS CLI commandes.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez la configuration du site web.

```
import {
  GetBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the website configuration for a bucket.
 * @param {{ bucketName }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    console.log(
      `Your bucket is set up to host a website with the following configuration:
\n${JSON.stringify(response, null, 2)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchWebsiteConfiguration"
    ) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}. The
bucket isn't configured as a website.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting website configuration for ${bucketName}.
${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketWebsite](#) à la section Référence des AWS SDK pour JavaScript API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie les détails des configurations de site Web statiques du compartiment S3 donné.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketWebsite](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie les détails des configurations de site Web statiques du compartiment S3 donné.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketWebsite](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetObject** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetObject`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Principes de base](#)

- [Récupérer un objet d'un compartiment s'il a été modifié](#)
- [Obtenir un objet depuis un point Multi-Region d'accès](#)
- [Démarrer avec le chiffrement](#)
- [Débuter avec le stockage d'objets](#)
- [Réalisation de demandes conditionnelles](#)
- [Suivi des chargements et des téléchargements](#)

.NET

SDK pour .NET (v4)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Shows how to download an object from an Amazon S3 bucket to the
/// local computer.
/// </summary>
/// <param name="bucketName">The name of the bucket where the object is
/// currently stored.</param>
/// <param name="objectName">The name of the object to download.</param>
/// <param name="filePath">The path, including filename, where the
/// downloaded object will be stored.</param>
/// <returns>A boolean value indicating the success or failure of the
/// download process.</returns>
public async Task<bool> DownloadObjectFromBucketAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    var request = new GetObjectRequest
    {
        BucketName = bucketName,
        Key = objectName,
    };
};
```

```
using GetObjectResponse response = await
    _amazonS3.GetObjectAsync(request);

try
{
    // Save object to local file
    await response.WriteResponseStreamToFileAsync($"{filePath}\
{objectName}", true, CancellationToken.None);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error saving {objectName}: {ex.Message}");
    return false;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS SDK pour .NET API.

SDK pour .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez un objet à l'aide d'une demande conditionnelle.

```
/// <summary>
/// Retrieves an object from Amazon S3 with a conditional request.
/// </summary>
/// <param name="objectKey">The key of the object to retrieve.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="conditionType">The type of condition: 'IfMatch',
'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
```

```
    /// <param name="etagConditionalValue">The value to use for the condition for
    etags.</param>
    /// <returns>True if the conditional read is successful, False otherwise.</
returns>
    public async Task<bool> GetObjectConditional(string objectKey, string
sourceBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
    {
        try
        {
            var getObjectRequest = new GetObjectRequest
            {
                BucketName = sourceBucket,
                Key = objectKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    getObjectRequest.EtagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    getObjectRequest.EtagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                default:
                    throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
            }

            var response = await _amazonS3.GetObjectAsync(getObjectRequest);
            var sampleBytes = new byte[20];
            await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
            _logger.LogInformation($"Conditional read
successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
```

```

        return true;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional read failed: Precondition failed");
        }
        else if (e.ErrorCode == "NotModified")
        {
            _logger.LogError("Conditional read failed: Object not modified");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS SDK pour .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {

```



```

printf "%s\n" "$*" 1>&2
}

#####
# function download_object_from_bucket
#
# This function downloads an object in a bucket to a file.
#
# Parameters:
#     $1 - The name of the bucket to download the object from.
#     $2 - The path and file name to store the downloaded bucket.
#     $3 - The key (name) of the object in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function download_object_from_bucket() {
    local bucket_name=$1
    local destination_file_name=$2
    local object_name=$3
    local response

    response=$(aws s3api get-object \
        --bucket "$bucket_name" \
        --key "$object_name" \
        "$destination_file_name")

    # shellcheck disable=SC2181
    if [[ ${?} -ne 0 ]]; then
        errecho "ERROR: AWS reports put-object operation failed.\n$response"
        return 1
    fi
}

```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
            << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

L'exemple suivant utilise la commande `get-object` pour télécharger un objet à partir d'Amazon S3 :

```
aws s3api get-object --bucket text-content --key dir/  
my_images.tar.bz2 my_images.tar.bz2
```

Notez que le paramètre `outfile` est spécifié sans nom d'option tel que « `--outfile` ». Le nom du fichier de sortie doit être le dernier paramètre de la commande.

L'exemple ci-dessous illustre l'utilisation de `--range` pour télécharger une plage d'octets spécifique d'un objet. Notez que les plages d'octets doivent comporter le préfixe « `bytes=` » :

```
aws s3api get-object --bucket text-content --key dir/my_data --  
range bytes=8888-9999 my_data_range
```

Pour plus d'informations sur l'extraction d'objets, consultez [Obtention d'objets](#) dans le Guide du développeur Amazon S3.

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"
```

```
"io"
"log"
"os"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
                objectKey, bucketName)
            err = noKey
        } else {
            log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
                objectKey, err)
        }
        return err
    }
    defer result.Body.Close()
```

```
file, err := os.Create(fileName)
if err != nil {
    log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
    return err
}
defer file.Close()
body, err := io.ReadAll(result.Body)
if err != nil {
    log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
err)
}
_, err = file.Write(body)
return err
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Lisez des données sous forme de tableau d'octets en utilisant un [S3Client](#).

```
/**
 * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
 * and writes them to a local file.
 *
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName    the key (or name) of the S3 object to retrieve
 * @param path       the local file path where the object's bytes will be
 * written
```

```

    * @return a {@link CompletableFuture} that completes when the object bytes
    have been written to the local file
    */
    public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
    keyName, String path) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        CompletableFuture<ResponseBytes<GetObjectResponse>> response =
        getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
        return response.thenAccept(objectBytes -> {
            try {
                byte[] data = objectBytes.asByteArray();
                Path filePath = Paths.get(path);
                Files.write(filePath, data);
                logger.info("Successfully obtained bytes from an S3 object");
            } catch (IOException ex) {
                throw new RuntimeException("Failed to write data to file", ex);
            }
        }).whenComplete((resp, ex) -> {
            if (ex != null) {
                throw new RuntimeException("Failed to get object bytes from S3",
ex);
            }
        });
    }
}

```

Utilisez un [S3TransferManager](#) pour [télécharger un objet](#) d'un compartiment S3 vers un fichier local. Consultez le [fichier complet](#) et le [test](#).

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

```

```

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String
bucketName,

                                String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFilePath))
            .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }

```

Lisez les étiquettes qui appartiennent à un objet à l'aide d'un [S3Client](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingRequest;
import software.amazon.awssdk.services.s3.model.GetObjectTaggingResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Tag;

import java.util.List;

/**
 * Before running this Java V2 code example, set up your development

```

```

    * environment, including your credentials.
    * <p>
    * For more information, see the following documentation topic:
    * <p>
    * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
    */

public class GetObjectTags {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listTags(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Lists the tags associated with an Amazon S3 object.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket that contains the object
     * @param keyName the key (name) of the S3 object
     */
    public static void listTags(S3Client s3, String bucketName, String keyName) {

```



```
    try {
        GetObjectTaggingRequest getTaggingRequest = GetObjectTaggingRequest
            .builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        GetObjectTaggingResponse tags =
s3.getObjectTagging(getTaggingRequest);
        List<Tag> tagSet = tags.tagSet();
        for (Tag tag : tagSet) {
            System.out.println(tag.key());
            System.out.println(tag.value());
        }

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

Obtenez une URL pour un objet en utilisant un [S3Client](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetUrlRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URL;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
```

```
public class GetObjectUrl {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
            <bucketName> <keyName>\s

        Where:
            bucketName - The Amazon S3 bucket name.
            keyName - A key name that represents the object.\s
        """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getUrl(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Retrieves the URL for a specific object in an Amazon S3 bucket.
     *
     * @param s3 the S3Client object used to interact with the Amazon S3 service
     * @param bucketName the name of the S3 bucket where the object is stored
     * @param keyName the name of the object for which the URL should be
     retrieved
     * @throws S3Exception if there is an error retrieving the URL for the
     specified object
     */
    public static void getUrl(S3Client s3, String bucketName, String keyName) {
        try {
            GetUrlRequest request = GetUrlRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();
```

```

        URL url = s3.utilities().getUrl(request);
        System.out.println("The URL for " + keyName + " is " + url);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

```

Obtenez un objet en utilisant l'objet client S3Presigner via un [S3Client](#).

```

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class GetObjectPresignedUrl {
    public static void main(String[] args) {
        final String USAGE = ""

```

```

Usage:
    <bucketName> <keyName>\s

Where:
    bucketName - The Amazon S3 bucket name.\s
    keyName - A key name that represents a text file.\s
""";

if (args.length != 2) {
    System.out.println(USAGE);
    System.exit(1);
}

String bucketName = args[0];
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Presigner presigner = S3Presigner.builder()
    .region(region)
    .build();

getPresignedUrl(presigner, bucketName, keyName);
presigner.close();
}

/**
 * Generates a pre-signed URL for an Amazon S3 object.
 *
 * @param presigner The {@link S3Presigner} instance to use for generating
the pre-signed URL.
 * @param bucketName The name of the Amazon S3 bucket where the object is
stored.
 * @param keyName The key name (file name) of the object in the Amazon S3
bucket.
 *
 * @throws S3Exception If there is an error interacting with the Amazon S3
service.
 * @throws IOException If there is an error opening the HTTP connection or
reading/writing the request/response.
 */
public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
    try {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()

```

```
        .bucket(bucketName)
        .key(keyName)
        .build();

    GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
        .signatureDuration(Duration.ofMinutes(60))
        .getObjectRequest(getObjectRequest)
        .build();

    PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
    String theUrl = presignedGetObjectRequest.url().toString();
    System.out.println("Presigned URL: " + theUrl);
    HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
    presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
        values.forEach(value -> {
            connection.addRequestProperty(header, value);
        });
    });

    // Send any request payload that the service needs (not needed when
    // isBrowserExecutable is true).
    if (presignedGetObjectRequest.signedPayload().isPresent()) {
        connection.setDoOutput(true);

        try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
            OutputStream httpOutputStream =
connection.getOutputStream()) {
            IoUtils.copy(signedPayload, httpOutputStream);
        }
    }

    // Download the result of executing the request.
    try (InputStream content = connection.getInputStream()) {
        System.out.println("Service returned response: ");
        IoUtils.copy(content, System.out);
    }

} catch (S3Exception | IOException e) {
    e.printStackTrace();
}
```

```
    }  
  }  
}
```

Obtenez un objet en utilisant un `ResponseTransformer` objet et [S3Client](#).

```
import software.amazon.awssdk.core.ResponseBytes;  
import software.amazon.awssdk.core.sync.ResponseTransformer;  
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.GetObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.GetObjectResponse;  
  
import java.io.File;  
import java.io.FileOutputStream;  
import java.io.IOException;  
import java.io.OutputStream;  
  
/**  
 * Before running this Java V2 code example, set up your development  
 * environment, including your credentials.  
 * <p>  
 * For more information, see the following documentation topic:  
 * <p>  
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html  
 */  
  
public class GetObjectData {  
    public static void main(String[] args) {  
        final String usage = ""  
  
            Usage:  
                <bucketName> <keyName> <path>  
  
            Where:  
                bucketName - The Amazon S3 bucket name.\s  
                keyName - The key name.\s  
                path - The path where the file is written to.\s  
            "";  
    }  
}
```

```
        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String path = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        getObjectBytes(s3, bucketName, keyName, path);
        s3.close();
    }

    /**
     * Retrieves the bytes of an object stored in an Amazon S3 bucket and saves
     * them to a local file.
     *
     * @param s3 The S3Client instance used to interact with the Amazon S3
     * service.
     * @param bucketName The name of the S3 bucket where the object is stored.
     * @param keyName The key (or name) of the S3 object.
     * @param path The local file path where the object's bytes will be saved.
     * @throws IOException If an I/O error occurs while writing the bytes to the
     * local file.
     * @throws S3Exception If an error occurs while retrieving the object from
     * the S3 bucket.
     */
    public static void getObjectBytes(S3Client s3, String bucketName, String
    keyName, String path) {
        try {
            GetObjectRequest objectRequest = GetObjectRequest
                .builder()
                .key(keyName)
                .bucket(bucketName)
                .build();

            ResponseBytes<GetObjectResponse> objectBytes =
                s3.getObject(objectRequest, ResponseTransformer.toBytes());
            byte[] data = objectBytes.asByteArray();
        }
```

```
        // Write the data to a local file.
        File myFile = new File(path);
        OutputStream os = new FileOutputStream(myFile);
        os.write(data);
        System.out.println("Successfully obtained bytes from an S3 object");
        os.close();

    } catch (IOException ex) {
        ex.printStackTrace();
    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Téléchargez l'objet.

```
import {
    GetObjectCommand,
    NoSuchKey,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
```



```

* @param {{ bucketName: string, key: string }}
*/
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log(str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

```

Téléchargez l'objet à condition que son ETag corresponde à celle fournie.

```

import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

```

```
/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";
```

```
const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

Téléchargez l'objet à condition que son ETag ne corresponde pas à celle fournie.

```
import {
  GetObjectCommand,
  NoSuchKey,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
```

```
*/
export const main = async ({ bucketName, key, eTag }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectCommand({
        Bucket: bucketName,
        Key: key,
        IfNoneMatch: eTag,
      }),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
```

```

        required: true,
    },
    key: {
        type: "string",
        required: true,
    },
    eTag: {
        type: "string",
        required: true,
    },
    },
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}

```

Téléchargez l'objet à condition qu'il ait été créé ou modifié dans un laps de temps donné.

```

import {
    GetObjectCommand,
    NoSuchKey,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
    const client = new S3Client({});
    const date = new Date();

```

```
date.setDate(date.getDate() - 1);
try {
  const response = await client.send(
    new GetObjectCommand({
      Bucket: bucketName,
      Key: key,
      IfModifiedSince: date,
    }),
  );
  // The Body object also has 'transformToByteArray' and 'transformToWebStream'
  methods.
  const str = await response.Body.transformToString();
  console.log("Success. Here is text of the file:", str);
} catch (caught) {
  if (caught instanceof NoSuchKey) {
    console.error(
      `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
```

```

        type: "string",
        required: true,
    },
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}

```

Téléchargez l'objet à condition qu'il n'ait pas été créé ou modifié dans un laps de temps donné.

```

import {
    GetObjectCommand,
    NoSuchKey,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
    const client = new S3Client({});
    const date = new Date();
    date.setDate(date.getDate() - 1);
    try {
        const response = await client.send(
            new GetObjectCommand({
                Bucket: bucketName,
                Key: key,
                IfUnmodifiedSince: date,
            })
        );
    } catch (error) {
        if (error instanceof S3ServiceException) {
            if (error.name === 'NoSuchKey') {
                // Object not found
            }
        }
    }
};

```

```
    }),
  );
  // The Body object also has 'transformToByteArray' and 'transformToWebStream'
  methods.
  const str = await response.Body.transformToString();
  console.log("Success. Here is text of the file:", str);
} catch (caught) {
  if (caught instanceof NoSuchKey) {
    console.error(
      `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
```



```
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS SDK pour JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getObjectBytes(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
        }
    }
}
```

```
        println("Successfully read $keyName from $bucketName")
    }
}
}
```

- Pour plus de détails sur l'API, consultez [GetObject](#) la section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez un objet.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $file = $this->s3client->getObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
    ]);
    $body = $file->get('Body');
    $body->rewind();
    echo "Downloaded the file and it begins with: {$body->read(26)}.\n";
} catch (Exception $exception) {
    echo "Failed to download $fileName from $this->bucketName with error:
" . $exception->getMessage();
    exit("Please fix error with file downloading before continuing.");
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS SDK pour PHP API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : Cette commande récupère l'élément « sample.txt » du bucket « amzn-s3-demo-bucket » et l'enregistre dans un fichier nommé « local-sample.txt » à l'emplacement actuel. Il n'est pas nécessaire que le fichier « local-sample.txt » existe pour que cette commande soit appelée.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Exemple 2 : Cette commande extrait le répertoire virtuel « DIR » du bucket « amzn-s3-demo-bucket » et l'enregistre dans un dossier nommé « » à l'emplacement actuel. Local-DIR Il n'est pas Local-DIR nécessaire que le dossier « » existe pour que cette commande soit appelée.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Exemple 3 : télécharge tous les objets dont les clés se terminent par « .json » à partir des compartiments dont le nom contient « config » vers les fichiers du dossier spécifié. Les clés d'objet sont utilisées pour définir les noms de fichiers.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : Cette commande récupère l'élément « sample.txt » du bucket « amzn-s3-demo-bucket » et l'enregistre dans un fichier nommé « local-sample.txt » à l'emplacement actuel. Il n'est pas nécessaire que le fichier « local-sample.txt » existe pour que cette commande soit appelée.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Exemple 2 : Cette commande extrait le répertoire virtuel « DIR » du bucket « amzn-s3-demo-bucket » et l'enregistre dans un dossier nommé « » à l'emplacement actuel. Local-DIR Il n'est pas Local-DIR nécessaire que le dossier « » existe pour que cette commande soit appelée.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Exemple 3 : télécharge tous les objets dont les clés se terminent par « .json » à partir des compartiments dont le nom contient « config » vers les fichiers du dossier spécifié. Les clés d'objet sont utilisées pour définir les noms de fichiers.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

Exemple 4 : télécharge un seul objet à l'aide du téléchargement parallèle en plusieurs parties pour améliorer les performances sur les fichiers volumineux.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt -UseMultipartDownload
```

Exemple 5 : télécharge un objet volumineux en mode RANGE avec une taille de pièce personnalisée de 64 Mo. Le mode RANGE utilise des requêtes HTTP par plage d'octets et fonctionne pour tous les objets, quelle que soit la manière dont ils ont été téléchargés.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Exemple 6 : télécharge tous les objets avec le préfixe clé « data/ » dans un dossier local en utilisant le téléchargement en plusieurs parties avec des téléchargements de fichiers simultanés.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Exemple 7 : télécharge un seul objet à l'aide d'un téléchargement en plusieurs parties avec 20 connexions simultanées pour un débit maximal sur les réseaux à bande passante élevée.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-
large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Exemple 8 : télécharge tous les objets avec le préfixe clé « data/ » dans un dossier local à l'aide du téléchargement en plusieurs parties, en continuant à corriger les défaillances individuelles des fichiers au lieu de les abandonner.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-
data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get(self):
        """
```

```

Gets the object.

:return: The object data in bytes.
"""
try:
    body = self.object.get()["Body"].read()
    logger.info(
        "Got object '%s' from bucket '%s'.",
        self.object.key,
        self.object.bucket_name,
    )
except ClientError:
    logger.exception(
        "Couldn't get object '%s' from bucket '%s'.",
        self.object.key,
        self.object.bucket_name,
    )
    raise
else:
    return body

```

Obtenez un objet à l'aide d'une demande conditionnelle.

```

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def get_object_conditional(
        self,

```

```

        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
        'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
                Key=object_key,
                **{condition_type: condition_value},
            )
            sample_bytes = response["Body"].read(20)
            print(
                f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
            )
            print(f"\t{sample_bytes}")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional read failed: Precondition failed")
            elif error_code == "304": # Not modified error code.
                print("\tConditional read failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise

```

- Pour plus de détails sur l'API, consultez [GetObject](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez un objet.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is
  # downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
  # successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

  wrapper = ObjectGetWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
end
```



```

obj_data = wrapper.get_object(target_path)
return unless obj_data

puts "Object #{object_key} (#{obj_data.content_length} bytes) downloaded to
#{target_path}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Obtenez un objet et signalez son état de chiffrement côté serveur.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
  # successful; otherwise nil.
  def object
    @object.get
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
    end
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
    object_key))
  obj_data = wrapper.get_object
  return unless obj_data
end

```

```

    encryption = obj_data.server_side_encryption.nil? ? 'no' :
    obj_data.server_side_encryption
    puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS SDK pour Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

async fn get_object(client: Client, opt: Opt) -> Result<usize, S3ExampleError> {
    trace!("bucket:      {}", opt.bucket);
    trace!("object:      {}", opt.object);
    trace!("destination: {}", opt.destination.display());

    let mut file = File::create(opt.destination.clone()).map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to initialize file for saving S3 download: {err:?}")
        ))
    })?;

    let mut object = client
        .get_object()
        .bucket(opt.bucket)
        .key(opt.object)
        .send()
        .await?;

    let mut byte_count = 0_usize;

```

```

    while let Some(bytes) = object.body.try_next().await.map_err(|err| {
        S3ExampleError::new(format!("Failed to read from S3 download stream:
{err:?}"))
    })? {
        let bytes_len = bytes.len();
        file.write_all(&bytes).map_err(|err| {
            S3ExampleError::new(format!(
                "Failed to write from S3 download stream to local file: {err:?}"
            ))
        })?;
        trace!("Intermediate write of {bytes_len}");
        byte_count += bytes_len;
    }

    Ok(byte_count)
}

```

- Pour plus de détails sur l'API, voir [GetObject](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_s3->getobject(          " oo_result is returned for
testing purposes. "
        iv_bucket = iv_bucket_name
        iv_key = iv_object_key ).
    DATA(lv_object_data) = oo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.

```

```
MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Swift

Kit SDK pour Swift

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import AWSS3  
  
public func downloadFile(bucket: String, key: String, to: String) async  
throws {  
    let fileUrl = URL(fileURLWithPath: to).appendingPathComponent(key)  
  
    let input = GetObjectInput(  
        bucket: bucket,  
        key: key  
    )  
    do {  
        let output = try await client.getObject(input: input)  
  
        guard let body = output.body else {  
            throw HandlerError.getObjectBody("GetObjectInput missing body.")  
        }  
  
        guard let data = try await body.readData() else {  
            throw HandlerError.readGetObjectBody("GetObjectInput unable to  
read data.")  
        }  
  
        try data.write(to: fileUrl)  
    }  
}
```

```
        catch {
            print("ERROR: ", dump(error, name: "Downloading a file. "))
            throw error
        }
    }
}
```

```
import AWSS3

public func readFile(bucket: String, key: String) async throws -> Data {
    let input = GetObjectInput(
        bucket: bucket,
        key: key
    )
    do {
        let output = try await client.getObject(input: input)

        guard let body = output.body else {
            throw HandlerError.getObjectBody("GetObjectInput missing body.")
        }

        guard let data = try await body.readData() else {
            throw HandlerError.readGetObjectBody("GetObjectInput unable to
read data.")
        }

        return data
    }
    catch {
        print("ERROR: ", dump(error, name: "Reading a file. "))
        throw error
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section AWS SDK pour la référence de l'API Swift.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetObjectAcl** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetObjectAcl`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Gérer les listes de contrôle d'accès \(ACL\)](#)

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
        s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
                    << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();
    }
```

```

        for (auto it = grants.begin(); it != grants.end(); it++) {
            std::cout << "For object " << objectKey << ": "
                << std::endl << std::endl;

            Aws::S3::Model::Grant grant = *it;
            Aws::S3::Model::Grantee grantee = grant.GetGrantee();

            if (grantee.TypeHasBeenSet()) {
                std::cout << "Type:          "
                    << getGranteeTypeString(grantee.GetType()) <<
std::endl;
            }

            if (grantee.DisplayNameHasBeenSet()) {
                std::cout << "Display name:  "
                    << grantee.GetDisplayName() << std::endl;
            }

            if (grantee.EmailAddressHasBeenSet()) {
                std::cout << "Email address: "
                    << grantee.GetEmailAddress() << std::endl;
            }

            if (grantee.IDHasBeenSet()) {
                std::cout << "ID:           "
                    << grantee.GetID() << std::endl;
            }

            if (grantee.URIHasBeenSet()) {
                std::cout << "URI:          "
                    << grantee.GetURI() << std::endl;
            }

            std::cout << "Permission:    " <<
                getPermissionString(grant.GetPermission()) <<
                std::endl << std::endl;
        }
    }

    return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable
string.

```

```
/*!  
 \param type: Type enumeration.  
 \return String: Human-readable string  
*/  
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {  
    switch (type) {  
        case Aws::S3::Model::Type::AmazonCustomerByEmail:  
            return "Email address of an AWS account";  
        case Aws::S3::Model::Type::CanonicalUser:  
            return "Canonical user ID of an AWS account";  
        case Aws::S3::Model::Type::Group:  
            return "Predefined Amazon S3 group";  
        case Aws::S3::Model::Type::NOT_SET:  
            return "Not set";  
        default:  
            return "Type unknown";  
    }  
}  
  
//! Routine which converts a built-in type enumeration to a human-readable  
string.  
/*!  
 \param permission: Permission enumeration.  
 \return String: Human-readable string  
*/  
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {  
    switch (permission) {  
        case Aws::S3::Model::Permission::FULL_CONTROL:  
            return "Can read this object's data and its metadata, "  
                "and read/write this object's permissions";  
        case Aws::S3::Model::Permission::NOT_SET:  
            return "Permission not set";  
        case Aws::S3::Model::Permission::READ:  
            return "Can read this object's data and its metadata";  
        case Aws::S3::Model::Permission::READ_ACP:  
            return "Can read this object's permissions";  
            // case Aws::S3::Model::Permission::WRITE // Not applicable.  
        case Aws::S3::Model::Permission::WRITE_ACP:  
            return "Can write this object's permissions";  
        default:  
            return "Permission unknown";  
    }  
}
```


- Pour plus de détails sur l'API, reportez-vous [GetObjectAcl](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante extrait la liste de contrôle d'accès d'un objet dans un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key index.html
```

Sortie :

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    },
    {
      "Grantee": {
        "URI": "http://acs.amazonaws.com/groups/global/AllUsers"
      },
      "Permission": "READ"
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectAcl](#) à la section Référence des AWS CLI commandes.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun getBucketACL(
    objectKey: String,
    bucketName: String,
) {
    val request =
        GetObjectAclRequest {
            bucket = bucketName
            key = objectKey
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.getObjectAcl(request)
        response.grants?.forEach { grant ->
            println("Grant permission is ${grant.permission}")
        }
    }
}
```

- Pour plus de détails sur l'API, consultez [GetObjectAcl](#) la section AWS SDK pour la référence de l'API Kotlin.

PowerShell

Outils pour PowerShell V5

Exemple 1 : la commande obtient les détails du propriétaire de l'objet S3.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
*).Owner
```

Sortie :

```

DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544

```

- Pour plus de détails sur l'API, reportez-vous [GetObjectAcl](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                                that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def get_acl(self):
        """
        Gets the ACL of the object.

        :return: The ACL of the object.
        """
        try:
            acl = self.object.Acl()

```

```

        logger.info(
            "Got ACL for object %s owned by %s.",
            self.object.key,
            acl.owner["DisplayName"],
        )
    except ClientError:
        logger.exception("Couldn't get ACL for object %s.", self.object.key)
        raise
    else:
        return acl

```

- Pour plus de détails sur l'API, consultez [GetObjectAcl](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_s3->getobjectacl(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key ).
    MESSAGE 'Retrieved object ACL.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [GetObjectAcl](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetObjectAttributes** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetObjectAttributes`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Utilisation de l'intégrité des objets Amazon S3](#)

C++

SDK pour C++

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of
the object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
&key,
```

```

        AwsDoc::S3::HASH_METHOD hashMethod,
        Aws::String &hashData,
        std::vector<Aws::String> *partHashes,
        const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
        client.GetObjectAttributes(
            request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
            outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
        client.GetObjectAttributes(
            request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
            outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:
                    break; // MD5 is not supported.
#pragma clang diagnostic pop

```

```

        case AwsDoc::S3::SHA1:
            hashData = result.GetChecksum().GetChecksumSHA1();
            break;
        case AwsDoc::S3::SHA256:
            hashData = result.GetChecksum().GetChecksumSHA256();
            break;
        case AwsDoc::S3::CRC32:
            hashData = result.GetChecksum().GetChecksumCRC32();
            break;
        case AwsDoc::S3::CRC32C:
            hashData = result.GetChecksum().GetChecksumCRC32C();
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            return false;
    }
} else {
    std::cerr << "Error retrieving object checksum attributes." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

if (nullptr != partHashes) {
    attributes.clear();
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
    request.SetObjectAttributes(attributes);
    outcome = client.GetObjectAttributes(request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
        for (const Aws::S3::Model::ObjectPart &part: parts) {
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                    break;
                case AwsDoc::S3::MD5: // MD5 is not supported.
                    break;
                case AwsDoc::S3::SHA1:
                    partHashes->push_back(part.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:
                    partHashes->push_back(part.GetChecksumSHA256());

```

```

        break;
    case AwsDoc::S3::CRC32:
        partHashes->push_back(part.GetChecksumCRC32());
        break;
    case AwsDoc::S3::CRC32C:
        partHashes->push_back(part.GetChecksumCRC32C());
        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        return false;
    }
}
} else {
    std::cerr << "Error retrieving object attributes for object
parts." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
}
}
return true;
}

```

- Pour plus de détails sur l'API, reportez-vous [GetObjectAttributes](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour extraire les métadonnées d'un objet sans renvoyer ce dernier

L'exemple `get-object-attributes` suivant extrait les métadonnées de l'objet `doc1.rtf`.

```

aws s3api get-object-attributes \
  --bucket amzn-s3-demo-bucket \
  --key doc1.rtf \
  --object-attributes "StorageClass" "ETag" "ObjectSize"

```

Sortie :


```
{
  "LastModified": "2022-03-15T19:37:31+00:00",
  "VersionId": "IuCPjXTDzHNf1dAuitVBIKJpF2p1fg4P",
  "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",
  "StorageClass": "STANDARD",
  "ObjectSize": 405
}
```

Pour plus d'informations, consultez [GetObjectAttributes](#) le manuel de référence des API Amazon S3.

- Pour plus de détails sur l'API, reportez-vous [GetObjectAttributes](#) à la section Référence des AWS CLI commandes.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetObjectLegalHold** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetObjectLegalHold`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Verrouillage d'objets Amazon S3](#)

.NET

SDK pour .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Get the legal hold details for an S3 object.
```

```

    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"Object legal hold for {objectKey} in
{bucketName}: " +
                $"{response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Unable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLegalHold](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour extraire le statut de conservation légale d'un objet

L'exemple `get-object-legal-hold` suivant extrait le statut de conservation légale de l'objet spécifié.

```

aws s3api get-object-legal-hold \
    --bucket amzn-s3-demo-bucket-with-object-lock \

```

```
--key doc1.rtf
```

Sortie :

```
{
  "LegalHold": {
    "Status": "ON"
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLegalHold](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
```

```
S3Client  *s3.Client
S3Manager *manager.Uploader
}

// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
    var status *types.ObjectLockLegalHoldStatus
    input := &s3.GetObjectLegalHoldInput{
        Bucket:    aws.String(bucket),
        Key:        aws.String(key),
        VersionId: aws.String(versionId),
    }

    output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
    if err != nil {
        var noSuchKeyErr *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noSuchKeyErr) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noSuchKeyErr
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "NoSuchObjectLockConfiguration":
                log.Printf("Object %s does not have an object lock configuration.\n", key)
                err = nil
            case "InvalidRequest":
                log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
                err = nil
            }
        }
    } else {
        status = &output.LegalHold.Status
    }

    return status, err
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLegalHold](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
    }

    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLegalHold](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import {
  GetObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get an object's current legal hold status.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetObjectLegalHoldCommand({
        Bucket: bucketName,
        Key: key,
        // Optionally, you can provide additional parameters
        // ExpectedBucketOwner: "<account ID that is expected to own the
        bucket>",
        // VersionId: "<the specific version id of the object to check>",
      }),
    );
    console.log(`Legal Hold Status: ${response.LegalHold.Status}`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
```

```
        `Error from S3 while getting legal hold status for ${key} in
        ${bucketName}. The bucket doesn't exist.`
    );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting legal hold status for ${key} in
            ${bucketName} from ${bucketName}. ${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
        key: {
            type: "string",
            required: true,
        },
    };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLegalHold](#) à la section Référence des AWS SDK pour JavaScript API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Placez un objet en détention légale.

```
def get_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Get the legal hold status of a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to get the legal hold status of.
    """
    print()
    logger.info("Getting legal hold status of file [%s] in bucket [%s]", key,
                bucket)
    try:
        response = s3_client.get_object_legal_hold(Bucket=bucket, Key=key)
        legal_hold_status = response["LegalHold"]["Status"]
        logger.debug(
            "Legal hold status of file [%s] in bucket [%s] is [%s]",
            key,
            bucket,
            legal_hold_status,
        )
    except Exception as e:
        logger.error(
            "Failed to get legal hold status of file [%s] in bucket [%s]: %s",
```



```

        key,
        bucket,
        e,
    )

```

- Pour plus de détails sur l'API, consultez [GetObjectLegalHold](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_s3->getobjectlegalhold(          " oo_result is returned
for testing purposes. "
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key ).
    MESSAGE 'Retrieved object legal hold status.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLegalHold](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetObjectLockConfiguration** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetObjectLockConfiguration`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Verrouillage d'objets Amazon S3](#)

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Get the object lock configuration details for an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to get details.</param>
/// <returns>The bucket's object lock configuration details.</returns>
public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
{
    try
    {
        var request = new GetObjectLockConfigurationRequest()
        {
            BucketName = bucketName
        };

        var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
        Console.WriteLine($"{\tBucket object lock config for {bucketName} in
{bucketName}: " +
                        $"{\n\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
```

```
        $"\\n\\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

        return response.ObjectLockConfiguration;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}'");
        return new ObjectLockConfiguration();
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLockConfiguration](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour extraire la configuration de verrouillage d'objet d'un compartiment

L'exemple `get-object-lock-configuration` suivant extrait la configuration de verrouillage d'objet du compartiment spécifié.

```
aws s3api get-object-lock-configuration \
--bucket amzn-s3-demo-bucket-with-object-lock
```

Sortie :

```
{
  "ObjectLockConfiguration": {
    "ObjectLockEnabled": "Enabled",
    "Rule": {
      "DefaultRetention": {
        "Mode": "COMPLIANCE",
        "Days": 50
      }
    }
  }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLockConfiguration](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectLockConfiguration retrieves the object lock configuration for an S3  
bucket.
```

```
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
string) (*types.ObjectLockConfiguration, error) {
    var lockConfig *types.ObjectLockConfiguration
    input := &s3.GetObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
    }

    output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
"ObjectLockConfigurationNotFoundError" {
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    } else {
        lockConfig = output.ObjectLockConfiguration
    }

    return lockConfig, err
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLockConfiguration](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .build();

    GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName+": ");
    System.out.println("\tEnabled:
"+response.objectLockConfiguration().objectLockEnabled());
    System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLockConfiguration](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import {
    GetObjectLockConfigurationCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Gets the Object Lock configuration for a bucket.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
```

```
const client = new S3Client({});

try {
  const { ObjectLockConfiguration } = await client.send(
    new GetObjectLockConfigurationCommand({
      Bucket: bucketName,
      // Optionally, you can provide additional parameters
      // ExpectedBucketOwner: "<account ID that is expected to own the
bucket>",
    }),
  );
  console.log(
    `Object Lock Configuration:\n${JSON.stringify(ObjectLockConfiguration)}`,
  );
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while getting object lock configuration for ${bucketName}.
The bucket doesn't exist.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while getting object lock configuration for ${bucketName}.
${caught.name}: ${caught.message}`,
    );
  } else {
    throw caught;
  }
}

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
```

```
        type: "string",
        required: true,
    },
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLockConfiguration](#) à la section Référence des AWS SDK pour JavaScript API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie la valeur « Enabled » si la configuration du verrouillage d'objet est activée pour le compartiment S3 donné.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
ObjectLockConfiguration.ObjectLockEnabled
```

Sortie :

```
Value
-----
Enabled
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLockConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie la valeur « Enabled » si la configuration du verrouillage d'objet est activée pour le compartiment S3 donné.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Sortie :

```
Value  
-----  
Enabled
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLockConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez la configuration du verrouillage d'objet.

```
def is_object_lock_enabled(s3_client, bucket: str) -> bool:  
    """  
    Check if object lock is enabled for a bucket.  
  
    Args:  
        s3_client: Boto3 S3 client.  
        bucket: The name of the bucket to check.  
  
    Returns:  
        True if object lock is enabled, False otherwise.  
    """
```

```

try:
    response = s3_client.get_object_lock_configuration(Bucket=bucket)
    return (
        "ObjectLockConfiguration" in response
        and response["ObjectLockConfiguration"]["ObjectLockEnabled"] ==
"Enabled"
    )
except s3_client.exceptions.ClientError as e:
    if e.response["Error"]["Code"] == "ObjectLockConfigurationNotFoundError":
        return False
    else:
        raise

```

- Pour plus de détails sur l'API, consultez [GetObjectLockConfiguration](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_s3->getobjectlockconfiguration(          " oo_result is
returned for testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved object lock configuration.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [GetObjectLockConfiguration](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetObjectRetention** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `GetObjectRetention`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Verrouillage d'objets Amazon S3](#)

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Get the retention period for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object retention details.</returns>
public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };
    }
```

```
var response = await _amazonS3.GetObjectRetentionAsync(request);
Console.WriteLine($"\\tObject retention for {objectKey} in
{bucketName}: " +
                    $"\\n\\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
return response.Retention;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tUnable to fetch object lock retention:
'{ex.Message}'");
    return new ObjectLockRetention();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectRetention](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour extraire la configuration de rétention d'un objet

L'exemple `get-object-retention` suivant extrait la configuration de rétention de l'objet spécifié.

```
aws s3api get-object-retention \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf
```

Sortie :

```
{
  "Retention": {
    "Mode": "GOVERNANCE",
    "RetainUntilDate": "2025-01-01T00:00:00.000Z"
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectRetention](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectRetention retrieves the object retention configuration for an S3  
// object.  
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key  
    string) (*types.ObjectLockRetention, error) {
```

```
var retention *types.ObjectLockRetention
input := &s3.GetObjectRetentionInput{
    Bucket: aws.String(bucket),
    Key:    aws.String(key),
}

output, err := actor.S3Client.GetObjectRetention(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have locking enabled.", bucket)
            err = nil
        }
    }
} else {
    retention = output.Retention
}

return retention, err
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectRetention](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("Object retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectRetention](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import {
  GetObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Log the "RetainUntilDate" for an object in an S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});

  try {
    const { Retention } = await client.send(
      new GetObjectRetentionCommand({
        Bucket: bucketName,
        Key: key,
      }),
    );
    console.log(
      `${key} in ${bucketName} will be retained until ${Retention.RetainUntilDate}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchObjectLockConfiguration"
    ) {
      console.warn(
```



```
    `The object "${key}" in the bucket "${bucketName}" does not have an
    ObjectLock configuration.` ,
  );
} else if (caught instanceof S3ServiceException) {
  console.error(
    `Error from S3 while getting object retention settings for
    "${bucketName}". ${caught.name}: ${caught.message}`,
  );
} else {
  throw caught;
}
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
  };
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectRetention](#) à la section Référence des AWS SDK pour JavaScript API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : la commande renvoie le mode et la date de conservation de l'objet.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectRetention](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : la commande renvoie le mode et la date de conservation de l'objet.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectRetention](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetObjectTagging** avec une CLI

Les exemples de code suivants illustrent comment utiliser `GetObjectTagging`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Démarrer avec les étiquettes](#)

CLI

AWS CLI

Pour extraire les balises attachées à un objet

L'exemple `get-object-tagging` suivant extrait les valeurs de la clé spécifiée de l'objet spécifié.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc1.rtf
```

Sortie :

```
{  
  "TagSet": [  
    {  
      "Value": "confidential",  
      "Key": "designation"  
    }  
  ]  
}
```

L'exemple `get-object-tagging` suivant tente d'extraire les balises de l'objet `doc2.rtf`, qui ne possède pas de balise.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc2.rtf
```

Sortie :

```
{  
  "TagSet": []  
}
```

L'exemple `get-object-tagging` suivant extrait les balises de l'objet `doc3.rtf`, qui possède plusieurs balises.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc3.rtf
```

Sortie :

```
{  
  "TagSet": [  
    {  
      "Value": "confidential",  
      "Key": "designation"  
    },  
    {  
      "Value": "finance",  
      "Key": "department"  
    },  
    {  
      "Value": "payroll",  
      "Key": "team"  
    }  
  ]  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectTagging](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : l'exemple renvoie les balises associées à l'objet présent dans le compartiment S3 donné.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Key  Value  
---  -  
test value
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectTagging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : l'exemple renvoie les balises associées à l'objet présent dans le compartiment S3 donné.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Sortie :

```
Key  Value
---  -
test value
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectTagging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **GetPublicAccessBlock** avec une CLI

Les exemples de code suivants illustrent comment utiliser `GetPublicAccessBlock`.

CLI

AWS CLI

Pour définir ou modifier la configuration du blocage de l'accès public d'un compartiment

L'exemple `get-public-access-block` suivant affiche la configuration du blocage de l'accès public du compartiment spécifié.

```
aws s3api get-public-access-block \
  --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
```

```
"PublicAccessBlockConfiguration": {  
  "IgnorePublicAcls": true,  
  "BlockPublicPolicy": true,  
  "BlockPublicAcls": true,  
  "RestrictPublicBuckets": true  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetPublicAccessBlock](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : la commande renvoie la configuration du bloc d'accès public du compartiment S3 donné.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetPublicAccessBlock](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : la commande renvoie la configuration du bloc d'accès public du compartiment S3 donné.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [GetPublicAccessBlock](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **HeadBucket** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser HeadBucket.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# function bucket_exists
#
# This function checks to see if the specified bucket already exists.
#
# Parameters:
#     $1 - The name of the bucket to check.
#
# Returns:
#     0 - If the bucket already exists.
#     1 - If the bucket doesn't exist.
#####
function bucket_exists() {
    local bucket_name
    bucket_name=$1

    # Check whether the bucket already exists.
    # We suppress all output - we're interested only in the return code.

    if aws s3api head-bucket \
        --bucket "$bucket_name" \
        >/dev/null 2>&1; then
        return 0 # 0 in Bash script means true.
    else
        return 1 # 1 in Bash script means false.
    fi
}
```

- Pour plus de détails sur l'API, reportez-vous [HeadBucket](#) à la section Référence des AWS CLI commandes.

CLI

AWS CLI

La commande suivante vérifie l'accès à un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

Si le compartiment existe et que vous y avez accès, aucune sortie n'est renvoyée. Dans le cas contraire, un message d'erreur s'affiche. Par exemple :

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- Pour plus de détails sur l'API, reportez-vous [HeadBucket](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
```



```
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
    (bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
    exists := true
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
                err = nil
            default:
                log.Printf("Either you don't have access to bucket %v or another error
                occurred. "+
                    "Here's what happened: %v\n", bucketName, err)
            }
        }
    } else {
        log.Printf("Bucket %v exists and you already own it.", bucketName)
    }

    return exists, err
}
```

- Pour plus de détails sur l'API, reportez-vous [HeadBucket](#) à la section Référence des AWS SDK pour Go API.

PowerShell

Outils pour PowerShell V5

Exemple 1 : Cette commande renvoie le résultat avec le code d'état HTTP 200 OK pour le bucket existant lorsque l'utilisateur est autorisé à y accéder. BucketArn le paramètre n'est pris en charge que pour les compartiments de répertoire S3.

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

Sortie :

```
AccessPointAlias    : False
BucketArn           :
BucketLocationName  :
BucketLocationType  :
BucketRegion        : us-east-2
ResponseMetadata     : Amazon.Runtime.ResponseMetadata
ContentLength        : 0
HttpStatusCode       : OK
```

Exemple 2 : Cette commande génère une erreur avec le code d'état HTTP NotFound pour un bucket inexistant.

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

Sortie :

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status Code NotFound. No further error information was returned by the service.
```

Exemple 3 : cette commande génère une erreur avec le code de statut HTTP Interdit pour le compartiment existant auquel l'utilisateur n'est pas autorisé à accéder.

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

Sortie :

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status Code Forbidden. No further error information was returned by the service.
```

- Pour plus de détails sur l'API, reportez-vous [HeadBucket](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def exists(self):
        """
        Determine whether the bucket exists and you have access to it.

        :return: True when the bucket exists; otherwise, False.
        """
        try:
```

```

        self.bucket.meta.client.head_bucket(Bucket=self.bucket.name)
        logger.info("Bucket %s exists.", self.bucket.name)
        exists = True
    except ClientError:
        logger.warning(
            "Bucket %s doesn't exist or you don't have access to it.",
            self.bucket.name,
        )
        exists = False
    return exists

```

- Pour plus de détails sur l'API, consultez [HeadBucket](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_s3->headbucket(          " oo_result is returned for
testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Bucket exists and you have access to it.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [HeadBucket](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **HeadObject** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser HeadObject.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Vérifiez si un objet existe](#)
- [Débuter avec le stockage d'objets](#)

CLI

AWS CLI

La commande suivante extrait les métadonnées d'un objet dans un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

Sortie :

```
{
  "AcceptRanges": "bytes",
  "ContentType": "text/html",
  "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
  "ContentLength": 77,
  "VersionId": "null",
  "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
  "Metadata": {}
}
```

- Pour plus de détails sur l'API, reportez-vous [HeadObject](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Déterminez le type de contenu d'un objet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class GetObjectContentType {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name.\s
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
    }
}
```

```
    }

    String bucketName = args[0];
    String keyName = args[1];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    getContentType(s3, bucketName, keyName);
    s3.close();
}

/**
 * Retrieves the content type of an object stored in an Amazon S3 bucket.
 *
 * @param s3 an instance of the {@link S3Client} class, which is used to
interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket where the object is stored
 * @param keyName the key (file name) of the object in the S3 bucket
 */
public static void getContentType(S3Client s3, String bucketName, String
keyName) {
    try {
        HeadObjectRequest objectRequest = HeadObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        HeadObjectResponse objectHead = s3.headObject(objectRequest);
        String type = objectHead.contentType();
        System.out.println("The object content type is " + type);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}
```

Obtenez le statut de restauration d'un objet.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

public class GetObjectRestoreStatus {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName>\s

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - A key name that represents the object.\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        checkStatus(s3, bucketName, keyName);
        s3.close();
    }

    /**
     * Checks the restoration status of an Amazon S3 object.
     *
     * @param s3          an instance of the {@link S3Client} class used to
     *                    interact with the Amazon S3 service
     * @param bucketName the name of the Amazon S3 bucket where the object is
     *                    stored
     * @param keyName     the name of the Amazon S3 object to be checked
     */
}
```



```
    * @throws S3Exception if an error occurs while interacting with the Amazon
    S3 service
    */
    public static void checkStatus(S3Client s3, String bucketName, String
    keyName) {
        try {
            HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)
                .build();

            HeadObjectResponse response = s3.headObject(headObjectRequest);
            System.out.println("The Amazon S3 object restoration status is " +
            response.restore());

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [HeadObject](#) à la section Référence des AWS SDK for Java 2.x API.

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object
```

```

# @param object [Aws::S3::Object] An Amazon S3 object.
def initialize(object)
  @object = object
end

# Checks whether the object exists.
#
# @return [Boolean] True if the object exists; otherwise false.
def exists?
  @object.exists?
rescue Aws::Errors::ServiceError => e
  puts "Couldn't check existence of object
#{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"
  false
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  exists = wrapper.exists?

  puts "Object #{@object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Pour plus de détails sur l'API, reportez-vous [HeadObject](#) à la section Référence des AWS SDK pour Ruby API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **ListBucketAnalyticsConfigurations** avec une CLI

Les exemples de code suivants illustrent comment utiliser **ListBucketAnalyticsConfigurations**.

CLI

AWS CLI

Pour extraire une liste des configurations d'analyse d'un compartiment

L'exemple `list-bucket-analytics-configurations` suivant extrait une liste des configurations d'analyse du compartiment spécifié.

```
aws s3api list-bucket-analytics-configurations \
  --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "AnalyticsConfigurationList": [
    {
      "StorageClassAnalysis": {},
      "Id": "1"
    }
  ],
  "IsTruncated": false
}
```

- Pour plus de détails sur l'API, reportez-vous [ListBucketAnalyticsConfigurations](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie les cent premières configurations d'analyse du compartiment S3 donné.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [ListBucketAnalyticsConfigurations](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie les cent premières configurations d'analyse du compartiment S3 donné.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [ListBucketAnalyticsConfigurations](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **ListBucketInventoryConfigurations** avec une CLI

Les exemples de code suivants illustrent comment utiliser **ListBucketInventoryConfigurations**.

CLI

AWS CLI

Pour renvoyer une liste des configurations d'inventaire d'un compartiment

L'exemple `list-bucket-inventory-configurations` suivant répertorie les configurations d'inventaire du compartiment spécifié.

```
aws s3api list-bucket-inventory-configurations \
  --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "InventoryConfigurationList": [
    {
      "IsEnabled": true,
      "Destination": {
```

```
        "S3BucketDestination": {
            "Format": "ORC",
            "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
            "AccountId": "123456789012"
        },
        "IncludedObjectVersions": "Current",
        "Id": "1",
        "Schedule": {
            "Frequency": "Weekly"
        }
    },
    {
        "IsEnabled": true,
        "Destination": {
            "S3BucketDestination": {
                "Format": "CSV",
                "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
                "AccountId": "123456789012"
            }
        },
        "IncludedObjectVersions": "Current",
        "Id": "2",
        "Schedule": {
            "Frequency": "Daily"
        }
    }
],
    "IsTruncated": false
}
```

- Pour plus de détails sur l'API, reportez-vous [ListBucketInventoryConfigurations](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie les cent premières configurations d'inventaire du compartiment S3 donné.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [ListBucketInventoryConfigurations](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie les cent premières configurations d'inventaire du compartiment S3 donné.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Pour plus de détails sur l'API, reportez-vous [ListBucketInventoryConfigurations](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **ListBuckets** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser ListBuckets.

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
namespace ListBucketsExample
{
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.S3;
    using Amazon.S3.Model;

    /// <summary>
    /// This example uses the AWS SDK for .NET to list the Amazon Simple Storage
```

```

/// Service (Amazon S3) buckets belonging to the default account.
/// </summary>
public class ListBuckets
{
    private static IAmazonS3 _s3Client;

    /// <summary>
    /// Get a list of the buckets owned by the default user.
    /// </summary>
    /// <param name="client">An initialized Amazon S3 client object.</param>
    /// <returns>The response from the ListingBuckets call that contains a
    /// list of the buckets owned by the default user.</returns>
    public static async Task<ListBucketsResponse> GetBuckets(IAmazonS3
client)
    {
        return await client.ListBucketsAsync();
    }

    /// <summary>
    /// This method lists the name and creation date for the buckets in
    /// the passed List of S3 buckets.
    /// </summary>
    /// <param name="bucketList">A List of S3 bucket objects.</param>
    public static void DisplayBucketList(List<S3Bucket> bucketList)
    {
        bucketList
            .ForEach(b => Console.WriteLine($"Bucket name: {b.BucketName},
created on: {b.CreationDate}"));
    }

    public static async Task Main()
    {
        // The client uses the AWS Region of the default user.
        // If the Region where the buckets were created is different,
        // pass the Region to the client constructor. For example:
        // _s3Client = new AmazonS3Client(RegionEndpoint.USEast1);
        _s3Client = new AmazonS3Client();
        var response = await GetBuckets(_s3Client);
        DisplayBucketList(response.Buckets);
    }
}
}

```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK pour .NET API.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante utilise la commande `list-buckets` pour afficher les noms de tous vos compartiments Amazon S3 (dans toutes les régions) :

```
aws s3api list-buckets --query "Buckets[].Name"
```

L'option de requête filtre la sortie de `list-buckets` pour n'afficher que les noms des compartiments.

Pour plus d'informations sur les compartiments, consultez [Utilisation des compartiments Amazon S3](#) dans le Guide du développeur Amazon S3.

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
```

```
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
error) {
    var err error
    var output *s3.ListBucketsOutput
    var buckets []types.Bucket
    bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
&s3.ListBucketsInput{})
    for bucketPaginator.HasMorePages() {
        output, err = bucketPaginator.NextPage(ctx)
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
                fmt.Println("You don't have permission to list buckets for this account.")
                err = apiErr
            } else {
                log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
            }
            break
        } else {
            buckets = append(buckets, output.Buckets...)
        }
    }
    return buckets, err
}
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.paginators.ListBucketsIterable;
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class ListBuckets {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        listAllBuckets(s3);
    }

    /**
     * Lists all the S3 buckets available in the current AWS account.
     */
}
```

```

    * @param s3 The {@link S3Client} instance to use for interacting with the
    Amazon S3 service.
    */
    public static void listAllBuckets(S3Client s3) {
        ListBucketsIterable response = s3.listBucketsPaginator();
        response.buckets().forEach(bucket ->
            System.out.println("Bucket Name: " + bucket.name()));
    }

```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Listez les compartiments.

```

import {
    paginateListBuckets,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * List the Amazon S3 buckets in your account.
 */
export const main = async () => {
    const client = new S3Client({});
    /** @type {?import('@aws-sdk/client-s3').Owner} */
    let Owner = null;

    /** @type {import('@aws-sdk/client-s3').Bucket[]} */
    const Buckets = [];

```

```
try {
  const paginator = paginateListBuckets({ client }, {});

  for await (const page of paginator) {
    if (!Owner) {
      Owner = page.Owner;
    }

    Buckets.push(...page.Buckets);
  }

  console.log(
    `${Owner.DisplayName} owns ${Buckets.length} bucket${
      Buckets.length === 1 ? "" : "s"
    }:`,
  );
  console.log(`${Buckets.map((b) => ` • ${b.Name}`).join("\n")}`);
} catch (caught) {
  if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while listing buckets.  ${caught.name}:
${caught.message}`,
    );
  } else {
    throw caught;
  }
}
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK pour JavaScript API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande renvoie tous les compartiments S3.

```
Get-S3Bucket
```

Exemple 2 : cette commande renvoie un compartiment nommé « test-files »

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande renvoie tous les compartiments S3.

```
Get-S3Bucket
```

Exemple 2 : cette commande renvoie un compartiment nommé « test-files »

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name
```

```
@staticmethod
def list(s3_resource):
    """
    Get the buckets in all Regions for the current account.

    :param s3_resource: A Boto3 S3 resource. This is a high-level resource in
Boto3
                        that contains collections and factory methods to
create
                        other high-level S3 sub-resources.
    :return: The list of buckets.
    """
    try:
        buckets = list(s3_resource.buckets.all())
        logger.info("Got buckets: %s.", buckets)
    except ClientError:
        logger.exception("Couldn't get buckets.")
        raise
    else:
        return buckets
```

- Pour plus de détails sur l'API, consultez [ListBuckets](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 resource actions.
```

```
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts 'Found these buckets:'
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section Référence des AWS SDK pour Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
async fn show_buckets(
    strict: bool,
    client: &Client,
    region: BucketLocationConstraint,
) -> Result<(), S3ExampleError> {
    let mut buckets = client.list_buckets().into_paginator().send();

    let mut num_buckets = 0;
    let mut in_region = 0;

    while let Some(Ok(output)) = buckets.next().await {
        for bucket in output.buckets() {
            num_buckets += 1;
            if strict {
                let r = client
                    .get_bucket_location()
                    .bucket(bucket.name().unwrap_or_default())
                    .send()
                    .await?;

                if r.location_constraint() == Some(&region) {
                    println!("{}", bucket.name().unwrap_or_default());
                    in_region += 1;
                }
            } else {
                println!("{}", bucket.name().unwrap_or_default());
            }
        }
    }

    println!();
    if strict {
```

```
println!(
    "Found {} buckets in the {} region out of a total of {} buckets.",
    in_region, region, num_buckets
);
} else {
    println!("Found {} buckets in all regions.", num_buckets);
}

Ok(())
}
```

- Pour plus de détails sur l'API, voir [ListBuckets](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.
    oo_result = lo_s3->listbuckets(          " oo_result is returned for
testing purposes. "
    ).
    DATA(lv_bucket_count) = lines( oo_result->get_buckets( ) ).
    MESSAGE |Retrieved { lv_bucket_count } buckets in all regions.| TYPE 'I'.
CATCH /aws1/cx_rt_generic.
    MESSAGE 'Unable to list buckets.' TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Swift

Kit SDK pour Swift

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import AWSS3

/// Return an array containing information about every available bucket.
///
/// - Returns: An array of ``S3ClientTypes.Bucket`` objects describing
///   each bucket.
public func getAllBuckets() async throws -> [S3ClientTypes.Bucket] {
    return try await client.listBuckets(input: ListBucketsInput())
}
```

- Pour plus de détails sur l'API, reportez-vous [ListBuckets](#) à la section AWS SDK pour la référence de l'API Swift.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **ListMultipartUploads** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser ListMultipartUploads.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Suppression des chargements partitionnés incomplets](#)

CLI

AWS CLI

La commande suivante répertorie tous les chargements partitionnés actifs d'un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api list-multipart-uploads --bucket amzn-s3-demo-bucket
```

Sortie :

```
{
  "Uploads": [
    {
      "Initiator": {
        "DisplayName": "username",
        "ID": "arn:aws:iam::0123456789012:user/username"
      },
      "Initiated": "2015-06-02T18:01:30.000Z",
      "UploadId":
        "dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3",
      "StorageClass": "STANDARD",
      "Key": "multipart/01",
      "Owner": {
        "DisplayName": "aws-account-name",
        "ID":
          "100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
      }
    }
  ],
  "CommonPrefixes": []
}
```

Les chargements partitionnés en cours entraînent des coûts de stockage dans Amazon S3. Terminez ou annulez un chargement partitionné actif pour supprimer ses parties de votre compte.

- Pour plus de détails sur l'API, reportez-vous [ListMultipartUploads](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */

public class ListMultipartUploads {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName>\s

            Where:
                bucketName - The name of the Amazon S3 bucket where an in-
            progress multipart upload is occurring.
            """;

        if (args.length != 1) {
            System.out.println(usage);
        }
    }
}
```

```
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();
    listUploads(s3, bucketName);
    s3.close();
}

/**
 * Lists the multipart uploads currently in progress in the specified Amazon
 * S3 bucket.
 *
 * @param s3 the S3Client object used to interact with Amazon S3
 * @param bucketName the name of the Amazon S3 bucket to list the multipart
 * uploads for
 */
public static void listUploads(S3Client s3, String bucketName) {
    try {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
            ListMultipartUploadsRequest.builder()
                .bucket(bucketName)
                .build();

        ListMultipartUploadsResponse response =
            s3.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            System.out.println("Upload in progress: Key = \"" + upload.key()
                + "\", id = " + upload.uploadId());
        }

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListMultipartUploads](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **ListObjectVersions** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser ListObjectVersions.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Débuter avec le stockage d'objets](#)
- [Utiliser les objets soumis au contrôle de version](#)

.NET

SDK pour .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example lists the versions of the objects in a version enabled
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class ListObjectVersions
{
    public static async Task Main()
```

```

    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region where your bucket is defined is different from
        // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
        // for the AWS Region to the client constructor like this:
        //     var client = new AmazonS3Client(RegionEndpoint.USWest2);
        IAmazonS3 client = new AmazonS3Client();
        await GetObjectListWithAllVersionsAsync(client, bucketName);
    }

    /// <summary>
    /// This method lists all versions of the objects within an Amazon S3
    /// version enabled bucket.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// ListVersionsAsync.</param>
    /// <param name="bucketName">The name of the version enabled Amazon S3
bucket
    /// for which you want to list the versions of the contained objects.</
param>
    public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
client, string bucketName)
    {
        try
        {
            // When you instantiate the ListVersionRequest, you can
            // optionally specify a key name prefix in the request
            // if you want a list of object versions of a specific object.

            // For this example we set a small limit in MaxKeys to return
            // a small list of versions.
            ListVersionsRequest request = new ListVersionsRequest()
            {
                BucketName = bucketName,
                MaxKeys = 2,
            };

            do
            {
                ListVersionsResponse response = await
client.ListVersionsAsync(request);

```



```
// Process response.
foreach (S3ObjectVersion entry in response.Versions)
{
    Console.WriteLine($"key: {entry.Key} size:
{entry.Size}");
}

// If response is truncated, set the marker to get the next
// set of keys.
if (response.IsTruncated)
{
    request.KeyMarker = response.NextKeyMarker;
    request.VersionIdMarker = response.NextVersionIdMarker;
}
else
{
    request = null;
}
}
while (request != null);
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error: '{ex.Message}'");
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListObjectVersions](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

La commande suivante extrait les informations de la version d'un objet dans un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix index.html
```

Sortie :

```
{
  "DeleteMarkers": [
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": true,
      "VersionId": "B2VsEK5saUNNHKc0AJj7hIE86RozToyq",
      "Key": "index.html",
      "LastModified": "2015-11-10T00:57:03.000Z"
    },
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
      "Key": "index.html",
      "LastModified": "2015-11-09T23:32:20.000Z"
    }
  ],
  "Versions": [
    {
      "LastModified": "2015-11-10T00:20:11.000Z",
      "VersionId": "Rb_l2T8UHDkFEwCgJjhlGPOZC0qJ.vpD",
      "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
      "StorageClass": "STANDARD",
      "Key": "index.html",
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "Size": 38
    },
    {
      "LastModified": "2015-11-09T23:26:41.000Z",
```

```

        "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAff",
        "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
        "StorageClass": "STANDARD",
        "Key": "index.html",
        "Owner": {
            "DisplayName": "my-username",
            "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
        },
        "IsLatest": false,
        "Size": 38
    },
    {
        "LastModified": "2015-11-09T22:50:50.000Z",
        "VersionId": "null",
        "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
        "StorageClass": "STANDARD",
        "Key": "index.html",
        "Owner": {
            "DisplayName": "my-username",
            "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
        },
        "IsLatest": false,
        "Size": 533823
    }
]
}

```

- Pour plus de détails sur l'API, reportez-vous [ListObjectVersions](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// ListObjectVersions lists all versions of all objects in a bucket.  
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)  
    ([]types.ObjectVersion, error) {  
    var err error  
    var output *s3.ListObjectVersionsOutput  
    var versions []types.ObjectVersion  
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}  
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)  
    for versionPaginator.HasMorePages() {  
        output, err = versionPaginator.NextPage(ctx)  
        if err != nil {  
            var noBucket *types.NoSuchBucket  
            if errors.As(err, &noBucket) {  
                log.Printf("Bucket %s does not exist.\n", bucket)  
                err = noBucket  
            }  
            break  
        } else {  
            versions = append(versions, output.Versions...)  
        }  
    }  
}
```

```
    }  
  }  
  return versions, err  
}
```

- Pour plus de détails sur l'API, reportez-vous [ListObjectVersions](#) à la section Référence des AWS SDK pour Go API.

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
async fn show_versions(client: &Client, bucket: &str) -> Result<(), Error> {  
    let resp = client.list_object_versions().bucket(bucket).send().await?;  
  
    for version in resp.versions() {  
        println!("{}", version.key().unwrap_or_default());  
        println!("  version ID: {}", version.version_id().unwrap_or_default());  
        println!();  
    }  
  
    Ok(())  
}
```

- Pour plus de détails sur l'API, voir [ListObjectVersions](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
    oo_result = lo_s3->listobjectversions(          " oo_result is returned  
for testing purposes. "  
    iv_bucket = iv_bucket_name  
    iv_prefix = iv_prefix ).  
    MESSAGE 'Retrieved object versions.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [ListObjectVersions](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **ListObjects** avec une CLI

Les exemples de code suivants illustrent comment utiliser ListObjects.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Créer une page web qui répertorie les objets Amazon S3](#)

CLI

AWS CLI

L'exemple suivant utilise la commande `list-objects` pour afficher le nom de tous les objets du compartiment spécifié :

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

L'exemple utilise l'argument `--query` pour filtrer la sortie `list-objects` jusqu'à la valeur de clé et la taille de chaque objet.

Pour plus d'informations sur les objets, consultez Utilisation des objets Amazon S3 dans le Guide du développeur Amazon S3.

- Pour plus de détails sur l'API, reportez-vous [ListObjects](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : Cette commande récupère les informations relatives à tous les éléments du compartiment « test-files ».

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Exemple 2 : Cette commande récupère les informations relatives à l'élément « sample.txt » depuis le compartiment « test-files ».

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemple 3 : Cette commande récupère les informations relatives à tous les éléments portant le préfixe « sample » à partir du compartiment « test-files ».

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Pour plus de détails sur l'API, reportez-vous [ListObjects](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : Cette commande récupère les informations relatives à tous les éléments du compartiment « test-files ».

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Exemple 2 : Cette commande récupère les informations relatives à l'élément « sample.txt » depuis le compartiment « test-files ».

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemple 3 : Cette commande récupère les informations relatives à tous les éléments portant le préfixe « sample » à partir du compartiment « test-files ».

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Pour plus de détails sur l'API, reportez-vous [ListObjects](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisez la **ListObjectsV2** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser ListObjectsV2.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Principes de base](#)
- [Suppression de tous les objets dans un compartiment](#)
- [Débuter avec le stockage d'objets](#)

.NET

SDK pour .NET (v4)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Shows how to list the objects in an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket for which to list.
/// <param name="printList">True to print out the list.
/// <returns>The collection of objects.</returns>
public async Task<List<S3Object>?> ListBucketContentsAsync(string bucketName,
bool printList = true)
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
        };

        if (printList)
        {
            Console.WriteLine("-----");
            Console.WriteLine($"Listing the contents of {bucketName}:");
            Console.WriteLine("-----");
        }

        var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
        {
            BucketName = bucketName,
        });
        var s3Objects = new List<S3Object>();
        await foreach (var response in listObjectsV2Paginator.Responses)
```

```
        {
            if (response.S3Objects != null)
            {
                s3objects.AddRange(response.S3Objects);
            }
        }

        if (printList)
        {
            Console.WriteLine($"Number of Objects: {s3objects.Count}");
            foreach (var entry in s3objects)
            {
                Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
            }
        }

        return s3objects;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error encountered on server.
        Message: '{ex.Message}' getting list of objects.");
        return null;
    }
}
```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans le manuel de référence des AWS SDK pour .NET API.

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Listez les objets avec un paginateur.

```
using System;
using System.Threading.Tasks;
```

```
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// The following example lists objects in an Amazon Simple Storage
/// Service (Amazon S3) bucket.
/// </summary>
public class ListObjectsPaginator
{
    private const string BucketName = "amzn-s3-demo-bucket";

    public static async Task Main()
    {
        IAmazonS3 s3Client = new AmazonS3Client();

        Console.WriteLine($"Listing the objects contained in {BucketName}:
\n");
        await ListingObjectsAsync(s3Client, BucketName);
    }

    /// <summary>
    /// This method uses a paginator to retrieve the list of objects in an
    /// an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">An Amazon S3 client object.</param>
    /// <param name="bucketName">The name of the S3 bucket whose objects
    /// you want to list.</param>
    public static async Task ListingObjectsAsync(IAmazonS3 client, string
bucketName)
    {
        var listObjectsV2Paginator = client.Paginators.ListObjectsV2(new
ListObjectsV2Request
        {
            BucketName = bucketName,
        });

        await foreach (var response in listObjectsV2Paginator.Responses)
        {
            Console.WriteLine($"HttpStatusCode: {response.HttpStatusCode}");
            Console.WriteLine($"Number of Keys: {response.KeyCount}");
            foreach (var entry in response.S3Objects)
            {
                Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
            }
        }
    }
}
```

```

    }
  }
}

```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans le manuel de référence des AWS SDK pour .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function list_items_in_bucket
#
# This function displays a list of the files in the bucket with each file's
# size. The function uses the --query parameter to retrieve only the key and
# size fields from the Contents collection.
#
# Parameters:
#     $1 - The name of the bucket.
#
# Returns:
#     The list of files in text format.
#
# And:

```

```
#      0 - If successful.
#      1 - If it fails.
#####
function list_items_in_bucket() {
    local bucket_name=$1
    local response

    response=$(aws s3api list-objects \
        --bucket "$bucket_name" \
        --output text \
        --query 'Contents[].{Key: Key, Size: Size}')

    # shellcheck disable=SC2181
    if [[ ${?} -eq 0 ]]; then
        echo "$response"
    else
        errecho "ERROR: AWS reports s3api list-objects operation failed.\n$response"
        return 1
    fi
}
```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans AWS CLI Command Reference.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);
```

```

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " object(s) found:" << std::endl;

    for (const auto &object: allObjects) {
        std::cout << " " << object.GetKey() << std::endl;
        keysResult.push_back(object.GetKey());
    }

    return true;
}

```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans le manuel de référence des AWS SDK pour C++ API.

CLI

AWS CLI

Pour obtenir la liste des objets d'un compartiment

L'exemple `list-objects-v2` suivant répertorie les objets du compartiment spécifié.

```
aws s3api list-objects-v2 \  
  --bucket amzn-s3-demo-bucket
```

Sortie :

```
{  
  "Contents": [  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"621503c373607d548b37cff8778d992c\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc1.rtf",  
      "Size": 391  
    },  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"a2cecc36ab7c7fe3a71a273b9d45b1b5\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc2.rtf",  
      "Size": 373  
    },  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"08210852f65a2e9cb999972539a64d68\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc3.rtf",  
      "Size": 399  
    },  
    {  
      "LastModified": "2019-11-05T23:11:50.000Z",  
      "ETag": "\"d1852dd683f404306569471af106988e\"",  
      "StorageClass": "STANDARD",  
      "Key": "doc4.rtf",  
      "Size": 6225  
    }  
  ]  
}
```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans AWS CLI Command Reference.

Go

Kit SDK pour Go V2

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}  
  
// ListObjects lists the objects in a bucket.
```



```
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
([]types.Object, error) {
    var err error
    var output *s3.ListObjectsV2Output
    input := &s3.ListObjectsV2Input{
        Bucket: aws.String(bucketName),
    }
    var objects []types.Object
    objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
    for objectPaginator.HasMorePages() {
        output, err = objectPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
            break
        } else {
            objects = append(objects, output.Contents...)
        }
    }
    return objects, err
}
```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans le manuel de référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

/**
 * Asynchronously lists all objects in the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket to list objects for
 * @return a {@link CompletableFuture} that completes when all objects have
 * been listed
 */
public CompletableFuture<Void> listAllObjectsAsync(String bucketName) {
    ListObjectsV2Request initialRequest = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .maxKeys(1)
        .build();

    ListObjectsV2Publisher paginator =
getAsyncClient().listObjectsV2Paginator(initialRequest);
    return paginator.subscribe(response -> {
        response.contents().forEach(s3Object -> {
            logger.info("Object key: " + s3Object.key());
        });
    }).thenRun(() -> {
        logger.info("Successfully listed all objects in the bucket: " +
bucketName);
    }).exceptionally(ex -> {
        throw new RuntimeException("Failed to list objects", ex);
    });
}

```

Lister les objets en utilisant la pagination.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.paginators.ListObjectsV2Iterable;

public class ListObjectsPaginated {
    public static void main(String[] args) {
        final String usage = ""

        Usage:
        <bucketName>\s

```

```

        Where:
            bucketName - The Amazon S3 bucket from which objects are read.\s
        """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    listBucketObjects(s3, bucketName);
    s3.close();
}

/**
 * Lists the objects in the specified S3 bucket.
 *
 * @param s3 the S3Client instance used to interact with Amazon S3
 * @param bucketName the name of the S3 bucket to list the objects from
 */
public static void listBucketObjects(S3Client s3, String bucketName) {
    try {
        ListObjectsV2Request listReq = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .maxKeys(1)
            .build();

        ListObjectsV2Iterable listRes = s3.listObjectsV2Paginator(listReq);
        listRes.stream()
            .flatMap(r -> r.contents().stream())
            .forEach(content -> System.out.println(" Key: " + content.key() +
" size = " + content.size()));

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

```
}
```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans le manuel de référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Répertoriez tous les objets dans votre compartiment. S'il existe plusieurs objets, `IsTruncated` et `NextContinuationToken` seront utilisés pour parcourir la liste complète.

```
import {
  S3Client,
  S3ServiceException,
  // This command supersedes the ListObjectsCommand and is the recommended way to
  list objects.
  paginateListObjectsV2,
} from "@aws-sdk/client-s3";

/**
 * Log all of the object keys in a bucket.
 * @param {{ bucketName: string, pageSize: string }}
 */
export const main = async ({ bucketName, pageSize }) => {
  const client = new S3Client({});
  /** @type {string[][]} */
  const objects = [];
  try {
    const paginator = paginateListObjectsV2(
      { client, /* Max items per page */ pageSize: Number.parseInt(pageSize) },
      { Bucket: bucketName },
    );
```

```

    for await (const page of paginator) {
        objects.push(page.Contents.map((o) => o.Key));
    }
    objects.forEach((objectList, pageNum) => {
        console.log(
            `Page ${pageNum + 1}\n-----\n${objectList.map((o) => `•
${o}`)}.join("\n")}\n`,
        );
    });
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while listing objects for "${bucketName}". The bucket
doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while listing objects for "${bucketName}".
${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans le manuel de référence des AWS SDK pour JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun listBucketObjects(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.listObjects(request)
        response.contents?.forEach { myObject ->
            println("The name of the key is ${myObject.key}")
            println("The object is ${myObject.size?.let { calKb(it) }} KBs")
            println("The owner is ${myObject.owner}")
        }
    }
}

private fun calKb(intValue: Long): Long = intValue / 1024
```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans le AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Listez les objets dans un compartiment.

```
$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

try {
    $contents = $this->s3client->listObjectsV2([
        'Bucket' => $this->bucketName,
    ]);
    echo "The contents of your bucket are: \n";
```

```

        foreach ($contents['Contents'] as $content) {
            echo $content['Key'] . "\n";
        }
    } catch (Exception $exception) {
        echo "Failed to list objects in $this->bucketName with error: " .
        $exception->getMessage();
        exit("Please fix error with listing objects before continuing.");
    }
}

```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans le manuel de référence des AWS SDK pour PHP API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    @staticmethod
    def list(bucket, prefix=None):
        """
        Lists the objects in a bucket, optionally filtered by a prefix.

```

```

        :param bucket: The bucket to query. This is a Boto3 Bucket resource.
        :param prefix: When specified, only objects that start with this prefix
are listed.
        :return: The list of objects.
        """
        try:
            if not prefix:
                objects = list(bucket.objects.all())
            else:
                objects = list(bucket.objects.filter(Prefix=prefix))
            logger.info(
                "Got objects %s from bucket '%s'", [o.key for o in objects],
bucket.name
            )
        except ClientError:
            logger.exception("Couldn't get objects for bucket '%s'.",
bucket.name)
            raise
        else:
            return objects

```

- Pour plus de détails sur l'API, reportez-vous à la section [ListObjectsV2](#) dans le manuel de référence de l'API AWS SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

```



```

# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
def initialize(bucket)
  @bucket = bucket
end

# Lists object in a bucket.
#
# @param max_objects [Integer] The maximum number of objects to list.
# @return [Integer] The number of objects listed.
def list_objects(max_objects)
  count = 0
  puts "The objects in #{@bucket.name} are:"
  @bucket.objects.each do |obj|
    puts "\t#{obj.key}"
    count += 1
    break if count == max_objects
  end
  count
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
#{e.message}"
  0
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans le manuel de référence des AWS SDK pour Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
pub async fn list_objects(client: &aws_sdk_s3::Client, bucket: &str) ->
    Result<(), S3ExampleError> {
    let mut response = client
        .list_objects_v2()
        .bucket(bucket.to_owned())
        .max_keys(10) // In this example, go 10 at a time.
        .into_paginator()
        .send();

    while let Some(result) = response.next().await {
        match result {
            Ok(output) => {
                for object in output.contents() {
                    println!("{}", object.key().unwrap_or("Unknown"));
                }
            }
            Err(err) => {
                eprintln!("{err:?}")
            }
        }
    }

    Ok(())
}
```

- Pour plus de détails sur l'API, voir la [ListObjectsversion V2](#) dans le AWS SDK pour la référence de l'API Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
    oo_result = lo_s3->listobjectsv2(          " oo_result is returned for  
testing purposes. "  
    iv_bucket = iv_bucket_name ).  
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous à la section [ListObjectsV2](#) du AWS SDK pour la référence de l'API SAP ABAP.

Swift

Kit SDK pour Swift

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import AWSS3  
  
public func listBucketFiles(bucket: String) async throws -> [String] {  
    do {  
        let input = ListObjectsV2Input(  
            bucket: bucket
```

```
)

// Use "Paginated" to get all the objects.
// This lets the SDK handle the 'continuationToken' in
"ListObjectsV2Output".
let output = client.listObjectsV2Paginated(input: input)
var names: [String] = []

for try await page in output {
    guard let objList = page.contents else {
        print("ERROR: listObjectsV2Paginated returned nil contents.")
        continue
    }

    for obj in objList {
        if let objName = obj.key {
            names.append(objName)
        }
    }
}

return names
}
catch {
    print("ERROR: ", dump(error, name: "Listing objects."))
    throw error
}
}
```

- Pour plus de détails sur l'API, consultez la section [ListObjectsV2](#) dans le AWS SDK pour la référence de l'API Swift.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketAccelerateConfiguration** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser **PutBucketAccelerateConfiguration**.

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
    /// The main method initializes the client object and sets the
    /// Amazon Simple Storage Service (Amazon S3) bucket name before
    /// calling EnableAccelerationAsync.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        const string bucketName = "amzn-s3-demo-bucket";

        await EnableAccelerationAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method sets the configuration to enable transfer acceleration
    /// for the bucket referred to in the bucketName parameter.
    /// </summary>
    /// <param name="client">An Amazon S3 client used to enable the
    /// acceleration on an Amazon S3 bucket.</param>
```

```

    the
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
    /// method will be enabling acceleration.</param>
    private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
    {
        try
        {
            var putRequest = new PutBucketAccelerateConfigurationRequest
            {
                BucketName = bucketName,
                AccelerateConfiguration = new AccelerateConfiguration
                {
                    Status = BucketAccelerateStatus.Enabled,
                },
            };
            await client.PutBucketAccelerateConfigurationAsync(putRequest);

            var getRequest = new GetBucketAccelerateConfigurationRequest
            {
                BucketName = bucketName,
            };
            var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

            Console.WriteLine($"Acceleration state = '{response.Status}' ");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error occurred. Message:'{ex.Message}' when
setting transfer acceleration");
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [PutBucketAccelerateConfiguration](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour activer la configuration accélérée d'un compartiment

L'exemple `put-bucket-accelerate-configuration` suivant active la configuration accélérée du compartiment spécifié.

```
aws s3api put-bucket-accelerate-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --accelerate-configuration Status=Enabled
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [PutBucketAccelerateConfiguration](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande active l'accélération du transfert pour le compartiment S3 donné.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketAccelerateConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande active l'accélération du transfert pour le compartiment S3 donné.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketAccelerateConfiguration](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketAcl** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutBucketAcl.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Gérer les listes de contrôle d'accès \(ACL\)](#)

.NET

SDK pour .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Creates an Amazon S3 bucket with an ACL to control access to the
/// bucket and the objects stored in it.
/// </summary>
/// <param name="client">The initialized client object used to create
/// an Amazon S3 bucket, with an ACL applied to the bucket.
/// </param>
/// <param name="region">The AWS Region where the bucket will be
created.</param>
/// <param name="newBucketName">The name of the bucket to create.</param>
/// <returns>A boolean value indicating success or failure.</returns>
public static async Task<bool> CreateBucketUseCannedACLAsync(IAmazonS3
client, S3Region region, string newBucketName)
{
    try
    {
```



```

        // Create a new Amazon S3 bucket with Canned ACL.
        var putBucketRequest = new PutBucketRequest()
        {
            BucketName = newBucketName,
            BucketRegion = region,
            CannedACL = S3CannedACL.LogDeliveryWrite,
        };

        PutBucketResponse putBucketResponse = await
client.PutBucketAsync(putBucketRequest);

        return putBucketResponse.HttpStatusCode ==
System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Amazon S3 error: {ex.Message}");
    }

    return false;
}

```

- Pour plus de détails sur l'API, reportez-vous [PutBucketAcl](#) à la section Référence des AWS SDK pour .NET API.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
&ownerID,
                                const Aws::String &granteePermission,

```

```
const Aws::String &granteeType, const Aws::String
&granteeID,
const Aws::String &granteeEmailAddress,
const Aws::String &granteeURI, const
Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
        s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
```

```

        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
                   << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
                  << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
enumeration.
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
 */

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type
enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
 */

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")

```

```
    return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketAcl](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Cet exemple accorde `full control` l'autorisation à deux AWS utilisateurs (`user1@example.com` et `user2@example.com`) et `read` l'autorisation à tout le monde :

```
aws s3api put-bucket-acl --bucket amzn-s3-demo-bucket --grant-full-  
control emailaddress=user1@example.com,emailaddress=user2@example.com --grant-  
read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

Consultez <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> pour plus de détails sur les ACL personnalisées (les commandes ACL `s3api`, par exemple `put-bucket-acl`, utilisent la même notation abrégée en arguments).

- Pour plus de détails sur l'API, reportez-vous [PutBucketAcl](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AccessControlPolicy;
import software.amazon.awssdk.services.s3.model.Grant;
import software.amazon.awssdk.services.s3.model.Permission;
import software.amazon.awssdk.services.s3.model.PutBucketAclRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.Type;

import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class SetAcl {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <id>\s

            Where:
                bucketName - The Amazon S3 bucket to grant permissions on.\s
                id - The ID of the owner of this bucket (you can get this value
from the AWS Management Console).
            """;

        if (args.length != 2) {
            System.out.println(usage);
            return;
        }

        String bucketName = args[0];
        String id = args[1];
        System.out.format("Setting access \n");
        System.out.println(" in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
```

```
        .region(region)
        .build();

        setBucketAcl(s3, bucketName, id);
        System.out.println("Done!");
        s3.close();
    }

    /**
     * Sets the Access Control List (ACL) for an Amazon S3 bucket.
     *
     * @param s3 the S3Client instance to be used for the operation
     * @param bucketName the name of the S3 bucket to set the ACL for
     * @param id the ID of the AWS user or account that will be granted full
control of the bucket
     * @throws S3Exception if an error occurs while setting the bucket ACL
     */
    public static void setBucketAcl(S3Client s3, String bucketName, String id) {
        try {
            Grant ownerGrant = Grant.builder()
                .grantee(builder -> builder.id(id)
                    .type(Type.CANONICAL_USER))
                .permission(Permission.FULL_CONTROL)
                .build();

            List<Grant> grantList2 = new ArrayList<>();
            grantList2.add(ownerGrant);

            AccessControlPolicy acl = AccessControlPolicy.builder()
                .owner(builder -> builder.id(id))
                .grants(grantList2)
                .build();

            PutBucketAclRequest putAclReq = PutBucketAclRequest.builder()
                .bucket(bucketName)
                .accessControlPolicy(acl)
                .build();

            s3.putBucketAcl(putAclReq);

        } catch (S3Exception e) {
            e.printStackTrace();
            System.exit(1);
        }
    }
}
```

```
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketAcl](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Effectuez une commande PUT pour l'ACL du compartiment.

```
import {  
  PutBucketAclCommand,  
  S3Client,  
  S3ServiceException,  
} from "@aws-sdk/client-s3";  
  
/**  
 * Grant read access to a user using their canonical AWS account ID.  
 *  
 * Most Amazon S3 use cases don't require the use of access control lists (ACLs).  
 * We recommend that you disable ACLs, except in unusual circumstances where  
 * you need to control access for each object individually. Consider a policy  
 * instead.  
 * For more information see https://docs.aws.amazon.com/AmazonS3/latest/userguide/bucket-policies.html.  
 * @param {{ bucketName: string, granteeCanonicalUserId: string,  
   ownerCanonicalUserId }}  
 */  
export const main = async ({  
  bucketName,  
  granteeCanonicalUserId,  
  ownerCanonicalUserId,  
}) => {
```

```

const client = new S3Client({});
const command = new PutBucketAclCommand({
  Bucket: bucketName,
  AccessControlPolicy: {
    Grants: [
      {
        Grantee: {
          // The canonical ID of the user. This ID is an obfuscated form of
          your AWS account number.
          // It's unique to Amazon S3 and can't be found elsewhere.
          // For more information, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/finding-canonical-user-id.html.
          ID: granteeCanonicalUserId,
          Type: "CanonicalUser",
        },
        // One of FULL_CONTROL | READ | WRITE | READ_ACP | WRITE_ACP
        // https://docs.aws.amazon.com/AmazonS3/latest/API/API\_Grant.html#AmazonS3-Type-Grant-Permission
        Permission: "READ",
      },
    ],
    Owner: {
      ID: ownerCanonicalUserId,
    },
  },
});

try {
  await client.send(command);
  console.log(`Granted READ access to ${bucketName}`);
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while setting ACL for bucket ${bucketName}. The bucket
      doesn't exist.`,
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while setting ACL for bucket ${bucketName}.
      ${caught.name}: ${caught.message}`,
    );
  }
}

```



```
    } else {  
        throw caught;  
    }  
}  
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [PutBucketAcl](#) à la section Référence des AWS SDK pour JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun setBucketAcl(  
    bucketName: String,  
    idVal: String,  
) {  
    val myGrant =  
        Grantee {  
            id = idVal  
            type = Type.CanonicalUser  
        }  
  
    val ownerGrant =  
        Grant {  
            grantee = myGrant  
            permission = Permission.FullControl  
        }  
  
    val grantList = mutableListOf<Grant>()  
    grantList.add(ownerGrant)  
  
    val ownerObj =
```

```
        Owner {
            id = idVal
        }

    val acl =
        AccessControlPolicy {
            owner = ownerObj
            grants = grantList
        }

    val request =
        PutBucketAclRequest {
            bucket = bucketName
            accessControlPolicy = acl
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.putBucketAcl(request)
        println("An ACL was successfully set on $bucketName")
    }
}
```

- Pour plus de détails sur l'API, consultez [PutBucketAcl](#) la section AWS SDK pour la référence de l'API Kotlin.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
```

```

        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def grant_log_delivery_access(self):
        """
        Grant the AWS Log Delivery group write access to the bucket so that
        Amazon S3 can deliver access logs to the bucket. This is the only
recommended
        use of an S3 bucket ACL.
        """
        try:
            acl = self.bucket.Acl()
            # Putting an ACL overwrites the existing ACL. If you want to preserve
            # existing grants, append new grants to the list of existing grants.
            grants = acl.grants if acl.grants else []
            grants.append(
                {
                    "Grantee": {
                        "Type": "Group",
                        "URI": "http://acs.amazonaws.com/groups/s3/LogDelivery",
                    },
                    "Permission": "WRITE",
                }
            )
            acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
            logger.info("Granted log delivery access to bucket '%s'",
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't add ACL to bucket '%s'.",
self.bucket.name)
            raise

```

- Pour plus de détails sur l'API, consultez [PutBucketAcl](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
  " Example: Grant log delivery access to a bucket  
  " iv_grantwrite = 'uri=http://acs.amazonaws.com/groups/s3/LogDelivery'  
  lo_s3->putbucketacl(  
    iv_bucket = iv_bucket_name  
    iv_grantwrite = iv_grantwrite ).  
  MESSAGE 'Bucket ACL updated.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketAcl](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketCors** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutBucketCors.

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Add CORS configuration to the Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used
/// to apply the CORS configuration to an Amazon S3 bucket.</param>
/// <param name="configuration">The CORS configuration to apply.</param>
private static async Task PutCORSConfigurationAsync(AmazonS3Client
client, CORSConfiguration configuration)
{
    PutCORSConfigurationRequest request = new
PutCORSConfigurationRequest()
    {
        BucketName = BucketName,
        Configuration = configuration,
    };

    _ = await client.PutCORSConfigurationAsync(request);
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketCors](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

L'exemple suivant active les demandes PUT, POST et DELETE provenant de `www.example.com`, et active les demandes GET provenant de n'importe quel domaine :

```
aws s3api put-bucket-cors --bucket amzn-s3-demo-bucket --cors-configuration file://cors.json
```

cors.json:

```
{
  "CORSRules": [
    {
      "AllowedOrigins": ["http://www.example.com"],
      "AllowedHeaders": ["*"],
      "AllowedMethods": ["PUT", "POST", "DELETE"],
      "MaxAgeSeconds": 3000,
      "ExposeHeaders": ["x-amz-server-side-encryption"]
    },
    {
      "AllowedOrigins": ["*"],
      "AllowedHeaders": ["Authorization"],
      "AllowedMethods": ["GET"],
      "MaxAgeSeconds": 3000
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketCors](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;

import java.util.ArrayList;
import java.util.List;
```

```
import software.amazon.awssdk.services.s3.model.GetBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.GetBucketCorsResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketCorsRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.CORSRule;
import software.amazon.awssdk.services.s3.model.CORSConfiguration;
import software.amazon.awssdk.services.s3.model.PutBucketCorsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class S3Cors {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setCorsInformation(s3, bucketName, accountId);
        getBucketCorsInformation(s3, bucketName, accountId);
    }
}
```

```
        deleteBucketCorsInformation(s3, bucketName, accountId);
        s3.close();
    }

    /**
     * Deletes the CORS (Cross-Origin Resource Sharing) configuration for an
     * Amazon S3 bucket.
     *
     * @param s3          the {@link S3Client} instance used to interact with
     *                    the Amazon S3 service
     * @param bucketName  the name of the Amazon S3 bucket for which the CORS
     *                    configuration should be deleted
     * @param accountId   the expected AWS account ID of the bucket owner
     *
     * @throws S3Exception if an error occurs while deleting the CORS
     *                    configuration for the bucket
     */
    public static void deleteBucketCorsInformation(S3Client s3, String
        bucketName, String accountId) {
        try {
            DeleteBucketCorsRequest bucketCorsRequest =
                DeleteBucketCorsRequest.builder()
                    .bucket(bucketName)
                    .expectedBucketOwner(accountId)
                    .build();

            s3.deleteBucketCors(bucketCorsRequest);

        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Retrieves the CORS (Cross-Origin Resource Sharing) configuration for the
     * specified S3 bucket.
     *
     * @param s3 the S3Client instance to use for the operation
     * @param bucketName the name of the S3 bucket to retrieve the CORS
     *                    configuration for
     * @param accountId the expected bucket owner's account ID
     *
     * @throws S3Exception if there is an error retrieving the CORS configuration
     */
}
```



```
    */
    public static void getBucketCorsInformation(S3Client s3, String bucketName,
String accountId) {
        try {
            GetBucketCorsRequest bucketCorsRequest =
GetBucketCorsRequest.builder()
                .bucket(bucketName)
                .expectedBucketOwner(accountId)
                .build();

            GetBucketCorsResponse corsResponse =
s3.getBucketCors(bucketCorsRequest);
            List<CORSRule> corsRules = corsResponse.corsRules();
            for (CORSRule rule : corsRules) {
                System.out.println("allowOrigins: " + rule.allowedOrigins());
                System.out.println("AllowedMethod: " + rule.allowedMethods());
            }

        } catch (S3Exception e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Sets the Cross-Origin Resource Sharing (CORS) rules for an Amazon S3
bucket.
     *
     * @param s3 The S3Client object used to interact with the Amazon S3 service.
     * @param bucketName The name of the S3 bucket to set the CORS rules for.
     * @param accountId The AWS account ID of the bucket owner.
     */
    public static void setCorsInformation(S3Client s3, String bucketName, String
accountId) {
        List<String> allowMethods = new ArrayList<>();
        allowMethods.add("PUT");
        allowMethods.add("POST");
        allowMethods.add("DELETE");

        List<String> allowOrigins = new ArrayList<>();
        allowOrigins.add("http://example.com");
        try {
            // Define CORS rules.
```

```
        CORSRule corsRule = CORSRule.builder()
            .allowedMethods(allowMethods)
            .allowedOrigins(allowOrigins)
            .build();

        List<CORSRule> corsRules = new ArrayList<>();
        corsRules.add(corsRule);
        CORSConfiguration configuration = CORSConfiguration.builder()
            .corsRules(corsRules)
            .build();

        PutBucketCorsRequest putBucketCorsRequest =
        PutBucketCorsRequest.builder()
            .bucket(bucketName)
            .corsConfiguration(configuration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketCors(putBucketCorsRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketCors](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Ajoutez une règle CORS.

```
import {
  PutBucketCorsCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Allows cross-origin requests to an S3 bucket by setting the CORS
 * configuration.
 * @param {{ bucketName: string }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    await client.send(
      new PutBucketCorsCommand({
        Bucket: bucketName,
        CORSConfiguration: {
          CORSRules: [
            {
              // Allow all headers to be sent to this bucket.
              AllowedHeaders: ["*"],
              // Allow only GET and PUT methods to be sent to this bucket.
              AllowedMethods: ["GET", "PUT"],
              // Allow only requests from the specified origin.
              AllowedOrigins: ["https://www.example.com"],
              // Allow the entity tag (ETag) header to be returned in the
              // response. The ETag header
              // The entity tag represents a specific version of the object. The
              // ETag reflects
              // changes only to the contents of an object, not its metadata.
              ExposeHeaders: ["ETag"],
              // How long the requesting browser should cache the preflight
              // response. After
              // this time, the preflight request will have to be made again.
              MaxAgeSeconds: 3600,
            },
          ],
        },
      })
    );
    console.log(`Successfully set CORS rules for bucket: ${bucketName}`);
  }
}
```

```
    } catch (caught) {
      if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
      ) {
        console.error(
          `Error from S3 while setting CORS rules for ${bucketName}. The bucket
          doesn't exist.`
        );
      } else if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while setting CORS rules for ${bucketName}.
          ${caught.name}: ${caught.message}`
        );
      } else {
        throw caught;
      }
    }
  };
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [PutBucketCors](#) à la section Référence des AWS SDK pour JavaScript API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
```

```
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_cors(self, cors_rules):
        """
        Apply CORS rules to the bucket. CORS rules specify the HTTP actions that
are
        allowed from other domains.

        :param cors_rules: The CORS rules to apply.
        """
        try:
            self.bucket.Cors().put(CORSConfiguration={"CORSRules": cors_rules})
            logger.info(
                "Put CORS rules %s for bucket '%s'.", cors_rules,
self.bucket.name
            )
        except ClientError:
            logger.exception("Couldn't put CORS rules for bucket %s.",
self.bucket.name)
            raise
```

- Pour plus de détails sur l'API, consultez [PutBucketCors](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
    #{e.message}"
    false
  end
end
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketCors](#) à la section Référence des AWS SDK pour Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
  " Example: Allow PUT, POST, DELETE methods from http://www.example.com  
  lo_s3->putbucketcors(  
    iv_bucket = iv_bucket_name  
    io_corsconfiguration = NEW /aws1/cl_s3_corsconfiguration(  
      it_corsrules = it_cors_rules ) ).  
  MESSAGE 'Bucket CORS configuration set.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketCors](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketEncryption** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutBucketEncryption.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Débuter avec le stockage d'objets](#)

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Set the bucket server side encryption to use AWSKMS with a customer-
managed key id.
/// </summary>
/// <param name="bucketName">Name of the bucket.</param>
/// <param name="kmsKeyId">The Id of the KMS Key.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> SetBucketServerSideEncryption(string
bucketName, string kmsKeyId)
{
    var serverSideEncryptionByDefault = new ServerSideEncryptionConfiguration
    {
        ServerSideEncryptionRules = new List<ServerSideEncryptionRule>
        {
            new ServerSideEncryptionRule
            {
                ServerSideEncryptionByDefault = new
ServerSideEncryptionByDefault
                {
                    ServerSideEncryptionAlgorithm =
ServerSideEncryptionMethod.AWSKMS,
                    ServerSideEncryptionKeyManagementServiceKeyId = kmsKeyId
                }
            }
        }
    };
    try
    {
        var encryptionResponse = await _s3Client.PutBucketEncryptionAsync(new
PutBucketEncryptionRequest
        {
            BucketName = bucketName,
```



```
        ServerSideEncryptionConfiguration =
serverSideEncryptionByDefault,
    });

    return encryptionResponse.HttpStatusCode == HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine(ex.ErrorCode == "AccessDenied"
        ? $"This account does not have permission to set encryption on
{bucketName}, please try again."
        : $"Unable to set bucket encryption for bucket {bucketName},
{ex.Message}");
}
return false;
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketEncryption](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour configurer le chiffrement côté serveur d'un compartiment

L'exemple `put-bucket-encryption` suivant définit le chiffrement AES256 comme valeur par défaut pour le compartiment spécifié.

```
aws s3api put-bucket-encryption \
    --bucket amzn-s3-demo-bucket \
    --server-side-encryption-configuration '{"Rules":
[{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [PutBucketEncryption](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : Cette commande active le chiffrement par défaut côté serveur AES256 avec les clés gérées Amazon S3 (SSE-S3) sur le compartiment donné.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
    ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketEncryption](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : Cette commande active le chiffrement par défaut côté serveur AES256 avec les clés gérées Amazon S3 (SSE-S3) sur le compartiment donné.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
    ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketEncryption](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketLifecycleConfiguration** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutBucketLifecycleConfiguration.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Suppression des chargements partitionnés incomplets](#)

- [Gestion de messages volumineux à l'aide de S3](#)
- [Utiliser les objets soumis au contrôle de version](#)

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Adds lifecycle configuration information to the S3 bucket named in
/// the bucketName parameter.
/// </summary>
/// <param name="client">The S3 client used to call the
/// PutLifecycleConfigurationAsync method.</param>
/// <param name="bucketName">A string representing the S3 bucket to
/// which configuration information will be added.</param>
/// <param name="configuration">A LifecycleConfiguration object that
/// will be applied to the S3 bucket.</param>
public static async Task AddExampleLifecycleConfigAsync(IAmazonS3 client,
string bucketName, LifecycleConfiguration configuration)
{
    var request = new PutLifecycleConfigurationRequest()
    {
        BucketName = bucketName,
        Configuration = configuration,
    };
    var response = await client.PutLifecycleConfigurationAsync(request);
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketLifecycleConfiguration](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

La commande suivante applique une configuration de cycle de vie à un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --  
lifecycle-configuration file://lifecycle.json
```

Le fichier `lifecycle.json` est un document JSON du dossier en cours qui spécifie deux règles :

```
{  
  "Rules": [  
    {  
      "ID": "Move rotated logs to Glacier",  
      "Prefix": "rotated/",  
      "Status": "Enabled",  
      "Transitions": [  
        {  
          "Date": "2015-11-10T00:00:00.000Z",  
          "StorageClass": "GLACIER"  
        }  
      ]  
    },  
    {  
      "Status": "Enabled",  
      "Prefix": "",  
      "NoncurrentVersionTransitions": [  
        {  
          "NoncurrentDays": 2,  
          "StorageClass": "GLACIER"  
        }  
      ],  
      "ID": "Move old versions to Glacier"  
    }  
  ]  
}
```

La première règle déplace les fichiers possédant le préfixe `rotated` dans Glacier à la date spécifiée. La seconde règle déplace les anciennes versions des objets dans Glacier

lorsqu'elles ne sont plus d'actualité. Pour en savoir plus sur les formats d'horodatage acceptables, consultez Spécification des valeurs des paramètres dans le Guide de l'utilisateur AWS CLI.

- Pour plus de détails sur l'API, reportez-vous [PutBucketLifecycleConfiguration](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.Transition;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.DeleteBucketLifecycleRequest;
import software.amazon.awssdk.services.s3.model.TransitionStorageClass;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
```

```

* <p>
* https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
*/

public class LifecycleConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <accountId>\s

            Where:
                bucketName - The Amazon Simple Storage Service (Amazon S3) bucket
to upload an object into.
                accountId - The id of the account that owns the Amazon S3 bucket.
            "";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String accountId = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setLifecycleConfig(s3, bucketName, accountId);
        getLifecycleConfig(s3, bucketName, accountId);
        deleteLifecycleConfig(s3, bucketName, accountId);
        System.out.println("You have successfully created, updated, and deleted a
Lifecycle configuration");
        s3.close();
    }

    /**
     * Sets the lifecycle configuration for an Amazon S3 bucket.
     *
     * @param s3          The Amazon S3 client to use for the operation.
     * @param bucketName  The name of the Amazon S3 bucket.
     * @param accountId   The expected owner of the Amazon S3 bucket.

```

```

    *
    * @throws S3Exception if there is an error setting the lifecycle
    configuration.
    */
    public static void setLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
        try {
            // Create a rule to archive objects with the "glacierobjects/" prefix
to the

            // S3 Glacier Flexible Retrieval storage class immediately.
            LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            Transition transition = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            LifecycleRule rule1 = LifecycleRule.builder()
                .id("Archive immediately rule")
                .filter(ruleFilter)
                .transitions(transition)
                .status(ExpirationStatus.ENABLED)
                .build();

            // Create a second rule.
            Transition transition2 = Transition.builder()
                .storageClass(TransitionStorageClass.GLACIER)
                .days(0)
                .build();

            List<Transition> transitionList = new ArrayList<>();
            transitionList.add(transition2);

            LifecycleRuleFilter ruleFilter2 = LifecycleRuleFilter.builder()
                .prefix("glacierobjects/")
                .build();

            LifecycleRule rule2 = LifecycleRule.builder()
                .id("Archive and then delete rule")
                .filter(ruleFilter2)
                .transitions(transitionList)
                .status(ExpirationStatus.ENABLED)

```

```
        .build();

        // Add the LifecycleRule objects to an ArrayList.
        ArrayList<LifecycleRule> ruleList = new ArrayList<>();
        ruleList.add(rule1);
        ruleList.add(rule2);

        BucketLifecycleConfiguration lifecycleConfiguration =
        BucketLifecycleConfiguration.builder()
            .rules(ruleList)
            .build();

        PutBucketLifecycleConfigurationRequest
        putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
            .builder()
            .bucket(bucketName)
            .lifecycleConfiguration(lifecycleConfiguration)
            .expectedBucketOwner(accountId)
            .build();

        s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Retrieves the lifecycle configuration for an Amazon S3 bucket and adds a
 * new lifecycle rule to it.
 *
 * @param s3 the S3Client instance used to interact with Amazon S3
 * @param bucketName the name of the Amazon S3 bucket
 * @param accountId the expected owner of the Amazon S3 bucket
 */
public static void getLifecycleConfig(S3Client s3, String bucketName, String
accountId) {
    try {
        GetBucketLifecycleConfigurationRequest
        getBucketLifecycleConfigurationRequest = GetBucketLifecycleConfigurationRequest
            .builder()
            .bucket(bucketName)
```



```
        .expectedBucketOwner(accountId)
        .build();

    GetBucketLifecycleConfigurationResponse response = s3

    .getBucketLifecycleConfiguration(getBucketLifecycleConfigurationRequest);
    List<LifecycleRule> newList = new ArrayList<>();
    List<LifecycleRule> rules = response.rules();
    for (LifecycleRule rule : rules) {
        newList.add(rule);
    }

    // Add a new rule with both a prefix predicate and a tag predicate.
    LifecycleRuleFilter ruleFilter = LifecycleRuleFilter.builder()
        .prefix("YearlyDocuments/")
        .build();

    Transition transition = Transition.builder()
        .storageClass(TransitionStorageClass.GLACIER)
        .days(3650)
        .build();

    LifecycleRule rule1 = LifecycleRule.builder()
        .id("NewRule")
        .filter(ruleFilter)
        .transitions(transition)
        .status(ExpirationStatus.ENABLED)
        .build();

    // Add the new rule to the list.
    newList.add(rule1);
    BucketLifecycleConfiguration lifecycleConfiguration =
    BucketLifecycleConfiguration.builder()
        .rules(newList)
        .build();

    PutBucketLifecycleConfigurationRequest
    putBucketLifecycleConfigurationRequest = PutBucketLifecycleConfigurationRequest
        .builder()
        .bucket(bucketName)
        .lifecycleConfiguration(lifecycleConfiguration)
        .expectedBucketOwner(accountId)
        .build();
```

```

s3.putBucketLifecycleConfiguration(putBucketLifecycleConfigurationRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}

/**
 * Deletes the lifecycle configuration for an Amazon S3 bucket.
 *
 * @param s3 the {@link S3Client} to use for the operation
 * @param bucketName the name of the S3 bucket
 * @param accountId the expected account owner of the S3 bucket
 *
 * @throws S3Exception if an error occurs while deleting the lifecycle
 * configuration
 */
public static void deleteLifecycleConfig(S3Client s3, String bucketName,
String accountId) {
    try {
        DeleteBucketLifecycleRequest deleteBucketLifecycleRequest =
DeleteBucketLifecycleRequest
        .builder()
        .bucket(bucketName)
        .expectedBucketOwner(accountId)
        .build();

        s3.deleteBucketLifecycle(deleteBucketLifecycleRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Pour plus de détails sur l'API, reportez-vous [PutBucketLifecycleConfiguration](#) à la section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_lifecycle_configuration(self, lifecycle_rules):
        """
        Apply a lifecycle configuration to the bucket. The lifecycle
        configuration can
            be used to archive or delete the objects in the bucket according to
        specified
            parameters, such as a number of days.

        :param lifecycle_rules: The lifecycle rules to apply.
        """
        try:
            self.bucket.LifecycleConfiguration().put(
                LifecycleConfiguration={"Rules": lifecycle_rules}
            )
            logger.info(
                "Put lifecycle rules %s for bucket '%s'.",
                lifecycle_rules,
                self.bucket.name,
            )
```

```
except ClientError:
    logger.exception(
        "Couldn't put lifecycle rules for bucket '%s'.", self.bucket.name
    )
    raise
```

- Pour plus de détails sur l'API, consultez [PutBucketLifecycleConfiguration](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.
    " Example: Expire objects with prefix 'logs/' after 30 days
    lo_s3->putbucketlifecycleconf(
        iv_bucket = iv_bucket_name
        io_lifecycleconfiguration = NEW /aws1/cl_s3_bucketlcconf(
            it_rules = it_lifecycle_rule ) ).
    MESSAGE 'Bucket lifecycle configuration set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketLifecycleConfiguration](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketLogging** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutBucketLogging.

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
using System;
using System.IO;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

/// <summary>
/// This example shows how to enable logging on an Amazon Simple Storage
/// Service (Amazon S3) bucket. You need to have two Amazon S3 buckets for
/// this example. The first is the bucket for which you wish to enable
/// logging, and the second is the location where you want to store the
/// logs.
/// </summary>
public class ServerAccessLogging
{
    private static IConfiguration _configuration = null!;

    public static async Task Main()
    {
        LoadConfig();

        string bucketName = _configuration["BucketName"];
        string logBucketName = _configuration["LogBucketName"];
        string logObjectKeyPrefix = _configuration["LogObjectKeyPrefix"];
        string accountId = _configuration["AccountId"];

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
```

```

        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        try
        {
            // Update bucket policy for target bucket to allow delivery of
logs to it.
            await SetBucketPolicyToAllowLogDelivery(
                client,
                bucketName,
                logBucketName,
                logObjectKeyPrefix,
                accountId);

            // Enable logging on the source bucket.
            await EnableLoggingAsync(
                client,
                bucketName,
                logBucketName,
                logObjectKeyPrefix);
        }
        catch (AmazonS3Exception e)
        {
            Console.WriteLine($"Error: {e.Message}");
        }
    }

    /// <summary>
    /// This method grants appropriate permissions for logging to the
    /// Amazon S3 bucket where the logs will be stored.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client which will be
used
    /// to apply the bucket policy.</param>
    /// <param name="sourceBucketName">The name of the source bucket.</param>
    /// <param name="logBucketName">The name of the bucket where logging
    /// information will be stored.</param>
    /// <param name="logPrefix">The logging prefix where the logs should be
delivered.</param>
    /// <param name="accountId">The account id of the account where the
source bucket exists.</param>
    /// <returns>Async task.</returns>
    public static async Task SetBucketPolicyToAllowLogDelivery(

```

```

        IAmazonS3 client,
        string sourceBucketName,
        string logBucketName,
        string logPrefix,
        string accountId)
    {
        var resourceArn = @""arn:aws:s3::" + logBucketName + "/" +
logPrefix + @"""";

        var newPolicy = @"{
            ""Statement"": [{
                ""Sid"": ""S3ServerAccessLogsPolicy"",
                ""Effect"": ""Allow"",
                ""Principal"": { ""Service"":
""logging.s3.amazonaws.com"" },
                ""Action"": [""s3:PutObject""],
                ""Resource"": ["" + resourceArn + @""],
                ""Condition"": {
                    ""ArnLike"": { ""aws:SourceArn"":
""arn:aws:s3::" + sourceBucketName + @"" },
                    ""StringEquals"": { ""aws:SourceAccount"": """" +
accountId + @"""" }
                }
            }]
        }";

        Console.WriteLine($"The policy to apply to bucket {logBucketName} to
enable logging.");
        Console.WriteLine(newPolicy);

        PutBucketPolicyRequest putRequest = new PutBucketPolicyRequest
        {
            BucketName = logBucketName,
            Policy = newPolicy,
        };
        await client.PutBucketPolicyAsync(putRequest);
        Console.WriteLine("Policy applied.");
    }

    /// <summary>
    /// This method enables logging for an Amazon S3 bucket. Logs will be
stored
    /// in the bucket you selected for logging. Selected prefix
    /// will be prepended to each log object.
    /// </summary>

```

```

    /// <param name="client">The initialized Amazon S3 client which will be
used
    /// to configure and apply logging to the selected Amazon S3 bucket.</
param>
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
you
    /// wish to enable logging.</param>
    /// <param name="logBucketName">The name of the Amazon S3 bucket where
logging
    /// information will be stored.</param>
    /// <param name="logObjectKeyPrefix">The prefix to prepend to each
    /// object key.</param>
    /// <returns>Async task.</returns>
    public static async Task EnableLoggingAsync(
        IAmazonS3 client,
        string bucketName,
        string logBucketName,
        string logObjectKeyPrefix)
    {
        Console.WriteLine($"Enabling logging for bucket {bucketName}.");
        var loggingConfig = new S3BucketLoggingConfig
        {
            TargetBucketName = logBucketName,
            TargetPrefix = logObjectKeyPrefix,
        };

        var putBucketLoggingRequest = new PutBucketLoggingRequest
        {
            BucketName = bucketName,
            LoggingConfig = loggingConfig,
        };
        await client.PutBucketLoggingAsync(putBucketLoggingRequest);
        Console.WriteLine($"Logging enabled.");
    }

    /// <summary>
    /// Loads configuration from settings files.
    /// </summary>
    public static void LoadConfig()
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
    }

```



```
        .AddJsonFile("settings.local.json", true) // Optionally, load
local settings.
        .Build();
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketLogging](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Exemple 1 : pour définir la journalisation des stratégies de compartiment

L'exemple `put-bucket-logging` suivant définit la stratégie de journalisation pour `amzn-s3-demo-bucket`. Tout d'abord, accordez l'autorisation au principal du service de journalisation dans votre stratégie de compartiment à l'aide de la commande `put-bucket-policy`.

```
aws s3api put-bucket-policy \
  --bucket amzn-s3-demo-bucket \
  --policy file://policy.json
```

Contenu de `policy.json` :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3ServerAccessLogsPolicy",
      "Effect": "Allow",
      "Principal": {"Service": "logging.s3.amazonaws.com"},
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/Logs/*",
      "Condition": {
        "ArnLike": {"aws:SourceARN": "arn:aws:s3:::SOURCE-BUCKET-NAME"},
        "StringEquals": {"aws:SourceAccount": "SOURCE-AWS-ACCOUNT-ID"}
      }
    }
  ]
}
```

```
}
```

Pour appliquer la stratégie de journalisation, utilisez `put-bucket-logging`.

```
aws s3api put-bucket-logging \  
  --bucket amzn-s3-demo-bucket \  
  --bucket-logging-status file:///logging.json
```

Contenu de `logging.json` :

```
{  
  "LoggingEnabled": {  
    "TargetBucket": "amzn-s3-demo-bucket",  
    "TargetPrefix": "Logs/"  
  }  
}
```

La commande `put-bucket-policy` est requise pour accorder les autorisations `s3:PutObject` au principal du service de journalisation.

Pour plus d'informations, consultez [Journalisation des accès au serveur Amazon S3](#) dans le Guide de l'utilisateur Amazon S3.

Exemple 2 : pour définir une stratégie de compartiment pour la journalisation des accès pour un seul utilisateur

L'exemple `put-bucket-logging` suivant définit la stratégie de journalisation pour `amzn-s3-demo-bucket`. L'AWS utilisateur `bob@example.com` aura un contrôle total sur les fichiers journaux, et personne d'autre n'y aura accès. Commencez par octroyer l'autorisation S3 avec `put-bucket-acl`.

```
aws s3api put-bucket-acl \  
  --bucket amzn-s3-demo-bucket \  
  --grant-write URI=http://acs.amazonaws.com/groups/s3/LogDelivery \  
  --grant-read-acp URI=http://acs.amazonaws.com/groups/s3/LogDelivery
```

Ensuite, appliquez la stratégie de journalisation avec `put-bucket-logging`.

```
aws s3api put-bucket-logging \  
  --bucket amzn-s3-demo-bucket \  
  --bucket-logging-status file:///logging.json
```

Contenu de logging.json :

```
{
  "LoggingEnabled": {
    "TargetBucket": "amzn-s3-demo-bucket",
    "TargetPrefix": "amzn-s3-demo-bucket-logs/",
    "TargetGrants": [
      {
        "Grantee": {
          "Type": "AmazonCustomerByEmail",
          "EmailAddress": "bob@example.com"
        },
        "Permission": "FULL_CONTROL"
      }
    ]
  }
}
```

La commande `put-bucket-acl` est requise pour accorder les autorisations nécessaires (écriture et lecture-acp) au système de livraison des journaux d'Amazon S3.

Pour plus d'informations, consultez [Journalisation des accès au serveur Amazon S3](#) dans le Guide du développeur Amazon S3.

- Pour plus de détails sur l'API, reportez-vous [PutBucketLogging](#) à la section Référence des AWS CLI commandes.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **PutBucketNotification** avec une CLI

Les exemples de code suivants illustrent comment utiliser `PutBucketNotification`.

CLI

AWS CLI

Il applique une configuration de notification à un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-configuration file://notification.json
```

`notification.json` est un fichier JSON du dossier en cours qui spécifie une rubrique SNS et un type d'événement à surveiller :

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

La rubrique SNS doit être associée à une politique IAM qui autorise Amazon S3 à y publier :

```
{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "example-statement-ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-west-2:123456789012:amzn-s3-demo-bucket",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
        }
      }
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketNotification](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : Cet exemple configure la configuration de la rubrique SNS pour l'événement S3 ObjectRemovedDelete et active les notifications pour le compartiment s3 donné

```
$topic = [Amazon.S3.Model.TopicConfiguration] @{
    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
    $topic
```

Exemple 2 : Cet exemple active les notifications ObjectCreatedAll pour le bucket donné qui l'envoient à la fonction Lambda.

```
$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
    LambdaFunctionConfiguration $lambdaConfig
```

Exemple 3 : cet exemple crée deux configurations Lambda différentes sur la base d'un suffixe clé différent et configure les deux en une seule commande.

```
#Lambda Config 1
```

```

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".ps1"}
            )
        }
    }
}

#Lambda Config 2

$secondLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = [Amazon.S3.EventType]::ObjectCreatedAll
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifysm"
    Id = "ObjectCreated-dada-json"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".json"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondLambdaConfig

```

- Pour plus de détails sur l'API, reportez-vous [PutBucketNotification](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : Cet exemple configure la configuration de la rubrique SNS pour l'événement S3 ObjectRemovedDelete et active les notifications pour le compartiment s3 donné

```

$topic = [Amazon.S3.Model.TopicConfiguration] @{
    Id = "delete-event"
    Topic = "arn:aws:sns:eu-west-1:123456789012:topic-1"
}

```

```

    Event = [Amazon.S3.EventType]::ObjectRemovedDelete
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -TopicConfiguration
    $topic

```

Exemple 2 : Cet exemple active les notifications ObjectCreatedAll pour le bucket donné qui l'envoient à la fonction Lambda.

```

$lambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:rdplock"
    Id = "ObjectCreated-Lambda"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".pem"}
            )
        }
    }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
    LambdaFunctionConfiguration $lambdaConfig

```

Exemple 3 : cet exemple crée deux configurations Lambda différentes sur la base d'un suffixe clé différent et configure les deux en une seule commande.

```

#Lambda Config 1

$firstLambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
    Events = "s3:ObjectCreated:*"
    FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifynet"
    Id = "ObjectCreated-dada-ps1"
    Filter = @{
        S3KeyFilter = @{
            FilterRules = @(
                @{Name="Prefix";Value="dada"}
                @{Name="Suffix";Value=".ps1"}
            )
        }
    }
}

```

```

    }
  }

#Lambda Config 2

$secondlambdaConfig = [Amazon.S3.Model.LambdaFunctionConfiguration] @{
  Events = [Amazon.S3.EventType]::ObjectCreatedAll
  FunctionArn = "arn:aws:lambda:eu-west-1:123456789012:function:verifyssm"
  Id = "ObjectCreated-dada-json"
  Filter = @{
    S3KeyFilter = @{
      FilterRules = @(
        @{Name="Prefix";Value="dada"}
        @{Name="Suffix";Value=".json"}
      )
    }
  }
}

Write-S3BucketNotification -BucketName amzn-s3-demo-bucket -
LambdaFunctionConfiguration $firstLambdaConfig,$secondlambdaConfig

```

- Pour plus de détails sur l'API, reportez-vous [PutBucketNotification](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketNotificationConfiguration** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser **PutBucketNotificationConfiguration**.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Traitement de notifications d'événements S3](#)
- [Envoyer des notifications d'événements à EventBridge](#)

.NET

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to enable notifications for an Amazon Simple
/// Storage Service (Amazon S3) bucket.
/// </summary>
public class EnableNotifications
{
    public static async Task Main()
    {
        const string bucketName = "amzn-s3-demo-bucket1";
        const string snsTopic = "arn:aws:sns:us-east-2:0123456789ab:bucket-
notify";
        const string sqsQueue = "arn:aws:sqs:us-
east-2:0123456789ab:Example_Queue";

        IAmazonS3 client = new AmazonS3Client(Amazon.RegionEndpoint.USEast2);
        await EnableNotificationAsync(client, bucketName, snsTopic,
sqsQueue);
    }

    /// <summary>
    /// This method makes the call to the PutBucketNotificationAsync method.
    /// </summary>
    /// <param name="client">An initialized Amazon S3 client used to call
    /// the PutBucketNotificationAsync method.</param>
    /// <param name="bucketName">The name of the bucket for which
    /// notifications will be turned on.</param>
```

```
/// <param name="snsTopic">The ARN for the Amazon Simple Notification
/// Service (Amazon SNS) topic associated with the S3 bucket.</param>
/// <param name="sqsQueue">The ARN of the Amazon Simple Queue Service
/// (Amazon SQS) queue to which notifications will be pushed.</param>
public static async Task EnableNotificationAsync(
    IAmazonS3 client,
    string bucketName,
    string snsTopic,
    string sqsQueue)
{
    try
    {
        // The bucket for which we are setting up notifications.
        var request = new PutBucketNotificationRequest()
        {
            BucketName = bucketName,
        };

        // Defines the topic to use when sending a notification.
        var topicConfig = new TopicConfiguration()
        {
            Events = new List<EventType> { EventType.ObjectCreatedCopy },
            Topic = snsTopic,
        };
        request.TopicConfigurations = new List<TopicConfiguration>
        {
            topicConfig,
        };
        request.QueueConfigurations = new List<QueueConfiguration>
        {
            new QueueConfiguration()
            {
                Events = new List<EventType>
{ EventType.ObjectCreatedPut },
                Queue = sqsQueue,
            },
        };

        // Now apply the notification settings to the bucket.
        PutBucketNotificationResponse response = await
client.PutBucketNotificationAsync(request);
    }
    catch (AmazonS3Exception ex)
    {
```

```
        Console.WriteLine($"Error: {ex.Message}");  
    }  
}  
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketNotificationConfiguration](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour activer les notifications spécifiées d'un compartiment

L'exemple `put-bucket-notification-configuration` suivant applique une configuration de notification à un compartiment nommé `amzn-s3-demo-bucket`. `notification.json` est un fichier JSON du dossier en cours qui spécifie une rubrique SNS et un type d'événement à surveiller.

```
aws s3api put-bucket-notification-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --notification-configuration file://notification.json
```

Contenu de `notification.json` :

```
{  
  "TopicConfigurations": [  
    {  
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-  
topic",  
      "Events": [  
        "s3:ObjectCreated:*"  
      ]  
    }  
  ]  
}
```

La rubrique SNS doit être associée à une politique IAM qui autorise Amazon S3 à y publier.

```
{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "example-statement-ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": [
        "SNS:Publish"
      ],
      "Resource": "arn:aws:sns:us-west-2:123456789012::s3-notification-
topic",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:s3:*:*:amzn-s3-demo-bucket"
        }
      }
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketNotificationConfiguration](#) à la section Référence des AWS CLI commandes.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketPolicy** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutBucketPolicy.

C++

Kit de développement logiciel (SDK) for C++

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                  const Aws::String &policyBody,
                                  const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
        Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
        s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}

//! Build a policy JSON string.
/*!
```

```

\param userArn: Aws user Amazon Resource Name (ARN).
    For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
\param bucketName: Name of a bucket.
\return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                           const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\", \n"
        "  \"Statement\": [\n"
        "    {\n"
        "      \"Sid\": \"1\", \n"
        "      \"Effect\": \"Allow\", \n"
        "      \"Principal\": {\n"
        "        \"AWS\": \"\"
        + userArn +
        "\"\"\"      }, \n"
        "      \"Action\": [ \"s3:getObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3::\"
        + bucketName +
        \"/*\" ] \n"
        "    } \n"
        "  ] \n"
        "}";
}

```

- Pour plus de détails sur l'API, reportez-vous [PutBucketPolicy](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Cet exemple permet à tous les utilisateurs de récupérer n'importe quel objet dans `amzn-s3-demo-bucket`, à l'exception de ceux du `MySecretFolder`. Il accorde `put` également une `delete` autorisation à l'utilisateur `root` du AWS compte `1234-5678-9012` :

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://policy.json
```

policy.json:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    },
    {
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/MySecretFolder/*"
    },
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:root"
      },
      "Action": [
        "s3:DeleteObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  ]
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketPolicy](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.List;

import com.fasterxml.jackson.core.JsonParser;
import com.fasterxml.jackson.databind.ObjectMapper;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */
public class SetBucketPolicy {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <polFile>

            Where:
```



```

        bucketName - The Amazon S3 bucket to set the policy on.
        polFile - A JSON file containing the policy (see the Amazon S3
Readme for an example).\s
        """;

    if (args.length != 2) {
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String polFile = args[1];
    String policyText = getBucketPolicyFromFile(polFile);
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setPolicy(s3, bucketName, policyText);
    s3.close();
}

/**
 * Sets the policy for an Amazon S3 bucket.
 *
 * @param s3          the {@link S3Client} object used to interact with the
Amazon S3 service
 * @param bucketName the name of the Amazon S3 bucket
 * @param policyText the text of the policy to be set on the bucket
 * @throws S3Exception if there is an error setting the bucket policy
 */
public static void setPolicy(S3Client s3, String bucketName, String
policyText) {
    System.out.println("Setting policy:");
    System.out.println("----");
    System.out.println(policyText);
    System.out.println("----");
    System.out.format("On Amazon S3 bucket: \"%s\"\n", bucketName);

    try {
        PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
            .bucket(bucketName)
            .policy(policyText)
            .build();
    }
}

```

```
s3.putBucketPolicy(policyReq);

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}

System.out.println("Done!");
}

/**
 * Retrieves the bucket policy from a specified file.
 *
 * @param policyFile the path to the file containing the bucket policy
 * @return the content of the bucket policy file as a string
 */
public static String getBucketPolicyFromFile(String policyFile) {
    StringBuilder fileText = new StringBuilder();
    try {
        List<String> lines = Files.readAllLines(Paths.get(policyFile),
StandardCharsets.UTF_8);
        for (String line : lines) {
            fileText.append(line);
        }

    } catch (IOException e) {
        System.out.format("Problem reading file: \"%s\"", policyFile);
        System.out.println(e.getMessage());
    }

    try {
        final JsonParser parser = new
ObjectMapper().getFactory().createParser(fileText.toString());
        while (parser.nextToken() != null) {
        }

    } catch (IOException jpe) {
        jpe.printStackTrace();
    }
    return fileText.toString();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketPolicy](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Ajoutez la politique.

```
import {
  PutBucketPolicyCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Grant an IAM role GetObject access to all of the objects
 * in the provided bucket.
 * @param {{ bucketName: string, iamRoleArn: string }}
 */
export const main = async ({ bucketName, iamRoleArn }) => {
  const client = new S3Client({});
  const command = new PutBucketPolicyCommand({
    // This is a resource-based policy. For more information on resource-based
    // policies,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/
    // access_policies.html#policies_resource-based.
    Policy: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
```

```

        AWS: iamRoleArn,
      },
      Action: "s3:GetObject",
      Resource: `arn:aws:s3:::${bucketName}/*`,
    },
  ],
}),
// Apply the preceding policy to this bucket.
Bucket: bucketName,
});

try {
  await client.send(command);
  console.log(
    `GetObject access to the bucket "${bucketName}" was granted to the provided
IAM role.`
  );
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "MalformedPolicy"
  ) {
    console.error(
      `Error from S3 while setting the bucket policy for the bucket
"${bucketName}". The policy was malformed.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while setting the bucket policy for the bucket
"${bucketName}". ${caught.name}: ${caught.message}`
    );
  } else {
    throw caught;
  }
}
};

```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [PutBucketPolicy](#) à la section Référence des AWS SDK pour JavaScript API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def put_policy(self, policy):
        """
        Apply a security policy to the bucket. Policies control users' ability
        to perform specific actions, such as listing the objects in the bucket.

        :param policy: The policy to apply to the bucket.
        """
        try:
            self.bucket.Policy().put(Policy=json.dumps(policy))
            logger.info("Put policy %s for bucket '%s'.", policy,
self.bucket.name)
        except ClientError:
            logger.exception("Couldn't apply policy to bucket '%s'.",
self.bucket.name)
            raise
```

- Pour plus de détails sur l'API, consultez [PutBucketPolicy](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
    #{e.message}"
    false
  end
end
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketPolicy](#) à la section Référence des AWS SDK pour Ruby API.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.
    " Example policy JSON string
    " iv_policy = '{"Version":"2012-10-17",          "Statement":
[{"Effect":"Allow","Principal":{"AWS":"arn:aws:iam::123456789012:user/
user"},"Action":["s3:GetObject"],"Resource":["arn:aws:s3:::bucketname/*"]}]}'
    lo_s3->putbucketpolicy(
        iv_bucket = iv_bucket_name
        iv_policy = iv_policy ).
    MESSAGE 'Bucket policy set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketPolicy](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketReplication** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutBucketReplication.

CLI

AWS CLI

Pour configurer la réplication d'un compartiment S3

L'exemple `put-bucket-replication` suivant applique une configuration de réplication au compartiment S3 spécifié.

```
aws s3api put-bucket-replication \  
  --bucket amzn-s3-demo-bucket1 \  
  --replication-configuration file://replication.json
```

Contenu de `replication.json` :

```
{  
  "Role": "arn:aws:iam::123456789012:role/s3-replication-role",  
  "Rules": [  
    {  
      "Status": "Enabled",  
      "Priority": 1,  
      "DeleteMarkerReplication": { "Status": "Disabled" },  
      "Filter" : { "Prefix": "" },  
      "Destination": {  
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket2"  
      }  
    }  
  ]  
}
```

La gestion des versions doit être activée dans le compartiment de destination. Le rôle spécifié doit être autorisé à écrire dans le compartiment de destination et avoir une relation de confiance qui permet à Amazon S3 d'endosser le rôle.

Exemple de politique d'autorisation de rôle :

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetReplicationConfiguration",  
        "s3:ListBucket"  
      ],  
      "Resource": [  
        "arn:aws:s3:::amzn-s3-demo-bucket1"  
      ]  
    }  
  ]  
}
```



```

    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObjectVersion",
        "s3:GetObjectVersionAcl",
        "s3:GetObjectVersionTagging"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket1/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ReplicateObject",
        "s3:ReplicateDelete",
        "s3:ReplicateTags"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket2/*"
    }
  ]
}

```

Exemple de politique de relation de confiance :

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

Cette commande ne produit aucune sortie.

Pour plus d'informations, consultez [Cette rubrique](#) dans le Guide de l'utilisateur Amazon Simple Storage Service Console.

- Pour plus de détails sur l'API, reportez-vous [PutBucketReplication](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Sets the replication configuration for an Amazon S3 bucket.
 *
 * @param s3Client          the S3Client instance to use for the operation
 * @param sourceBucketName the name of the source bucket
 * @param destBucketName   the name of the destination bucket
 * @param destinationBucketARN the Amazon Resource Name (ARN) of the
destination bucket
 * @param roleARN          the ARN of the IAM role to use for the
replication configuration
 */
public static void setReplication(S3Client s3Client, String sourceBucketName,
String destBucketName, String destinationBucketARN, String roleARN) {
    try {
        Destination destination = Destination.builder()
            .bucket(destinationBucketARN)
            .storageClass(StorageClass.STANDARD)
            .build();

        // Define a prefix filter for replication.
        ReplicationRuleFilter ruleFilter = ReplicationRuleFilter.builder()
            .prefix("documents/")
            .build();

        // Define delete marker replication setting.
        DeleteMarkerReplication deleteMarkerReplication =
DeleteMarkerReplication.builder()
            .status(DeleteMarkerReplicationStatus.DISABLED)
```

```
        .build();

// Create the replication rule.
ReplicationRule replicationRule = ReplicationRule.builder()
    .priority(1)
    .filter(ruleFilter)
    .status(ReplicationRuleStatus.ENABLED)
    .deleteMarkerReplication(deleteMarkerReplication)
    .destination(destination)
    .build();

List<ReplicationRule> replicationRuleList = new ArrayList<>();
replicationRuleList.add(replicationRule);

// Define the replication configuration with IAM role.
ReplicationConfiguration configuration =
ReplicationConfiguration.builder()
    .role(roleARN)
    .rules(replicationRuleList)
    .build();

// Apply the replication configuration to the source bucket.
PutBucketReplicationRequest replicationRequest =
PutBucketReplicationRequest.builder()
    .bucket(sourceBucketName)
    .replicationConfiguration(configuration)
    .build();

s3Client.putBucketReplication(replicationRequest);
System.out.println("Replication configuration set successfully.");

} catch (IllegalArgumentException e) {
    System.err.println("Configuration error: " + e.getMessage());
} catch (S3Exception e) {
    System.err.println("S3 Exception: " +
e.awsErrorDetails().errorMessage());
    System.err.println("Status Code: " + e.statusCode());
    System.err.println("Error Code: " + e.awsErrorDetails().errorCode());

} catch (SdkException e) {
    System.err.println("SDK Exception: " + e.getMessage());
}
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketReplication](#) à la section Référence des AWS SDK for Java 2.x API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : Cet exemple définit une configuration de réplication avec une règle unique permettant de répliquer dans le compartiment « amzn-s3-demo-bucket » tous les nouveaux objets créés avec le préfixe de nom clé « » dans le compartiment « amzn-s3-demo-bucket ». TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Exemple 2 : Cet exemple définit une configuration de réplication avec plusieurs règles permettant de répliquer vers le compartiment « amzn-s3-demo-bucket » tous les nouveaux objets créés avec le préfixe de nom de clé « » ou « ». TaxDocs OtherDocs Les préfixes de clés ne doivent pas se chevaucher.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }
```

```
$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

Exemple 3 : Cet exemple met à jour la configuration de réplication sur le compartiment spécifié afin de désactiver la règle contrôlant la réplication des objets portant le préfixe de nom clé « » vers le compartiment TaxDocs « amzn-s3-demo-bucket ».

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketReplication](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : Cet exemple définit une configuration de réplication avec une règle unique permettant de répliquer dans le compartiment « amzn-s3-demo-bucket » tous les nouveaux objets créés avec le préfixe de nom clé « » dans le compartiment « amzn-s3-demo-bucket ». TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Exemple 2 : Cet exemple définit une configuration de réplication avec plusieurs règles permettant de répliquer vers le compartiment « amzn-s3-demo-bucket » tous les nouveaux objets créés avec le préfixe de nom de clé « » ou « ». TaxDocs OtherDocs Les préfixes de clés ne doivent pas se chevaucher.

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$rule2 = New-Object Amazon.S3.Model.ReplicationRule
$rule2.ID = "Rule-2"
$rule2.Status = "Enabled"
$rule2.Prefix = "OtherDocs"
$rule2.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }
```

```
$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1,$rule2
}

Write-S3BucketReplication @params
```

Exemple 3 : Cet exemple met à jour la configuration de réplication sur le compartiment spécifié afin de désactiver la règle contrôlant la réplication des objets portant le préfixe de nom clé « » vers le compartiment TaxDocs « amzn-s3-demo-bucket ».

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Disabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketReplication](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **PutBucketRequestPayment** avec une CLI

Les exemples de code suivants illustrent comment utiliser PutBucketRequestPayment.

CLI

AWS CLI

Exemple 1 : pour activer la configuration « Paiement par le demandeur » d'un compartiment

L'exemple `put-bucket-request-payment` suivant active `requester` pays pour le compartiment spécifié.

```
aws s3api put-bucket-request-payment \  
  --bucket amzn-s3-demo-bucket \  
  --request-payment-configuration '{"Payer":"Requester"}'
```

Cette commande ne produit aucune sortie.

Exemple 2 : pour désactiver la configuration « Paiement par le demandeur » d'un compartiment

L'exemple `put-bucket-request-payment` suivant désactive `requester` pays pour le compartiment spécifié.

```
aws s3api put-bucket-request-payment \  
  --bucket amzn-s3-demo-bucket \  
  --request-payment-configuration '{"Payer":"BucketOwner"}'
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [PutBucketRequestPayment](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : met à jour la configuration du paiement des demandes pour le compartiment nommé « `amzn-s3-demo-bucket` » afin que le téléchargement soit facturé à la personne demandant des téléchargements depuis le compartiment. Par défaut, le propriétaire du compartiment paie les téléchargements. Pour rétablir la valeur par défaut du paiement de la demande, utilisez « `BucketOwner` » pour le paramètre `RequestPaymentConfiguration_Payer`.


```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketRequestPayment](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : met à jour la configuration du paiement des demandes pour le compartiment nommé « amzn-s3-demo-bucket » afin que le téléchargement soit facturé à la personne demandant des téléchargements depuis le compartiment. Par défaut, le propriétaire du compartiment paie les téléchargements. Pour rétablir la valeur par défaut du paiement de la demande, utilisez « BucketOwner » pour le paramètre RequestPaymentConfiguration _Payer.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketRequestPayment](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisation de **PutBucketTagging** avec une CLI

Les exemples de code suivants illustrent comment utiliser PutBucketTagging.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Débuter avec le stockage d'objets](#)

CLI

AWS CLI

La commande suivante applique une configuration de balisage à un compartiment nommé amzn-s3-demo-bucket :

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://tagging.json
```

Le fichier `tagging.json` est un document JSON du dossier en cours qui spécifie des balises :

```
{
  "TagSet": [
    {
      "Key": "organization",
      "Value": "marketing"
    }
  ]
}
```

Ou appliquez une configuration de balisage à `amzn-s3-demo-bucket` directement depuis la ligne de commande :

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging 'TagSet=[{Key=organization,Value=marketing}]'
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketTagging](#) à la section Référence des AWS CLI commandes.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande applique deux balises à un compartiment nommé **cloudtrail-test-2018** : une balise avec une clé `Stage` et une valeur `Test`, et une balise avec une clé `Environment` et une valeur `Alpha`. Pour vérifier que les balises ont été ajoutées au compartiment, exécutez **Get-S3BucketTagging -BucketName bucket_name**. Les résultats doivent indiquer les balises que vous avez appliquées au compartiment lors de la première commande. Notez que **Write-S3BucketTagging** remplace la totalité du jeu de balises existant sur un compartiment. Pour ajouter ou supprimer des balises individuelles, exécutez les applets de commande Resource Groups et Tagging API, ainsi que **Add-RGResourceTag** et **Remove-RGResourceTag**. Vous pouvez également utiliser l'éditeur de balises dans la console AWS de gestion pour gérer les balises de compartiment S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";  
    Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Exemple 2 : cette commande dirige un compartiment nommé **cloudtrail-test-2018** vers l'applet de commande **Write-S3BucketTagging**. Il applique Stage:Production des balises et Department:Finance au compartiment. Notez que **Write-S3BucketTagging** remplace la totalité du jeu de balises existant sur un compartiment.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging  
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";  
    Value="Finance" } )
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketTagging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande applique deux balises à un compartiment nommé **cloudtrail-test-2018** : une balise avec une clé Stage et une valeur Test, et une balise avec une clé Environment et une valeur Alpha. Pour vérifier que les balises ont été ajoutées au compartiment, exécutez **Get-S3BucketTagging -BucketName bucket_name**. Les résultats doivent indiquer les balises que vous avez appliquées au compartiment lors de la première commande. Notez que **Write-S3BucketTagging** remplace la totalité du jeu de balises existant sur un compartiment. Pour ajouter ou supprimer des balises individuelles, exécutez les applets de commande Resource Groups et Tagging API, ainsi que **Add-RGResourceTag** et **Remove-RGResourceTag**. Vous pouvez également utiliser l'éditeur de balises dans la console AWS de gestion pour gérer les balises de compartiment S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";  
    Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Exemple 2 : cette commande dirige un compartiment nommé **cloudtrail-test-2018** vers l'applet de commande **Write-S3BucketTagging**. Il applique Stage:Production des balises et Department:Finance au compartiment. Notez que **Write-S3BucketTagging** remplace la totalité du jeu de balises existant sur un compartiment.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging  
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";  
    Value="Finance" } )
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketTagging](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketVersioning** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutBucketVersioning.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Débuter avec le stockage d'objets](#)

CLI

AWS CLI

La commande suivante active la gestion des versions sur un compartiment nommé amzn-s3-demo-bucket :

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled
```

La commande suivante active la gestion des versions et utilise un code MFA.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled --mfa "SERIAL 123456"
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketVersioning](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Enables bucket versioning for the specified S3 bucket.
 *
 * @param s3Client the S3 client to use for the operation
 * @param bucketName the name of the S3 bucket to enable versioning for
 */
public static void enableBucketVersioning(S3Client s3Client, String
bucketName){
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    s3Client.putBucketVersioning(versioningRequest);
    System.out.println("Bucket versioning has been enabled for "+bucketName);
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketVersioning](#) à la section Référence des AWS SDK for Java 2.x API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : la commande active la gestion des versions pour le compartiment S3 donné.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketVersioning](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : la commande active la gestion des versions pour le compartiment S3 donné.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketVersioning](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
  " Example: Enable versioning on a bucket  
  " iv_status = 'Enabled'  
  lo_s3->putbucketversioning(  
    iv_bucket = iv_bucket_name  
    io_versioningconfiguration = NEW /aws1/cl_s3_versioningconf(  
      iv_status = iv_status ) ).  
  MESSAGE 'Bucket versioning enabled.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.
```

```
MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketVersioning](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketWebsite** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutBucketWebsite.

.NET

SDK pour .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Put the website configuration.  
PutBucketWebsiteRequest putRequest = new  
PutBucketWebsiteRequest()  
{  
    BucketName = bucketName,  
    WebsiteConfiguration = new WebsiteConfiguration()  
    {  
        IndexDocumentSuffix = indexDocumentSuffix,  
        ErrorDocument = errorDocument,  
    },  
};  
PutBucketWebsiteResponse response = await  
client.PutBucketWebsiteAsync(putRequest);
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketWebsite](#) à la section Référence des AWS SDK pour .NET API.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::String &indexPage, const Aws::String
                                   &errorPage,
                                   const Aws::S3::S3ClientConfiguration
                                   &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
        client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                   << outcome.GetError().GetMessage() << std::endl;
    } else {
```



```
        std::cout << "Success: Set website configuration for bucket '"  
                    << bucketName << "'" << std::endl;  
    }  
  
    return outcome.IsSuccess();  
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketWebsite](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Il applique une configuration de site web statique à un compartiment nommé `amzn-s3-demo-bucket` :

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-  
configuration file://website.json
```

Le fichier `website.json` est un document JSON du dossier en cours qui spécifie les pages d'index et d'erreur du site web :

```
{  
  "IndexDocument": {  
    "Suffix": "index.html"  
  },  
  "ErrorDocument": {  
    "Key": "error.html"  
  }  
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketWebsite](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.IndexDocument;
import software.amazon.awssdk.services.s3.model.PutBucketWebsiteRequest;
import software.amazon.awssdk.services.s3.model.WebsiteConfiguration;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.regions.Region;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
 * started.html
 */

public class SetWebsiteConfiguration {
    public static void main(String[] args) {
        final String usage = ""

            Usage:    <bucketName> [indexdoc]\s

            Where:
                bucketName    - The Amazon S3 bucket to set the website
configuration on.\s
                indexdoc    - The index document, ex. 'index.html'
                            If not specified, 'index.html' will be set.

            """;

        if (args.length != 1) {
```

```
        System.out.println(usage);
        System.exit(1);
    }

    String bucketName = args[0];
    String indexDoc = "index.html";
    Region region = Region.US_EAST_1;
    S3Client s3 = S3Client.builder()
        .region(region)
        .build();

    setWebsiteConfig(s3, bucketName, indexDoc);
    s3.close();
}

/**
 * Sets the website configuration for an Amazon S3 bucket.
 *
 * @param s3 The {@link S3Client} instance to use for the AWS SDK operations.
 * @param bucketName The name of the S3 bucket to configure.
 * @param indexDoc The name of the index document to use for the website
 * configuration.
 */
public static void setWebsiteConfig(S3Client s3, String bucketName, String
indexDoc) {
    try {
        WebsiteConfiguration websiteConfig = WebsiteConfiguration.builder()
            .indexDocument(IndexDocument.builder().suffix(indexDoc).build())
            .build();

        PutBucketWebsiteRequest pubWebsiteReq =
PutBucketWebsiteRequest.builder()
            .bucket(bucketName)
            .websiteConfiguration(websiteConfig)
            .build();

        s3.putBucketWebsite(pubWebsiteReq);
        System.out.println("The call was successful");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketWebsite](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Définissez la configuration du site web.

```
import {
  PutBucketWebsiteCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Configure an Amazon S3 bucket to serve a static website.
 * Website access must also be granted separately. For more information
 * on setting the permissions for website access, see
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/
 * WebsiteAccessPermissionsReqd.html.
 */
@param {{ bucketName: string }}
*/
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutBucketWebsiteCommand({
    Bucket: bucketName,
    WebsiteConfiguration: {
      ErrorDocument: {
        // The object key name to use when a 4XX class error occurs.
        Key: "error.html",
      },
    },
  });
```

```
    },
    IndexDocument: {
      // A suffix that is appended to a request when the request is
      // for a directory.
      Suffix: "index.html",
    },
  },
});

try {
  await client.send(command);
  console.log(
    `The bucket "${bucketName}" has been configured as a static website.`
  );
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while configuring the bucket "${bucketName}" as a static
website. The bucket doesn't exist.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while configuring the bucket "${bucketName}" as a static
website. ${caught.name}: ${caught.message}`
    );
  } else {
    throw caught;
  }
}
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [PutBucketWebsite](#) à la section Référence des AWS SDK pour JavaScript API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : la commande active l'hébergement du site Web pour le compartiment donné avec le document d'index « index.html » et le document d'erreur « error.html ».

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'  
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketWebsite](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : la commande active l'hébergement du site Web pour le compartiment donné avec le document d'index « index.html » et le document d'erreur « error.html ».

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'  
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketWebsite](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
require 'aws-sdk-s3'  
  
# Wraps Amazon S3 bucket website actions.  
class BucketWebsiteWrapper  
  attr_reader :bucket_website
```

```
# @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
configured with an existing bucket.
def initialize(bucket_website)
  @bucket_website = bucket_website
end

# Sets a bucket as a static website.
#
# @param index_document [String] The name of the index document for the
website.
# @param error_document [String] The name of the error document to show for 4XX
errors.
# @return [Boolean] True when the bucket is configured as a website; otherwise,
false.
def set_website(index_document, error_document)
  @bucket_website.put(
    website_configuration: {
      index_document: { suffix: index_document },
      error_document: { key: error_document }
    }
  )
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
why: #{e.message}"
  false
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

  wrapper = BucketWebsiteWrapper.new(Aws::S3::BucketWebsite.new(bucket_name))
  return unless wrapper.set_website(index_document, error_document)

  puts "Successfully configured bucket #{bucket_name} as a static website."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketWebsite](#) à la section Référence des AWS SDK pour Ruby API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutObject** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutObject.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans les exemples de code suivants :

- [Principes de base](#)
- [Débuter avec le stockage d'objets](#)
- [Réalisation de demandes conditionnelles](#)
- [Suivi des chargements et des téléchargements](#)
- [Utilisation de l'intégrité des objets Amazon S3](#)

.NET

SDK pour .NET (v4)

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>  
/// Shows how to upload a file from the local computer to an Amazon S3  
/// bucket.
```



```
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to which the object
/// will be uploaded.</param>
/// <param name="objectName">The object to upload.</param>
/// <param name="filePath">The path, including file name, of the object
/// on the local computer to upload.</param>
/// <returns>A boolean value indicating the success or failure of the
/// upload procedure.</returns>
public async Task<bool> UploadFileAsync(
    string bucketName,
    string objectName,
    string filePath)
{
    try
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
        };

        var response = await _amazonS3.PutObjectAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error uploading {objectName}: {ex.Message}");
        return false;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS SDK pour .NET API.

SDK pour .NET

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Chargez un objet avec un chiffrement côté serveur.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket with server-side encryption enabled.
/// </summary>
public class ServerSideEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "samplefile.txt";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USWest2.
        IAmazonS3 client = new AmazonS3Client();

        await WritingAnObjectAsync(client, bucketName, keyName);
    }

    /// <summary>
    /// Upload a sample object include a setting for encryption.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// to upload a file and apply server-side encryption.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// encrypted object will reside.</param>
    /// <param name="keyName">The name for the object that you want to
    /// create in the supplied bucket.</param>
    public static async Task WritingAnObjectAsync(IAmazonS3 client, string
bucketName, string keyName)
    {
        try
        {
            var putRequest = new PutObjectRequest
            {
                BucketName = bucketName,
```

```

        Key = keyName,
        ContentBody = "sample text",
        ServerSideEncryptionMethod =
ServerSideEncryptionMethod.AES256,
    };

    var putResponse = await client.PutObjectAsync(putRequest);

    // Determine the encryption state of an object.
    GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest
    {
        BucketName = bucketName,
        Key = keyName,
    };
    GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);
    ServerSideEncryptionMethod objectEncryption =
response.ServerSideEncryptionMethod;

    Console.WriteLine($"Encryption method used: {0}",
objectEncryption.ToString());
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: '{ex.Message}' when writing an
object");
    }
}
}

```

Placez un objet à l'aide d'une demande conditionnelle.

```

/// <summary>
/// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
/// </summary>
/// <param name="objectKey">The key of the object to upload.</param>
/// <param name="bucket">The source bucket of the object.</param>
/// <param name="content">The content to upload as a string.</param>

```

```
/// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
{
    try
    {
        var putObjectRequest = new PutObjectRequest
        {
            BucketName = bucket,
            Key = objectKey,
            ContentBody = content,
            IfNoneMatch = "*"
        };

        var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
        _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
        return putResult.ETag;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional write failed: Precondition
failed");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return string.Empty;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS SDK pour .NET API.

Bash

AWS CLI avec le script Bash

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function copy_file_to_bucket
#
# This function creates a file in the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket to copy the file to.
#     $2 - The path and file name of the local file to copy to the bucket.
#     $3 - The key (name) to call the copy of the file in the bucket.
#
# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function copy_file_to_bucket() {
    local response bucket_name source_file destination_file_name
    bucket_name=$1
    source_file=$2
    destination_file_name=$3

    response=$(aws s3api put-object \
        --bucket "$bucket_name" \
        --body "$source_file" \
```

```
--key "$destination_file_name")

# shellcheck disable=SC2181
if [[ ${?} -ne 0 ]]; then
    errecho "ERROR: AWS reports put-object operation failed.\n$response"
    return 1
fi
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS CLI commandes.

C++

SDK pour C++

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
    needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                       fileName.c_str(),
                                       std::ios_base::in |
                                       std::ios_base::binary);
```

```
if (!*inputData) {
    std::cerr << "Error unable to read file " << fileName << std::endl;
    return false;
}

request.SetBody(inputData);

Aws::S3::Model::PutObjectOutcome outcome =
    s3Client.PutObject(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: putObject: " <<
        outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Added object '" << fileName << "' to bucket '"
        << bucketName << "'.";
}

return outcome.IsSuccess();
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

Exemple 1 : pour charger un objet dans Amazon S3

La commande `put-object` suivante charge un objet dans Amazon S3.

```
aws s3api put-object \
    --bucket amzn-s3-demo-bucket \
    --key my-dir/MySampleImage.png \
    --body MySampleImage.png
```

Pour plus d'informations sur le téléchargement d'objets, consultez [Uploading Objects < http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html >](http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html) dans le manuel Amazon S3 Developer Guide.

Exemple 2 : pour charger un fichier vidéo dans Amazon S3

La commande `put-object` suivante charge un fichier vidéo.

```
aws s3api put-object \  
  --bucket amzn-s3-demo-bucket \  
  --key my-dir/big-video-file.mp4 \  
  --body /media/videos/f-sharp-3-data-services.mp4
```

Pour plus d'informations sur le téléchargement d'objets, consultez [Uploading Objects](http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html) < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html> > dans le manuel Amazon S3 Developer Guide.

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Placez un objet dans un compartiment à l'aide de l'API de bas niveau.

```
import (  
  "bytes"  
  "context"  
  "errors"  
  "fmt"  
  "io"  
  "log"  
  "os"  
  "time"  
  
  "github.com/aws/aws-sdk-go-v2/aws"  
  "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
  "github.com/aws/aws-sdk-go-v2/service/s3"
```



```
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
            Body:   file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
                log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
                    fileName, bucketName, objectKey, err)
            }
        } else {
            err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                    aws.String(objectKey)}, time.Minute)
            if err != nil {
```

```
    log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
  }
}
}
return err
}
```

Chargez un objet dans un compartiment à l'aide d'un gestionnaire de transfert.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
    string, contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:      aws.String(bucket),
        Key:         aws.String(key),
        Body:        bytes.NewReader([]byte(contents)),
```

```
ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
}
output, err := actor.S3Manager.Upload(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
} else {
    err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
            bucket)
    } else {
        outKey = *output.Key
    }
}
return outKey, err
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Chargez un fichier dans un compartiment à l'aide d'un [S3Client](#).

```

/**
 * Uploads a local file to an AWS S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket to upload the file to
 * @param key         the key (object name) to use for the uploaded file
 * @param objectPath  the local file path of the file to be uploaded
 * @return a {@link CompletableFuture} that completes with the {@link
PutObjectResponse} when the upload is successful, or throws a {@link
RuntimeException} if the upload fails
 */
public CompletableFuture<PutObjectResponse> uploadLocalFileAsync(String
bucketName, String key, String objectPath) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<PutObjectResponse> response =
getAsyncClient().putObject(objectRequest,
AsyncRequestBody.fromFile(Paths.get(objectPath)));
    return response.whenComplete((resp, ex) -> {
        if (ex != null) {
            throw new RuntimeException("Failed to upload file", ex);
        }
    });
}

```

Utilisez un [S3TransferManager](#) pour [charger un fichier](#) dans un bucket. Consultez le [fichier complet](#) et le [test](#).

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileUpload;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;
import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.util.UUID;

```

```
public String uploadFile(S3TransferManager transferManager, String
bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

Chargez un objet dans un compartiment et définissez des étiquettes à l'aide d'un [S3Client](#).

```
/**
 * Puts tags on an Amazon S3 object.
 *
 * @param s3 An {@link S3Client} object that represents the Amazon S3 client.
 * @param bucketName The name of the Amazon S3 bucket.
 * @param objectKey The key of the Amazon S3 object.
 * @param objectPath The file path of the object to be uploaded.
 */
public static void putS3ObjectTags(S3Client s3, String bucketName, String
objectKey, String objectPath) {
    try {
        Tag tag1 = Tag.builder()
            .key("Tag 1")
            .value("This is tag 1")
            .build();

        Tag tag2 = Tag.builder()
            .key("Tag 2")
            .value("This is tag 2")
            .build();

        List<Tag> tags = new ArrayList<>();
        tags.add(tag1);
        tags.add(tag2);
    }
}
```

```
        Tagging allTags = Tagging.builder()
            .tagSet(tags)
            .build();

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .tagging(allTags)
            .build();

        s3.putObject(putOb,
RequestBody.fromBytes(getObjectFile(objectPath)));

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}

/**
 * Updates the tags associated with an object in an Amazon S3 bucket.
 *
 * @param s3 an instance of the S3Client class, which is used to interact
with the Amazon S3 service
 * @param bucketName the name of the S3 bucket containing the object
 * @param objectKey the key (or name) of the object in the S3 bucket
 * @throws S3Exception if there is an error updating the object's tags
 */
public static void updateObjectTags(S3Client s3, String bucketName, String
objectKey) {
    try {
        GetObjectTaggingRequest taggingRequest =
GetObjectTaggingRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectTaggingResponse getTaggingRes =
s3.getObjectTagging(taggingRequest);
        List<Tag> obTags = getTaggingRes.tagSet();
        for (Tag sinTag : obTags) {
            System.out.println("The tag key is: " + sinTag.key());
            System.out.println("The tag value is: " + sinTag.value());
        }
    }
```

```
// Replace the object's tags with two new tags.
Tag tag3 = Tag.builder()
    .key("Tag 3")
    .value("This is tag 3")
    .build();

Tag tag4 = Tag.builder()
    .key("Tag 4")
    .value("This is tag 4")
    .build();

List<Tag> tags = new ArrayList<>();
tags.add(tag3);
tags.add(tag4);

Tagging updatedTags = Tagging.builder()
    .tagSet(tags)
    .build();

PutObjectTaggingRequest taggingRequest1 =
PutObjectTaggingRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .tagging(updatedTags)
    .build();

s3.putObjectTagging(taggingRequest1);
GetObjectTaggingResponse getTaggingRes2 =
s3.getObjectTagging(taggingRequest);
List<Tag> modTags = getTaggingRes2.tagSet();
for (Tag sinTag : modTags) {
    System.out.println("The tag key is: " + sinTag.key());
    System.out.println("The tag value is: " + sinTag.value());
}

} catch (S3Exception e) {
    System.err.println(e.getMessage());
    System.exit(1);
}

}

/**
 * Retrieves the contents of a file as a byte array.
```

```
*
* @param filePath the path of the file to be read
* @return a byte array containing the contents of the file, or null if an
error occurs
*/
private static byte[] getObjectFile(String filePath) {
    FileInputStream fileInputStream = null;
    byte[] byteArray = null;

    try {
        File file = new File(filePath);
        byteArray = new byte[(int) file.length()];
        fileInputStream = new FileInputStream(file);
        fileInputStream.read(byteArray);

    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if (fileInputStream != null) {
            try {
                fileInputStream.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    return byteArray;
}
}
```

Chargez un objet dans un compartiment et définissez les métadonnées à l'aide d'un [S3Client](#).

```
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.File;
import java.util.HashMap;
```



```
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class PutObjectMetadata {
    public static void main(String[] args) {
        final String USAGE = ""

            Usage:
                <bucketName> <objectKey> <objectPath>\s

            Where:
                bucketName - The Amazon S3 bucket to upload an object into.
                objectKey - The object to upload (for example, book.pdf).
                objectPath - The path where the file is located (for example, C:/
AWS/book2.pdf).\s
            """;

        if (args.length != 3) {
            System.out.println(USAGE);
            System.exit(1);
        }

        String bucketName = args[0];
        String objectKey = args[1];
        String objectPath = args[2];
        System.out.println("Putting object " + objectKey + " into bucket " +
bucketName);
        System.out.println("  in bucket: " + bucketName);
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        putS3Object(s3, bucketName, objectKey, objectPath);
        s3.close();
    }
}
```

```

/**
 * Uploads an object to an Amazon S3 bucket with metadata.
 *
 * @param s3 the S3Client object used to interact with the Amazon S3 service
 * @param bucketName the name of the S3 bucket to upload the object to
 * @param objectKey the name of the object to be uploaded
 * @param objectPath the local file path of the object to be uploaded
 */
public static void putS3Object(S3Client s3, String bucketName, String
objectKey, String objectPath) {
    try {
        Map<String, String> metadata = new HashMap<>();
        metadata.put("author", "Mary Doe");
        metadata.put("version", "1.0.0.0");

        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .metadata(metadata)
            .build();

        s3.putObject(putOb, RequestBody.fromFile(new File(objectPath)));
        System.out.println("Successfully placed " + objectKey + " into bucket
" + bucketName);

    } catch (S3Exception e) {
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
}

```

Chargez un objet dans un compartiment et définissez une valeur de conservation de l'objet à l'aide d'un [S3Client](#).

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.S3Exception;

```

```
import java.time.Instant;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.ZoneOffset;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class PutObjectRetention {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <key> <bucketName>\s

            Where:
                key - The name of the object (for example, book.pdf).\s
                bucketName - The Amazon S3 bucket name that contains the object
(for example, bucket1).\s
            """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String key = args[0];
        String bucketName = args[1];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        setRentionPeriod(s3, key, bucketName);
        s3.close();
    }
}
```

```

/**
 * Sets the retention period for an object in an Amazon S3 bucket.
 *
 * @param s3      the S3Client object used to interact with the Amazon S3
service
 * @param key     the key (name) of the object in the S3 bucket
 * @param bucket  the name of the S3 bucket where the object is stored
 *
 * @throws S3Exception if an error occurs while setting the object retention
period
 */
public static void setRetentionPeriod(S3Client s3, String key, String bucket) {
    try {
        LocalDate localDate = LocalDate.parse("2020-07-17");
        LocalDateTime localDateTime = localDate.atStartOfDay();
        Instant instant = localDateTime.toInstant(ZoneOffset.UTC);

        ObjectLockRetention lockRetention = ObjectLockRetention.builder()
            .mode("COMPLIANCE")
            .retainUntilDate(instant)
            .build();

        PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
            .bucket(bucket)
            .key(key)
            .bypassGovernanceRetention(true)
            .retention(lockRetention)
            .build();

        // To set Retention on an object, the Amazon S3 bucket must support
object
        // locking, otherwise an exception is thrown.
        s3.putObjectRetention(retentionRequest);
        System.out.print("An object retention configuration was successfully
placed on the object");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Chargez l'objet.

```
import { readFile } from "node:fs/promises";

import {
  PutObjectCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Upload a file to an S3 bucket.
 * @param {{ bucketName: string, key: string, filePath: string }}
 */
export const main = async ({ bucketName, key, filePath }) => {
  const client = new S3Client({});
  const command = new PutObjectCommand({
    Bucket: bucketName,
    Key: key,
    Body: await readFile(filePath),
  });

  try {
    const response = await client.send(command);
    console.log(response);
  } catch (caught) {
    if (
```

```

        caught instanceof S3ServiceException &&
        caught.name === "EntityTooLarge"
    ) {
        console.error(
            `Error from S3 while uploading object to ${bucketName}. \
The object was too large. To upload objects larger than 5GB, use the S3 console \
(160GB max) \
or the multipart upload API (5TB max).`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while uploading object to ${bucketName}. ${caught.name}: \
${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

Chargez l'objet à condition que son ETag corresponde à celle fournie.

```

import {
    GetObjectCommand,
    NoSuchKey,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Get a single object from a specified S3 bucket.
 * @param {{ bucketName: string, key: string, eTag: string }}
 */
export const main = async ({ bucketName, key, eTag }) => {
    const client = new S3Client({});

    try {
        const response = await client.send(
            new GetObjectCommand({
                Bucket: bucketName,
                Key: key,

```

```
        IfMatch: eTag,
      )),
    );
    // The Body object also has 'transformToByteArray' and 'transformToWebStream'
    methods.
    const str = await response.Body.transformToString();
    console.log("Success. Here is text of the file:", str);
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(
        `Error from S3 while getting object "${key}" from "${bucketName}". No
such key exists.`
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while getting object from ${bucketName}. ${caught.name}:
${caught.message}`
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    key: {
      type: "string",
      required: true,
    },
    eTag: {
      type: "string",
      required: true,
    },
  };
};
```

```
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).
- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS SDK pour JavaScript API.

Kotlin

SDK pour Kotlin

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
suspend fun putS3Object(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
```



```

        bucket = bucketName
        key = objectKey
        metadata = metadataVal
        body = File(objectPath).asByteStream()
    }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

```

- Pour plus de détails sur l'API, consultez [PutObject](#) la section AWS SDK pour la référence de l'API Kotlin.

PHP

Kit SDK pour PHP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Chargez un objet dans un compartiment.

```

$s3client = new Aws\S3\S3Client(['region' => 'us-west-2']);

$fileName = __DIR__ . "/local-file-" . uniqid();
try {
    $this->s3client->putObject([
        'Bucket' => $this->bucketName,
        'Key' => $fileName,
        'SourceFile' => __DIR__ . '/testfile.txt'
    ]);
    echo "Uploaded $fileName to $this->bucketName.\n";
} catch (Exception $exception) {
    echo "Failed to upload $fileName with error: " . $exception-
    >getMessage();
    exit("Please fix error with file upload before continuing.");
}

```

```
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS SDK pour PHP API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : cette commande charge le fichier unique « local-sample.txt » sur Amazon S3, créant un objet avec la clé « sample.txt » dans le compartiment « test-files ».

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-sample.txt
```

Exemple 2 : cette commande charge le fichier unique « sample.txt » sur Amazon S3, créant un objet avec la clé « sample.txt » dans le compartiment « test-files ». Si le paramètre -Key n'est pas fourni, le nom du fichier est utilisé comme clé d'objet S3.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Exemple 3 : cette commande télécharge le fichier unique « local-sample.txt » sur Amazon S3, créant un objet avec la clé « prefix/to/sample.txt » dans le compartiment « test-files ».

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -File .\local-sample.txt
```

Exemple 4 : Cette commande télécharge tous les fichiers du sous-répertoire « Scripts » vers le bucket « test-files » et applique le préfixe de clé commun « » à chaque objet. SampleScripts Chaque fichier téléchargé aura une clé « » où le SampleScripts/filename « nom de fichier » varie.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix SampleScripts\
```

Exemple 5 : Cette commande télécharge tous les fichiers*.ps1 du répertoire local « Scripts » vers le bucket « test-files » et applique le préfixe de clé commun « » à chaque objet.

SampleScripts Chaque fichier téléchargé aura une clé « » où le SampleScripts/filename.ps1 « nom de fichier » varie.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
SampleScripts\ -SearchPattern *.ps1
```

Exemple 6 : cette commande crée un nouvel objet S3 contenant la chaîne de contenu spécifiée avec la clé « sample.txt ».

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object
contents"
```

Exemple 7 : cette commande charge le fichier spécifié (le nom du fichier est utilisé comme clé) et applique les balises spécifiées au nouvel objet.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Exemple 8 : cette commande charge de manière récursive le dossier spécifié et applique les balises spécifiées à tous les nouveaux objets.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : cette commande charge le fichier unique « local-sample.txt » sur Amazon S3, créant un objet avec la clé « sample.txt » dans le compartiment « test-files ».

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Exemple 2 : cette commande charge le fichier unique « sample.txt » sur Amazon S3, créant un objet avec la clé « sample.txt » dans le compartiment « test-files ». Si le paramètre -Key n'est pas fourni, le nom du fichier est utilisé comme clé d'objet S3.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Exemple 3 : cette commande télécharge le fichier unique « local-sample.txt » sur Amazon S3, créant un objet avec la clé « prefix/to /sample.txt » dans le compartiment « test-files ».

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -  
File .\local-sample.txt
```

Exemple 4 : Cette commande télécharge tous les fichiers du sous-répertoire « Scripts » vers le bucket « test-files » et applique le préfixe de clé commun « » à chaque objet. SampleScripts Chaque fichier téléchargé aura une clé « » où le SampleScripts/filename « nom de fichier » varie.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\
```

Exemple 5 : Cette commande télécharge tous les fichiers*.ps1 du répertoire local « Scripts » vers le bucket « test-files » et applique le préfixe de clé commun « » à chaque objet. SampleScripts Chaque fichier téléchargé aura une clé « » où le SampleScripts/filename.ps1 « nom de fichier » varie.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Exemple 6 : cette commande crée un nouvel objet S3 contenant la chaîne de contenu spécifiée avec la clé « sample.txt ».

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Exemple 7 : cette commande charge le fichier spécifié (le nom du fichier est utilisé comme clé) et applique les balises spécifiées au nouvel objet.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Exemple 8 : cette commande charge de manière récursive le dossier spécifié et applique les balises spécifiées à tous les nouveaux objets.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"  
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                           that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key

    def put(self, data):
        """
        Upload data to the object.

        :param data: The data to upload. This can either be bytes or a string.
        When this
                           argument is a string, it is interpreted as a file name,
        which is
                           opened in read bytes mode.
        """
        put_data = data
        if isinstance(data, str):
            try:
                put_data = open(data, "rb")
            except IOError:
```

```

        logger.exception("Expected file name or binary data, got '%s'.",
data)
        raise

    try:
        self.object.put(Body=put_data)
        self.object.wait_until_exists()
        logger.info(
            "Put object '%s' to bucket '%s'.",
            self.object.key,
            self.object.bucket_name,
        )
    except ClientError:
        logger.exception(
            "Couldn't put object '%s' to bucket '%s'.",
            self.object.key,
            self.object.bucket_name,
        )
        raise
    finally:
        if getattr(put_data, "close", None):
            put_data.close()

```

Chargez un objet à l'aide d'une demande conditionnelle.

```

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

```

```
def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
    """
    Uploads an object to Amazon S3 with a conditional request. Prevents
    overwrite
    using an IfNoneMatch condition for the object key.

    :param object_key: The key of the object to upload.
    :param source_bucket: The source bucket of the object.
    :param data: The data to upload.
    """
    try:
        self.s3.put_object(
            Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
        )
        print(
            f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
        )
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional write failed: Precondition failed")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise
```

- Pour plus de détails sur l'API, consultez [PutObject](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Ruby

Kit SDK pour Ruby

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Chargez un fichier à l'aide d'un téléchargeur géré (Object.upload_file).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
#{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

  wrapper = ObjectUploadFileWrapper.new(Aws::S3::Object.new(bucket_name,
    object_key))
  return unless wrapper.upload_file(file_path)

  puts "File #{file_path} successfully uploaded to #{bucket_name}:#{object_key}."
end

run_demo if $PROGRAM_NAME == __FILE__
```


Téléchargez un fichier à l'aide de Object.put.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
#{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

Chargez un fichier à l'aide du chiffrement côté serveur Object.put et ajoutez-le.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{@object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

  wrapper = ObjectPutSseWrapper.new(Aws::S3::Object.new(bucket_name,
    object_content))
  return unless wrapper.put_object_encrypted(object_content, encryption)

  puts "Put your content into #{bucket_name}:#{object_key} and encrypted it with
    #{encryption}."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section Référence des AWS SDK pour Ruby API.

Rust

SDK pour Rust

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
pub async fn upload_object(
    client: &aws_sdk_s3::Client,
    bucket_name: &str,
    file_name: &str,
    key: &str,
) -> Result<aws_sdk_s3::operation::put_object::PutObjectOutput, S3ExampleError> {
    let body =
        aws_sdk_s3::primitives::ByteStream::from_path(std::path::Path::new(file_name)).await;
    client
        .put_object()
        .bucket(bucket_name)
        .key(key)
        .body(body.unwrap())
        .send()
        .await
        .map_err(S3ExampleError::from)
}
```

- Pour plus de détails sur l'API, voir [PutObject](#) la section de référence de l'API AWS SDK for Rust.

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

"Get contents of file from application server."
DATA lv_body TYPE xstring.
OPEN DATASET iv_file_name FOR INPUT IN BINARY MODE.
READ DATASET iv_file_name INTO lv_body.
CLOSE DATASET iv_file_name.

"Upload/put an object to an S3 bucket."
TRY.
    lo_s3->putobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_file_name
        iv_body = lv_body ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Swift

Kit SDK pour Swift

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

import AWSS3
import Smithy

public func uploadFile(bucket: String, key: String, file: String) async
throws {
    let fileUrl = URL(fileURLWithPath: file)
    do {
        let fileData = try Data(contentsOf: fileUrl)

```

```
        let dataStream = ByteStream.data(fileData)

        let input = PutObjectInput(
            body: dataStream,
            bucket: bucket,
            key: key
        )

        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object."))
        throw error
    }
}
```

```
import AWSS3
import Smithy

public func createFile(bucket: String, key: String, withData data: Data)
async throws {
    let dataStream = ByteStream.data(data)

    let input = PutObjectInput(
        body: dataStream,
        bucket: bucket,
        key: key
    )

    do {
        _ = try await client.putObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Putting an object."))
        throw error
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObject](#) à la section AWS SDK pour la référence de l'API Swift.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutObjectAcl** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutObjectAcl.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Gérer les listes de contrôle d'accès \(ACL\)](#)

C++

SDK pour C++

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
    &objectKey, const Aws::String &ownerID,
                                const Aws::String &granteePermission, const
    Aws::String &granteeType,
                                const Aws::String &granteeID, const Aws::String
    &granteeEmailAddress,
                                const Aws::String &granteeURI, const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
```

```

        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutObjectAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::PutObjectAclOutcome outcome =
        s3Client.PutObjectAcl(request);

    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
            << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type
enumeration.

```

```

    /*!
    \param access: Human readable string.
    \return Permission: Permission enumeration.
    */
    Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
        if (access == "FULL_CONTROL")
            return Aws::S3::Model::Permission::FULL_CONTROL;
        if (access == "WRITE")
            return Aws::S3::Model::Permission::WRITE;
        if (access == "READ")
            return Aws::S3::Model::Permission::READ;
        if (access == "WRITE_ACP")
            return Aws::S3::Model::Permission::WRITE_ACP;
        if (access == "READ_ACP")
            return Aws::S3::Model::Permission::READ_ACP;
        return Aws::S3::Model::Permission::NOT_SET;
    }

    //! Routine which converts a human-readable string to a built-in type
    enumeration.
    /*!
    \param type: Human readable string.
    \return Type: Type enumeration.
    */
    Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
        if (type == "Amazon customer by email")
            return Aws::S3::Model::Type::AmazonCustomerByEmail;
        if (type == "Canonical user")
            return Aws::S3::Model::Type::CanonicalUser;
        if (type == "Group")
            return Aws::S3::Model::Type::Group;
        return Aws::S3::Model::Type::NOT_SET;
    }

```

- Pour plus de détails sur l'API, reportez-vous [PutObjectAcl](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante accorde l'`read` autorisation `full control` à deux AWS utilisateurs (`user1@example.com` et `user2@example.com`) et à tout le monde :

```
aws s3api put-object-acl --bucket amzn-s3-demo-bucket --key file.txt --grant-  
full-control emailaddress=user1@example.com,emailaddress=user2@example.com --  
grant-read uri=http://acs.amazonaws.com/groups/global/AllUsers
```

Consultez <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> pour plus de détails sur les ACL personnalisées (les commandes ACL `s3api`, par exemple `put-object-acl`, utilisent la même notation abrégée en arguments).

- Pour plus de détails sur l'API, reportez-vous [PutObjectAcl](#) à la section Référence des AWS CLI commandes.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class ObjectWrapper:
    """Encapsulates S3 object actions."""

    def __init__(self, s3_object):
        """
        :param s3_object: A Boto3 Object resource. This is a high-level resource
        in Boto3
                                that wraps object actions in a class-like structure.
        """
        self.object = s3_object
        self.key = self.object.key
```

```
def put_acl(self, email):
    """
    Applies an ACL to the object that grants read access to an AWS user
    identified
    by email address.

    :param email: The email address of the user to grant access.
    """
    try:
        acl = self.object.Acl()
        # Putting an ACL overwrites the existing ACL, so append new grants
        # if you want to preserve existing grants.
        grants = acl.grants if acl.grants else []
        grants.append(
            {
                "Grantee": {"Type": "AmazonCustomerByEmail", "EmailAddress":
email},
                "Permission": "READ",
            }
        )
        acl.put(AccessControlPolicy={"Grants": grants, "Owner": acl.owner})
        logger.info("Granted read access to %s.", email)
    except ClientError:
        logger.exception("Couldn't add ACL to object '%s'.", self.object.key)
        raise
```

- Pour plus de détails sur l'API, consultez [PutObjectAcl](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
  " Example: Grant read access to an AWS user  
  " iv_grantread = 'emailAddress=user@example.com'  
  lo_s3->putobjectacl(  
    iv_bucket = iv_bucket_name  
    iv_key = iv_object_key  
    iv_grantread = iv_grantread ).  
  MESSAGE 'Object ACL updated.' TYPE 'I'.  
CATCH /aws1/cx_s3_nosuchbucket.  
  MESSAGE 'Bucket does not exist.' TYPE 'E'.  
CATCH /aws1/cx_s3_nosuchkey.  
  MESSAGE 'Object key does not exist.' TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectAcl](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutObjectLegalHold** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutObjectLegalHold.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Verrouillage d'objets Amazon S3](#)

.NET

SDK pour .NET

 Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>
/// Set or modify a legal hold on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="holdStatus">The On or Off status for the legal hold.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectLegalHold(string bucketName,
    string objectKey, ObjectLockLegalHoldStatus holdStatus)
{
    try
    {
        var request = new PutObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            LegalHold = new ObjectLockLegalHold()
            {
                Status = holdStatus
            }
        };

        var response = await _amazonS3.PutObjectLegalHoldAsync(request);
        Console.WriteLine($"\\tModified legal hold for {objectKey} in
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tError modifying legal hold: '{ex.Message}'");
        return false;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLegalHold](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour appliquer une conservation légale à un objet

L'exemple `put-object-legal-hold` suivant définit une conservation légale sur l'objet `doc1.rtf`.

```
aws s3api put-object-legal-hold \  
  --bucket amzn-s3-demo-bucket-with-object-lock \  
  --key doc1.rtf \  
  --legal-hold Status=ON
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [PutObjectLegalHold](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (  
  "bytes"  
  "context"  
  "errors"  
  "fmt"  
  "log"  
  "time"  
  
  "github.com/aws/aws-sdk-go-v2/aws"  
  "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
  "github.com/aws/aws-sdk-go-v2/service/s3"
```

```
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }

    _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLegalHold](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus sur GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLegalHold](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import {
  PutObjectLegalHoldCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Apply a legal hold configuration to the specified object.
 * @param {{ bucketName: string, objectKey: string, legalHoldStatus: "ON" |
 * "OFF" }}
 */
export const main = async ({ bucketName, objectKey, legalHoldStatus }) => {
  if (!["OFF", "ON"].includes(legalHoldStatus.toUpperCase())) {
    throw new Error(
      "Invalid parameter. legalHoldStatus must be 'ON' or 'OFF'."
    );
  }

  const client = new S3Client({});
  const command = new PutObjectLegalHoldCommand({
    Bucket: bucketName,
    Key: objectKey,
    LegalHold: {
      // Set the status to 'ON' to place a legal hold on the object.
      // Set the status to 'OFF' to remove the legal hold.
      Status: legalHoldStatus,
    },
  });

  try {
    await client.send(command);
    console.log(
```



```

    `Legal hold status set to "${legalHoldStatus}" for "${objectKey}" in
    "${bucketName}"`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
        "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying legal hold status for "${objectKey}" in
        "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
    objectKey: {
      type: "string",
      required: true,
    },
    legalHoldStatus: {
      type: "string",
      default: "ON",
    },
  },

```

```

};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
}

```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLegalHold](#) à la section Référence des AWS SDK pour JavaScript API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Placez un objet en détention légale.

```

def set_legal_hold(s3_client, bucket: str, key: str) -> None:
    """
    Set a legal hold on a specific file in a bucket.

    Args:
        s3_client: Boto3 S3 client.
        bucket: The name of the bucket containing the file.
        key: The key of the file to set the legal hold on.
    """
    print()

```

```

logger.info("Setting legal hold on file [%s] in bucket [%s]", key, bucket)
try:
    before_status = "OFF"
    after_status = "ON"
    s3_client.put_object_legal_hold(
        Bucket=bucket, Key=key, LegalHold={"Status": after_status}
    )
    logger.debug(
        "Legal hold set successfully on file [%s] in bucket [%s]", key,
bucket
    )
    _print_legal_hold_update(bucket, key, before_status, after_status)
except Exception as e:
    logger.error(
        "Failed to set legal hold on file [%s] in bucket [%s]: %s", key,
bucket, e
    )

```

- Pour plus de détails sur l'API, consultez [PutObjectLegalHold](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

TRY.

```

" Example: Set legal hold status to ON
" iv_status = 'ON'
lo_s3->putobjectlegalhold(
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key
    io_legalhold = NEW /aws1/cl_s3_objlocklegalhold(

```

```

        iv_status = iv_status ) ).
    MESSAGE 'Object legal hold status set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
    MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLegalHold](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutObjectLockConfiguration** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutObjectLockConfiguration.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Verrouillage d'objets Amazon S3](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Définissez la configuration du verrouillage d'objet d'un compartiment.

```

/// <summary>
/// Enable object lock on an existing bucket.

```

```
/// </summary>
/// <param name="bucketName">The name of the bucket to modify.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableObjectLockOnBucket(string bucketName)
{
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
            },
        };

        var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"{bucketName}\tAdded an object lock policy to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
        return false;
    }
}
```

Définissez la période de rétention par défaut d'un compartiment.

```

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()
            {
                BucketName = bucketName,
                ObjectLockConfiguration = new ObjectLockConfiguration()
                {
                    ObjectLockEnabled = new ObjectLockEnabled(enabledString),
                    Rule = new ObjectLockRule()
                    {
                        DefaultRetention = new DefaultRetention()
                        {
                            Mode = retention,
                            Days = timeDifference.Days // Can be specified in
days or years but not both.
                        }
                    }
                }
            };

```

```
        var response = await
        _amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"\\tAdded a default retention to bucket
        {bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tError modifying object lock: '{ex.Message}'");
        return false;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLockConfiguration](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour définir la configuration de verrouillage d'objet d'un compartiment

L'exemple `put-object-lock-configuration` suivant définit un verrouillage d'objet de 50 jours sur le compartiment spécifié.

```
aws s3api put-object-lock-configuration \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --object-lock-configuration '{ "ObjectLockEnabled": "Enabled", "Rule":
  { "DefaultRetention": { "Mode": "COMPLIANCE", "Days": 50 }}}'
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [PutObjectLockConfiguration](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Définissez la configuration du verrouillage d'objet d'un compartiment.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
    string) error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
```



```

    MFADelete: types.MFADeleteDisabled,
    Status:    types.BucketVersioningStatusEnabled,
  },
}
_, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
    return err
}

input := &s3.PutObjectLockConfigurationInput{
    Bucket: aws.String(bucket),
    ObjectLockConfiguration: &types.ObjectLockConfiguration{
        ObjectLockEnabled: types.ObjectLockEnabledEnabled,
    },
}
_, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
}

return err
}

```

Définissez la période de rétention par défaut d'un compartiment.

```

import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"

```

```

"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// ModifyDefaultBucketRetention modifies the default retention period of an
// existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }

    _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }
}

```

```
    return err
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLockConfiguration](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Définissez la configuration du verrouillage d'objet d'un compartiment.

```
// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();

        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        // Enable versioning on the bucket.
        getClient().putBucketVersioning(putBucketVersioningRequest);
        PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
                .build())
    }
```

```
        .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on
"+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
    }
}
```

Définissez la période de rétention par défaut d'un compartiment.

```
// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .mfaDelete(MFADelete.DISABLED)
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention rention = DefaultRetention.builder()
        .days(1)
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(rention)
        .build();

    ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .rule(lockRule)
```

```
        .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .objectLockConfiguration(objectLockConfiguration)
        .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName
+ ".");
    }
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLockConfiguration](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Définissez la configuration du verrouillage d'objet d'un compartiment.

```
import {
    PutObjectLockConfigurationCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Enable S3 Object Lock for an Amazon S3 bucket.
 * After you enable Object Lock on a bucket, you can't
 * disable Object Lock or suspend versioning for that bucket.
 * @param {{ bucketName: string, enabled: boolean }}
 */
```

```
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  const command = new PutObjectLockConfigurationCommand({
    Bucket: bucketName,
    // The Object Lock configuration that you want to apply to the specified
    bucket.
    ObjectLockConfiguration: {
      ObjectLockEnabled: "Enabled",
    },
  });

  try {
    await client.send(command);
    console.log(`Object Lock for "${bucketName}" enabled.`);
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchBucket"
    ) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
        bucket "${bucketName}". The bucket doesn't exist.`,
      );
    } else if (caught instanceof S3ServiceException) {
      console.error(
        `Error from S3 while modifying the object lock configuration for the
        bucket "${bucketName}". ${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
```

```

        type: "string",
        required: true,
    },
};
const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}

```

Définissez la période de rétention par défaut d'un compartiment.

```

import {
    PutObjectLockConfigurationCommand,
    S3Client,
    S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Change the default retention settings for an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, retentionDays: string }}
 */
export const main = async ({ bucketName, retentionDays }) => {
    const client = new S3Client({});

    try {
        await client.send(
            new PutObjectLockConfigurationCommand({
                Bucket: bucketName,
                // The Object Lock configuration that you want to apply to the specified
                bucket.
                ObjectLockConfiguration: {
                    ObjectLockEnabled: "Enabled",

```

```

        Rule: {
            // The default Object Lock retention mode and period that you want to
            apply
            // to new objects placed in the specified bucket. Bucket settings
            require
            // both a mode and a period. The period can be either Days or Years
            but
            // you must select one.
            DefaultRetention: {
                // In governance mode, users can't overwrite or delete an object
                version
                // or alter its lock settings unless they have special permissions.
                With
                // governance mode, you protect objects against being deleted by
                most users,
                // but you can still grant some users permission to alter the
                retention settings
                // or delete the objects if necessary.
                Mode: "GOVERNANCE",
                Days: Number.parseInt(retentionDays),
            },
        },
    },
    }),
);
console.log(
    `Set default retention mode to "GOVERNANCE" with a retention period of
    ${retentionDays} day(s).`,
);
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while setting the default object retention for a bucket.
            The bucket doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while setting the default object retention for a bucket.
            ${caught.name}: ${caught.message}`,
        );
    } else {

```



```
        throw caught;
    }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
    isMain,
    validateArgs,
} from "@aws-doc-sdk-examples/lib/utils/util-node.js";

const loadArgs = () => {
    const options = {
        bucketName: {
            type: "string",
            required: true,
        },
        retentionDays: {
            type: "string",
            required: true,
        },
    };
};

const results = parseArgs({ options });
const { errors } = validateArgs({ options }, results);
return { errors, results };
};

if (isMain(import.meta.url)) {
    const { errors, results } = loadArgs();
    if (!errors) {
        main(results.values);
    } else {
        console.error(errors.join("\n"));
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLockConfiguration](#) à la section Référence des AWS SDK pour JavaScript API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Placez la configuration du verrouillage d'objet.

```
s3_client.put_object_lock_configuration(  
    Bucket=bucket,  
    ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":  
{}},  
)
```

- Pour plus de détails sur l'API, consultez [PutObjectLockConfiguration](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
    " Example: Enable object lock with default retention  
    " iv_enabled = 'Enabled'  
    lo_s3->putobjectlockconfiguration(  
        iv_bucket = iv_bucket_name  
        io_objectlockconfiguration = NEW /aws1/cl_s3_objectlockconf(  
            iv_objectlockenabled = iv_enabled ) ).  
    MESSAGE 'Object lock configuration set.' TYPE 'I'.
```

```
CATCH /aws1/cx_s3_nosuchbucket.  
    MESSAGE 'Bucket does not exist.' TYPE 'E'.  
ENDTRY.
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectLockConfiguration](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutObjectRetention** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser PutObjectRetention.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Verrouillage d'objets Amazon S3](#)

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/// <summary>  
/// Set or modify a retention period on an object in an S3 bucket.  
/// </summary>  
/// <param name="bucketName">The bucket of the object.</param>  
/// <param name="objectKey">The key of the object.</param>  
/// <param name="retention">The retention mode.</param>  
/// <param name="retainUntilDate">The date retention expires.</param>
```

```

    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
        string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [PutObjectRetention](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour définir la configuration de rétention d'un objet pour un objet

L'exemple `put-object-retention` suivant définit la configuration de rétention de l'objet spécifié jusqu'au 01/01/2025.

```
aws s3api put-object-retention \  
  --bucket amzn-s3-demo-bucket-with-object-lock \  
  --key doc1.rtf \  
  --retention '{ "Mode": "GOVERNANCE", "RetainUntilDate":  
  "2025-01-01T00:00:00" }'
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [PutObjectRetention](#) à la section Référence des AWS CLI commandes.

Go

Kit SDK pour Go V2

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import (  
  "bytes"  
  "context"  
  "errors"  
  "fmt"  
  "log"  
  "time"  
  
  "github.com/aws/aws-sdk-go-v2/aws"  
  "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
  "github.com/aws/aws-sdk-go-v2/service/s3"  
  "github.com/aws/aws-sdk-go-v2/service/s3/types"  
  "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
  S3Client *s3.Client  
  S3Manager *manager.Uploader
```

```
}

// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
    string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
    error {
    input := &s3.PutObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        Retention: &types.ObjectLockRetention{
            Mode:           retentionMode,
            RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
        },
        BypassGovernanceRetention: aws.Bool(true),
    }

    _, err := actor.S3Client.PutObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectRetention](#) à la section Référence des AWS SDK pour Go API.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time
    zone.
    ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);

    // Print the formatted date string.
    System.out.println("Formatted Date: " + humanReadableDate);
    ObjectLockRetention retention = ObjectLockRetention.builder()
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .retainUntilDate(futureInstant)
        .build();

    PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .retention(retention)
        .build();
}
```

```
getClient().putObjectRetention(retentionRequest);
System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectRetention](#) à la section Référence des AWS SDK for Java 2.x API.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import {
  PutObjectRetentionCommand,
  S3Client,
  S3ServiceException,
} from "@aws-sdk/client-s3";

/**
 * Place a 24-hour retention period on an object in an Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const client = new S3Client({});
  const command = new PutObjectRetentionCommand({
    Bucket: bucketName,
    Key: key,
    BypassGovernanceRetention: false,
    Retention: {
      // In governance mode, users can't overwrite or delete an object version
      // or alter its lock settings unless they have special permissions. With
      // governance mode, you protect objects against being deleted by most
      users,
```



```
// but you can still grant some users permission to alter the retention
settings
// or delete the objects if necessary.
Mode: "GOVERNANCE",
RetainUntilDate: new Date(new Date().getTime() + 24 * 60 * 60 * 1000),
},
});

try {
  await client.send(command);
  console.log("Object Retention settings updated.");
} catch (caught) {
  if (
    caught instanceof S3ServiceException &&
    caught.name === "NoSuchBucket"
  ) {
    console.error(
      `Error from S3 while modifying the governance mode and retention period
on an object. The bucket doesn't exist.`
    );
  } else if (caught instanceof S3ServiceException) {
    console.error(
      `Error from S3 while modifying the governance mode and retention period
on an object. ${caught.name}: ${caught.message}`
    );
  } else {
    throw caught;
  }
}
};

// Call function if run directly
import { parseArgs } from "node:util";
import {
  isMain,
  validateArgs,
} from "@aws-doc-sdk-examples/lib/utills/util-node.js";

const loadArgs = () => {
  const options = {
    bucketName: {
      type: "string",
      required: true,
    },
  },
```

```
    key: {
      type: "string",
      required: true,
    },
  };
  const results = parseArgs({ options });
  const { errors } = validateArgs({ options }, results);
  return { errors, results };
};

if (isMain(import.meta.url)) {
  const { errors, results } = loadArgs();
  if (!errors) {
    main(results.values);
  } else {
    console.error(errors.join("\n"));
  }
}
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectRetention](#) à la section Référence des AWS SDK pour JavaScript API.

PowerShell

Outils pour PowerShell V4

Exemple 1 : la commande active le mode de rétention de la gouvernance jusqu'à la date du 31 décembre 2019 à 00 h 00 m 00 s pour l'objet « testfile.txt » dans le compartiment S3 donné.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectRetention](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V4).

Outils pour PowerShell V5

Exemple 1 : la commande active le mode de rétention de la gouvernance jusqu'à la date du 31 décembre 2019 à 00 h 00 m 00 s pour l'objet « testfile.txt » dans le compartiment S3 donné.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -  
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Pour plus de détails sur l'API, reportez-vous [PutObjectRetention](#) à la section Référence des Outils AWS pour PowerShell applets de commande (V5).

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Placez un objet en rétention.

```
s3_client.put_object_retention(  
    Bucket=bucket,  
    Key=key,  
    VersionId=version_id,  
    Retention={"Mode": "GOVERNANCE", "RetainUntilDate":  
far_future_date},  
    BypassGovernanceRetention=True,  
)
```

- Pour plus de détails sur l'API, consultez [PutObjectRetention](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
  " Example: Set retention mode to GOVERNANCE for 30 days
  " iv_mode = 'GOVERNANCE'
  " iv_retain_date should be a timestamp in the future
  lo_s3->putobjectretention(
    iv_bucket = iv_bucket_name
    iv_key = iv_object_key
    io_retention = NEW /aws1/cl_s3_objectlockret(
      iv_mode = iv_mode
      iv_retainuntildate = iv_retain_date )
    iv_bypassgovernanceretention = abap_true ).
  MESSAGE 'Object retention set.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
  MESSAGE 'Bucket does not exist.' TYPE 'E'.
CATCH /aws1/cx_s3_nosuchkey.
  MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

```

- Pour plus de détails sur l'API, reportez-vous [PutObjectRetention](#) à la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **RestoreObject** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser `RestoreObject`.

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
using System;
```

```

using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to restore an archived object in an Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class RestoreArchivedObject
{
    public static void Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string objectKey = "archived-object.txt";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        IAmazonS3 client = new AmazonS3Client(bucketRegion);
        RestoreObjectAsync(client, bucketName, objectKey).Wait();
    }

    /// <summary>
    /// This method restores an archived object from an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// RestoreObjectAsync.</param>
    /// <param name="bucketName">A string representing the name of the
    /// bucket where the object was located before it was archived.</param>
    /// <param name="objectKey">A string representing the name of the
    /// archived object to restore.</param>
    public static async Task RestoreObjectAsync(IAmazonS3 client, string
bucketName, string objectKey)
    {
        try
        {
            var restoreRequest = new RestoreObjectRequest
            {
                BucketName = bucketName,
                Key = objectKey,
                Days = 2,
            };

```

```

        RestoreObjectResponse response = await
client.RestoreObjectAsync(restoreRequest);

        // Check the status of the restoration.
        await CheckRestorationStatusAsync(client, bucketName, objectKey);
    }
    catch (AmazonS3Exception amazonS3Exception)
    {
        Console.WriteLine($"Error: {amazonS3Exception.Message}");
    }
}

/// <summary>
/// This method retrieves the status of the object's restoration.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// GetObjectMetadataAsync.</param>
/// <param name="bucketName">A string representing the name of the Amazon
/// S3 bucket which contains the archived object.</param>
/// <param name="objectKey">A string representing the name of the
/// archived object you want to restore.</param>
public static async Task CheckRestorationStatusAsync(IAmazonS3 client,
string bucketName, string objectKey)
{
    GetObjectMetadataRequest metadataRequest = new
GetObjectMetadataRequest()
    {
        BucketName = bucketName,
        Key = objectKey,
    };

    GetObjectMetadataResponse response = await
client.GetObjectMetadataAsync(metadataRequest);

    var restStatus = response.RestoreInProgress ? "in-progress" :
"finished or failed";
    Console.WriteLine($"Restoration status: {restStatus}");
}
}

```

- Pour plus de détails sur l'API, reportez-vous [RestoreObject](#) à la section Référence des AWS SDK pour .NET API.

CLI

AWS CLI

Pour créer une demande de restauration d'un objet

L'exemple `restore-object` suivant restaure l'objet Amazon S3 Glacier spécifié du compartiment `my-glacier-bucket` pendant 10 jours.

```
aws s3api restore-object \  
  --bucket my-glacier-bucket \  
  --key doc1.rtf \  
  --restore-request Days=10
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [RestoreObject](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import software.amazon.awssdk.regions.Region;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.RestoreRequest;  
import software.amazon.awssdk.services.s3.model.GlacierJobParameters;  
import software.amazon.awssdk.services.s3.model.RestoreObjectRequest;  
import software.amazon.awssdk.services.s3.model.S3Exception;  
import software.amazon.awssdk.services.s3.model.Tier;
```

```
/*
 * For more information about restoring an object, see "Restoring an archived
 * object" at
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/restoring-objects.html
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class RestoreObject {
    public static void main(String[] args) {
        final String usage = ""

            Usage:
                <bucketName> <keyName> <expectedBucketOwner>

            Where:
                bucketName - The Amazon S3 bucket name.\s
                keyName - The key name of an object with a Storage class value of
                Glacier.\s
                expectedBucketOwner - The account that owns the bucket (you can
                obtain this value from the AWS Management Console).\s
            "";

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String bucketName = args[0];
        String keyName = args[1];
        String expectedBucketOwner = args[2];
        Region region = Region.US_EAST_1;
        S3Client s3 = S3Client.builder()
            .region(region)
            .build();

        restoreS3Object(s3, bucketName, keyName, expectedBucketOwner);
        s3.close();
    }
}
```



```

/**
 * Restores an S3 object from the Glacier storage class.
 *
 * @param s3                an instance of the {@link S3Client} to be used
for interacting with Amazon S3
 * @param bucketName       the name of the S3 bucket where the object is
stored
 * @param keyName           the key (object name) of the S3 object to be
restored
 * @param expectedBucketOwner the AWS account ID of the expected bucket
owner
 */
public static void restoreS3Object(S3Client s3, String bucketName, String
keyName, String expectedBucketOwner) {
    try {
        RestoreRequest restoreRequest = RestoreRequest.builder()
            .days(10)

.glacierJobParameters(GlacierJobParameters.builder().tier(Tier.STANDARD).build())
            .build();

        RestoreObjectRequest objectRequest = RestoreObjectRequest.builder()
            .expectedBucketOwner(expectedBucketOwner)
            .bucket(bucketName)
            .key(keyName)
            .restoreRequest(restoreRequest)
            .build();

        s3.restoreObject(objectRequest);

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
}

```

- Pour plus de détails sur l'API, reportez-vous [RestoreObject](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **SelectObjectContent** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser SelectObjectContent.

CLI

AWS CLI

Pour filtrer le contenu d'un objet Amazon S3 sur la base d'une instruction SQL

L'exemple select-object-content suivant filtre l'objet my-data-file.csv sur la base de l'instruction SQL spécifiée et envoie la sortie dans un fichier.

```
aws s3api select-object-content \
  --bucket amzn-s3-demo-bucket \
  --key my-data-file.csv \
  --expression "select * from s3object limit 100" \
  --expression-type 'SQL' \
  --input-serialization '{"CSV": {}, "CompressionType": "NONE"}' \
  --output-serialization '{"CSV": {}}' "output.csv"
```

Cette commande ne produit aucune sortie.

- Pour plus de détails sur l'API, reportez-vous [SelectObjectContent](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

L'exemple suivant montre une requête utilisant un objet JSON. L'[exemple complet](#) montre également l'utilisation d'un objet CSV.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.async.BlockingInputStreamAsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.CSVInput;
import software.amazon.awssdk.services.s3.model.CSVOutput;
import software.amazon.awssdk.services.s3.model.CompressionType;
import software.amazon.awssdk.services.s3.model.ExpressionType;
import software.amazon.awssdk.services.s3.model.FileHeaderInfo;
import software.amazon.awssdk.services.s3.model.InputSerialization;
import software.amazon.awssdk.services.s3.model.JSONInput;
import software.amazon.awssdk.services.s3.model.JSONOutput;
import software.amazon.awssdk.services.s3.model.JSONType;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.OutputSerialization;
import software.amazon.awssdk.services.s3.model.Progress;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.SelectObjectContentRequest;
import
    software.amazon.awssdk.services.s3.model.SelectObjectContentResponseHandler;
import software.amazon.awssdk.services.s3.model.Stats;

import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;

public class SelectObjectContentExample {
    static final Logger logger =
        LoggerFactory.getLogger(SelectObjectContentExample.class);
    static String BUCKET_NAME = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
    static final S3AsyncClient s3AsyncClient = S3AsyncClient.create();
    static String FILE_CSV = "csv";
    static String FILE_JSON = "json";
    static String URL_CSV = "https://raw.githubusercontent.com/mledoze/countries/
        master/dist/countries.csv";
```

```
static String URL_JSON = "https://raw.githubusercontent.com/mledoze/
countries/master/dist/countries.json";

public static void main(String[] args) {
    SelectObjectContentExample selectObjectContentExample = new
SelectObjectContentExample();
    try {
        SelectObjectContentExample.setUp();
        selectObjectContentExample.runSelectObjectContentMethodForJSON();
        selectObjectContentExample.runSelectObjectContentMethodForCSV();
    } catch (SdkException e) {
        logger.error(e.getMessage(), e);
        System.exit(1);
    } finally {
        SelectObjectContentExample.tearDown();
    }
}

EventStreamInfo runSelectObjectContentMethodForJSON() {
    // Set up request parameters.
    final String queryExpression = "select * from s3object[*][*] c where
c.area < 350000";
    final String fileType = FILE_JSON;

    InputSerialization inputSerialization = InputSerialization.builder()
        .json(JSONInput.builder().type(JSONType.DOCUMENT).build())
        .compressionType(CompressionType.NONE)
        .build();

    OutputSerialization outputSerialization = OutputSerialization.builder()
        .json(JSONOutput.builder().recordDelimiter(null).build())
        .build();

    // Build the SelectObjectContentRequest.
    SelectObjectContentRequest select = SelectObjectContentRequest.builder()
        .bucket(BUCKET_NAME)
        .key(FILE_JSON)
        .expression(queryExpression)
        .expressionType(ExpressionType.SQL)
        .inputSerialization(inputSerialization)
        .outputSerialization(outputSerialization)
        .build();

    EventStreamInfo eventStreamInfo = new EventStreamInfo();
```

```

        // Call the selectObjectContent method with the request and a response
        handler.
        // Supply an EventStreamInfo object to the response handler to gather
        records and information from the response.
        s3AsyncClient.selectObjectContent(select,
        buildResponseHandler(eventStreamInfo)).join();

        // Log out information gathered while processing the response stream.
        long recordCount = eventStreamInfo.getRecords().stream().mapToInt(record
        ->
            record.split("\n").length
        ).sum();
        logger.info("Total records {}: {}", fileType, recordCount);
        logger.info("Visitor onRecords for fileType {} called {} times",
        fileType, eventStreamInfo.getCountOnRecordsCalled());
        logger.info("Visitor onStats for fileType {}, {}", fileType,
        eventStreamInfo.getStats());
        logger.info("Visitor onContinuations for fileType {}, {}", fileType,
        eventStreamInfo.getCountContinuationEvents());
        return eventStreamInfo;
    }

    static SelectObjectContentResponseHandler
    buildResponseHandler(EventStreamInfo eventStreamInfo) {
        // Use a Visitor to process the response stream. This visitor logs
        information and gathers details while processing.
        final SelectObjectContentResponseHandler.Visitor visitor =
        SelectObjectContentResponseHandler.Visitor.builder()
            .onRecords(r -> {
                logger.info("Record event received.");
                eventStreamInfo.addRecord(r.payload().asUtf8String());
                eventStreamInfo.incrementOnRecordsCalled();
            })
            .onCont(ce -> {
                logger.info("Continuation event received.");
                eventStreamInfo.incrementContinuationEvents();
            })
            .onProgress(pe -> {
                Progress progress = pe.details();
                logger.info("Progress event received:\n bytesScanned:
        {} \n bytesProcessed: {} \n bytesReturned: {}",
                    progress.bytesScanned(),
                    progress.bytesProcessed(),
                    progress.bytesReturned());
            })
    }

```

```
        })
        .onEnd(ee -> logger.info("End event received. "))
        .onStats(se -> {
            logger.info("Stats event received.");
            eventStreamInfo.addStats(se.details());
        })
        .build();

    // Build the SelectObjectContentResponseHandler with the visitor that
    // processes the stream.
    return SelectObjectContentResponseHandler.builder()
        .subscriber(visitor).build();
}

// The EventStreamInfo class is used to store information gathered while
// processing the response stream.
static class EventStreamInfo {
    private final List<String> records = new ArrayList<>();
    private Integer countOnRecordsCalled = 0;
    private Integer countContinuationEvents = 0;
    private Stats stats;

    void incrementOnRecordsCalled() {
        countOnRecordsCalled++;
    }

    void incrementContinuationEvents() {
        countContinuationEvents++;
    }

    void addRecord(String record) {
        records.add(record);
    }

    void addStats(Stats stats) {
        this.stats = stats;
    }

    public List<String> getRecords() {
        return records;
    }

    public Integer getCountOnRecordsCalled() {
        return countOnRecordsCalled;
    }
}
```

```
    }

    public Integer getCountContinuationEvents() {
        return countContinuationEvents;
    }

    public Stats getStats() {
        return stats;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [SelectObjectContent](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **UploadPart** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser UploadPart.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action dans son contexte dans les exemples de code suivants :

- [Utiliser les totaux de contrôle](#)
- [Utilisation de l'intégrité des objets Amazon S3](#)

C++

SDK pour C++

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
checksum algorithm,
                                ignored when it is an empty string.
    \param body: An shared_ptr IOSTream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
&bucket,
                                                    const Aws::String &key,
                                                    const Aws::String
&uploadID,
                                                    int partNumber,

Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                    const Aws::String
&calculatedHash,
                                                    const
std::shared_ptr<Aws::IOStream> &body,
                                                    const Aws::S3::S3Client
&client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);

```



```

        break;
    case Aws::S3::Model::ChecksumAlgorithm::CRC32:
        request.SetChecksumCRC32(calculatedHash);
        break;
    case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
        request.SetChecksumCRC32C(calculatedHash);
        break;
    case Aws::S3::Model::ChecksumAlgorithm::SHA1:
        request.SetChecksumSHA1(calculatedHash);
        break;
    case Aws::S3::Model::ChecksumAlgorithm::SHA256:
        request.SetChecksumSHA256(calculatedHash);
        break;
    }
}

return client.UploadPart(request);
}

```

- Pour plus de détails sur l'API, reportez-vous [UploadPart](#) à la section Référence des AWS SDK pour C++ API.

CLI

AWS CLI

La commande suivante charge la première partie d'un chargement partitionné initié par la commande `create-multipart-upload` :

```
aws s3api upload-part --bucket amzn-s3-demo-bucket --key
'multipart/01' --part-number 1 --body part01 --upload-id
'dfRtDYU0WMCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6YjlX.cph0gtNBtJ8P'
```

L'option `body` prend le nom ou le chemin d'un fichier local à charger (n'utilisez pas le préfixe `file ://`). La taille minimale est de 5 Mo. L'ID de chargement est renvoyé par `create-multipart-upload` et peut également être récupéré avec `list-multipart-uploads`. Le compartiment et la clé sont spécifiés lorsque vous créez le chargement partitionné.

Sortie :

```
{
```

```
"ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

Enregistrez la valeur ETag de chaque partie pour plus tard. Vous en aurez besoin pour effectuer le chargement partitionné.

- Pour plus de détails sur l'API, reportez-vous [UploadPart](#) à la section Référence des AWS CLI commandes.

Rust

SDK pour Rust

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
```

```

        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

upload_parts.push(
    CompletedPart::builder()
        .e_tag(upload_part_res.e_tag.unwrap_or_default())
        .part_number(part_number)
        .build(),
);
}

```

```

// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;

```

```

// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;

```

- Pour plus de détails sur l'API, voir [UploadPart](#) la section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **UploadPartCopy** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser UploadPartCopy.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Effectuer une copie en plusieurs parties](#)

CLI

AWS CLI

Pour charger une partie d'un objet en copiant les données d'un objet existant comme source de données

L'exemple upload-part-copy suivant charge une partie d'un objet en copiant les données d'un objet existant comme source de données.

```
aws s3api upload-part-copy \  
  --bucket amzn-s3-demo-bucket \  
  --key "Map_Data_June.mp4" \  
  --copy-source "amzn-s3-demo-bucket/copy_of_Map_Data_June.mp4" \  
  --part-number 1 \  
  --upload-  
id "bq0tdE1CDpWQYRPLHuNG50xAT6pA5D.m_RiBy0gg0H6b13pVRY7QjvLlf75iFdJqp_2wztk5hvpUM2SesXgrz"
```

Sortie :

```
{  
  "CopyPartResult": {
```

```

        "LastModified": "2019-12-13T23:16:03.000Z",
        "ETag": "\"711470fc377698c393d94aed6305e245\""
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [UploadPartCopy](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

    public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
        CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
            .bucket(toBucket)
            .key(key)
            .build();

        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId) // Use the valid uploadId.
                    .partNumber(1) // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023") // Adjust range as needed
                    .build();

```

```
        return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
    })
    .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
    .whenComplete((uploadPartCopyResponse, exception) -> {
        if (exception != null) {
            // Handle any exceptions.
            logger.error("Error during upload part copy: " +
exception.getMessage());
        } else {
            // Successfully completed the upload part copy.
            System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
        }
    });
    return null;
}
```

- Pour plus de détails sur l'API, reportez-vous [UploadPartCopy](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Scénarios pour Amazon S3 utilisant AWS Kits SDK

Les exemples de code suivants vous montrent comment implémenter des scénarios courants dans Amazon S3 avec des AWS SDK. Ces scénarios vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions dans Amazon S3 ou en les combinant avec d'autres Services AWS. Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code.

Les scénarios ciblent un niveau d'expérience intermédiaire pour vous aider à comprendre les actions de service dans leur contexte.

Exemples

- [Vérification de l'existence d'un compartiment](#)
- [Vérifiez si un objet existe](#)

- [Convertissez du texte en parole et de nouveau en texte à l'aide d'un AWS Kit SDK](#)
- [Créez une URL présignée pour Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Création d'une application de gestion des ressources photographiques permettant aux utilisateurs de gérer les photos à l'aide d'étiquettes](#)
- [Une page Web répertoriant les objets Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Créer une application Amazon Textract Explorer](#)
- [Supprimez tous les objets d'un compartiment Amazon S3 donné à l'aide d'un AWS Kit SDK](#)
- [Supprimez les chargements partitionnés incomplets vers Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Déterminez le PPE dans les images avec Amazon Rekognition à l'aide d'un AWS Kit SDK](#)
- [Déterminez les entités dans le texte extrait d'une image à l'aide d'un AWS Kit SDK](#)
- [Déterminez les visages dans une image à l'aide d'un AWS Kit SDK](#)
- [Déterminez des objets dans des images avec Amazon Rekognition à l'aide d'un AWS Kit SDK](#)
- [Déterminez les personnes et les objets dans une vidéo avec Amazon Rekognition à l'aide d'un AWS Kit SDK](#)
- [Téléchargement des « répertoires » S3 à partir d'un compartiment Amazon Simple Storage Service \(Amazon S3\)](#)
- [Téléchargez tous les objets d'un compartiment Amazon Simple Storage Service \(Amazon S3\) dans un répertoire local.](#)
- [Téléchargez un flux de taille inconnue à partir d'un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Obtenez un objet Amazon S3 à partir d'un point d'accès Multi-Region à l'aide d'un AWS Kit SDK](#)
- [Obtenez un objet depuis un compartiment Amazon S3 à l'aide d'un AWS SDK, spécification d'un en-tête If-Modified-Since](#)
- [Commencez à chiffrer les objets Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Commencez à utiliser les balises pour les objets Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Débuter avec les clusters de traitement des mégadonnées](#)
- [Commencer à gérer la configuration](#)
- [Commencer à extraire le texte d'un document](#)
- [Commencer à utiliser les magasins de fonctionnalités d'apprentissage automatique](#)
- [Débuter avec le stockage d'objets](#)
- [Commencer à utiliser l'analyse des requêtes](#)
- [Utilisez les fonctionnalités de verrouillage d'objets d'Amazon S3 à l'aide d'un AWS Kit SDK](#)

- [Effectuez des demandes conditionnelles Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Gérez les listes de contrôle d'accès \(ACL\) pour les compartiments Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Gérez les messages Amazon SQS volumineux à l'aide d'Amazon S3 avec un AWS Kit SDK](#)
- [Gérez des objets Amazon S3 versionnés par lots à l'aide d'une fonction Lambda à l'aide d'un AWS Kit SDK](#)
- [Analysez les URI Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Effectuez une copie en plusieurs parties d'un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Recevez et traitez les notifications d'événements Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Enregistrez les informations EXIF et autres informations d'image à l'aide d'un AWS Kit SDK](#)
- [Envoyez des notifications d'événements S3 à Amazon à EventBridge l'aide d'un AWS Kit SDK](#)
- [Suivez le chargement ou le téléchargement d'un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Transformation des données pour votre application avec S3 Object Lambda](#)
- [Exemples d'approches pour les tests unitaires et d'intégration avec un AWS Kit SDK](#)
- [Charger récursivement un répertoire local dans un compartiment Amazon Simple Storage Service \(Amazon S3\)](#)
- [Chargez ou téléchargez des fichiers volumineux vers et depuis Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Chargez un flux de taille inconnue vers un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Utilisez des checksums pour travailler avec un objet Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Utilisez les fonctionnalités d'intégrité des objets d'Amazon S3 à l'aide d'un AWS Kit SDK](#)
- [Travaillez avec des objets versionnés Amazon S3 à l'aide d'un AWS Kit SDK](#)

Vérification de l'existence d'un compartiment

L'exemple de code suivant montre comment vérifier si un compartiment existe.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Vous pouvez utiliser la `doesBucketExists` méthode suivante en remplacement de la méthode [AmazonS3Client BucketExist #does V2 \(String\) du SDK for Java V1](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.http.HttpStatusCode;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.utils.Validate;

public class DoesBucketExist {
    private static final Logger logger =
        LoggerFactory.getLogger(DoesBucketExist.class);

    public static void main(String[] args) {
        DoesBucketExist doesBucketExist = new DoesBucketExist();

        final S3Client s3SyncClient = S3Client.builder().build();
        final String bucketName = "amzn-s3-demo-bucket"; // Replace with the
        bucket name that you want to check.

        boolean exists = doesBucketExist.doesBucketExist(bucketName,
            s3SyncClient);
        logger.info("Bucket exists: {}", exists);
    }

    /**
     * Checks if the specified bucket exists. Amazon S3 buckets are named in a
     * global namespace; use this method to
     * determine if a specified bucket name already exists, and therefore can't
     * be used to create a new bucket.
     */
}
```

```

    * <p>
    * Internally this method uses the <a
    * href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#getBucketAcl(java.util.function.Consumer)">S3Client.getBucketAcl(String)</
a>
    * operation to determine whether the bucket exists.
    * <p>
    * This method is equivalent to the AWS SDK for Java V1's <a
    * href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/
com/amazonaws/services/s3/AmazonS3Client.html#doesBucketExistV2-
java.lang.String-">AmazonS3Client#doesBucketExistV2(String)</a>.
    *
    * @param bucketName The name of the bucket to check.
    * @param s3SyncClient An <code>S3Client</code> instance. The method checks
for the bucket in the AWS Region
    *
    * configured on the instance.
    * @return The value true if the specified bucket exists in Amazon S3; the
value false if there is no bucket in
    *
    * Amazon S3 with that name.
    */
    public boolean doesBucketExist(String bucketName, S3Client s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
empty string.", "");
            s3SyncClient.getBucketAcl(r -> r.bucket(bucketName));
            return true;
        } catch (AwsServiceException ase) {
            // A redirect error or an AccessDenied exception means the bucket
exists but it's not in this region
            // or we don't have permissions to it.
            if ((ase.statusCode() == HttpStatus.MOVED_PERMANENTLY) ||
"AccessDenied".equals(ase.awsErrorDetails().errorCode())) {
                return true;
            }
            if (ase.statusCode() == HttpStatus.NOT_FOUND) {
                return false;
            }
            throw ase;
        }
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [GetBucketAcl](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Vérifiez si un objet existe

L'exemple de code suivant montre comment vérifier l'existence d'un objet.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Vous pouvez utiliser la `doesObjectExist` méthode suivante en remplacement de la méthode [AmazonS3Client ObjectExist #does \(String, String\) du SDK pour Java V1](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.NoSuchKeyException;
import software.amazon.awssdk.utils.Validate;

public class DoesObjectExist {
    private static final Logger logger =
        LoggerFactory.getLogger(DoesObjectExist.class);

    public static void main(String[] args) {
        DoesObjectExist doesObjectExist = new DoesObjectExist();

        final S3Client s3SyncClient = S3Client.builder().build();
```

```

        final String bucketName = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
        final String key = "my-key"; // Replace with your object key.

        boolean exists = doesObjectExist.doesObjectExist(bucketName, key,
s3SyncClient);
        logger.info("Object exists: {}", exists);
    }

    /**
     * Checks if the specified object exists in the specified bucket.
     * <p>
     * Internally this method uses the
     * <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/awssdk/
services/s3/
S3Client.html#headObject(java.util.function.Consumer)">S3Client.headObject</a>
     * operation to determine whether the object exists.
     * <p>
     * This method is equivalent to the AWS SDK for Java V1's
     * <a href="https://docs.aws.amazon.com/AWSJavaSDK/latest/javadoc/com/
amazonaws/services/s3/AmazonS3Client.html#doesObjectExist-java.lang.String-
java.lang.String-">AmazonS3Client#doesObjectExist(String, String)</a>.
     * <p>
     * <b>Note:</b> This method returns {@code false} only when S3 responds with
404 (NoSuchKey). If the caller
     * does not have {@code s3:ListBucket} permission on the bucket, S3 may
return 403 instead of 404 for
     * non-existent objects, which will be thrown as an exception.
     *
     * @param bucketName The name of the bucket containing the object.
     * @param key The key of the object to check.
     * @param s3SyncClient An {@code S3Client} instance.
     * @return {@code true} if the object exists; {@code false} if it does not
exist.
     */
    public boolean doesObjectExist(String bucketName, String key, S3Client
s3SyncClient) {
        try {
            Validate.notEmpty(bucketName, "The bucket name must not be null or an
empty string.", "");
            Validate.notEmpty(key, "The object key must not be null or an empty
string.", "");
            s3SyncClient.headObject(HeadObjectRequest.builder()
                .bucket(bucketName)

```

```
        .key(key)
        .build());
    return true;
} catch (NoSuchKeyException e) {
    return false;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [HeadObject](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Convertissez du texte en parole et de nouveau en texte à l'aide d'un AWS Kit SDK

L'exemple de code suivant illustre comment :

- Utilisez Amazon Polly pour synthétiser un fichier d'entrée en texte brut (UTF-8) en un fichier audio.
- Chargez le fichier audio sur un compartiment Amazon S3.
- Utilisez Amazon Transcribe pour convertir le fichier audio en texte.
- Affichez le texte.

Rust

SDK pour Rust

Utilisez Amazon Polly pour synthétiser un fichier d'entrée en texte brut (UTF-8) en fichier audio, chargez le fichier audio dans un compartiment Amazon S3, utilisez Amazon Transcribe pour convertir ce fichier audio en texte et affichez le texte.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Polly

- Amazon S3
- Amazon Transcribe

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Créez une URL présignée pour Amazon S3 à l'aide d'un AWS Kit SDK

Les exemples de code suivants montrent comment créer une URL présignée pour Amazon S3 et charger un objet.

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Générez une URL présignée qui peut exécuter une action Amazon S3 pour une durée limitée.

```
using System;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

public class GenPresignedUrl
{
    public static void Main()
    {
        const string bucketName = "amzn-s3-demo-bucket";
        const string objectKey = "sample.txt";

        // Specify how long the presigned URL lasts, in hours
        const double timeoutDuration = 12;

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
```

```

        // pass the Region to the constructor for the client. For
        // example: new AmazonS3Client(RegionEndpoint.USEast1);

        // If using the Region us-east-1, and server-side encryption with AWS
        KMS, you must specify Signature Version 4.
        // Region us-east-1 defaults to Signature Version 2 unless explicitly
        set to Version 4 as shown below.
        // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingAWSSDK.html#specify-signature-version
        // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/TAWSConfigsS3.html
        AWSConfigsS3.UseSignatureVersion4 = true;
        IAmazonS3 s3Client = new AmazonS3Client(RegionEndpoint.USEast1);

        string urlString = GeneratePresignedURL(s3Client, bucketName,
        objectKey, timeoutDuration);
        Console.WriteLine($"The generated URL is: {urlString}.");
    }

    /// <summary>
    /// Generate a presigned URL that can be used to access the file named
    /// in the objectKey parameter for the amount of time specified in the
    /// duration parameter.
    /// </summary>
    /// <param name="client">An initialized S3 client object used to call
    /// the GetPresignedUrl method.</param>
    /// <param name="bucketName">The name of the S3 bucket containing the
    /// object for which to create the presigned URL.</param>
    /// <param name="objectKey">The name of the object to access with the
    /// presigned URL.</param>
    /// <param name="duration">The length of time for which the presigned
    /// URL will be valid.</param>
    /// <returns>A string representing the generated presigned URL.</returns>
    public static string GeneratePresignedURL(IAmazonS3 client, string
    bucketName, string objectKey, double duration)
    {
        string urlString = string.Empty;
        try
        {
            var request = new GetPreSignedUrlRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Expires = DateTime.UtcNow.AddHours(duration),
            }
        }
    }

```

```
        };
        urlString = client.GetPreSignedURL(request);
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error: '{ex.Message}'");
    }

    return urlString;
}
}
```

Générez une URL présignée et effectuez un chargement en utilisant cette URL.

```
using System;
using System.IO;
using System.Net.Http;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to upload an object to an Amazon Simple Storage
/// Service (Amazon S3) bucket using a presigned URL. The code first
/// creates a presigned URL and then uses it to upload an object to an
/// Amazon S3 bucket using that URL.
/// </summary>
public class UploadUsingPresignedURL
{
    private static HttpClient httpClient = new HttpClient();

    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "samplefile.txt";
        string filePath = $"source\\{keyName}";

        // Specify how long the signed URL will be valid in hours.
        double timeoutDuration = 12;
```



```

        // Specify the AWS Region of your Amazon S3 bucket. If it is
        // different from the Region defined for the default user,
        // pass the Region to the constructor for the client. For
        // example: new AmazonS3Client(RegionEndpoint.USEast1);

        // If using the Region us-east-1, and server-side encryption with AWS
        KMS, you must specify Signature Version 4.
        // Region us-east-1 defaults to Signature Version 2 unless explicitly
        set to Version 4 as shown below.
        // For more details, see https://docs.aws.amazon.com/AmazonS3/latest/userguide/UsingAWSSDK.html#specify-signature-version
        // and https://docs.aws.amazon.com/sdkfornet/v3/apidocs/items/Amazon/TAWSConfigsS3.html
        AWSConfigsS3.UseSignatureVersion4 = true;
        IAmazonS3 client = new AmazonS3Client(RegionEndpoint.USEast1);

        var url = GeneratePreSignedURL(client, bucketName, keyName,
        timeoutDuration);
        var success = await UploadObject(filePath, url);

        if (success)
        {
            Console.WriteLine("Upload succeeded.");
        }
        else
        {
            Console.WriteLine("Upload failed.");
        }
    }

    /// <summary>
    /// Uploads an object to an Amazon S3 bucket using the presigned URL
    passed in
    /// the url parameter.
    /// </summary>
    /// <param name="filePath">The path (including file name) to the local
    /// file you want to upload.</param>
    /// <param name="url">The presigned URL that will be used to upload the
    /// file to the Amazon S3 bucket.</param>
    /// <returns>A Boolean value indicating the success or failure of the
    /// operation, based on the HttpResponseMessage.</returns>
    public static async Task<bool> UploadObject(string filePath, string url)
    {
        using var streamContent = new StreamContent(

```

```

        new FileStream(filePath, FileMode.Open, FileAccess.Read));

        var response = await httpClient.PutAsync(url, streamContent);
        return response.IsSuccessStatusCode;
    }

    /// <summary>
    /// Generates a presigned URL which will be used to upload an object to
    /// an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetPreSignedURL.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
    /// presigned URL will point.</param>
    /// <param name="objectKey">The name of the file that will be uploaded.</
param>
    /// <param name="duration">How long (in hours) the presigned URL will
    /// be valid.</param>
    /// <returns>The generated URL.</returns>
    public static string GeneratePreSignedURL(
        IAmazonS3 client,
        string bucketName,
        string objectKey,
        double duration)
    {
        var request = new GetPreSignedUrlRequest
        {
            BucketName = bucketName,
            Key = objectKey,
            Verb = HttpVerb.PUT,
            Expires = DateTime.UtcNow.AddHours(duration),
        };

        string url = client.GetPreSignedURL(request);
        return url;
    }
}

```

SDK pour .NET (v4)

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez et utilisez des URL POST présignées pour les téléchargements directs par navigateur.

```
/// <summary>
/// Scenario demonstrating the complete workflow for presigned POST URLs:
/// 1. Create an S3 bucket
/// 2. Create a presigned POST URL
/// 3. Upload a file using the presigned POST URL
/// 4. Clean up resources
/// </summary>
public class CreatePresignedPostBasics
{
    public static ILogger<CreatePresignedPostBasics> _logger = null!;
    public static S3Wrapper _s3Wrapper = null!;
    public static UiMethods _uiMethods = null!;
    public static IHttpClientFactory _httpClientFactory = null!;
    public static bool _isInteractive = true;
    public static string? _bucketName;
    public static string? _objectKey;

    /// <summary>
    /// Set up the services and logging.
    /// </summary>
    /// <param name="host">The IHost instance.</param>
    public static void SetupServices(IHost host)
    {
        var loggerFactory = LoggerFactory.Create(builder =>
        {
            builder.AddConsole();
        });
        _logger = new Logger<CreatePresignedPostBasics>(loggerFactory);

        _s3Wrapper = host.Services.GetRequiredService<S3Wrapper>();
        _httpClientFactory =
            host.Services.GetRequiredService<IHttpClientFactory>();
        _uiMethods = new UiMethods();
    }
}
```

```
}

/// <summary>
/// Perform the actions defined for the Amazon S3 Presigned POST scenario.
/// </summary>
/// <param name="args">Command line arguments.</param>
/// <returns>A Task object.</returns>
public static async Task Main(string[] args)
{
    _isInteractive = !args.Contains("--non-interactive");

    // Set up dependency injection for Amazon S3
    using var host =
        Microsoft.Extensions.Hosting.Host.CreateDefaultBuilder(args)
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonS3>()
                    .AddTransient<S3Wrapper>()
                    .AddHttpClient()
            )
            .Build();

    SetupServices(host);

    try
    {
        // Display overview
        _uiMethods.DisplayOverview();
        _uiMethods.PressEnter(_isInteractive);

        // Step 1: Create bucket
        await CreateBucketAsync();
        _uiMethods.PressEnter(_isInteractive);

        // Step 2: Create presigned URL
        _uiMethods.DisplayTitle("Step 2: Create presigned POST URL");
        var response = await CreatePresignedPostAsync();
        _uiMethods.PressEnter(_isInteractive);

        // Step 3: Display URL and fields
        _uiMethods.DisplayTitle("Step 3: Presigned POST URL details");
        DisplayPresignedPostFields(response);
        _uiMethods.PressEnter(_isInteractive);

        // Step 4: Upload file
```

```

        _uiMethods.DisplayTitle("Step 4: Upload test file using presigned
POST URL");
        await UploadFileAsync(response);
        _uiMethods.PressEnter(_isInteractive);

        // Step 5: Verify file exists
        await VerifyFileExistsAsync();
        _uiMethods.PressEnter(_isInteractive);

        // Step 6: Cleanup
        _uiMethods.DisplayTitle("Step 6: Clean up resources");
        await CleanupAsync();

        _uiMethods.DisplayTitle("S3 Presigned POST Scenario completed
successfully!");
        _uiMethods.PressEnter(_isInteractive);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error in scenario");
        Console.WriteLine($"Error: {ex.Message}");

        // Attempt cleanup if there was an error
        if (!string.IsNullOrEmpty(_bucketName))
        {
            _uiMethods.DisplayTitle("Cleaning up resources after error");
            await _s3Wrapper.DeleteBucketAsync(_bucketName);
            Console.WriteLine($"Cleaned up bucket: {_bucketName}");
        }
    }
}

/// <summary>
/// Create an S3 bucket for the scenario.
/// </summary>
private static async Task CreateBucketAsync()
{
    _uiMethods.DisplayTitle("Step 1: Create an S3 bucket");

    // Generate a default bucket name for the scenario
    var defaultBucketName = $"presigned-post-demo-
{DateTime.Now:yyyyMMddHHmmss}".ToLower();

    // Prompt user for bucket name or use default in non-interactive mode

```

```
        _bucketName = _uiMethods.GetUserInput(
            $"Enter S3 bucket name (or press Enter for '{defaultBucketName}'):",
            defaultBucketName,
            _isInteractive);

    // Basic validation to ensure bucket name is not empty
    if (string.IsNullOrEmpty(_bucketName))
    {
        _bucketName = defaultBucketName;
    }

    Console.WriteLine($"Creating bucket: {_bucketName}");

    await _s3Wrapper.CreateBucketAsync(_bucketName);

    Console.WriteLine($"Successfully created bucket: {_bucketName}");
}

/// <summary>
/// Create a presigned POST URL.
/// </summary>
private static async Task<CreatePresignedPostResponse>
CreatePresignedPostAsync()
{
    _objectKey = "example-upload.txt";
    var expiration = DateTime.UtcNow.AddMinutes(10); // Short expiration for
the demo

    Console.WriteLine($"Creating presigned POST URL for {_bucketName}/
{_objectKey}");
    Console.WriteLine($"Expiration: {expiration} UTC");

    var s3Client = _s3Wrapper.GetS3Client();

    var response = await _s3Wrapper.CreatePresignedPostAsync(
        s3Client, _bucketName!, _objectKey, expiration);

    Console.WriteLine("Successfully created presigned POST URL");
    return response;
}

/// <summary>
/// Upload a file using the presigned POST URL.
```

```
    /// </summary>
    private static async Task UploadFileAsync(CreatePresignedPostResponse
response)
    {

        // Create a temporary test file to upload
        string testFilePath = Path.GetTempFileName();
        string testContent = "This is a test file for the S3 presigned POST
scenario.";

        await File.WriteAllTextAsync(testFilePath, testContent);
        Console.WriteLine($"Created test file at: {testFilePath}");

        // Upload the file using the presigned POST URL
        Console.WriteLine("\nUploading file using the presigned POST URL...");
        var uploadResult = await UploadFileWithPresignedPostAsync(response,
testFilePath);

        // Display the upload result
        if (uploadResult.Success)
        {
            Console.WriteLine($"Upload successful! Status code:
{uploadResult.StatusCode}");
        }
        else
        {
            Console.WriteLine($"Upload failed with status code:
{uploadResult.StatusCode}");
            Console.WriteLine($"Error: {uploadResult.Response}");
            throw new Exception("File upload failed");
        }

        // Clean up the temporary file
        File.Delete(testFilePath);
        Console.WriteLine("Temporary file deleted");
    }

    /// <summary>
    /// Helper method to upload a file using a presigned POST URL.
    /// </summary>
    private static async Task<(bool Success, HttpStatusCode StatusCode, string
Response)> UploadFileWithPresignedPostAsync(
        CreatePresignedPostResponse response,
        string filePath)
```

```
{
    try
    {
        _logger.LogInformation("Uploading file {filePath} using presigned
POST URL", filePath);

        using var httpClient = _httpClientFactory.CreateClient();
        using var formContent = new MultipartFormDataContent();

        // Add all the fields from the presigned POST response
        foreach (var field in response.Fields)
        {
            formContent.Add(new StringContent(field.Value), field.Key);
        }

        // Add the file content
        var fileStream = File.OpenRead(filePath);
        var fileName = Path.GetFileName(filePath);
        var fileContent = new StreamContent(fileStream);
        fileContent.Headers.ContentType = new MediaTypeHeaderValue("text/
plain");
        formContent.Add(fileContent, "file", fileName);

        // Send the POST request
        var httpResponse = await httpClient.PostAsync(response.Url,
formContent);
        var responseContent = await httpResponse.Content.ReadAsStringAsync();

        // Log and return the result
        _logger.LogInformation("Upload completed with status code
{statusCode}", httpResponse.StatusCode);

        return (httpResponse.IsSuccessStatusCode, httpResponse.StatusCode,
responseContent);
    }
    catch (Exception ex)
    {
        _logger.LogError(ex, "Error uploading file");
        return (false, HttpStatusCode.InternalServerError, ex.Message);
    }
}

/// <summary>
/// Verify that the uploaded file exists in the S3 bucket.
```



```
/// </summary>
private static async Task VerifyFileExistsAsync()
{
    _uiMethods.DisplayTitle("Step 5: Verify uploaded file exists");

    Console.WriteLine($"Checking if file exists at {_bucketName}/
{_objectKey}...");

    try
    {
        var metadata = await _s3Wrapper.GetObjectMetadataAsync(_bucketName!,
            _objectKey!);

        Console.WriteLine($"File verification successful! File exists in the
bucket.");
        Console.WriteLine($"File size: {metadata.ContentLength} bytes");
        Console.WriteLine($"File type: {metadata.Headers.ContentType}");
        Console.WriteLine($"Last modified: {metadata.LastModified}");
    }
    catch (AmazonS3Exception ex) when (ex.StatusCode ==
        System.Net.HttpStatusCode.NotFound)
    {
        Console.WriteLine($"Error: File was not found in the bucket.");
        throw;
    }
}

private static void DisplayPresignedPostFields(CreatePresignedPostResponse
response)
{
    Console.WriteLine($"Presigned POST URL: {response.Url}");
    Console.WriteLine("Form fields to include:");

    foreach (var field in response.Fields)
    {
        Console.WriteLine($" {field.Key}: {field.Value}");
    }
}

/// <summary>
/// Clean up resources created by the scenario.
/// </summary>
private static async Task CleanupAsync()
{

```

```
        if (!string.IsNullOrEmpty(_bucketName))
        {
            Console.WriteLine($"Deleting bucket {_bucketName} and its
contents...");
            bool result = await _s3Wrapper.DeleteBucketAsync(_bucketName);

            if (result)
            {
                Console.WriteLine("Bucket deleted successfully");
            }
            else
            {
                Console.WriteLine("Failed to delete bucket - it may have been
already deleted");
            }
        }
    }
}
```

C++

SDK pour C++

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Générez une URL présignée pour télécharger un objet.

```
//! Routine which demonstrates creating a pre-signed URL to download an object
from an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
    \param bucketName: Name of the bucket.
    \param key: Name of an object key.
    \param expirationSeconds: Expiration in seconds for pre-signed URL.
    \param clientConfig: Aws client configuration.
    \return Aws::String: A pre-signed URL.
*/
```

```

Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String
    &bucketName,
                                                    const Aws::String &key,
                                                    uint64_t expirationSeconds,
                                                    const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_GET,
                                                    expirationSeconds);
}

```

Téléchargez à l'aide de libcurl.

```

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
    \param presignedURL: A pre-signed URL to get an object from a bucket.
    \param resultString: A string to hold the result.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                    Aws::String &resultString) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }
}

```

```

    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
std::endl;
        return false;
    }

    return true;
}

```

Générez une URL présignée pour charger un objet.

```

//! Routine which demonstrates creating a pre-signed URL to upload an object to
an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
    \param bucketName: Name of the bucket.
    \param key: Name of an object key.

```

```

    \param clientConfig: Aws client configuration.
    \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String
    &bucketName,
                                                    const Aws::String &key,
                                                    uint64_t expirationSeconds,
                                                    const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_PUT,
                                                    expirationSeconds);
}

```

Chargez à l'aide de libcurl.

```

static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
    *userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
    \param presignedURL: A pre-signed URL to put an object in a bucket.
    \param data: Body of the putObject request.
    \return bool: Function succeeded.
*/

```

```
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                  const Aws::String &data) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }
}
```

```
result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_URL" << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
    return false;
}

result = curl_easy_perform(curl);

if (result != CURLE_OK) {
    std::cerr << "Failed to perform CURL request" << std::endl;
    return false;
}

std::string outString = outWriteString.str();
if (outString.empty()) {
    std::cout << "Successfully put object." << std::endl;
    return true;
} else {
    std::cout << "A server error was encountered, output:\n" << outString
                << std::endl;
    return false;
}
}
```

Go

Kit SDK pour Go V2

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez des fonctions qui enveloppent les actions de présignature S3.

```
import (
    "context"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
// actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
// S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
// client.
type Presigner struct {
    PresignClient *s3.PresignClient
}

// GetObject makes a presigned request that can be used to get an object from a
// bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
(*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignGetObject(ctx,
        &s3.GetObjectInput{
            Bucket: aws.String(bucketName),
            Key:   aws.String(objectKey),
        }, func(opts *s3.PresignOptions) {
            opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return request, err
}
```



```
}

// PutObject makes a presigned request that can be used to put an object in a
// bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
(*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignPutObject(ctx,
        &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
        }, func(opts *s3.PresignOptions) {
            opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return request, err
}

// DeleteObject makes a presigned request that can be used to delete an object
// from a bucket.
func (presigner Presigner) DeleteObject(ctx context.Context, bucketName string,
    objectKey string) (*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignDeleteObject(ctx,
        &s3.DeleteObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
        })
    if err != nil {
        log.Printf("Couldn't get a presigned request to delete object %v. Here's why:
        %v\n", objectKey, err)
    }
    return request, err
}
```

```
func (presigner Presigner) PresignPostObject(ctx context.Context, bucketName
    string, objectKey string, lifetimeSecs int64) (*s3.PresignedPostRequest, error)
{
    request, err := presigner.PresignClient.PresignPostObject(ctx,
    &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    }, func(options *s3.PresignPostOptions) {
        options.Expires = time.Duration(lifetimeSecs) * time.Second
    })
    if err != nil {
        log.Printf("Couldn't get a presigned post request to put %v:%v. Here's why: %v
    \n", bucketName, objectKey, err)
    }
    return request, nil
}
```

Exécutez un exemple interactif qui génère et utilise des URL présignées pour charger, télécharger et supprimer un objet S3.

```
import (
    "bytes"
    "context"
    "io"
    "log"
    "mime/multipart"
    "net/http"
    "os"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)
```

```
// RunPresigningScenario is an interactive example that shows you how to get
// presigned
// HTTP requests that you can use to move data into and out of Amazon Simple
// Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and
// can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local
//    file to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the
//    object to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../..demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
// mocked
// during testing.
func RunPresigningScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner, httpRequester IHttpRequester) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := questioner.(*demotools.MockQuestioner)
            if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
                log.Println(r)
            }
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon S3 presigning demo.")
    log.Println(strings.Repeat("-", 88))
}
```

```
s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}
presignClient := s3.NewPresignClient(s3Client)
presigner := actions.Presigner{PresignClient: presignClient}

bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
    "you own or one you want to create:", demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
    demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedPutRequest.Method,
presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
    panic(err)
}
```

```
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Now we'll create a new request to put the same object using a
presigned post request")
questioner.Ask("Press Enter when you're ready.")
presignPostRequest, err := presigner.PresignPostObject(ctx, bucketName,
uploadKey, 60)
if err != nil {
```

```
    panic(err)
}
log.Printf("Got a presigned post request to url %v with values %v\n",
presignPostRequest.URL, presignPostRequest.Values)
log.Println("Using net/http multipart to send the request...")
uploadFile, err = os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
multiPartResponse, err := sendMultipartRequest(presignPostRequest.URL,
presignPostRequest.Values, uploadFile, uploadKey, httpRequester)
if err != nil {
    panic(err)
}
log.Printf("Presign post object %v with presigned URL returned %v.", uploadKey,
multiPartResponse.StatusCode)

log.Println("Let's presign a request to delete the object.")
questioner.Ask("Press Enter when you're ready.")
presignedDelRequest, err := presigner.DeleteObject(ctx, bucketName, uploadKey)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedDelRequest.Method,
presignedDelRequest.URL)
log.Println("Using net/http to send the request...")
delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.\n",
presignedDelRequest.Method,
uploadKey, delResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

Définissez un wrapper de requête HTTP utilisé par l'exemple pour effectuer des requêtes HTTP.

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
// during
// unit testing.
type IHttpRequester interface {
    Get(url string) (resp *http.Response, err error)
    Post(url, contentType string, body io.Reader) (resp *http.Response, err error)
    Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
    error)
    Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
// scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
    return http.Get(url)
}
func (httpReq HttpRequester) Post(url, contentType string, body io.Reader) (resp
    *http.Response, err error) {
    postRequest, err := http.NewRequest("POST", url, body)
    if err != nil {
        return nil, err
    }
    postRequest.Header.Set("Content-Type", contentType)
    return http.DefaultClient.Do(postRequest)
}

func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
    (resp *http.Response, err error) {
    putRequest, err := http.NewRequest("PUT", url, body)
    if err != nil {
        return nil, err
    }
    putRequest.ContentLength = contentLength
    return http.DefaultClient.Do(putRequest)
}
func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
    {
```

```
delRequest, err := http.NewRequest("DELETE", url, nil)
if err != nil {
    return nil, err
}
return http.DefaultClient.Do(delRequest)
}
```

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Voici trois exemples de création d'URL présignées et d'utilisation des URL avec des bibliothèques clientes HTTP :

- Une requête GET HTTP qui utilise l'URL de trois bibliothèques clientes HTTP
- Une requête PUT HTTP avec des métadonnées dans les en-têtes qui utilise l'URL de trois bibliothèques clientes HTTP
- Une requête PUT HTTP avec des paramètres de requête qui utilise l'URL d'une bibliothèque cliente HTTP

Générez une URL pré-signée pour un objet, puis téléchargez-le (requête GET).

Importations.

```
import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
```



```
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.utils.IoUtils;

import java.io.ByteArrayOutputStream;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.UUID;
```

Générez l'URL.

```
/* Create a pre-signed URL to download an object in a subsequent GET request.
 */
public String createPresignedGetUrl(String bucketName, String keyName) {
    try (S3Presigner presigner = S3Presigner.create()) {

        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .build();

        GetObjectPresignRequest presignRequest =
            GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL will
                    expire in 10 minutes.
                .getObjectRequest(objectRequest)
                .build();
```

```

        PresignedGetObjectRequest presignedRequest =
presigner.presignGetObject(presignRequest);
        logger.info("Presigned URL: [{}]",
presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Téléchargez l'objet en utilisant l'une des trois approches suivantes.

Utilisez la classe JDK `URLConnection` (depuis la version 1.1) pour effectuer le téléchargement.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Utilisez la classe JDK `HttpClient` (depuis v11) pour effectuer le téléchargement.

```

/* Use the JDK HttpClient (since v11) class to do the download. */

```

```
public byte[] useHttpClientToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
    ByteArrayOutputStream(); // Capture the response body to a byte array.

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpResponse<InputStream> response = httpClient.send(requestBuilder
            .uri(presignedUrl.toURI())
            .GET()
            .build(),
            HttpResponse.BodyHandlers.ofInputStream());

        IoUtils.copy(response.body(), byteArrayOutputStream);

        logger.info("HTTP response code is " + response.statusCode());
    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}
```

Utilisez la classe AWS SDK for SdkHttpClient Java pour effectuer le téléchargement.

```
/* Use the AWS SDK for Java SdkHttpClient class to do the download. */
public byte[] useSdkHttpClientToGet(String presignedUrlString) {

    ByteArrayOutputStream byteArrayOutputStream = new
    ByteArrayOutputStream(); // Capture the response body to a byte array.
    try {
        URL presignedUrl = new URL(presignedUrlString);
        SdkHttpRequest request = SdkHttpRequest.builder()
            .method(SdkHttpMethod.GET)
            .uri(presignedUrl.toURI())
            .build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .build();
    }
```

```

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            response.responseBody().ifPresentOrElse(
                abortableInputStream -> {
                    try {
                        IoUtils.copy(abortableInputStream,
byteArrayOutputStream);
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                },
                () -> logger.error("No response body."));

            logger.info("HTTP Response code is {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Générez une URL pré-signée avec des métadonnées dans les en-têtes pour un téléchargement, puis chargez un fichier (requête PUT).

Importations.

```

import com.example.s3.util.PresignUrlUtils;
import org.slf4j.Logger;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;

```

```
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.io.OutputStream;
import java.io.RandomAccessFile;
import java.net.HttpURLConnection;
import java.net.URISyntaxException;
import java.net.URL;
import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;
```

Générez l'URL.

```
/* Create a presigned URL to use in a subsequent PUT request */
public String createPresignedUrl(String bucketName, String keyName,
    Map<String, String> metadata) {
    try (S3Presigner presigner = S3Presigner.create()) {

        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)
            .metadata(metadata)
            .build();

        PutObjectPresignRequest presignRequest =
        PutObjectPresignRequest.builder()
            .signatureDuration(Duration.ofMinutes(10)) // The URL
            expires in 10 minutes.
            .putObjectRequest(objectRequest)
            .build();
```

```

        PresignedPutObjectRequest presignedRequest =
presigner.presignPutObject(presignRequest);
        String myURL = presignedRequest.url().toString();
        logger.info("Presigned URL to upload a file to: [{}]", myURL);
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Chargez l'objet de fichier en utilisant l'une des trois approches suivantes.

Utilisez la classe JDK `URLConnection` (depuis la version 1.1) pour effectuer le chargement.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the upload. */
public void useHttpURLConnectionToPut(String presignedUrlString, File
fileToPut, Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());
    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setDoOutput(true);
        metadata.forEach((k, v) -> connection.setRequestProperty("x-amz-
meta-" + k, v));
        connection.setRequestMethod("PUT");
        OutputStream out = connection.getOutputStream();

        try (RandomAccessFile file = new RandomAccessFile(fileToPut, "r");
            FileChannel inChannel = file.getChannel()) {
            ByteBuffer buffer = ByteBuffer.allocate(8192); //Buffer size is
8k

            while (inChannel.read(buffer) > 0) {
                buffer.flip();
                for (int i = 0; i < buffer.limit(); i++) {
                    out.write(buffer.get());
                }
            }
        }
    }
}

```

```

        buffer.clear();
    }
} catch (IOException e) {
    logger.error(e.getMessage(), e);
}

out.close();
connection.getResponseCode();
logger.info("HTTP response code is " + connection.getResponseCode());

} catch (S3Exception | IOException e) {
    logger.error(e.getMessage(), e);
}
}

```

Utilisez la classe JDK `HttpClient` (depuis la version 11) pour effectuer le chargement.

```

/* Use the JDK HttpClient (since v11) class to do the upload. */
public void useHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    HttpRequest.Builder requestBuilder = HttpRequest.newBuilder();
    metadata.forEach((k, v) -> requestBuilder.header("x-amz-meta-" + k, v));

    HttpClient httpClient = HttpClient.newHttpClient();
    try {
        final HttpResponse<Void> response = httpClient.send(requestBuilder
            .uri(new URL(presignedUrlString).toURI())

        .PUT(HttpRequest.BodyPublishers.ofFile(Path.of(fileToPut.toURI()))
            .build(),
            HttpResponse.BodyHandlers.discarding());

        logger.info("HTTP response code is " + response.statusCode());

    } catch (URISyntaxException | InterruptedException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}

```

Utilisez la `SdkHttpClient` classe AWS for Java V2 pour effectuer le téléchargement.

```
/* Use the AWS SDK for Java V2 SdkHttpClient class to do the upload. */
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut,
Map<String, String> metadata) {
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());
        // Add headers
        metadata.forEach((k, v) -> requestBuilder.putHeader("x-amz-meta-" +
k, v));
        // Finish building the request.
        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

Générez une URL pré-signée avec des paramètres de requête pour un chargement, puis chargez un fichier (requête PUT).

Importations.

```
import com.example.s3.util.PresignUrlUtils;
```



```

import org.slf4j.Logger;
import software.amazon.awssdk.awscore.AwsRequestOverrideConfiguration;
import software.amazon.awssdk.core.internal.sync.FileContentStreamProvider;
import software.amazon.awssdk.http.HttpExecuteRequest;
import software.amazon.awssdk.http.HttpExecuteResponse;
import software.amazon.awssdk.http.SdkHttpClient;
import software.amazon.awssdk.http.SdkHttpMethod;
import software.amazon.awssdk.http.SdkHttpRequest;
import software.amazon.awssdk.http.apache.ApacheHttpClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedPutObjectRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PutObjectPresignRequest;

import java.io.File;
import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;
import java.time.Duration;
import java.util.Map;
import java.util.UUID;

```

Générez l'URL.

```

/**
 * Creates a presigned URL to use in a subsequent HTTP PUT request. The code
 * adds query parameters
 * to the request instead of using headers. By using query parameters, you
 * do not need to add the
 * the parameters as headers when the PUT request is eventually sent.
 *
 * @param bucketName Bucket name where the object will be uploaded.
 * @param keyName Key name of the object that will be uploaded.
 * @param queryParams Query string parameters to be added to the presigned
 * URL.
 * @return
 */

```

```

    public String createPresignedUrl(String bucketName, String keyName,
    Map<String, String> queryParams) {
        try (S3Presigner presigner = S3Presigner.create()) {
            // Create an override configuration to store the query parameters.
            AwsRequestOverrideConfiguration.Builder overrideConfigurationBuilder
            = AwsRequestOverrideConfiguration.builder();

            queryParams.forEach(overrideConfigurationBuilder::putRawQueryParameter);

            PutObjectRequest objectRequest = PutObjectRequest.builder()
                .bucket(bucketName)
                .key(keyName)

            .overrideConfiguration(overrideConfigurationBuilder.build()) // Add the override
            configuration.
                .build();

            PutObjectPresignRequest presignRequest =
            PutObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // The URL
            expires in 10 minutes.
                .putObjectRequest(objectRequest)
                .build();

            PresignedPutObjectRequest presignedRequest =
            presigner.presignPutObject(presignRequest);
            String myURL = presignedRequest.url().toString();
            logger.info("Presigned URL to upload a file to: [{}]", myURL);
            logger.info("HTTP method: [{}]",
            presignedRequest.httpRequest().method());

            return presignedRequest.url().toExternalForm();
        }
    }
}

```

Utilisez la `SdkHttpClient` classe AWS for Java V2 pour effectuer le téléchargement.

```

/**
 * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT
 request. Since the

```

```
    * URL contains the query parameters, no headers are needed for metadata, SSE
    settings, or ACL settings.
    *
    * @param presignedUrlString The URL for the PUT request.
    * @param fileToPut File to upload
    */
    public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut)
    {
        logger.info("Begin [{}] upload", fileToPut.toString());

        try {
            URL presignedUrl = new URL(presignedUrlString);

            SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
                .method(SdkHttpMethod.PUT)
                .uri(presignedUrl.toURI());

            SdkHttpRequest request = requestBuilder.build();

            HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
                .request(request)
                .contentStreamProvider(new
                FileContentStreamProvider(fileToPut.toPath()))
                .build();

            try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
                HttpExecuteResponse response =
                sdkHttpClient.prepareRequest(executeRequest).call();
                logger.info("Response code: {}",
                response.httpResponse().statusCode());
            }
        } catch (URISyntaxException | IOException e) {
            logger.error(e.getMessage(), e);
        }
    }
}
```

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez une URL présignée pour charger un objet dans un compartiment.

```
import https from "node:https";

import { XMLParser } from "fast-xml-parser";
import { PutObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
  S3RequestPresigner,
} from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(
    new HttpRequest({ ...url, method: "PUT" })
  );
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new PutObjectCommand({ Bucket: bucket, Key: key });
```

```
    return getSignedUrl(client, command, { expiresIn: 3600 });
  };

  /**
   * Make a PUT request to the provided URL.
   *
   * @param {string} url
   * @param {string} data
   */
  const put = (url, data) => {
    return new Promise((resolve, reject) => {
      const req = https.request(
        url,
        { method: "PUT", headers: { "Content-Length": new Blob([data]).size } },
        (res) => {
          let responseBody = "";
          res.on("data", (chunk) => {
            responseBody += chunk;
          });
          res.on("end", () => {
            const parser = new XMLParser();
            if (res.statusCode >= 200 && res.statusCode <= 299) {
              resolve(parser.parse(responseBody, true));
            } else {
              reject(parser.parse(responseBody, true));
            }
          });
        },
      );
      req.on("error", (err) => {
        reject(err);
      });
      req.write(data);
      req.end();
    });
  };

  /**
   * Create two presigned urls for uploading an object to an S3 bucket.
   * The first presigned URL is created with credentials from the shared INI file
   * in the current environment. The second presigned URL is created using an
   * existing S3Client instance that has already been provided with credentials.
   * @param {{ bucketName: string, key: string, region: string }}
   */
```

```
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      key,
      region,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });

    // After you get the presigned URL, you can provide your own file
    // data. Refer to put() above.
    console.log("Calling PUT using presigned URL without client");
    await put(noClientUrl, "Hello World");

    console.log("Calling PUT using presigned URL with client");
    await put(clientUrl, "Hello World");

    console.log("\nDone. Check your S3 console.");
  } catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
      console.error(
        `There was an error getting your credentials. Are your local credentials
        configured?\n${caught.name}: ${caught.message}`,
      );
    } else {
      throw caught;
    }
  }
};
```

Créez une URL présignée pour télécharger un objet à partir d'un compartiment.

```
import { GetObjectCommand, S3Client } from "@aws-sdk/client-s3";
import { fromIni } from "@aws-sdk/credential-providers";
import { HttpRequest } from "@smithy/protocol-http";
import {
  getSignedUrl,
```

```

    S3RequestPresigner,
  } from "@aws-sdk/s3-request-presigner";
import { parseUrl } from "@smithy/url-parser";
import { formatUrl } from "@aws-sdk/util-format-url";
import { Hash } from "@smithy/hash-node";

const createPresignedUrlWithoutClient = async ({ region, bucket, key }) => {
  const url = parseUrl(`https://${bucket}.s3.${region}.amazonaws.com/${key}`);
  const presigner = new S3RequestPresigner({
    credentials: fromIni(),
    region,
    sha256: Hash.bind(null, "sha256"),
  });

  const signedUrlObject = await presigner.presign(new HttpRequest(url));
  return formatUrl(signedUrlObject);
};

const createPresignedUrlWithClient = ({ region, bucket, key }) => {
  const client = new S3Client({ region });
  const command = new GetObjectCommand({ Bucket: bucket, Key: key });
  return getSignedUrl(client, command, { expiresIn: 3600 });
};

/**
 * Create two presigned urls for downloading an object from an S3 bucket.
 * The first presigned URL is created with credentials from the shared INI file
 * in the current environment. The second presigned URL is created using an
 * existing S3Client instance that has already been provided with credentials.
 * @param {{ bucketName: string, key: string, region: string }}
 */
export const main = async ({ bucketName, key, region }) => {
  try {
    const noClientUrl = await createPresignedUrlWithoutClient({
      bucket: bucketName,
      region,
      key,
    });

    const clientUrl = await createPresignedUrlWithClient({
      bucket: bucketName,
      region,
      key,
    });
  }
};

```

```
console.log("Presigned URL without client");
console.log(noClientUrl);
console.log("\n");

console.log("Presigned URL with client");
console.log(clientUrl);
} catch (caught) {
    if (caught instanceof Error && caught.name === "CredentialsProviderError") {
        console.error(
            `There was an error getting your credentials. Are your local credentials
            configured?\n${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK pour JavaScript](#).

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez une demande présignée `GetObject` et utilisez l'URL pour télécharger un objet.

```
suspend fun getObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): String {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
```



```
        bucket = bucketName
        key = keyName
    }

    // Presign the GetObject request.
    val presignedRequest = s3.presignGetObject(unsignedRequest, 24.hours)

    // Use the URL from the presigned HttpRequest in a subsequent HTTP GET
    request to retrieve the object.
    val objectContents = URL(presignedRequest.url.toString()).readText()

    return objectContents
}
```

Créez une demande présignée `GetObject` avec des options avancées.

```
suspend fun getObjectPresignedMoreOptions(
    s3: S3Client,
    bucketName: String,
    keyName: String,
): HttpRequest {
    // Create a GetObjectRequest.
    val unsignedRequest =
        GetObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the GetObject request.
    val presignedRequest =
        s3.presignGetObject(unsignedRequest, signer = CrtAwsSigner) {
            signingDate = Instant.now() + 12.hours // Presigned request can be
            used 12 hours from now.
            algorithm = AwsSigningAlgorithm.SIGV4_ASYMMETRIC
            signatureType = AwsSignatureType.HTTP_REQUEST_VIA_QUERY_PARAMS
            expiresAfter = 8.hours // Presigned request expires 8 hours later.
        }
    return presignedRequest
}
```

Créez une demande présignée `PutObject` et utilisez-la pour charger un objet.

```
suspend fun putObjectPresigned(
    s3: S3Client,
    bucketName: String,
    keyName: String,
    content: String,
) {
    // Create a PutObjectRequest.
    val unsignedRequest =
        PutObjectRequest {
            bucket = bucketName
            key = keyName
        }

    // Presign the request.
    val presignedRequest = s3.presignPutObject(unsignedRequest, 24.hours)

    // Use the URL and any headers from the presigned HttpRequest in a subsequent
    // HTTP PUT request to retrieve the object.
    // Create a PUT request using the OKHttpClient API.
    val putRequest =
        Request
            .Builder()
            .url(presignedRequest.url.toString())
            .apply {
                presignedRequest.headers.forEach { key, values ->
                    header(key, values.joinToString(", "))
                }
            }
            .put(content.toRequestBody())
            .build()

    val response = OkHttpClient().newCall(putRequest).execute()
    assert(response.isSuccessful)
}
```

- Pour en savoir plus, consultez [Guide du développeur d'AWS SDK pour Kotlin](#).

PHP

Kit SDK pour PHP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
namespace S3;
use Aws\Exception\AwsException;
use AwsUtilities\PrintableLineBreak;
use AwsUtilities\TestableReadline;
use DateTime;

require 'vendor/autoload.php';

class PresignedURL
{
    use PrintableLineBreak;
    use TestableReadline;

    public function run()
    {
        $s3Service = new S3Service();

        $expiration = new DateTime("+20 minutes");
        $linebreak = $this->getLineBreak();

        echo $linebreak;
        echo ("Welcome to the Amazon S3 presigned URL demo.\n");
        echo $linebreak;

        $bucket = $this->testable_readline("First, please enter the name of the
S3 bucket to use: ");
        $key = $this->testable_readline("Next, provide the key of an object in
the given bucket: ");
        echo $linebreak;
        $command = $s3Service->getClient()->getCommand('GetObject', [
            'Bucket' => $bucket,
            'Key' => $key,
```

```

    });
    try {
        $preSignedUrl = $s3Service->preSignedUrl($command, $expiration);
        echo "Your preSignedUrl is \n$preSignedUrl\nand will be good for the
next 20 minutes.\n";
        echo $linebreak;
        echo "Thanks for trying the Amazon S3 presigned URL demo.\n";
    } catch (AwsException $exception) {
        echo $linebreak;
        echo "Something went wrong: $exception";
        die();
    }
}

$runner = new PresignedURL();
$runner->run();

namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)
    {
        if ($client) {
            $this->client = $client;
        } else {
            $this->client = new S3Client([
                'version' => 'latest',
                'region' => 'us-west-2',
            ]);
        }
    }

```

```
    }
    $this->verbose = $verbose;
}

public function setVerbose($verbose)
{
    $this->verbose = $verbose;
}

public function isVerbose(): bool
{
    return $this->verbose;
}

public function getClient(): S3Client
{
    return $this->client;
}

public function setClient(S3Client $client)
{
    $this->client = $client;
}

public function emptyAndDeleteBucket($bucketName, array $args = [])
{
    try {
        $objects = $this->listAllObjects($bucketName, $args);
        $this->deleteObjects($bucketName, $objects, $args);
        if ($this->verbose) {
            echo "Deleted all objects and folders from $bucketName.\n";
        }
        $this->deleteBucket($bucketName, $args);
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "\nPlease fix error with bucket deletion before continuing.
\n";
        }
        throw $exception;
    }
}
```

```
public function createBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->createBucket($parameters);
        if ($this->verbose) {
            echo "Created the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket creation before continuing.";
        }
        throw $exception;
    }
}

public function putObject(string $bucketName, string $key, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
    $args);
    try {
        $this->client->putObject($parameters);
        if ($this->verbose) {
            echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object uploading before continuing.";
        }
        throw $exception;
    }
}
```

```
public function getObject(string $bucketName, string $key, array $args = []):
Result
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $object = $this->client->getObject($parameters);
        if ($this->verbose) {
            echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object downloading before
continuing.";
        }
        throw $exception;
    }
    return $object;
}

public function copyObject($bucketName, $key, $copySource, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
    try {
        $this->client->copyObject($parameters);
        if ($this->verbose) {
            echo "Copied the object from: $copySource in $bucketName to:
$key.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object copying before continuing.";
        }
        throw $exception;
    }
}
```

```
}

    public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
        try {
            $objects = $this->client->listObjectsV2($parameters);
            if ($this->verbose) {
                echo "Retrieved the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
                echo "Please fix error with list objects before continuing.";
            }
            throw $exception;
        }
        return $objects;
    }

    public function listAllObjects($bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);

        $contents = [];
        $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

        foreach ($paginator as $result) {
            if($result['KeyCount'] == 0){
                break;
            }
            foreach ($result['Contents'] as $object) {
                $contents[] = $object;
            }
        }
        return $contents;
    }
}
```



```

    public function deleteObjects(string $bucketName, array $objects, array $args
= [])
    {
        $listOfObjects = array_map(
            function ($object) {
                return ['Key' => $object];
            },
            array_column($objects, 'Key')
        );
        if(!$listOfObjects){
            return;
        }

        $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
        try {
            $this->client->deleteObjects($parameters);
            if ($this->verbose) {
                echo "Deleted the list of objects from: $bucketName.\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
                echo "Please fix error with object deletion before continuing.";
            }
            throw $exception;
        }
    }

    public function deleteBucket(string $bucketName, array $args = [])
    {
        $parameters = array_merge(['Bucket' => $bucketName], $args);
        try {
            $this->client->deleteBucket($parameters);
            if ($this->verbose) {
                echo "Deleted the bucket named: $bucketName.\n";
            }
        } catch (AwsException $exception) {

```

```
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket deletion before continuing.";
        }
        throw $exception;
    }
}

public function deleteObject(string $bucketName, string $fileName, array
$args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
    try {
        $this->client->deleteObject($parameters);
        if ($this->verbose) {
            echo "Deleted the object named: $fileName from $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}

public function listBuckets(array $args = [])
{
    try {
        $buckets = $this->client->listBuckets($args);
        if ($this->verbose) {
            echo "Retrieved all " . count($buckets) . "\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
        }
    }
}
```

```

        echo "Please fix error with bucket lists before continuing.";
    }
    throw $exception;
}
return $buckets;
}

public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
{
    $request = $this->client->createPresignedRequest($command, $expires,
$options);
    try {
        $presignedUrl = (string)$request->getUri();
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
            echo "Please fix error with presigned urls before continuing.";
        }
        throw $exception;
    }
    return $presignedUrl;
}

public function createSession(string $bucketName)
{
    try{
        $result = $this->client->createSession([
            'Bucket' => $bucketName,
        ]);
        return $result;
    }catch(S3Exception $caught){
        if($caught->getAwsErrorType() == "NoSuchBucket"){
            echo "The specified bucket does not exist.";
        }
        throw $caught;
    }
}

```

```
}
```

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Générez une URL présignée qui peut exécuter une action S3 pour une durée limitée. Utilisez le package Requests (Requêtes) pour effectuer une requête avec l'URL.

```
import argparse
import logging
import boto3
from botocore.exceptions import ClientError
import requests

logger = logging.getLogger(__name__)

def generate_presigned_url(s3_client, client_method, method_parameters,
    expires_in):
    """
    Generate a presigned Amazon S3 URL that can be used to perform an action.

    :param s3_client: A Boto3 Amazon S3 client.
    :param client_method: The name of the client method that the URL performs.
    :param method_parameters: The parameters of the specified client method.
    :param expires_in: The number of seconds the presigned URL is valid for.
    :return: The presigned URL.
    """
    try:
        url = s3_client.generate_presigned_url(
            ClientMethod=client_method, Params=method_parameters,
            ExpiresIn=expires_in
```

```
    )
    logger.info("Got presigned URL: %s", url)
except ClientError:
    logger.exception(
        "Couldn't get a presigned URL for client method '%s'.", client_method
    )
    raise
return url

def usage_demo():
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon S3 presigned URL demo.")
    print("-" * 88)

    parser = argparse.ArgumentParser()
    parser.add_argument("bucket", help="The name of the bucket.")
    parser.add_argument(
        "key",
        help="For a GET operation, the key of the object in Amazon S3. For a "
        "PUT operation, the name of a file to upload.",
    )
    parser.add_argument("action", choices=("get", "put"), help="The action to perform.")
    args = parser.parse_args()

    s3_client = boto3.client("s3")
    client_action = "get_object" if args.action == "get" else "put_object"
    url = generate_presigned_url(
        s3_client, client_action, {"Bucket": args.bucket, "Key": args.key}, 1000
    )

    print("Using the Requests package to send a request to the URL.")
    response = None
    if args.action == "get":
        response = requests.get(url)
        if response.status_code == 200:
            with open(args.key.split("/")[-1], 'wb') as object_file:
                object_file.write(response.content)
    elif args.action == "put":
        print("Putting data to the URL.")
        try:
```

```

        with open(args.key, "rb") as object_file:
            object_text = object_file.read()
            response = requests.put(url, data=object_text)
    except FileNotFoundError:
        print(
            f"Couldn't find {args.key}. For a PUT operation, the key must be
the "
            f"name of a file that exists on your computer."
        )

    if response is not None:
        print(f"Status: {response.status_code}\nReason: {response.reason}")

    print("-" * 88)

if __name__ == "__main__":
    usage_demo()

```

Générez une requête POST présignée pour charger un fichier.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def generate_presigned_post(self, object_key, expires_in):
        """
        Generate a presigned Amazon S3 POST request to upload a file.
        A presigned POST can be used for a limited time to let someone without an
AWS
        account upload a file to a bucket.

        :param object_key: The object key to identify the uploaded object.

```

```

        :param expires_in: The number of seconds the presigned POST is valid.
        :return: A dictionary that contains the URL and form fields that contain
                   required access data.
        """
        try:
            response = self.bucket.meta.client.generate_presigned_post(
                Bucket=self.bucket.name, Key=object_key, ExpiresIn=expires_in
            )
            logger.info("Got presigned POST URL: %s", response["url"])
        except ClientError:
            logger.exception(
                "Couldn't get a presigned POST URL for bucket '%s' and object
                '%s'",
                self.bucket.name,
                object_key,
            )
            raise
        return response

```

Ruby

Kit SDK pour Ruby

Note

Il y en a plus à ce sujet [GitHub](#). Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

require 'aws-sdk-s3'
require 'net/http'

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)

```

```
url = bucket.object(object_key).presigned_url(:put)
puts "Created presigned URL: #{url}"
URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
  why: #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
    http.send_request('PUT', presigned_url.request_uri, object_content,
    'content_type' => '')
  end

  case response
  when Net::HTTPSuccess
    puts 'Content uploaded!'
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```


Rust

SDK pour Rust

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez des demandes de présignature pour les objets S3 GET.

```
/// Generate a URL for a presigned GET request.
async fn get_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
) -> Result<(), Box<dyn Error>> {
    let expires_in = Duration::from_secs(expires_in);
    let presigned_request = client
        .get_object()
        .bucket(bucket)
        .key(object)
        .presigned(PresigningConfig::expires_in(expires_in?))
        .await?;

    println!("Object URI: {}", presigned_request.uri());
    let valid_until = chrono::offset::Local::now() + expires_in;
    println!("Valid until: {valid_until}");

    Ok(())
}
```

Créez des demandes de présignature pour les objets S3 PUT.

```
async fn put_object(
    client: &Client,
    bucket: &str,
    object: &str,
    expires_in: u64,
```

```

) -> Result<String, S3ExampleError> {
    let expires_in: std::time::Duration =
std::time::Duration::from_secs(expires_in);
    let expires_in: aws_sdk_s3::presigning::PresigningConfig =
    PresigningConfig::expires_in(expires_in).map_err(|err| {
        S3ExampleError::new(format!(
            "Failed to convert expiration to PresigningConfig: {err:?}")
        ))
    })?;
    let presigned_request = client
        .put_object()
        .bucket(bucket)
        .key(object)
        .presigned(expires_in)
        .await?;

    Ok(presigned_request.uri().into())
}

```

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez des demandes de présignature pour les objets S3 GET.

```

" iv_bucket_name is the bucket name
" iv_key is the object name like "myfile.txt"

DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

"Upload a nice Hello World file to an S3 bucket."
TRY.
    DATA(lv_contents) = cl_abap_codepage=>convert_to( 'Hello, World' ).
    lo_s3->putobject(

```

```

        iv_bucket = iv_bucket_name
        iv_key = iv_key
        iv_body = lv_contents
        iv_contenttype = 'text/plain' ).
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" now generate a presigned URL with a 600-second expiration
DATA(lo_presigner) = lo_s3->get_presigner( iv_expires_sec = 600 ).
" the presigner getobject() method has the same signature as
" lo_s3->getobject(), but it doesn't actually make the call.
" to the service. It just prepares a presigned URL for a future call
DATA(lo_presigned_req) = lo_presigner->getobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key ).

" You can provide this URL to a web page, user, email etc so they
" can retrieve the file. The URL will expire in 10 minutes.
ov_url = lo_presigned_req->get_url( ).

```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Création d'une application de gestion des ressources photographiques permettant aux utilisateurs de gérer les photos à l'aide d'étiquettes

Les exemples de code suivants montrent comment créer une application sans serveur permettant aux utilisateurs de gérer des photos à l'aide d'étiquettes.

.NET

SDK pour .NET

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur [AWS Community](#).

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK pour C++

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur [AWS Community](#).

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK pour Java 2.x

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur [AWS Community](#).

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK pour JavaScript (v3)

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur [AWS Community](#).

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK pour Kotlin

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur [AWS Community](#).

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

Kit SDK pour PHP

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur [AWS Community](#).

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK pour Rust

Montre comment développer une application de gestion de ressources photographiques qui détecte les étiquettes dans les images à l'aide d'Amazon Rekognition et les stocke pour les récupérer ultérieurement.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Pour explorer en profondeur l'origine de cet exemple, consultez l'article sur [AWS Community](#).

Les services utilisés dans cet exemple

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Une page Web répertoriant les objets Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment répertorier des objets Amazon S3 dans une page web.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Le code suivant est le composant React pertinent qui appelle le AWS SDK. Une version exécutable de l'application contenant ce composant se trouve sur le lien précédent GitHub .

```
import { useEffect, useState } from "react";
import {
  ListObjectsCommand,
  type ListObjectsCommandOutput,
  S3Client,
} from "@aws-sdk/client-s3";
import { fromCognitoIdentityPool } from "@aws-sdk/credential-providers";
import "./App.css";

function App() {
  const [objects, setObjects] = useState<
    Required<ListObjectsCommandOutput>["Contents"]
  >([]);

  useEffect(() => {
    const client = new S3Client({
      region: "us-east-1",
      // Unless you have a public bucket, you'll need access to a private bucket.
      // One way to do this is to create an Amazon Cognito identity pool, attach
      // a role to the pool,
      // and grant the role access to the 's3:GetObject' action.
      //
      // You'll also need to configure the CORS settings on the bucket to allow
      // traffic from
      // this example site. Here's an example configuration that allows all
      // origins. Don't
      // do this in production.
      // [
      // {

```



```

    //    "AllowedHeaders": ["*"],
    //    "AllowedMethods": ["GET"],
    //    "AllowedOrigins": ["*"],
    //    "ExposeHeaders": [],
    //  },
  //]
  //
  credentials: fromCognitoIdentityPool({
    clientConfig: { region: "us-east-1" },
    identityPoolId: "<YOUR_IDENTITY_POOL_ID>",
  }),
});
const command = new ListObjectsCommand({ Bucket: "bucket-name" });
client.send(command).then(({ Contents }) => setObjects(Contents || []));
}, []);

return (
  <div className="App">
    {objects.map((o) => (
      <div key={o.ETag}>{o.Key}</div>
    ))}
  </div>
);
}

export default App;

```

- Pour plus de détails sur l'API, reportez-vous [ListObjects](#) à la section Référence des AWS SDK pour JavaScript API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Créer une application Amazon Textract Explorer

Les exemples de code suivants expliquent comment explorer la sortie Amazon Textract via une application interactive.

JavaScript

SDK pour JavaScript (v3)

Montre comment utiliser le AWS SDK pour JavaScript pour créer une application React qui utilise Amazon Textract pour extraire des données d'une image de document et les afficher sur une page Web interactive. Cet exemple s'exécute dans un navigateur Web et nécessite une identité Amazon Cognito authentifiée pour les informations d'identification. Il utilise Amazon Simple Storage Service (Amazon S3) pour le stockage et, pour les notifications, il interroge une file d'attente Amazon Simple Queue Service (Amazon SQS) abonnée à une rubrique Amazon Simple Notification Service (Amazon SNS).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

Kit SDK for Python (Boto3)

Montre comment utiliser Amazon Textract pour détecter des éléments de texte, de formulaire et de tableau dans une image de document. AWS SDK pour Python (Boto3) L'image d'entrée et la sortie d'Amazon Textract sont affichées dans une application Tkinter qui vous permet d'explorer les éléments détectés.

- Soumettez une image de document à Amazon Textract et explorez la sortie des éléments détectés.
- Soumettez des images directement à Amazon Textract ou via un compartiment Amazon Simple Storage Service (Amazon S3).
- Utilisez des API asynchrones pour démarrer une tâche qui publie une notification dans une rubrique Amazon Simple Notification Service (Amazon SNS) lorsque le travail est terminé.

- Interrogez un service Amazon Simple Queue Service (Amazon SQS) pour obtenir un message de fin de tâche et affichez les résultats.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Supprimez tous les objets d'un compartiment Amazon S3 donné à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment supprimer tous les objets d'un compartiment Amazon S3.

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez tous les objets d'un compartiment Amazon S3 donné.

```
import {
  DeleteObjectsCommand,
  paginateListObjectsV2,
```

```
S3Client,
} from "@aws-sdk/client-s3";

/**
 * @param {{ bucketName: string }} config
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});
  try {
    console.log(`Deleting all objects in bucket: ${bucketName}`);

    const paginator = paginateListObjectsV2(
      { client },
      {
        Bucket: bucketName,
      },
    );

    const objectKeys = [];
    for await (const { Contents } of paginator) {
      objectKeys.push(...Contents.map((obj) => ({ Key: obj.Key })));
    }

    const deleteCommand = new DeleteObjectsCommand({
      Bucket: bucketName,
      Delete: { Objects: objectKeys },
    });

    await client.send(deleteCommand);

    console.log(`All objects deleted from bucket: ${bucketName}`);
  } catch (caught) {
    if (caught instanceof Error) {
      console.error(
        `Failed to empty ${bucketName}. ${caught.name}: ${caught.message}`,
      );
    }
  }
};

// Call function if run directly.
import { fileURLToPath } from "node:url";
import { parseArgs } from "node:util";
```

```
if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const options = {
    bucketName: {
      type: "string",
    },
  };

  const { values } = parseArgs({ options });
  main(values);
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour JavaScript .
 - [DeleteObjects](#)
 - [ListObjectsV2](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Supprimez les chargements partitionnés incomplets vers Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment supprimer ou arrêter les chargements partitionnés Amazon S3.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Pour arrêter les chargements partitionnés en cours ou incomplets pour une quelconque raison, vous pouvez obtenir une liste des téléchargements, puis les supprimer comme indiqué dans l'exemple suivant.

```

/**
 * Aborts all incomplete multipart uploads from the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket
 */
public static void abortIncompleteMultipartUploadsFromList(String bucketName)
{
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(upload.key())
            .expectedBucketOwner(accountId)
            .uploadId(upload.uploadId())
            .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}

```

Pour supprimer des chargements partitionnés incomplets initiés avant ou après une date, vous pouvez supprimer les chargements partitionnés de manière sélective en fonction d'un moment donné, comme indiqué dans l'exemple suivant.

```

/**
 * Aborts incomplete multipart uploads older than the specified point in
time.

```

```

    *
    * @param bucketName the name of the S3 bucket
    * @param pointInTime the cutoff time; uploads initiated before this are
    aborted
    */
    static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
    Instant pointInTime) {
        ListMultipartUploadsRequest listMultipartUploadsRequest =
    ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

        ListMultipartUploadsResponse response =
    s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();

        AbortMultipartUploadRequest abortMultipartUploadRequest;
        for (MultipartUpload upload : uploads) {
            logger.info("Found multipartUpload with upload ID [{}], initiated
    [{}]", upload.uploadId(), upload.initiated());
            if (upload.initiated().isBefore(pointInTime)) {
                abortMultipartUploadRequest =
    AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

                AbortMultipartUploadResponse abortMultipartUploadResponse =
    s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if
    (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
    aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }
}

```

Si vous avez accès à l'ID de chargement après avoir commencé un chargement partitionné, vous pouvez supprimer le chargement en cours à l'aide de cet ID.

```

/**
 * Aborts a multipart upload using the upload ID.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 */
static void abortMultipartUploadUsingUploadId(String bucketName, String key)
{
    String uploadId = startUploadReturningUploadId(bucketName, key);
    AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
    }
}

```

Pour supprimer systématiquement les chargements partitionnés incomplets datant de plus d'un certain nombre de jours, configurez une configuration du cycle de vie du compartiment pour le compartiment. L'exemple suivant montre comment créer une règle pour supprimer les chargements incomplets datant de plus de 7 jours.

```

/**
 * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
 *
 * @param bucketName the name of the S3 bucket
 */
static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
    Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());
}

```



```
PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
    .bucket(bucketName)
    .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
    } else {
        logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
    }
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [PutBucketLifecycleConfiguration](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Détectez le PPE dans les images avec Amazon Rekognition à l'aide d'un AWS Kit SDK

Les exemples de code suivants montrent comment créer une application qui utilise Amazon Rekognition afin de détecter l'équipement de protection individuelle (EPI) dans les images.

Java

SDK pour Java 2.x

Montre comment créer une AWS Lambda fonction qui détecte les images à l'aide d'un équipement de protection individuelle.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Détectez les entités dans le texte extrait d'une image à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment utiliser Amazon Comprehend pour détecter des entités dans du texte extrait par Amazon Textract à partir d'une image stockée dans Amazon S3.

Python

Kit SDK for Python (Boto3)

Montre comment utiliser le AWS SDK pour Python (Boto3) dans un bloc-notes Jupyter pour détecter des entités dans du texte extrait d'une image. Cet exemple utilise Amazon Textract pour extraire le texte d'une image stockée dans Amazon Simple Storage Service (Amazon S3) et Amazon Comprehend pour détecter les entités dans le texte extrait.

Cet exemple est un carnet Jupyter et doit être exécuté dans un environnement qui peut accueillir des carnets. Pour savoir comment exécuter cet exemple à l'aide d'Amazon SageMaker AI, consultez les instructions fournies dans [TextextractAndComprehendNotebook.ipynb](#).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Détecter les visages dans une image à l'aide d'un AWS Kit SDK

L'exemple de code suivant illustre comment :

- Enregistrez une image dans un compartiment Amazon S3.
- Utilisez Amazon Rekognition pour détecter les détails du visage, tels que la tranche d'âge, le sexe et l'émotion (sourire, etc.).
- Affichez ces détails.

Rust

SDK pour Rust

Enregistrez l'image dans un compartiment Amazon S3 avec un préfixe uploads (chargement), utilisez Amazon Rekognition pour détecter les détails du visage, tels que la tranche d'âge, le sexe et l'émotion (sourire, etc.) et affichez ces détails.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Détectez des objets dans des images avec Amazon Rekognition à l'aide d'un AWS Kit SDK

Les exemples de code suivants montrent comment créer une application qui utilise Amazon Rekognition afin de détecter des objets par catégorie dans des images.

.NET

SDK pour .NET

Montre comment utiliser l'API Java Amazon Rekognition afin de créer une application qui, avec Amazon Rekognition, permet d'identifier des objets par catégorie dans des images stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK pour Java 2.x

Montre comment utiliser l'API Java Amazon Rekognition afin de créer une application qui, avec Amazon Rekognition, permet d'identifier des objets par catégorie dans des images stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK pour JavaScript (v3)

Montre comment utiliser Amazon Rekognition AWS SDK pour JavaScript pour créer une application qui utilise Amazon Rekognition pour identifier les objets par catégorie dans des images situées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Découvrez comment :

- Créer un utilisateur non authentifié à l'aide d'Amazon Cognito.
- Analyser les images à la recherche d'objets à l'aide d'Amazon Rekognition.
- Vérifier une adresse e-mail pour Amazon SES.
- Envoyer une notification par e-mail à l'aide d'Amazon SES.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK pour Kotlin

Montre comment utiliser l'API Kotlin Amazon Rekognition afin de créer une application qui, avec Amazon Rekognition, permet d'identifier des objets par catégorie dans des images stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK pour Python (Boto3)

Vous montre comment utiliser le AWS SDK pour Python (Boto3) pour créer une application Web qui vous permet d'effectuer les opérations suivantes :

- Chargez les photos dans un compartiment Amazon Simple Storage Service (Amazon S3).
- Utilisez Amazon Rekognition pour analyser et étiqueter les photos.
- Utilisez Amazon Simple Email Service (Amazon SES) pour envoyer des rapports d'analyse d'images par e-mail.

Cet exemple contient deux composants principaux : une page Web écrite avec React et un service REST écrit en Python construit avec Flask-RESTful. JavaScript

Vous pouvez utiliser la page web React pour :

- Affichez une liste d'images stockées dans votre compartiment S3.
- Chargez des images depuis votre ordinateur dans votre compartiment S3.
- Affichez des images et des étiquettes qui identifient les éléments détectés dans l'image.
- Obtenez un rapport de toutes les images de votre compartiment S3 et envoyez un e-mail du rapport.

La page web appelle le service REST. Le service envoie des demandes à AWS pour effectuer les opérations suivantes :

- Obtenez et filtrez la liste des images de votre compartiment S3.
- Chargez des photos dans votre compartiment S3.
- Utilisez Amazon Rekognition pour analyser des photos individuelles et obtenir une liste d'étiquettes qui identifient les éléments détectés sur la photo.
- Analysez toutes les photos de votre compartiment S3 et utilisez Amazon SES pour envoyer un rapport par e-mail.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Détectez les personnes et les objets dans une vidéo avec Amazon Rekognition à l'aide d'un AWS Kit SDK

Les exemples de code suivants montrent comment détecter des personnes et des objets dans une vidéo avec Amazon Rekognition.

Java

SDK pour Java 2.x

Montre comment utiliser l'API Java Amazon Rekognition afin de créer une application qui détecte les visages et les objets dans des vidéos stockées dans un compartiment Amazon Simple Storage Service (Amazon S3). L'application envoie à l'administrateur une notification par e-mail contenant les résultats à l'aide d'Amazon Simple Email Service (Amazon SES).

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK pour Python (Boto3)

Utilisez Amazon Rekognition pour détecter des visages, des objets et des personnes dans des vidéos en démarrant des tâches de détection asynchrone. Cet exemple montre également comment configurer Amazon Rekognition pour notifier une rubrique Amazon Simple Notification Service (Amazon SNS) lorsque les tâches sont terminées et abonner une file d'attente Amazon Simple Queue Service (Amazon SQS) à la rubrique. Lorsque la file d'attente reçoit un message concernant une tâche, elle est récupérée et les résultats sont affichés.

Il est préférable de visionner cet exemple sur GitHub. Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Téléchargement des « répertoires » S3 à partir d'un compartiment Amazon Simple Storage Service (Amazon S3)

L'exemple de code suivant montre comment télécharger et filtrer le contenu des « répertoires » du compartiment Amazon S3.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Cet exemple montre comment utiliser le [S3TransferManager](#) in AWS SDK for Java 2.x pour télécharger des « répertoires » depuis un compartiment Amazon S3. Il montre également comment l'utiliser [DownloadFilters](#) dans la demande.

```
/**
 * For standard buckets, S3 provides the illusion of a directory structure
 * through the use of keys. When you upload
 *   * an object to an S3 bucket, you specify a key, which is essentially the
 * "path" to the object. The key can contain
 *   * forward slashes ("/") to make it appear as if the object is stored in a
 * directory structure, but this is just a
 *   * logical representation, not an actual directory.
 *   * <p><pre>
 * In this example, our S3 bucket contains the following objects:
 *
 *   * folder1/file1.txt
 *   * folder1/file2.txt
 *   * folder1/file3.txt
 *   * folder2/file1.txt
 *   * folder2/file2.txt
 *   * folder2/file3.txt
 *   * folder3/file1.txt
 *   * folder3/file2.txt
 *   * folder3/file3.txt
 *
 * When method `downloadS3Directories` is invoked with
 * `destinationPathURI` set to `/test`, the downloaded
 * directory looks like:
 *
 *   * |- test
 *     *   |- folder1
 *       *     |- file1.txt
```

```

*      |- file2.txt
*      |- file3.txt
*      |- folder3
*      |- file1.txt
*      |- file2.txt
*      |- file3.txt
* </pre>
*
* @param transferManager    An S3TransferManager instance.
* @param destinationPathURI local directory to hold the downloaded S3
'directories' and files.
* @param bucketName        The S3 bucket that contains the 'directories' to
download.
* @return The number of objects (files, in this case) that were downloaded.
*/
public Integer downloadS3Directories(S3TransferManager transferManager,
                                   URI destinationPathURI, String
bucketName) {

    // Define the filters for which 'directories' we want to download.
    DownloadFilter folder1Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder1/");
    DownloadFilter folder3Filter = (S3Object s3Object) ->
s3Object.key().startsWith("folder3/");
    DownloadFilter folderFilter = s3Object ->
folder1Filter.or(folder3Filter).test(s3Object);

    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
                                .destination(Paths.get(destinationPathURI))
                                .bucket(bucketName)
                                .filter(folderFilter)
                                .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    Integer numFilesInFolder1 =
Paths.get(destinationPathURI).resolve("folder1").toFile().list().length;
    Integer numFilesInFolder3 =
Paths.get(destinationPathURI).resolve("folder3").toFile().list().length;

    try {
        assert numFilesInFolder1 == 3;
        assert numFilesInFolder3 == 3;
    }
}

```

```
        assert !
Paths.get(destinationPathURI).resolve("folder2").toFile().exists(); // `folder2`
was not downloaded.
    } catch (AssertionError e) {
        logger.error("An assertion failed.");
    }

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object failed to transfer  [{}]",
fail.exception().getMessage()));
    return numFilesInFolder1 + numFilesInFolder3;
}
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Téléchargez tous les objets d'un compartiment Amazon Simple Storage Service (Amazon S3) dans un répertoire local.

L'exemple de code suivant montre comment télécharger tous les objets d'un compartiment Amazon Simple Storage Service (Amazon S3) dans un répertoire local.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez un [S3TransferManager](#) pour [télécharger tous les objets S3](#) dans le même compartiment S3. Consultez le [fichier complet](#) et le [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
```

```
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DirectoryDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadDirectoryRequest;
import java.io.IOException;
import java.net.URI;
import java.net.URISyntaxException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.HashSet;
import java.util.Set;
import java.util.UUID;
import java.util.stream.Collectors;

    public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
        URI destinationPathURI, String bucketName) {
        DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
        .destination(Paths.get(destinationPathURI))
        .bucket(bucketName)
        .build());
        CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

        completedDirectoryDownload.failedTransfers()
            .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryDownload.failedTransfers().size();
    }
```

- Pour plus de détails sur l'API, reportez-vous [DownloadDirectory](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Téléchargez un flux de taille inconnue à partir d'un objet Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment télécharger un flux de taille inconnue à partir d'un objet Amazon S3.

Swift

Kit SDK pour Swift

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams

/// Download a file from the specified bucket.
///
/// - Parameters:
///   - bucket: The Amazon S3 bucket name to get the file from.
///   - key: The name (or path) of the file to download from the bucket.
///   - destPath: The pathname on the local filesystem at which to store
///     the downloaded file.
func downloadFile(bucket: String, key: String, destPath: String?) async
throws {
    let fileURL: URL

    // If no destination path was provided, use the key as the name to use
    // for the file in the downloads folder.

    if destPath == nil {
        do {
            try fileURL = FileManager.default.url(
```

```
        for: .downloadsDirectory,
        in: .userDomainMask,
        appropriateFor: URL(string: key),
        create: true
    ).appendingPathComponent(key)
} catch {
    throw TransferError.directoryError
}
} else {
    fileURL = URL(fileURLWithPath: destPath!)
}

let config = try await S3Client.S3ClientConfiguration(region: region)
let s3Client = S3Client(config: config)

// Create a `FileHandle` referencing the local destination. Then
// create a `ByteStream` from that.

FileManager.default.createFile(atPath: fileURL.path, contents: nil,
attributes: nil)
let fileHandle = try FileHandle(forWritingTo: fileURL)

// Download the file using `GetObject`.

let getInput = GetObjectInput(
    bucket: bucket,
    key: key
)

do {
    let getOutput = try await s3Client.getObject(input: getInput)

    guard let body = getOutput.body else {
        throw TransferError.downloadError("Error: No data returned for
download")
    }

    // If the body is returned as a `Data` object, write that to the
    // file. If it's a stream, read the stream chunk by chunk,
    // appending each chunk to the destination file.

    switch body {
    case .data:
        guard let data = try await body.readData() else {
```

```
        throw TransferError.downloadError("Download error")
    }

    // Write the `Data` to the file.

    do {
        try data.write(to: fileURL)
    } catch {
        throw TransferError.writeError
    }
    break

case .stream(let stream as ReadableStream):
    while (true) {
        let chunk = try await stream.readAsync(upToCount: 5 * 1024 *
1024)

        guard let chunk = chunk else {
            break
        }

        // Write the chunk to the destination file.

        do {
            try fileHandle.write(contentsOf: chunk)
        } catch {
            throw TransferError.writeError
        }
    }

    break
default:
    throw TransferError.downloadError("Received data is unknown
object type")
    }
    } catch {
        throw TransferError.downloadError("Error downloading the file:
\\(error)")
    }

    print("File downloaded to \\(fileURL.path).")
}
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Obtenez un objet Amazon S3 à partir d'un point d' Multi-Region accès à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment obtenir un objet depuis un point Multi-Region d'accès.

Kotlin

SDK pour Kotlin

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Configurez le client S3 pour utiliser l'algorithme de signature asymétrique Sigv4 (Sigv4a).

```
suspend fun createS3Client(): S3Client {  
    // Configure your S3Client to use the Asymmetric SigV4 (SigV4a)  
    signing algorithm.  
    val sigV4aScheme = SigV4AsymmetricAuthScheme(DefaultAwsSigner)  
    val s3 = S3Client.fromEnvironment {  
        authSchemes = listOf(sigV4aScheme)  
    }  
    return s3  
}
```

Utilisez l'ARN du point d' Multi-Region accès au lieu d'un nom de compartiment pour récupérer l'objet.

```
suspend fun getObjectFromMrap(  
    s3: S3Client,  
    mrpArn: String,  
    keyName: String,  
): String? {
```



```
val request = GetObjectRequest {  
    bucket = mrapArn // Use the ARN instead of the bucket name for object  
    operations.  
    key = keyName  
}  
  
var stringObj: String? = null  
s3.getObject(request) { resp ->  
    stringObj = resp.body?.decodeToString()  
    if (stringObj != null) {  
        println("Successfully read $keyName from $mrapArn")  
    }  
}  
return stringObj  
}
```

- Pour en savoir plus, consultez [Guide du développeur d'AWS SDK pour Kotlin](#).
- Pour plus de détails sur l'API, consultez [GetObject](#) la section AWS SDK pour la référence de l'API Kotlin.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Obtenez un objet depuis un compartiment Amazon S3 à l'aide d'un AWS SDK, spécification d'un en-tête If-Modified-Since

L'exemple de code suivant montre comment lire les données d'un objet dans un compartiment S3, mais uniquement si ce compartiment n'a pas été modifié depuis la dernière extraction.

Rust

SDK pour Rust

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
use aws_sdk_s3::{
    error::SdkError,
    primitives::{ByteStream, DateTime, DateTimeFormat},
    Client,
};
use s3_code_examples::error::S3ExampleError;
use tracing::{error, warn};

const KEY: &str = "key";
const BODY: &str = "Hello, world!";

/// Demonstrate how `if-modified-since` reports that matching objects haven't
/// changed.
///
/// # Steps
/// - Create a bucket.
/// - Put an object in the bucket.
/// - Get the bucket headers.
/// - Get the bucket headers again but only if modified.
/// - Delete the bucket.
#[tokio::main]
async fn main() -> Result<(), S3ExampleError> {
    tracing_subscriber::fmt::init();

    // Get a new UUID to use when creating a unique bucket name.
    let uuid = uuid::Uuid::new_v4();

    // Load the AWS configuration from the environment.
    let client = Client::new(&aws_config::load_from_env().await);

    // Generate a unique bucket name using the previously generated UUID.
    // Then create a new bucket with that name.
    let bucket_name = format!("if-modified-since-{{uuid}}");
    client
        .create_bucket()
        .bucket(bucket_name.clone())
        .send()
        .await?;

    // Create a new object in the bucket whose name is `KEY` and whose
    // contents are `BODY`.
    let put_object_output = client
        .put_object()
```

```
.bucket(bucket_name.as_str())
.key(KEY)
.body(ByteStream::from_static(BODY.as_bytes()))
.send()
.await;

// If the `PutObject` succeeded, get the eTag string from it. Otherwise,
// report an error and return an empty string.
let e_tag_1 = match put_object_output {
    Ok(put_object) => put_object.e_tag.unwrap(),
    Err(err) => {
        error!("{err:?}");
        String::new()
    }
};

// Request the object's headers.
let head_object_output = client
    .head_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .send()
    .await;

// If the `HeadObject` request succeeded, create a tuple containing the
// values of the headers `last-modified` and `etag`. If the request
// failed, return the error in a tuple instead.
let (last_modified, e_tag_2) = match head_object_output {
    Ok(head_object) => (
        Ok(head_object.last_modified().cloned().unwrap()),
        head_object.e_tag.unwrap(),
    ),
    Err(err) => (Err(err), String::new()),
};

warn!("last modified: {last_modified:?}");
assert_eq!(
    e_tag_1, e_tag_2,
    "PutObject and first GetObject had differing eTags"
);

println!("First value of last_modified: {last_modified:?}");
println!("First tag: {}\n", e_tag_1);
```

```
// Send a second `HeadObject` request. This time, the `if_modified_since`
// option is specified, giving the `last_modified` value returned by the
// first call to `HeadObject`.
//
// Since the object hasn't been changed, and there are no other objects in
// the bucket, there should be no matching objects.

let head_object_output = client
    .head_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .if_modified_since(last_modified.unwrap())
    .send()
    .await;

// If the `HeadObject` request succeeded, the result is a tuple containing
// the `last_modified` and `e_tag_1` properties. This is _not_ the expected
// result.
//
// The _expected_ result of the second call to `HeadObject` is an
// `SdkError::ServiceError` containing the HTTP error response. If that's
// the case and the HTTP status is 304 (not modified), the output is a
// tuple containing the values of the HTTP `last-modified` and `etag`
// headers.
//
// If any other HTTP error occurred, the error is returned as an
// `SdkError::ServiceError`.

let (last_modified, e_tag_2) = match head_object_output {
    Ok(head_object) => (
        Ok(head_object.last_modified().cloned().unwrap()),
        head_object.e_tag.unwrap(),
    ),
    Err(err) => match err {
        SdkError::ServiceError(err) => {
            // Get the raw HTTP response. If its status is 304, the
            // object has not changed. This is the expected code path.
            let http = err.raw();
            match http.status().as_u16() {
                // If the HTTP status is 304: Not Modified, return a
                // tuple containing the values of the HTTP
                // `last-modified` and `etag` headers.
                304 => (
                    Ok(DateTime::from_str(
```

```

        http.headers().get("last-modified").unwrap(),
        DateTimeFormat::HttpDate,
    )
    .unwrap()),
    http.headers().get("etag").map(|t| t.into()).unwrap(),
),
// Any other HTTP status code is returned as an
// `SdkError::ServiceError`.
_ => (Err(SdkError::ServiceError(err)), String::new()),
}
}
// Any other kind of error is returned in a tuple containing the
// error and an empty string.
_ => (Err(err), String::new()),
},
};

warn!("last modified: {last_modified:?}");
assert_eq!(
    e_tag_1, e_tag_2,
    "PutObject and second HeadObject had different eTags"
);

println!("Second value of last modified: {last_modified:?}");
println!("Second tag: {}", e_tag_2);

// Clean up by deleting the object and the bucket.
client
    .delete_object()
    .bucket(bucket_name.as_str())
    .key(KEY)
    .send()
    .await?;

client
    .delete_bucket()
    .bucket(bucket_name.as_str())
    .send()
    .await?;

Ok(())
}

```

- Pour plus de détails sur l'API, voir [GetObject](#) la section de référence de l'API AWS SDK for Rust.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Commencez à chiffrer les objets Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment démarrer avec le chiffrement des objets Amazon S3.

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class SSEClientEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "exampleobject.txt";
        string copyTargetKeyName = "examplecopy.txt";

        // If the AWS Region defined for your default user is different
```

```

        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USSouth1.
        IAmazonS3 client = new AmazonS3Client();

        try
        {
            // Create an encryption key.
            Aes aesEncryption = Aes.Create();
            aesEncryption.KeySize = 256;
            aesEncryption.GenerateKey();
            string base64Key = Convert.ToBase64String(aesEncryption.Key);

            // Upload the object.
            PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

            // Download the object and verify that its contents match what
you uploaded.
            await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

            // Get object metadata and verify that the object uses AES-256
encryption.
            await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

            // Copy both the source and target objects using server-side
encryption with
            // an encryption key.
            await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
    }

    /// <summary>
    /// Uploads an object to an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call

```

```

    /// PutObjectAsync.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
    /// object will be uploaded.</param>
    /// <param name="keyName">The name of the object to upload to the Amazon
S3
    /// bucket.</param>
    /// <param name="base64Key">The encryption key.</param>
    /// <returns>The PutObjectRequest object for use by
DownloadObjectAsync.</returns>
    public static async Task<PutObjectRequest> UploadObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string base64Key)
    {
        PutObjectRequest putObjectRequest = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
            ContentBody = "sample text",
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };
        PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
        return putObjectRequest;
    }

    /// <summary>
    /// Downloads an encrypted object from an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetObjectAsync.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
object
    /// is located.</param>
    /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
    /// <param name="base64Key">The encryption key used to encrypt the
    /// object.</param>
    /// <param name="putObjectRequest">The PutObjectRequest used to upload

```



```

    /// the object.</param>
    public static async Task DownloadObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string base64Key,
        PutObjectRequest putObjectRequest)
    {
        GetObjectRequest getObjectRequest = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,

            // Provide encryption information for the object stored in Amazon
S3.
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };

        using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (string.Compare(putObjectRequest.ContentBody, content) == 0)
                {
                    Console.WriteLine("Object content is same as we uploaded");
                }
                else
                {
                    Console.WriteLine("Error...Object content is not same.");
                }

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                {
                    Console.WriteLine("Object encryption method is AES256, same
as we set");
                }
                else
                {

```

```

        Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
    }
}

/// <summary>
/// Retrieves the metadata associated with an Amazon S3 object.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used
/// to call GetObjectMetadataAsync.</param>
/// <param name="bucketName">The name of the Amazon S3 bucket containing
the
/// object for which we want to retrieve metadata.</param>
/// <param name="keyName">The name of the object for which we wish to
/// retrieve the metadata.</param>
/// <param name="base64Key">The encryption key associated with the
/// object.</param>
public static async Task GetObjectMetadataAsync(
    IAmazonS3 client,
    string bucketName,
    string keyName,
    string base64Key)
{
    GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
    {
        BucketName = bucketName,
        Key = keyName,

        // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
        ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
        ServerSideEncryptionCustomerProvidedKey = base64Key,
    };

    GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
    Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
}

/// <summary>

```

```

    /// Copies an encrypted object from one Amazon S3 bucket to another.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// CopyObjectAsync.</param>
    /// <param name="bucketName">The Amazon S3 bucket containing the object
    /// to copy.</param>
    /// <param name="keyName">The name of the object to copy.</param>
    /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
    /// will be copied.</param>
    /// <param name="aesEncryption">The encryption type to use.</param>
    /// <param name="base64Key">The encryption key to use.</param>
    public static async Task CopyObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string copyTargetKeyName,
        Aes aesEncryption,
        string base64Key)
    {
        aesEncryption.GenerateKey();
        string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

        CopyObjectRequest copyRequest = new CopyObjectRequest
        {
            SourceBucket = bucketName,
            SourceKey = keyName,
            DestinationBucket = bucketName,
            DestinationKey = copyTargetKeyName,

            // Information about the source object's encryption.
            CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

            // Information about the target object's encryption.
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
        };
        await client.CopyObjectAsync(copyRequest);
    }
}

```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour .NET .
 - [CopyObject](#)
 - [GetObject](#)
 - [GetObjectMetadata](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Commencez à utiliser les balises pour les objets Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment démarrer avec des étiquettes pour les objets Amazon S3.

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
/// Service (Amazon S3) objects.
```

```

/// </summary>
public class ObjectTag
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "newobject.txt";
        string filePath = @"**** file path ****";

        // Specify your bucket region (an example region is shown).
        RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

        var client = new AmazonS3Client(bucketRegion);
        await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
    }

    /// <summary>
    /// This method uploads an object with tags. It then shows the tag
    /// values, changes the tags, and shows the new tags.
    /// </summary>
    /// <param name="client">The Initialized Amazon S3 client object used
    /// to call the methods to create and change an objects tags.</param>
    /// <param name="bucketName">A string representing the name of the
    /// bucket where the object will be stored.</param>
    /// <param name="keyName">A string representing the key name of the
    /// object to be tagged.</param>
    /// <param name="filePath">The directory location and file name of the
    /// object to be uploaded to the Amazon S3 bucket.</param>
    public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
    {
        try
        {
            // Create an object with tags.
            var putRequest = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = keyName,
                FilePath = filePath,
                TagSet = new List<Tag>
                {
                    new Tag { Key = "Keyx1", Value = "Value1" },
                    new Tag { Key = "Keyx2", Value = "Value2" },
                },
            },

```

```
};

PutObjectResponse response = await
client.PutObjectAsync(putRequest);

// Now retrieve the new object's tags.
GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
{
    BucketName = bucketName,
    Key = keyName,
};

GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

// Display the tag values.
objectTags.Tagging
    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

Tagging newTagSet = new Tagging()
{
    TagSet = new List<Tag>
    {
        new Tag { Key = "Key3", Value = "Value3" },
        new Tag { Key = "Key4", Value = "Value4" },
    },
};

PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
{
    BucketName = bucketName,
    Key = keyName,
    Tagging = newTagSet,
};

PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

// Retrieve the tags again and show the values.
GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
```

```
        {
            BucketName = bucketName,
            Key = keyName,
        };
        GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

        objectTags2.Tagging
            .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine(
            $"Error: '{ex.Message}'");
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectTagging](#) à la section Référence des AWS SDK pour .NET API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Débuter avec les clusters de traitement des mégadonnées

L'exemple de code suivant illustre comment :

- Création d'une paire de clés EC2
- Configurez le stockage et préparez votre application
- nettoyer des ressources ;

Bash

AWS CLI avec le script Bash

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code](#).

```
#!/bin/bash

# EMR Getting Started Tutorial Script
# This script automates the steps in the Amazon EMR Getting Started tutorial

set -euo pipefail

# Security: Set strict mode and trap errors
trap 'handle_error "Script interrupted or command failed"' ERR

# Set up logging with secure permissions
LOG_FILE="emr-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EMR Getting Started Tutorial Script"
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "${BUCKET_NAME:-}" ]; then echo "- S3 Bucket: $BUCKET_NAME"; fi
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- EMR Cluster: $CLUSTER_ID"; fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
```



```
cleanup() {
    echo ""
    echo "=====
    echo "CLEANUP IN PROGRESS"
    echo "=====
    echo "Starting cleanup process..."

    # Terminate EMR cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Terminating EMR cluster: $CLUSTER_ID"
        aws emr terminate-clusters --cluster-ids "$CLUSTER_ID" 2>/dev/null ||
true

        echo "Waiting for cluster to terminate..."
        aws emr wait cluster-terminated --cluster-id "$CLUSTER_ID" 2>/dev/null ||
true

        echo "Cluster terminated successfully."
    fi

    # Delete S3 bucket and contents if it exists and is not shared
    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" != "true" ];
then
        echo "Deleting S3 bucket contents: $BUCKET_NAME"
        aws s3 rm "s3://$BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket: $BUCKET_NAME"
        aws s3 rb "s3://$BUCKET_NAME" 2>/dev/null || true
    fi

    # Remove temporary key pair file if created by this script
    if [ -f "${KEY_NAME_FILE:-}" ]; then
        rm -f "$KEY_NAME_FILE"
        echo "Removed temporary key pair file."
    fi

    echo "Cleanup completed."
}

# Validate AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed"
fi

# Test AWS credentials
```

```
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS credentials are not configured or invalid"
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || true)

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="emr-${RANDOM_ID}"
fi
echo "Using bucket name: $BUCKET_NAME"

# Create S3 bucket with security best practices
echo "Creating S3 bucket: $BUCKET_NAME"
aws s3 mb "s3://$BUCKET_NAME" --region "${AWS_REGION:-us-east-1}" || handle_error
"Failed to create S3 bucket"

# Tag the bucket
aws s3api put-bucket-tagging --bucket "$BUCKET_NAME" \
    --tagging 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=emr-
gs}]'

# Enable bucket versioning for safety
aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled || true

# Block public access to bucket
aws s3api put-public-access-block --bucket "$BUCKET_NAME" \
    --public-access-block-configuration \

"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
|| true
```

```
# Enable encryption on bucket
aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' || true

echo "S3 bucket created successfully with security best practices."

# Create PySpark script
echo "Creating PySpark script: health_violations.py"
cat > health_violations.py << 'EOL'
import argparse

from pyspark.sql import SparkSession

def calculate_red_violations(data_source, output_uri):
    """
    Processes sample food establishment inspection data and queries the data to
    find the top 10 establishments
    with the most Red violations from 2006 to 2020.

    :param data_source: The URI of your food establishment data CSV, such as
    's3://emr-tutorial-bucket/food-establishment-data.csv'.
    :param output_uri: The URI where output is written, such as 's3://emr-
    tutorial-bucket/restaurant_violation_results'.
    """
    with SparkSession.builder.appName("Calculate Red Health
    Violations").getOrCreate() as spark:
        # Load the restaurant violation CSV data
        if data_source is not None:
            restaurants_df = spark.read.option("header", "true").csv(data_source)

        # Create an in-memory DataFrame to query
        restaurants_df.createOrReplaceTempView("restaurant_violations")

        # Create a DataFrame of the top 10 restaurants with the most Red
        violations
        top_red_violation_restaurants = spark.sql("""SELECT name, count(*) AS
        total_red_violations
        FROM restaurant_violations
```

```
        WHERE violation_type = 'RED'
        GROUP BY name
        ORDER BY total_red_violations DESC LIMIT 10""")

    # Write the results to the specified output URI
    top_red_violation_restaurants.write.option("header",
"true").mode("overwrite").csv(output_uri)

if __name__ == "__main__":
    parser = argparse.ArgumentParser()
    parser.add_argument(
        '--data_source', help="The URI for you CSV restaurant data, like an S3
bucket location.")
    parser.add_argument(
        '--output_uri', help="The URI where output is saved, like an S3 bucket
location.")
    args = parser.parse_args()

    calculate_red_violations(args.data_source, args.output_uri)
EOL

# Secure the script file
chmod 600 health_violations.py

# Upload PySpark script to S3
echo "Uploading PySpark script to S3"
aws s3 cp health_violations.py "s3://$BUCKET_NAME/" --sse AES256 || handle_error
"Failed to upload PySpark script"
echo "PySpark script uploaded successfully."

# Download and prepare sample data
echo "Downloading sample data"
curl -sS -o food_establishment_data.zip "https://docs.aws.amazon.com/emr/latest/
ManagementGuide/samples/food_establishment_data.zip" || handle_error "Failed to
download sample data"

# Verify downloaded file
if [ ! -f food_establishment_data.zip ] || [ ! -s food_establishment_data.zip ];
then
    handle_error "Downloaded file is empty or missing"
fi

unzip -o food_establishment_data.zip || handle_error "Failed to unzip sample
data"
```

```
echo "Sample data downloaded and extracted successfully."

# Secure the sample data file
chmod 600 food_establishment_data.csv

# Upload sample data to S3
echo "Uploading sample data to S3"
aws s3 cp food_establishment_data.csv "s3://$BUCKET_NAME/" --sse AES256 ||
  handle_error "Failed to upload sample data"
echo "Sample data uploaded successfully."

# Clean up sensitive local files
rm -f food_establishment_data.zip health_violations.py

# Create IAM default roles for EMR
echo "Creating IAM default roles for EMR"
aws emr create-default-roles 2>/dev/null || true
echo "IAM default roles created successfully."

# Check if EC2 key pair exists
echo "Checking for EC2 key pair"
KEY_PAIRS=$(aws ec2 describe-key-pairs --query "KeyPairs[*].KeyName" --output
  text 2>/dev/null || true)

if [ -z "$KEY_PAIRS" ]; then
  echo "No EC2 key pairs found. Creating a new key pair..."
  KEY_NAME="emr-tutorial-key-${RANDOM_ID}"
  KEY_NAME_FILE="${KEY_NAME}.pem"
  aws ec2 create-key-pair --key-name "$KEY_NAME" \
    --tag-specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-
smith},{Key=tutorial,Value=emr-gs}]' \
    --query "KeyMaterial" --output text > "$KEY_NAME_FILE"
  chmod 400 "$KEY_NAME_FILE"
  echo "Created new key pair: $KEY_NAME"
else
  # Use the first available key pair
  KEY_NAME=$(echo "$KEY_PAIRS" | awk '{print $1}')
  echo "Using existing key pair: $KEY_NAME"
fi

# Launch EMR cluster with security best practices
echo "Launching EMR cluster with Spark"
CLUSTER_RESPONSE=$(aws emr create-cluster \
  --name "EMR Tutorial Cluster" \
```

```
--release-label emr-6.10.0 \  
--applications Name=Spark \  
--ec2-attributes KeyName="$KEY_NAME" \  
--instance-type m5.xlarge \  
--instance-count 3 \  
--use-default-roles \  
--log-uri "s3://$BUCKET_NAME/logs/" \  
--ebs-root-volume-size 100 \  
--tags Key=project,Value=doc-smith Key=tutorial,Value=emr-gs \  
--security-configuration "EMR-Tutorial-SecurityConfig" 2>/dev/null || true)  
  
# Check for errors in the response  
if echo "$CLUSTER_RESPONSE" | grep -i "error" > /dev/null; then  
    handle_error "Failed to create EMR cluster: $CLUSTER_RESPONSE"  
fi  
  
# Extract cluster ID using jq if available, otherwise use alternative parsing  
if command -v jq &> /dev/null; then  
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | jq -r '.ClusterId // empty')  
else  
    CLUSTER_ID=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterId"[[:space:]]*:  
[[:space:]]*"^[^"]*" | grep -o 'j-[A-Z0-9]*' || true)  
fi  
  
if [ -z "$CLUSTER_ID" ] || [ "$CLUSTER_ID" == "null" ]; then  
    handle_error "Failed to extract cluster ID from response: $CLUSTER_RESPONSE"  
fi  
  
echo "EMR cluster created with ID: $CLUSTER_ID"  
  
# Wait for cluster to be ready  
echo "Waiting for cluster to be ready (this may take several minutes)..."  
aws emr wait cluster-running --cluster-id "$CLUSTER_ID" || handle_error "Cluster  
failed to reach running state"  
  
# Check if cluster is in WAITING state  
CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --query  
"Cluster.Status.State" --output text)  
if [ "$CLUSTER_STATE" != "WAITING" ]; then  
    echo "Waiting for cluster to reach WAITING state..."  
    WAIT_COUNT=0  
    MAX_WAIT=120  
    while [ "$CLUSTER_STATE" != "WAITING" ]; do  
        if [ $WAIT_COUNT -ge $MAX_WAIT ]; then
```

```

        handle_error "Cluster did not reach WAITING state within timeout
period"
    fi
    sleep 30
    CLUSTER_STATE=$(aws emr describe-cluster --cluster-id "$CLUSTER_ID" --
query "Cluster.Status.State" --output text)
    echo "Current cluster state: $CLUSTER_STATE"

    # Check for error states
    if [[ "$CLUSTER_STATE" == "TERMINATED_WITH_ERRORS" || "$CLUSTER_STATE" ==
"TERMINATED" ]]; then
        handle_error "Cluster entered error state: $CLUSTER_STATE"
    fi
    WAIT_COUNT=$((WAIT_COUNT + 1))
done
fi

echo "Cluster is now in WAITING state and ready to accept work."

# Submit Spark application as a step
echo "Submitting Spark application as a step"
STEP_RESPONSE=$(aws emr add-steps \
    --cluster-id "$CLUSTER_ID" \
    --steps Type=Spark,Name="Health Violations
Analysis",ActionOnFailure=CONTINUE,Args=["s3://$BUCKET_NAME/
health_violations.py","--data_source","s3://$BUCKET_NAME/
food_establishment_data.csv","--output_uri","s3://$BUCKET_NAME/results/"])

# Check for errors in the response
if echo "$STEP_RESPONSE" | grep -i "error" > /dev/null; then
    handle_error "Failed to submit step: $STEP_RESPONSE"
fi

# Extract step ID using appropriate method
if command -v jq &> /dev/null; then
    STEP_ID=$(echo "$STEP_RESPONSE" | jq -r '.StepIds[0] // empty')
else
    STEP_ID=$(echo "$STEP_RESPONSE" | grep -o 's-[A-Z0-9]*' | head -1 || true)
fi

if [ -z "$STEP_ID" ] || [ "$STEP_ID" == "null" ]; then
    echo "Full step response: $STEP_RESPONSE"
    handle_error "Failed to extract valid step ID from response"
fi

```

```
echo "Step submitted with ID: $STEP_ID"

# Wait for step to complete with timeout
echo "Waiting for step to complete (this may take several minutes)..."
aws emr wait step-complete --cluster-id "$CLUSTER_ID" --step-id "$STEP_ID" ||
  handle_error "Step failed to complete"

# Check step status
STEP_STATE=$(aws emr describe-step --cluster-id "$CLUSTER_ID" --step-id
"$STEP_ID" --query "Step.Status.State" --output text)
if [ "$STEP_STATE" != "COMPLETED" ]; then
  handle_error "Step did not complete successfully. Final state: $STEP_STATE"
fi

echo "Step completed successfully."

# View results
echo "Listing output files in S3"
aws s3 ls "s3://$BUCKET_NAME/results/" || handle_error "Failed to list output
files"

# Download results
echo "Downloading results file"
RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-[0-9]*\.csv"
| head -1 || true)
if [ -z "$RESULT_FILE" ]; then
  echo "No result file found with pattern 'part-[0-9]*.csv'. Trying to find any
  CSV file..."
  RESULT_FILE=$(aws s3 ls "s3://$BUCKET_NAME/results/" | grep -o "part-.*\.csv"
  | head -1 || true)
  if [ -z "$RESULT_FILE" ]; then
    echo "Listing all files in results directory:"
    aws s3 ls "s3://$BUCKET_NAME/results/"
    handle_error "No result file found in the output directory"
  fi
fi

aws s3 cp "s3://$BUCKET_NAME/results/$RESULT_FILE" ./results.csv --sse AES256 ||
  handle_error "Failed to download results file"
chmod 600 ./results.csv

echo "Results downloaded to results.csv"
echo "Top 10 establishments with the most red violations:"
```



```
cat results.csv

# Display SSH connection information
echo ""
echo "To connect to the cluster via SSH, use the following command:"
echo "aws emr ssh --cluster-id $CLUSTER_ID --key-pair-file ${KEY_NAME_FILE:-./}
${KEY_NAME}.pem)"

# Display summary of created resources
echo ""
echo "======"
echo "RESOURCES CREATED"
echo "======"
echo "- S3 Bucket: $BUCKET_NAME"
echo "- EMR Cluster: $CLUSTER_ID"
echo "- Results file: results.csv"
if [ -f "${KEY_NAME_FILE:-}" ]; then
    echo "- EC2 Key Pair: $KEY_NAME (saved to ${KEY_NAME_FILE})"
fi

# Perform cleanup
cleanup

echo "Script completed successfully."
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des commandes de l'AWS CLI .

- [AddSteps](#)
- [Capuchon](#)
- [CreateCluster](#)
- [CreateDefaultRoles](#)
- [CreateKeyPair](#)
- [DescribeCluster](#)
- [DescribeKeyPairs](#)
- [DescribeStep](#)
- [Ls](#)
- [Mb](#)
- [Rb](#)

- [Rm](#)
- [SSH](#)
- [TerminateClusters](#)
- [Attente](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Commencer à gérer la configuration

L'exemple de code suivant illustre comment :

- Créer un compartiment Amazon S3
- Créer une rubrique Amazon SNS
- Création d'un rôle IAM pour Config
- Configurer l'enregistreur de configuration Config
- Configurer le canal de diffusion Config
- Démarrez l'enregistreur de configuration
- Vérifiez la configuration de Config

Bash

AWS CLI avec le script Bash

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code](#).

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI
```

```
# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
"true" ]; then
        echo "Deleting delivery channel..."
    fi
}
```

```

    aws configservice delete-delivery-channel --delivery-channel-name
"$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
"true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi

    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
true
        fi
    fi
}

```

```

# Function to display created resources
display_resources() {
    echo ""
    echo "=====
    echo "CREATED RESOURCES"
    echo "=====
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "=====
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"

```

```
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1" # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true"
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
```

```
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
  Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^']*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-{$RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^']*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
  Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"
```

```
echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15
```



```
# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-records 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
    "name": "${CONFIG_RECORDER_NAME}",
    "roleARN": "${ROLE_ARN}",
    "recordingMode": {
        "recordingFrequency": "CONTINUOUS"
    }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
    "allSupported": true,
    "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"
```

```

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
    Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
    "name": "${DELIVERY_CHANNEL_NAME}",
    "s3BucketName": "${S3_BUCKET_NAME}",
    "snsTopicARN": "${SNS_TOPIC_ARN}",
    "configSnapshotDeliveryProperties": {
        "deliveryFrequency": "Six_Hours"
    }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

```

```
    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"
```

```
echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-records)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "====="
echo "CLEANUP CONFIRMATION"
echo "====="
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
    echo "Resources will not be cleaned up. You can manually clean them up later."
fi

echo "Script completed successfully!"
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des commandes de l'AWS CLI .
 - [AttachRolePolicy](#)
 - [CreateBucket](#)
 - [CreateRole](#)
 - [CreateTopic](#)
 - [DeleteBucket](#)
 - [DeleteConfigurationRecorder](#)

- [DeleteDeliveryChannel](#)
- [DeleteRole](#)
- [DeleteRolePolicy](#)
- [DeleteTopic](#)
- [DescribeConfigurationRecorderStatus](#)
- [DescribeConfigurationRecorders](#)
- [DescribeDeliveryChannels](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [PutConfigurationRecorder](#)
- [PutDeliveryChannel](#)
- [PutPublicAccessBlock](#)
- [PutRolePolicy](#)
- [Rm](#)
- [StartConfigurationRecorder](#)
- [StopConfigurationRecorder](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Commencer à extraire le texte d'un document

L'exemple de code suivant illustre comment :

- Création d'un compartiment S3
- Charger un document sur S3
- nettoyer des ressources ;

Bash

AWS CLI avec le script Bash

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code](#).

```
#!/bin/bash

# Amazon Textract Getting Started Tutorial Script
# This script demonstrates how to use Amazon Textract to analyze document text

set -euo pipefail

# Set up logging with restricted permissions
LOG_FILE="textract-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "====="
echo "Amazon Textract Getting Started Tutorial"
echo "====="
echo "This script will guide you through using Amazon Textract to analyze
document text."
echo ""

# Function to check for errors in command output and exit code
check_error() {
    local exit_code=$1
    local output=$2
    local cmd=$3

    if [ $exit_code -ne 0 ] || echo "$output" | grep -i "error" > /dev/null; then
        echo "ERROR: Command failed: $cmd"
        echo "$output" | sed 's/\(aws_secret_access_key\|Authorization\|X-Amz-
Security-Token\).*\/1=***REDACTED***\/g'
        cleanup_on_error
        exit 1
    fi
}
```

```
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    # Clean up temporary JSON files
    if [ -f "document.json" ]; then
        rm -f document.json
    fi

    if [ -f "features.json" ]; then
        rm -f features.json
    fi

    if [ -n "${DOCUMENT_NAME:-}" ] && [ -n "${BUCKET_NAME:-}" ]; then
        echo "Deleting document from S3..."
        aws s3 rm "s3://${BUCKET_NAME}/${DOCUMENT_NAME}" || echo "Failed to
delete document"
    fi

    if [ -n "${BUCKET_NAME:-}" ] && [ "${BUCKET_IS_SHARED:-false}" = "false" ];
then
        echo "Deleting S3 bucket..."
        aws s3 rb "s3://${BUCKET_NAME}" --force || echo "Failed to delete bucket"
    fi
}

# Set up trap for cleanup on exit
trap cleanup_on_error EXIT

# Verify AWS CLI is installed and configured
echo "Verifying AWS CLI configuration..."
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed."
    exit 1
fi

AWS_CONFIG_OUTPUT=$(aws configure list 2>&1)
AWS_CONFIG_STATUS=$?
if [ $AWS_CONFIG_STATUS -ne 0 ]; then
    echo "ERROR: AWS CLI is not properly configured."
```

```
    echo "$AWS_CONFIG_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization
\).*\/1=***REDACTED***/'g'
    exit 1
fi

# Verify AWS region is configured and supports Textract
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Please run 'aws configure' to set a
default region."
    exit 1
fi

# Check if Textract is available in the configured region
echo "Checking if Amazon Textract is available in region $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-${RANDOM_ID}"
fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
```



```

    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
{"SSEAlgorithm": "AES256"}]}'} 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text
50,50 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAyAAAAAFcSJAAAADU1EQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."

```

```
UPLOAD_OUTPUT=$(aws s3 cp ".$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256
2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp ".$DOCUMENT_NAME "s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues
cat > document.json << 'EOF'
{
  "S3Object": {
    "Bucket": "BUCKET_PLACEHOLDER",
    "Name": "DOCUMENT_PLACEHOLDER"
  }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES","FORMS","SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
  echo "ERROR: Document analysis failed"
  echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*\1=***REDACTED***/'
  exit 1
fi
```

```
# Save the analysis results to a file with restricted permissions
echo "$ANALYZE_OUTPUT" > textract-analysis-results.json
chmod 600 textract-analysis-results.json
echo "Analysis results saved to textract-analysis-results.json"
RESOURCES_CREATED+=("Local file: textract-analysis-results.json")

# Display a summary of the analysis
echo ""
echo "====="
echo "Analysis Summary"
echo "====="
PAGES=$(echo "$ANALYZE_OUTPUT" | grep -o '"Pages": [0-9]*' | head -1 | awk
'{{print $2}}')
echo "Document pages: $PAGES"

BLOCKS_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType":' | wc -l)
echo "Total blocks detected: $BLOCKS_COUNT"

# Count different block types using jq if available, fallback to grep
PAGE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "PAGE"' | wc -l ||
echo 0)
LINE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "LINE"' | wc -l ||
echo 0)
WORD_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "WORD"' | wc -l ||
echo 0)
TABLE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "TABLE"' | wc -l ||
echo 0)
CELL_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "CELL"' | wc -l ||
echo 0)
KEY_VALUE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "KEY_VALUE_SET"'
| wc -l || echo 0)
SIGNATURE_COUNT=$(echo "$ANALYZE_OUTPUT" | grep -o '"BlockType": "SIGNATURE"' |
wc -l || echo 0)

echo "Pages: $PAGE_COUNT"
echo "Lines of text: $LINE_COUNT"
echo "Words: $WORD_COUNT"
echo "Tables: $TABLE_COUNT"
echo "Table cells: $CELL_COUNT"
echo "Key-value pairs: $KEY_VALUE_COUNT"
echo "Signatures: $SIGNATURE_COUNT"
echo ""
```

```

# Cleanup confirmation
echo ""
echo "=====
echo "RESOURCES CREATED"
echo "=====
for resource in "${RESOURCES_CREATED[@]}"; do
    echo "- $resource"
done
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Cleaning up resources..."

# Delete document from S3
echo "Deleting document from S3..."
DELETE_DOC_OUTPUT=$(aws s3 rm "s3://$BUCKET_NAME/$DOCUMENT_NAME" 2>&1)
DELETE_DOC_STATUS=$?
echo "$DELETE_DOC_OUTPUT"
check_error $DELETE_DOC_STATUS "$DELETE_DOC_OUTPUT" "aws s3 rm s3://$BUCKET_NAME/
$DOCUMENT_NAME"

# Delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Deleting S3 bucket..."
    DELETE_BUCKET_OUTPUT=$(aws s3 rb "s3://$BUCKET_NAME" --force 2>&1)
    DELETE_BUCKET_STATUS=$?
    echo "$DELETE_BUCKET_OUTPUT"
    check_error $DELETE_BUCKET_STATUS "$DELETE_BUCKET_OUTPUT" "aws s3 rb s3://
$BUCKET_NAME --force"
fi

# Delete local JSON files
rm -f document.json features.json

echo "Cleanup complete. The analysis results file (textract-analysis-
results.json) has been kept."

echo ""
echo "=====
echo "Tutorial complete!"
echo "=====
echo "You have successfully analyzed a document using Amazon Textract."
echo "The analysis results are available in textract-analysis-results.json"

```

```
echo ""  
  
trap - EXIT
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des commandes de l'AWS CLI .
 - [AnalyzeDocument](#)
 - [Capuchon](#)
 - [Aide](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Commencer à utiliser les magasins de fonctionnalités d'apprentissage automatique

L'exemple de code suivant illustre comment :

- Définir des autorisations IAM
- Création d'un rôle SageMaker d'exécution
- Création de groupes de fonctionnalités
- nettoyer des ressources ;

Bash

AWS CLI avec le script Bash

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code](#).

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
```

```

        --output text)
    echo "Current status: ${status}"

    if [ "$status" = "Failed" ]; then
        handle_error "Feature group ${feature_group_name} creation failed"
    fi
done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
"$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)

```

```

        echo "Unknown resource type: $resource_type"
        ;;
    esac
done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
    Key=tutorial,Value=sagemaker-featurestore 2>&1

```



```
# Attach necessary policies
echo "Attaching policies to role..." >&2

# SageMaker execution policy
local policy1_result=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

if echo "$policy1_result" | grep -i "error" > /dev/null; then
    handle_error "Failed to attach SageMaker policy: $policy1_result"
fi

# S3 access policy
local policy2_result=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

if echo "$policy2_result" | grep -i "error" > /dev/null; then
    handle_error "Failed to attach S3 policy: $policy2_result"
fi

# Get account ID for role ARN
local account_id=$(aws sts get-caller-identity --query Account --output text)
local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

echo "Role ARN: $role_arn" >&2
echo "Waiting 10 seconds for role to propagate..." >&2
sleep 10

# Return only the role ARN to stdout
echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"
fi
```

```
# Validate the role ARN
ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\\///')
ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
    echo "Creating a new role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
```

```
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
            --region "$REGION" 2>&1)
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
    fi
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
    'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
    "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
```

```

        exit 1
    fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-`${RANDOM_ID}`"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://${S3_BUCKET_NAME}/${PREFIX}"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

```

```

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
  --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
  --record-identifier-feature-name "customer_id" \
  --event-time-feature-name "EventTime" \
  --feature-definitions '[
    {"FeatureName": "customer_id", "FeatureType": "Integral"},
    {"FeatureName": "order_id", "FeatureType": "String"},
    {"FeatureName": "order_date", "FeatureType": "String"},
    {"FeatureName": "product", "FeatureType": "String"},
    {"FeatureName": "quantity", "FeatureType": "Integral"},
    {"FeatureName": "amount", "FeatureType": "Fractional"},
    {"FeatureName": "EventTime", "FeatureType": "Fractional"}
  ]' \
  --online-store-config '{"EnableOnlineStore": true}' \
  --offline-store-config '{
    "S3StorageConfig": {
      "S3Uri": "s3://'$S3_BUCKET_NAME'/'$PREFIX'"
    },
    "DisableGlueTableCreation": false
  }' \
  --role-arn "$ROLE_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to create orders feature group: $ORDERS_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

```

```
# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "573291"},
    {"FeatureName": "name", "ValueAsString": "John Doe"},
    {"FeatureName": "age", "ValueAsString": "35"},
    {"FeatureName": "address", "ValueAsString": "123 Main St"},
    {"FeatureName": "membership_type", "ValueAsString": "premium"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "109382"},
    {"FeatureName": "name", "ValueAsString": "Jane Smith"},
    {"FeatureName": "age", "ValueAsString": "28"},
    {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
    {"FeatureName": "membership_type", "ValueAsString": "standard"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
```

```

--feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
--record '[
    {"FeatureName": "customer_id", "ValueAsString": "573291"},
    {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
    {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
    {"FeatureName": "product", "ValueAsString": "Laptop"},
    {"FeatureName": "quantity", "ValueAsString": "1"},
    {"FeatureName": "amount", "ValueAsString": "1299.99"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"

```

```

CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
| grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

```



```

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "======"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "======"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "======"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "======"
    echo "CLEANUP CONFIRMATION"
    echo "======"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else

```

```

        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
$ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"

```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des commandes de l'AWS CLI .
 - [AttachRolePolicy](#)
 - [CreateBucket](#)
 - [CreateFeatureGroup](#)
 - [CreateRole](#)
 - [DeleteBucket](#)
 - [DeleteFeatureGroup](#)
 - [DeleteRole](#)
 - [DescribeFeatureGroup](#)
 - [DetachRolePolicy](#)
 - [GetCallerIdentity](#)
 - [GetRole](#)
 - [ListFeatureGroups](#)
 - [PutPublicAccessBlock](#)
 - [Rm](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Débuter avec le stockage d'objets

L'exemple de code suivant illustre comment :

- Créez votre premier compartiment S3
- Charger un objet
- Activation de la gestion des versions
- Configurer le chiffrement par défaut
- Ajoutez des tags à votre bucket
- Répertoire les objets et les versions
- nettoyer des ressources ;

Bash

AWS CLI avec le script Bash

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code](#).

```
#!/bin/bash
# S3 Getting Started - Create a bucket, upload and download objects, copy to a
# folder prefix, enable versioning, configure encryption and public access
# blocking, tag the bucket, list objects and versions, and clean up.

set -eE
set -o pipefail

# =====
# Prerequisites check
# =====

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
"$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured. Run 'aws configure' or set
AWS_DEFAULT_REGION."
```

```

        exit 1
    fi

    # Verify AWS credentials are configured
    if ! aws sts get-caller-identity &>/dev/null; then
        echo "ERROR: AWS credentials not configured or invalid. Run 'aws configure'."
        exit 1
    fi

    # =====
    # Setup: logging, temp directory, resource tracking
    # =====

    UNIQUE_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
    # Check for shared prereq bucket
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || true)
    if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
        BUCKET_NAME="$PREREQ_BUCKET"
        BUCKET_IS_SHARED=true
        echo "Using shared bucket: $BUCKET_NAME"
    else
        BUCKET_IS_SHARED=false
        BUCKET_NAME="s3api-`${UNIQUE_ID}`"
    fi

    TEMP_DIR=$(mktemp -d)
    trap 'rm -rf "$TEMP_DIR"' EXIT
    LOG_FILE="${TEMP_DIR}/s3-gettingstarted.log"
    CREATED_RESOURCES=()

    exec > >(tee -a "$LOG_FILE") 2>&1

    echo "======"
    echo "S3 Getting Started"
    echo "======"
    echo "Bucket name: ${BUCKET_NAME}"
    echo "Temp directory: ${TEMP_DIR}"
    echo "Log file: ${LOG_FILE}"
    echo ""

    # =====

```

```

# Helper functions
# =====

get_region() {
    echo "${AWS_REGION:-${AWS_DEFAULT_REGION:-${CONFIGURED_REGION}}}"
}

delete_object_versions() {
    local bucket=$1
    local query=$2

    local versions
    versions=$(aws s3api list-object-versions \
        --bucket "$bucket" \
        --query "$query" \
        --output json 2>&1) || return 0

    if [ -z "$versions" ] || [ "$versions" = "null" ] || [ "$versions" = "[]" ];
    then
        return 0
    fi

    echo "$versions" | jq -r '.[ ] | "\(.Key)\t\(.VersionId)"" 2>/dev/null | while
IFS=$'\t' read -r key version_id; do
    if [ -n "$key" ] && [ "$key" != "null" ]; then
        aws s3api delete-object --bucket "$bucket" --key "$key" --version-id
"$version_id" >/dev/null 2>&1 || true
    fi
done

    return 0
}

# =====
# Error handling and cleanup functions
# =====

cleanup() {
    echo ""
    echo "=====
    echo "CLEANUP"
    echo "=====

    if [ "$BUCKET_IS_SHARED" = "false" ]; then

```

```

    echo "Deleting all object versions in bucket..."

    delete_object_versions "$BUCKET_NAME" "Versions[.
{Key:Key,VersionId:VersionId}" || true

    delete_object_versions "$BUCKET_NAME" "DeleteMarkers[.
{Key:Key,VersionId:VersionId}" || true

    echo "Deleting bucket: ${BUCKET_NAME}"
    if ! aws s3api delete-bucket --bucket "$BUCKET_NAME" 2>/dev/null; then
        echo "WARNING: Failed to delete bucket ${BUCKET_NAME}"
    fi

    # Clean up logs bucket
    LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
    if aws s3api head-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/null; then
        echo "Deleting log bucket: ${LOG_TARGET_BUCKET}"
        if ! aws s3api delete-bucket --bucket "$LOG_TARGET_BUCKET" 2>/dev/
null; then
            echo "WARNING: Failed to delete bucket ${LOG_TARGET_BUCKET}"
        fi
    fi
else
    echo "Keeping shared bucket: ${BUCKET_NAME}"
fi

echo ""
echo "Cleanup complete."
}

handle_error() {
    local line_number=$1
    echo ""
    echo "======"
    echo "ERROR on line ${line_number}"
    echo "======"
    echo ""
    echo "Resources created before error:"
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        for RESOURCE in "${CREATED_RESOURCES[@]}"; do
            echo "  - ${RESOURCE}"
        done
    else
        echo "  (none)"
    fi
}

```

```

    fi
    echo ""
    echo "Attempting cleanup..."
    cleanup
    exit 1
}

trap 'handle_error "$LINENO"' ERR

# =====
# Step 1: Create a bucket
# =====

echo "Step 1: Creating bucket ${BUCKET_NAME}..."
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        if ! aws s3api create-bucket --bucket "$BUCKET_NAME" >/dev/null 2>&1;
        then
            echo "ERROR: Failed to create bucket $BUCKET_NAME"
            exit 1
        fi
    else
        if ! aws s3api create-bucket \
            --bucket "$BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
        2>&1; then
            echo "ERROR: Failed to create bucket $BUCKET_NAME in region $REGION"
            exit 1
        fi
    fi
    CREATED_RESOURCES+=("s3:bucket:${BUCKET_NAME}")
    echo "Bucket created."

    if ! aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {

```

```

        "Key": "tutorial",
        "Value": "s3-gettingstarted"
    }
]
}' >/dev/null 2>&1; then
    echo "WARNING: Failed to tag bucket"
fi
fi
echo ""

# =====
# Step 2: Upload a sample text file
# =====

echo "Step 2: Uploading a sample text file..."

SAMPLE_FILE="${TEMP_DIR}/sample.txt"
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is a sample file for the getting started tutorial.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload sample.txt"
    exit 1
fi
echo "File uploaded."
echo ""

# =====
# Step 3: Download the object
# =====

echo "Step 3: Downloading the object..."

DOWNLOAD_FILE="${TEMP_DIR}/downloaded-sample.txt"
if ! aws s3api get-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    "$DOWNLOAD_FILE" >/dev/null 2>&1; then

```



```

        echo "ERROR: Failed to download sample.txt"
        exit 1
    fi
    echo "Downloaded to: ${DOWNLOAD_FILE}"
    echo "Contents:"
    cat "$DOWNLOAD_FILE"
    echo ""

    # =====
    # Step 4: Copy the object to a folder prefix
    # =====

    echo "Step 4: Copying object to a folder prefix..."

    if ! aws s3api copy-object \
        --bucket "$BUCKET_NAME" \
        --copy-source "${BUCKET_NAME}/sample.txt" \
        --key "backup/sample.txt" \
        --server-side-encryption AES256 \
        --metadata-directive COPY >/dev/null 2>&1; then
        echo "ERROR: Failed to copy object to backup/sample.txt"
        exit 1
    fi
    echo "Object copied to backup/sample.txt."
    echo ""

    # =====
    # Step 5: Enable versioning and upload a second version
    # =====

    echo "Step 5: Enabling versioning..."

    if ! aws s3api put-bucket-versioning \
        --bucket "$BUCKET_NAME" \
        --versioning-configuration Status=Enabled >/dev/null 2>&1; then
        echo "ERROR: Failed to enable versioning"
        exit 1
    fi
    echo "Versioning enabled."

    echo "Uploading a second version of sample.txt..."
    cat > "$SAMPLE_FILE" << 'EOF'
    Hello, Amazon S3! This is version 2 of the sample file.
    EOF

```

```

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted,version=2" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload second version of sample.txt"
    exit 1
fi
echo "Second version uploaded."
echo ""

# =====
# Step 6: Configure SSE-S3 encryption
# =====

echo "Step 6: Configuring SSE-S3 default encryption..."

if ! aws s3api put-bucket-encryption \
    --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": true
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to configure SSE-S3 encryption"
    exit 1
fi
echo "SSE-S3 encryption configured."
echo ""

# =====
# Step 7: Block all public access
# =====

echo "Step 7: Blocking all public access..."

if ! aws s3api put-public-access-block \

```

```

    --bucket "$BUCKET_NAME" \
    --public-access-block-configuration '{
        "BlockPublicAcls": true,
        "IgnorePublicAcls": true,
        "BlockPublicPolicy": true,
        "RestrictPublicBuckets": true
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to block public access"
    exit 1
fi
echo "Public access blocked."
echo ""

# =====
# Step 8: Configure bucket logging
# =====

echo "Step 8: Configuring bucket logging..."

LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    REGION=$(get_region)
    if [ "$REGION" = "us-east-1" ]; then
        aws s3api create-bucket --bucket "$LOG_TARGET_BUCKET" >/dev/null 2>&1 ||
true
    else
        aws s3api create-bucket \
            --bucket "$LOG_TARGET_BUCKET" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
2>&1 || true
    fi

    if ! aws s3api put-bucket-tagging \
        --bucket "$LOG_TARGET_BUCKET" \
        --tagging '{
            "TagSet": [
                {
                    "Key": "project",
                    "Value": "doc-smith"
                },
                {
                    "Key": "tutorial",
                    "Value": "s3-gettingstarted"
                }
            ]
        }' >/dev/null 2>&1; then
        echo "ERROR: Failed to configure bucket logging"
        exit 1
    fi
fi

```

```

        }
    ]
}' >/dev/null 2>&1; then
    echo "WARNING: Failed to tag log bucket"
fi

aws s3api put-bucket-acl --bucket "$LOG_TARGET_BUCKET" --acl log-delivery-
write 2>/dev/null || true

if ! aws s3api put-bucket-logging \
    --bucket "$BUCKET_NAME" \
    --bucket-logging-status '{
        "LoggingEnabled": {
            "TargetBucket": "'$LOG_TARGET_BUCKET'",
            "TargetPrefix": "logs/"
        }
    }' >/dev/null 2>&1; then
    echo "WARNING: Failed to configure bucket logging"
else
    echo "Bucket logging configured."
fi
else
    echo "Skipping logging configuration for shared bucket."
fi
echo ""

# =====
# Step 9: Tag the bucket
# =====

echo "Step 9: Tagging the bucket..."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \
    --tagging '{
        "TagSet": [
            {
                "Key": "project",
                "Value": "doc-smith"
            },
            {
                "Key": "tutorial",
                "Value": "s3-gettingstarted"
            },
        ]
    }' >/dev/null 2>&1; then
    echo "WARNING: Failed to tag bucket"
fi

```

```

        {
            "Key": "Environment",
            "Value": "Tutorial"
        },
        {
            "Key": "Project",
            "Value": "S3-GettingStarted"
        },
        {
            "Key": "ManagedBy",
            "Value": "Bash-Tutorial"
        }
    ]
}' >/dev/null 2>&1; then
echo "ERROR: Failed to tag bucket"
exit 1
fi
echo "Bucket tagged."

echo "Verifying tags..."
if ! aws s3api get-bucket-tagging --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to retrieve bucket tags"
fi
echo ""

# =====
# Step 10: List objects and versions
# =====

echo "Step 10: Listing objects..."

if ! aws s3api list-objects-v2 --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list objects"
fi
echo ""

echo "Listing object versions..."

if ! aws s3api list-object-versions --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list object versions"
fi
echo ""

# =====

```

```
# Step 11: Cleanup
# =====

echo ""
echo "=====
echo "TUTORIAL COMPLETE"
echo "=====
echo ""
echo "Resources created:"
if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
    for RESOURCE in "${CREATED_RESOURCES[@}"; do
        echo "  - ${RESOURCE}"
    done
else
    echo "  (none)"
fi
echo ""
echo "=====
echo "CLEANUP"
echo "=====
echo "Cleaning up all created resources..."
cleanup

echo ""
echo "Done."
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des commandes de l'AWS CLI .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [HeadObject](#)
 - [ListObjectVersions](#)
 - [ListObjectsV2](#)
 - [PutBucketEncryption](#)
 - [PutBucketTagging](#)
 - [PutBucketVersioning](#)

- [PutObject](#)
- [PutPublicAccessBlock](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Commencer à utiliser l'analyse des requêtes

L'exemple de code suivant illustre comment :

- Création d'un compartiment S3 pour les résultats des requêtes
- Créer une base de données
- Création d'une table
- Exécuter une requête
- Création et utilisation de requêtes nommées
- nettoyer des ressources ;

Bash

AWS CLI avec le script Bash

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code](#).

```
#!/bin/bash

# Amazon Athena Getting Started Script
# This script demonstrates how to use Amazon Athena with AWS CLI
# It creates a database, table, runs queries, and manages named queries

set -euo pipefail

# Security: Validate AWS credentials are configured
if ! aws sts get-caller-identity &>/dev/null; then
```

```

    echo "ERROR: AWS credentials not configured or invalid"
    exit 1
fi

# Security: Restrict umask to prevent world-readable files
umask 0077

# Set up logging with restricted permissions
LOG_FILE="athena-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Athena Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${NAMED_QUERY_ID:-}" ]; then
        echo "- Named Query: $NAMED_QUERY_ID"
    fi
    if [ -n "${DATABASE_NAME:-}" ]; then
        echo "- Database: $DATABASE_NAME"
        if [ -n "${TABLE_NAME:-}" ]; then
            echo "- Table: $TABLE_NAME in $DATABASE_NAME"
        fi
    fi
    if [ -n "${S3_BUCKET:-}" ]; then
        echo "- S3 Bucket: $S3_BUCKET"
    fi

    echo "Exiting..."
    exit 1
}

# Security: Validate bucket name format
validate_bucket_name() {
    local bucket_name="$1"
    if [[ ! "$bucket_name" =~ ^[a-z0-9][a-z0-9.-]*[a-z0-9]$ ]] ||
[ ${#bucket_name} -lt 3 ] || [ ${#bucket_name} -gt 63 ]; then
        return 1
    fi
}

```



```

    return 0
}

# Security: Validate database and table names
validate_identifier() {
    local identifier="$1"
    if [[ ! "$identifier" =~ ^[a-zA-Z_][a-zA-Z0-9_]*$ ]]; then
        return 1
    fi
    return 0
}

# Security: Safely generate random identifier
if ! command -v openssl &>/dev/null; then
    RANDOM_ID=$(head -c 6 /dev/urandom | od -An -tx1 | tr -d ' ')
else
    RANDOM_ID=$(openssl rand -hex 6)
fi

# Security: Validate random ID format
if [[ ! "$RANDOM_ID" =~ ^[a-f0-9]{12}$ ]]; then
    handle_error "Failed to generate valid random ID"
fi

# Check for shared prereq bucket with proper error handling
PREREQ_BUCKET=""
if aws cloudformation describe-stacks --stack-name tutorial-prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
    text 2>/dev/null | grep -qv "^$"; then
    PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-
prereqs-bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --
output text 2>/dev/null)
fi

if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET="athena-${RANDOM_ID}"
fi

```

```
if ! validate_bucket_name "$S3_BUCKET"; then
    handle_error "Invalid S3 bucket name: $S3_BUCKET"
fi

DATABASE_NAME="mydatabase"
TABLE_NAME="cloudfront_logs"

if ! validate_identifrier "$DATABASE_NAME"; then
    handle_error "Invalid database name: $DATABASE_NAME"
fi

if ! validate_identifrier "$TABLE_NAME"; then
    handle_error "Invalid table name: $TABLE_NAME"
fi

# Get the current AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No AWS region found in configuration, defaulting to $AWS_REGION"
fi

# Security: Validate region format - expanded regex for newer regions
if [[ ! "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]] && [[ ! "$AWS_REGION" =~
    ^[a-z]+-[a-z]+-[0-9]{1}$ ]]; then
    echo "WARNING: Region format may be invalid: $AWS_REGION"
fi

echo "Using AWS Region: $AWS_REGION"

# Create S3 bucket for Athena query results
echo "Creating S3 bucket for Athena query results: $S3_BUCKET"
if [ "$BUCKET_IS_SHARED" = false ]; then
    CREATE_BUCKET_RESULT=$(aws s3 mb "s3://$S3_BUCKET" --region "$AWS_REGION"
2>&1)
    if echo "$CREATE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        handle_error "Failed to create S3 bucket: $CREATE_BUCKET_RESULT"
    fi

    aws s3api put-bucket-tagging \
        --bucket "$S3_BUCKET" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-athena-gs}]'
```

```

# Security: Enable S3 bucket encryption with KMS validation
echo "Enabling default encryption on S3 bucket..."
if ! aws s3api put-bucket-encryption \
    --bucket "$S3_BUCKET" \
    --server-side-encryption-configuration '{
        "Rules": [{
            "ApplyServerSideEncryptionByDefault": {
                "SSEAlgorithm": "AES256"
            }
        }]
    }' 2>&1; then
    echo "Warning: Could not enable encryption on bucket"
fi

# Security: Block public access
echo "Blocking public access to S3 bucket..."
if ! aws s3api put-public-access-block \
    --bucket "$S3_BUCKET" \
    --public-access-block-configuration \

"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1; then
    echo "Warning: Could not block public access on bucket"
fi

# Security: Enable versioning for data protection
echo "Enabling versioning on S3 bucket..."
if ! aws s3api put-bucket-versioning \
    --bucket "$S3_BUCKET" \
    --versioning-configuration Status=Enabled 2>&1; then
    echo "Warning: Could not enable versioning on bucket"
fi

echo "S3 bucket created successfully: $S3_BUCKET"
fi

# Step 1: Create a database
echo "Step 1: Creating Athena database: $DATABASE_NAME"
CREATE_DB_RESULT=$(aws athena start-query-execution \
    --query-string "CREATE DATABASE IF NOT EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$CREATE_DB_RESULT" | grep -qi "error\|failed"; then

```

```
    handle_error "Failed to create database: $CREATE_DB_RESULT"
fi

QUERY_ID=$(echo "$CREATE_DB_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/
null || echo "$CREATE_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from database creation response"
fi
echo "Database creation query ID: $QUERY_ID"

# Wait for database creation to complete
echo "Waiting for database creation to complete..."
WAIT_TIMEOUT=60
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Database creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Database creation failed with status: $QUERY_STATUS"
    fi
    echo "Database creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Database creation timed out"
fi

# Verify the database was created
echo "Verifying database creation..."
LIST_DB_RESULT=$(aws athena list-databases --catalog-name AwsDataCatalog --region
"$AWS_REGION" 2>&1)
if echo "$LIST_DB_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list databases: $LIST_DB_RESULT"
fi
echo "$LIST_DB_RESULT"
```

[illegible]

```
# Wait for table creation to complete
echo "Waiting for table creation to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Table creation completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Table creation failed with status: $QUERY_STATUS"
    fi
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
```

```

--result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
--region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
|| echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
        --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
"$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"

```

```

fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi

DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/
dev/null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi

```



```
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    \
    --region "$AWS_REGION" 2>&1)
if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
|| echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
```

```

    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Named query execution failed with status: $QUERY_STATUS"
    fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "=====
echo "RESOURCES CREATED"
echo "=====
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "=====

# Auto-confirm cleanup
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"

```

```

echo "=====
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
"$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else
        echo "Named query deleted successfully."
    fi

    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
        --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*'
| cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Table dropped successfully."
                    break

```

```

        elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
            echo "Warning: Table deletion failed with status:
$QUERY_STATUS"
            break
        fi
        echo "Table deletion in progress, status: $QUERY_STATUS"
        sleep 2
        ((ELAPSED+=2))
    done
fi
fi

# Drop database
echo "Dropping database: $DATABASE_NAME"
DROP_DB_RESULT=$(aws athena start-query-execution \
    --query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
    echo "Warning: Failed to drop database: $DROP_DB_RESULT"
else
    QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
cut -d'"' -f4)
    if [ -n "$QUERY_ID" ]; then
        echo "Waiting for database deletion to complete..."

        ELAPSED=0
        while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
            QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
            if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                echo "Database dropped successfully."
                break
            elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
                echo "Warning: Database deletion failed with status:
$QUERY_STATUS"
                break
            fi
        done
    fi
fi

```

```
        echo "Database deletion in progress, status: $QUERY_STATUS"
        sleep 2
        ((ELAPSED+=2))
    done
fi
fi

# Empty and delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Emptying S3 bucket: $S3_BUCKET"
    EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
    if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
    else
        echo "S3 bucket emptied successfully."
    fi

    echo "Deleting S3 bucket: $S3_BUCKET"
    DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
    if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
    else
        echo "S3 bucket deleted successfully."
    fi
else
    echo "Skipping S3 bucket deletion (shared resource)"
fi

# Security: Remove downloaded query results
if [ -f "./query-results.csv" ]; then
    if command -v shred &>/dev/null; then
        shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
    else
        rm -f "./query-results.csv"
    fi
    echo "Query results file securely removed."
fi

echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des commandes de l'AWS CLI .
 - [Capuchon](#)
 - [CreateNamedQuery](#)
 - [DeleteNamedQuery](#)
 - [GetNamedQuery](#)
 - [GetQueryExecution](#)
 - [GetQueryResults](#)
 - [ListDatabases](#)
 - [ListNamedQueries](#)
 - [ListTableMetadata](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)
 - [StartQueryExecution](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisez les fonctionnalités de verrouillage d'objets d'Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment travailler avec les fonctionnalités de verrouillage d'objet S3.

.NET

SDK pour .NET

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif illustrant les fonctionnalités de verrouillage d'objet d'Amazon S3.

```
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ObjectLockScenario;

public static class S3ObjectLockWorkflow
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

        This .NET example performs the following tasks:
        1. Create test Amazon Simple Storage Service (S3) buckets with different
        lock policies.
        2. Upload sample objects to each bucket.
        3. Set some Legal Hold and Retention Periods on objects and buckets.
        4. Investigate lock policies by viewing settings or attempting to delete
        or overwrite objects.
        5. Clean up objects and buckets.
    */

    public static S3ActionsWrapper _s3ActionsWrapper = null!;
    public static IConfiguration _configuration = null!;
    private static string _resourcePrefix = null!;
    private static string noLockBucketName = null!;
    private static string lockEnabledBucketName = null!;
    private static string retentionAfterCreationBucketName = null!;
    private static List<string> bucketNames = new List<string>();
    private static List<string> fileNames = new List<string>();

    public static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
```

```
using var host = Host.CreateDefaultBuilder(args)
    .ConfigureLogging(logging =>
        logging.AddFilter("System", LogLevel.Debug)
        .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
        .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
    .ConfigureServices((_, services) =>
        services.AddAWSService<IAmazonS3>()
        .AddTransient<S3ActionsWrapper>()
    )
    .Build();

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load settings from .json file.
    .AddJsonFile("settings.local.json",
        true) // Optionally, load local settings.
    .Build();

ConfigurationSetup();

ServicesSetup(host);

try
{
    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
    Console.WriteLine(new string('-', 80));
    await Setup(true);

    await DemoActionChoices();

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Cleaning up resources.");
    Console.WriteLine(new string('-', 80));
    await Cleanup(true);

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Amazon S3 Object Locking Scenario is complete.");
    Console.WriteLine(new string('-', 80));
}
catch (Exception ex)
```



```

        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem: {ex.Message}");
            await Cleanup(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
    }

    /// <summary>
    /// Any setup operations needed.
    /// </summary>
    public static void ConfigurationSetup()
    {
        _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

        noLockBucketName = _resourcePrefix + "-no-lock";
        lockEnabledBucketName = _resourcePrefix + "-lock-enabled";
        retentionAfterCreationBucketName = _resourcePrefix + "-retention-after-creation";

        bucketNames.Add(noLockBucketName);
        bucketNames.Add(lockEnabledBucketName);
        bucketNames.Add(retentionAfterCreationBucketName);
    }

    // <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Setup(bool interactive)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create
several S3\n" +

```

```
        "buckets and files to demonstrate working with S3 locking features.
\n");

    Console.WriteLine(new string('-', 80));
    Console.WriteLine("Press Enter when you are ready to start.");
    if (interactive)
        Console.ReadLine();

    Console.WriteLine("\nS3 buckets can be created either with or without
object lock enabled.");
    await _s3ActionsWrapper.CreateBucketWithObjectLock(noLockBucketName,
false);
    await _s3ActionsWrapper.CreateBucketWithObjectLock(lockEnabledBucketName,
true);
    await
_s3ActionsWrapper.CreateBucketWithObjectLock(retentionAfterCreationBucketName,
false);

    Console.WriteLine("Press Enter to continue.");
    if (interactive)
        Console.ReadLine();

    Console.WriteLine("\nA bucket can be configured to use object locking
with a default retention period.");
    await
_s3ActionsWrapper.ModifyBucketDefaultRetention(retentionAfterCreationBucketName,
true,
        ObjectLockRetentionMode.Governance, DateTime.UtcNow.AddDays(1));

    Console.WriteLine("Press Enter to continue.");
    if (interactive)
        Console.ReadLine();

    Console.WriteLine("\nObject lock policies can also be added to existing
buckets.");
    await _s3ActionsWrapper.EnableObjectLockOnBucket(lockEnabledBucketName);

    Console.WriteLine("Press Enter to continue.");
    if (interactive)
        Console.ReadLine();

    // Upload some files to the buckets.
    Console.WriteLine("\nNow let's add some test files:");
    var fileName = _configuration["exampleFileName"] ?? "exampleFile.txt";
```

```
int fileCount = 2;
// Create the file if it does not already exist.
if (!File.Exists(fileName))
{
    await using StreamWriter sw = File.CreateText(fileName);
    await sw.WriteLineAsync(
        "This is a sample file for uploading to a bucket.");
}

foreach (var bucketName in bucketNames)
{
    for (int i = 0; i < fileCount; i++)
    {
        var numberedFileName = Path.GetFileNameWithoutExtension(fileName)
+ i + Path.GetExtension(fileName);
        fileNames.Add(numberedFileName);
        await _s3ActionsWrapper.UploadFileAsync(bucketName,
numberedFileName, fileName);
    }
}
Console.WriteLine("Press Enter to continue.");
if (interactive)
    Console.ReadLine();

if (!interactive)
    return true;
Console.WriteLine("\nNow we can set some object lock policies on
individual files:");
foreach (var bucketName in bucketNames)
{
    for (int i = 0; i < fileNames.Count; i++)
    {
        // No modifications to the objects in the first bucket.
        if (bucketName != bucketNames[0])
        {
            var exampleFileName = fileNames[i];
            switch (i)
            {
                case 0:
                {
                    var question =
                        $"Would you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                    if (GetYesNoResponse(question))
```

```

        {
            // Set a legal hold.
            await
            _s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
            ObjectLockLegalHoldStatus.On);

        }
        break;
    }
    case 1:
    {
        var question =
            $"\\nWould you like to add a 1 day Governance
            retention period to {exampleFileName} in {bucketName}? (y/n)" +
            "\\nReminder: Only a user with the
            s3:BypassGovernanceRetention permission will be able to delete this file or its
            bucket until the retention period has expired.";
        if (GetYesNoResponse(question))
        {
            // Set a Governance mode retention period for
            1 day.

            await
            _s3ActionsWrapper.ModifyObjectRetentionPeriod(
                bucketName, exampleFileName,
                ObjectLockRetentionMode.Governance,
                DateTime.UtcNow.AddDays(1));
        }
        break;
    }
}

}

}

}

Console.WriteLine(new string('-', 80));
return true;
}

// <summary>
/// List all of the current buckets and objects.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>The list of buckets and objects.</returns>
public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)

```

```
{
    var allObjects = new List<S3ObjectVersion>();
    foreach (var bucketName in bucketNames)
    {
        var objectsInBucket = await
        _s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
        foreach (var objectKey in objectsInBucket.Versions)
        {
            allObjects.Add(objectKey);
        }
    }

    if (interactive)
    {
        Console.WriteLine("\nCurrent buckets and objects:\n");
        int i = 0;
        foreach (var bucketObject in allObjects)
        {
            i++;
            Console.WriteLine(
                $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
        }
    }

    return allObjects;
}

/// <summary>
/// Present the user with the demo action choices.
/// </summary>
/// <returns>Async task.</returns>
public static async Task<bool> DemoActionChoices()
{
    var choices = new string[]{
        "List all files in buckets.",
        "Attempt to delete a file.",
        "Attempt to delete a file with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the object and bucket retention settings for a file.",
        "View the legal hold settings for a file.",
        "Finish the scenario."};

    var choice = 0;
```

```
// Keep asking the user until they choose to move on.
while (choice != 6)
{
    Console.WriteLine(new string('-', 80));
    choice = GetChoiceResponse(
        "\nExplore the S3 locking features by selecting one of the
following choices:"
        , choices);
    Console.WriteLine(new string('-', 80));
    switch (choice)
    {
        case 0:
        {
            await ListBucketsAndObjects(true);
            break;
        }
        case 1:
        {
            Console.WriteLine("\nEnter the number of the object to
delete:");

            var allFiles = await ListBucketsAndObjects(true);
            var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
            await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
            break;
        }
        case 2:
        {
            Console.WriteLine("\nEnter the number of the object to
delete:");

            var allFiles = await ListBucketsAndObjects(true);
            var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
            await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
            break;
        }
        case 3:
        {
            var allFiles = await ListBucketsAndObjects(true);
```

```

        Console.WriteLine("\nEnter the number of the object to
overwrite:");

        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        // Create the file if it does not already exist.
        if (!File.Exists(allFiles[fileChoice].Key))
        {
            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a
bucket.");
        }
        await
_s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
        break;
    }
    case 4:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object and
bucket to view:");

        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
        await
_s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
        break;
    }
    case 5:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object to
view:");

        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
        break;
    }
}

```

```

    }
}
return true;
}

// <summary>
/// Clean up the resources from the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Cleanup(bool interactive)
{
    Console.WriteLine(new string('-', 80));

    if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
    {
        // Remove all locks and delete all buckets and objects.
        var allFiles = await ListBucketsAndObjects(false);
        foreach (var fileInfo in allFiles)
        {
            // Check for a legal hold.
            var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
            if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
            {
                await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
            }

            // Check for a retention period.
            var retention = await
_s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);
            var hasRetentionPeriod = retention?.Mode ==
ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
DateTime.UtcNow.Date;
            await
_s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
hasRetentionPeriod, fileInfo.VersionId);
        }

        foreach (var bucketName in bucketNames)
        {

```



```

        await _s3ActionsWrapper.DeleteBucketByName(bucketName);
    }

    }
    else
    {
        Console.WriteLine(
            "Ok, we'll leave the resources intact.\n" +
            "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
        );
    }

    Console.WriteLine(new string('-', 80));
    return true;
}

/// <summary>
/// Helper method to get a yes or no response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <returns>True if the user responds with a yes.</returns>
private static bool GetYesNoResponse(string question)
{
    Console.WriteLine(question);
    var ynResponse = Console.ReadLine();
    var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
    return response;
}

/// <summary>
/// Helper method to get a choice response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <param name="choices">The choices to print on the console.</param>
/// <returns>The index of the selected choice</returns>
private static int GetChoiceResponse(string? question, string[] choices)
{
    if (question != null)
    {
        Console.WriteLine(question);
    }
}

```

```
        for (int i = 0; i < choices.Length; i++)
        {
            Console.WriteLine($"{i + 1}. {choices[i]}");
        }
    }

    var choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > choices.Length)
    {
        var choice = Console.ReadLine();
        Int32.TryParse(choice, out choiceNumber);
    }

    return choiceNumber - 1;
}
}
```

Une classe encapsuleur pour les fonctions S3.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }
}
```

```
}

/// <summary>
/// Create a new Amazon S3 bucket with object lock actions.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <param name="enableObjectLock">True to enable object lock on the
bucket.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
enableObjectLock)
{
    Console.WriteLine($"\\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true,
            ObjectLockEnabledForBucket = enableObjectLock,
        };

        var response = await _amazonS3.PutBucketAsync(request);

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Enable object lock on an existing bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket to modify.</param>
/// <returns>True if successful.</returns>
public async Task<bool> EnableObjectLockOnBucket(string bucketName)
{
    try
    {
        // First, enable Versioning on the bucket.
```

```

        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

var request = new PutObjectLockConfigurationRequest()
{
    BucketName = bucketName,
    ObjectLockConfiguration = new ObjectLockConfiguration()
    {
        ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
    },
};

var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
Console.WriteLine($"{bucketName}\tAdded an object lock policy to bucket
{bucketName}.");
return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
    return false;
}
}

/// <summary>
/// Set or modify a retention period on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="retention">The retention mode.</param>
/// <param name="retainUntilDate">The date retention expires.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
    string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)

```

```

    {
        try
        {
            var request = new PutObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                Retention = new ObjectLockRetention()
                {
                    Mode = retention,
                    RetainUntilDate = retainUntilDate
                }
            };

            var response = await _amazonS3.PutObjectRetentionAsync(request);
            Console.WriteLine($"\\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Set or modify a retention period on an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to modify.</param>
    /// <param name="retention">The retention mode.</param>
    /// <param name="retainUntilDate">The date for retention until.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
    {
        var enabledString = enableObjectLock ? "Enabled" : "Disabled";
        var timeDifference = retainUntilDate.Subtract(DateTime.Now);
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()

```

```

        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

var request = new PutObjectLockConfigurationRequest()
{
    BucketName = bucketName,
    ObjectLockConfiguration = new ObjectLockConfiguration()
    {
        ObjectLockEnabled = new ObjectLockEnabled(enabledString),
        Rule = new ObjectLockRule()
        {
            DefaultRetention = new DefaultRetention()
            {
                Mode = retention,
                Days = timeDifference.Days // Can be specified in
days or years but not both.
            }
        }
    }
};

var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
Console.WriteLine($"\\tAdded a default retention to bucket
{bucketName}.");
return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tError modifying object lock: '{ex.Message}'");
    return false;
}
}

/// <summary>
/// Get the retention period for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>

```

```

    /// <param name="objectKey">The object key.</param>
    /// <returns>The object retention details.</returns>
    public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectRetentionRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectRetentionAsync(request);
            Console.WriteLine($"\\tObject retention for {objectKey} in
{bucketName}: " +
                $"\\n\\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
            return response.Retention;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tUnable to fetch object lock retention:
'{ex.Message}'");
            return new ObjectLockRetention();
        }
    }

    /// <summary>
    /// Set or modify a legal hold on an object in an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The key of the object.</param>
    /// <param name="holdStatus">The On or Off status for the legal hold.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ModifyObjectLegalHold(string bucketName,
        string objectKey, ObjectLockLegalHoldStatus holdStatus)
    {
        try
        {
            var request = new PutObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey,

```

```

        LegalHold = new ObjectLockLegalHold()
        {
            Status = holdStatus
        }
    };

    var response = await _amazonS3.PutObjectLegalHoldAsync(request);
    Console.WriteLine($"\\tModified legal hold for {objectKey} in
{bucketName}.");
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tError modifying legal hold: '{ex.Message}'");
    return false;
}
}

/// <summary>
/// Get the legal hold details for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object legal hold details.</returns>
public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };

        var response = await _amazonS3.GetObjectLegalHoldAsync(request);
        Console.WriteLine($"\\tObject legal hold for {objectKey} in
{bucketName}: " +
            $"\\n\\tStatus: {response.LegalHold.Status}");
        return response.LegalHold;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch legal hold: '{ex.Message}'");
    }
}

```



```

        return new ObjectLockLegalHold();
    }
}

/// <summary>
/// Get the object lock configuration details for an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to get details.</param>
/// <returns>The bucket's object lock configuration details.</returns>
public async Task<ObjectLockConfiguration>
GetBucketObjectLockConfiguration(string bucketName)
{
    try
    {
        var request = new GetObjectLockConfigurationRequest()
        {
            BucketName = bucketName
        };

        var response = await
_amazonS3.GetObjectLockConfigurationAsync(request);
        Console.WriteLine($"\\tBucket object lock config for {bucketName} in
{bucketName}: " +
            $"\\n\\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
            $"\\n\\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

        return response.ObjectLockConfiguration;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}'");
        return new ObjectLockConfiguration();
    }
}

/// <summary>
/// Upload a file from the local computer to an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectName">The object to upload.</param>

```

```
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.</returns>
    public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
            ChecksumAlgorithm = ChecksumAlgorithm.SHA256
        };

        var response = await _amazonS3.PutObjectAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"\\tSuccessfully uploaded {objectName} to
{bucketName}.");
            return true;
        }
        else
        {
            Console.WriteLine($"\\tCould not upload {objectName} to
{bucketName}.");
            return false;
        }
    }

    /// <summary>
    /// List bucket objects and versions.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <returns>The list of objects and versions.</returns>
    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }
}
```

```

    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey, bool hasRetention, string? versionId = null)
    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Unable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>

```

```
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <param name="versionId">Optional versionId.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteBucketByName(string bucketName)
{
    try
    {
        var request = new DeleteBucketRequest() { BucketName = bucketName, };
        var response = await _amazonS3.DeleteBucketAsync(request);
        Console.WriteLine($"\\tDelete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
ex.Message);
        return false;
    }
}
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour .NET .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Go

Kit SDK pour Go V2

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif illustrant les fonctionnalités de verrouillage d'objet d'Amazon S3.

```
import (  
    "context"  
    "fmt"  
    "log"  
    "strings"  
  
    "s3_object_lock/actions"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"  
)  
  
// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.  
type ObjectLockScenario struct {  
    questioner demotools.IQuestioner  
    resources  Resources  
    s3Actions  *actions.S3Actions  
    sdkConfig  aws.Config  
}  
  
// NewObjectLockScenario constructs a new ObjectLockScenario instance.  
func NewObjectLockScenario(sdkConfig aws.Config, questioner  
    demotools.IQuestioner) ObjectLockScenario {  
    scenario := ObjectLockScenario{  
        questioner: questioner,
```

```

resources: Resources{},
s3Actions: &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
sdkConfig: sdkConfig,
}
scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
scenario.resources.init(scenario.s3Actions, questioner)
return scenario
}

type nameLocked struct {
    name    string
    locked  bool
}

var createInfo = []nameLocked{
    {"standard-bucket", false},
    {"lock-bucket", true},
    {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
    log.Println("Let's create some S3 buckets to use for this workflow.")
    success := false
    for !success {
        prefix := scenario.questioner.Ask(
            "This example creates three buckets. Enter a prefix to name your buckets\n(remember bucket names must be globally unique):")

        for _, info := range createInfo {
            log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
            bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
                fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
            if err != nil {
                switch err.(type) {
                case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
                    log.Printf("Couldn't create bucket %s.\n", bucketName)
                default:
                    panic(err)
                }
            }
            break
        }
        scenario.resources.demoBuckets[info.name] = &DemoBucket{
            name:    bucketName,

```

```
    objectKeys: []string{},
  }
  log.Printf("Created bucket %s.\n", bucketName)
}

if len(scenario.resources.demoBuckets) < len(createInfo) {
  scenario.resources.deleteBuckets(ctx)
} else {
  success = true
}
}

log.Println("S3 buckets created.")
log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
  log.Println("\nA bucket can be configured to use object locking.")
  scenario.questioner.Ask("Press Enter to continue.")

  var err error
  bucket := scenario.resources.demoBuckets["retention-bucket"]
  err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
  if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
      log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
    default:
      panic(err)
    }
  } else {
    log.Printf("Object locking enabled on bucket %s.", bucket.name)
  }

  log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a
// bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
context.Context) {
  log.Println("\nA bucket can be configured to use object locking with a default
retention period.")
}
```

```

bucket := scenario.resources.demoBuckets["retention-bucket"]
retentionPeriod := scenario.questioner.AskInt("Enter the default retention
period in days: ")
err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
types.ObjectLockRetentionModeGovernance)
if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
        log.Printf("Couldn't configure a default retention period on bucket %s.\n",
bucket.name)
    default:
        panic(err)
    }
} else {
    log.Printf("Default retention policy set on bucket %s with %d day retention
period.", bucket.name, retentionPeriod)
    bucket.retentionEnabled = true
}

log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
    log.Println("Uploading test objects to S3 buckets.")

    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        for i := 0; i < 2; i++ {
            key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
fmt.Sprintf("example-%d", i),
            fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
            if err != nil {
                switch err.(type) {
                case *types.NoSuchBucket:
                    log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
                default:
                    panic(err)
                }
            } else {
                log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
                bucket.objectKeys = append(bucket.objectKeys, key)
            }
        }
    }
}

```



```

    }
    }
}

scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
// objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
context.Context) {
log.Println("Now let's set object lock configurations on individual objects.")

buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
scenario.resources.demoBuckets["retention-bucket"]}
for _, bucket := range buckets {
for index, objKey := range bucket.objectKeys {
switch index {
case 0:
if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
to %s in %s (y/n)? ", objKey, bucket.name), "y") {
err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
types.ObjectLockLegalHoldStatusOn)
if err != nil {
switch err.(type) {
case *types.NoSuchKey:
log.Printf("Couldn't set legal hold on %s.\n", objKey)
default:
panic(err)
}
} else {
log.Printf("Legal hold set on %s.\n", objKey)
}
}
case 1:
q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
%s in %s?\n"+
"Reminder: Only a user with the s3:BypassGovernanceRetention permission is
able to delete this object\n"+
"or its bucket until the retention period has expired. (y/n) ", objKey,
bucket.name)
if scenario.questioner.AskBool(q, "y") {

```

```

    err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
types.ObjectLockRetentionModeGovernance, 1)
    if err != nil {
        switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
bucket.name)
            default:
                panic(err)
        }
    } else {
        log.Printf("Retention period set to 1 for %s.", objKey)
        bucket.retentionEnabled = true
    }
}
}
}
}
log.Println(strings.Repeat("-", 88))
}

const (
    ListAll = iota
    DeleteObject
    DeleteRetentionObject
    OverwriteObject
    ViewRetention
    ViewLegalHold
    Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
// object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
    log.Println("Now you can interact with the objects to explore the object lock
configurations.")
    interactiveChoices := []string{
        "List all objects and buckets.",
        "Attempt to delete an object.",
        "Attempt to delete an object with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the retention settings for an object.",
        "View the legal hold settings for an object.",
        "Finish the workflow."}

```

```
choice := ListAll
for choice != Finish {
    objList := scenario.GetAllObjects(ctx)
    objChoices := scenario.makeObjectChoiceList(objList)
    choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
interactiveChoices)
    switch choice {
    case ListAll:
        log.Println("The current objects in the example buckets are:")
        for _, objChoice := range objChoices {
            log.Println("\t", objChoice)
        }
    case DeleteObject, DeleteRetentionObject:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
delete:\n", objChoices)
        obj := objList[objChoice]
        deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
obj.versionId, choice == DeleteRetentionObject)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Println("Nothing to delete.")
            default:
                panic(err)
            }
        } else if deleted {
            log.Printf("Object %s deleted.\n", obj.key)
        }
    case OverwriteObject:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
        obj := objList[objChoice]
        _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Println("Couldn't upload to nonexistent bucket.")
            default:
                panic(err)
            }
        } else {
            log.Printf("Uploaded new content to object %s.\n", obj.key)
        }
    }
}
```

```
}
case ViewRetention:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
    obj := objList[objChoice]
    retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
obj.key)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Printf("Can't get retention configuration for %s.\n", obj.key)
        default:
            panic(err)
        }
    } else if retention != nil {
        log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
    } else {
        log.Printf("Object %s does not have object retention configured.\n", obj.key)
    }
case ViewLegalHold:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
    obj := objList[objChoice]
    legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
obj.key, obj.versionId)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
        default:
            panic(err)
        }
    } else if legalHold != nil {
        log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
    } else {
        log.Printf("Object %s does not have legal hold configured.", obj.key)
    }
case Finish:
    log.Println("Let's clean up.")
}
log.Println(strings.Repeat("-", 88))
}
}
```

```

type BucketKeyVersionId struct {
    bucket    string
    key       string
    versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
// them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
[]BucketKeyVersionId {
    var objectList []BucketKeyVersionId
    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Printf("Couldn't get object versions for %s.\n", bucket.name)
            default:
                panic(err)
            }
        } else {
            for _, version := range versions {
                objectList = append(objectList,
                    BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
                    *version.VersionId})
            }
        }
    }
    return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
// are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
[]BucketKeyVersionId) []string {
    choices := make([]string, len(bucketObjects))
    for i := 0; i < len(bucketObjects); i++ {
        choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
            bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
    }
    return choices
}

```

```

}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := scenario.questioner.(*demotools.MockQuestioner)
            if isMock || scenario.questioner.AskBool("Do you want to see the full error message (y/n)?", "y") {
                log.Println(r)
            }
            scenario.resources.Cleanup(ctx)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
    log.Println(strings.Repeat("-", 88))

    scenario.CreateBuckets(ctx)
    scenario.EnableLockOnBucket(ctx)
    scenario.SetDefaultRetentionPolicy(ctx)
    scenario.UploadTestObjects(ctx)
    scenario.SetObjectLockConfigurations(ctx)
    scenario.InteractWithObjects(ctx)

    scenario.resources.Cleanup(ctx)

    log.Println(strings.Repeat("-", 88))
    log.Println("Thanks for watching!")
    log.Println(strings.Repeat("-", 88))
}

```

Définissez une structure qui encapsule les actions S3 utilisées dans cet exemple.

```

import (
    "bytes"
    "context"
    "errors"

```

```
"fmt"
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking
// enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    }

    if enableObjectLock {
        input.ObjectLockEnabledForBucket = aws.Bool(true)
    }

    _, err := actor.S3Client.CreateBucket(ctx, input)
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", bucket)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", bucket)
        }
    }
}
```

```

    err = exists
}
} else {
    err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
    }
}

return bucket, err
}

// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
    var status *types.ObjectLockLegalHoldStatus
    input := &s3.GetObjectLegalHoldInput{
        Bucket:    aws.String(bucket),
        Key:        aws.String(key),
        VersionId: aws.String(versionId),
    }

    output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
    if err != nil {
        var noSuchKeyErr *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noSuchKeyErr) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noSuchKeyErr
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "NoSuchObjectLockConfiguration":
                log.Printf("Object %s does not have an object lock configuration.\n", key)
                err = nil
            case "InvalidRequest":
                log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
                err = nil
            }
        }
    } else {
        status = &output.LegalHold.Status
    }
}

```



```
}

return status, err
}

// GetObjectLockConfiguration retrieves the object lock configuration for an S3
// bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
string) (*types.ObjectLockConfiguration, error) {
    var lockConfig *types.ObjectLockConfiguration
    input := &s3.GetObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
    }

    output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
"ObjectLockConfigurationNotFoundError" {
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    } else {
        lockConfig = output.ObjectLockConfiguration
    }

    return lockConfig, err
}

// GetObjectRetention retrieves the object retention configuration for an S3
// object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
string) (*types.ObjectLockRetention, error) {
    var retention *types.ObjectLockRetention
    input := &s3.GetObjectRetentionInput{
        Bucket: aws.String(bucket),
```

```

    Key:    aws.String(key),
}

output, err := actor.S3Client.GetObjectRetention(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have locking enabled.", bucket)
            err = nil
        }
    }
} else {
    retention = output.Retention
}

return retention, err
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:    aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }

    _, err := actor.S3Client.PutObjectLegalHold(ctx, input)

```

```
if err != nil {
    var noKey *types.NoSuchKey
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    }
}

return err
}

// ModifyDefaultBucketRetention modifies the default retention period of an
// existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }

    _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}
```

```
// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
string) error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
            MFADelete: types.MFADeleteDisabled,
            Status:    types.BucketVersioningStatusEnabled,
        },
    }
    _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
        return err
    }

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: types.ObjectLockEnabledEnabled,
        },
    }
    _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}

// PutObjectRetention sets the object retention configuration for an S3 object.
```

```

func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
error {
input := &s3.PutObjectRetentionInput{
    Bucket: aws.String(bucket),
    Key:    aws.String(key),
    Retention: &types.ObjectLockRetention{
        Mode:          retentionMode,
        RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
    },
    BypassGovernanceRetention: aws.Bool(true),
}

_, err := actor.S3Client.PutObjectRetention(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    }
}

return err
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
string, contents string) (string, error) {
var outKey string
input := &s3.PutObjectInput{
    Bucket:          aws.String(bucket),
    Key:            aws.String(key),
    Body:           bytes.NewReader([]byte(contents)),
    ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
}
output, err := actor.S3Manager.Upload(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
}

```

```

    } else {
        err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
            Bucket: aws.String(bucket),
            Key:     aws.String(key),
        }, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
                bucket)
        } else {
            outKey = *output.Key
        }
    }
    return outKey, err
}

```

```

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
    ([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
    var versions []types.ObjectVersion
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
    for versionPaginator.HasMorePages() {
        output, err = versionPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
            break
        } else {
            versions = append(versions, output.Versions...)
        }
    }
    return versions, err
}

```

```

// DeleteObject deletes an object from a bucket.

```

```
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "AccessDenied":
                log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
                err = nil
            case "InvalidArgument":
                if bypassGovernance {
                    log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
                    err = nil
                }
            }
        }
    } else {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
        } else {
            deleted = true
        }
    }
}
```

```
    return deleted, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &types.Delete{
            Objects: objects,
            Quiet:   aws.Bool(true),
        },
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
    if err != nil || len(delOut.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucket)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
        }
        } else if len(delOut.Errors) > 0 {
            for _, outErr := range delOut.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *delOut.Errors[0].Message)
        }
    } else {
        for _, delObjs := range delOut.Deleted {
            err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
                time.Minute)
            if err != nil {

```



```
    log.Printf("Failed attempt to wait for object %s to be deleted.\n",
*del0bjs.Key)
    } else {
        log.Printf("Deleted %s.\n", *del0bjs.Key)
    }
}
}
return err
}
```

Nettoyez les ressources.

```
import (
    "context"
    "log"
    "s3_object_lock/actions"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
    name            string
    retentionEnabled bool
    objectKeys      []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
// and handles
// cleanup when the scenario finishes.
type Resources struct {
    demoBuckets map[string]*DemoBucket

    s3Actions    *actions.S3Actions
    questioner   demotools.IQuestioner
}
```

```
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
    demotools.IQuestioner) {
    resources.s3Actions = s3Actions
    resources.questioner = questioner
    resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
            log.Println("Use the AWS Management Console to remove any remaining resources"
                " +
                "that were created for this scenario.")
        }
    }()

    wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
        resources that were created "+
        "during this demo (y/n)?", "y")
    if !wantDelete {
        log.Println("Be sure to remove resources when you're done with them to avoid
            unexpected charges!")
        return
    }

    log.Println("Removing objects from S3 buckets and deleting buckets...")
    resources.deleteBuckets(ctx)
    //resources.deleteRetentionObjects(resources.retentionBucket,
    resources.retentionObjects)

    log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
    for _, info := range createInfo {
        bucket := resources.demoBuckets[info.name]
        resources.deleteObjects(ctx, bucket)
    }
}
```

```

_, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
    Bucket: aws.String(bucket.name),
})
if err != nil {
    panic(err)
}
}
for _, info := range createInfo {
    bucket := resources.demoBuckets[info.name]
    err := s3.NewBucketNotExistsWaiter(resources.s3Actions.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket.name)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
bucket.name)
    } else {
        log.Printf("Deleted %s.\n", bucket.name)
    }
}
resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket
*DemoBucket) {
    lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
bucket.name)
    if err != nil {
        panic(err)
    }
    versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("No objects to get from %s.\n", bucket.name)
        default:
            panic(err)
        }
    }
    delObjects := make([]types.ObjectIdentifier, len(versions))
    for i, version := range versions {
        if lockConfig != nil && lockConfig.ObjectLockEnabled ==
types.ObjectLockEnabledEnabled {
            status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
*version.Key, *version.VersionId)

```

```
    if err != nil {
        switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Couldn't determine legal hold status for %s in %s.\n",
*version.Key, bucket.name)
            default:
                panic(err)
        }
    } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
        err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
*version.VersionId, types.ObjectLockLegalHoldStatusOff)
        if err != nil {
            switch err.(type) {
                case *types.NoSuchKey:
                    log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
bucket.name)
                default:
                    panic(err)
            }
        }
    }
}
del0bjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.DeleteObjects(ctx, bucket.name, del0bjects,
bucket.retentionEnabled)
if err != nil {
    switch err.(type) {
        case *types.NoSuchBucket:
            log.Println("Nothing to delete.")
        default:
            panic(err)
    }
}
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour Go .
 - [GetObjectLegalHold](#)

- [GetObjectLockConfiguration](#)
- [GetObjectRetention](#)
- [PutObjectLegalHold](#)
- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif illustrant les fonctionnalités de verrouillage d'objet d'Amazon S3.

```
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import java.io.BufferedWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.util.stream.Collectors;

/*
Before running this Java V2 code example, set up your development
environment, including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html

This Java example performs the following tasks:
    1. Create test Amazon Simple Storage Service (S3) buckets with different lock
    policies.
    2. Upload sample objects to each bucket.
    3. Set some Legal Hold and Retention Periods on objects and buckets.
```

4. Investigate lock policies by viewing settings or attempting to delete or overwrite objects.
5. Clean up objects and buckets.

```
*/
```

```
public class S3ObjectLockWorkflow {

    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    static String bucketName;
    static S3LockActions s3LockActions;
    private static final List<String> bucketNames = new ArrayList<>();
    private static final List<String> fileNames = new ArrayList<>();

    public static void main(String[] args) {
        final String usage = ""
            Usage:
                <bucketName> \s

        Where:
            bucketName - The Amazon S3 bucket name.
        """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
        s3LockActions = new S3LockActions();
        bucketName = args[0];
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Simple Storage Service (S3)
Object Locking Feature Scenario.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        configurationSetup();
        System.out.println(DASHES);

        System.out.println(DASHES);
        setup();
        System.out.println("Setup is complete. Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);
    }
}
```

```
        System.out.println(DASHES);
        System.out.println("Lets present the user with choices.");
        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        demoActionChoices() ;
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Would you like to clean up the resources? (y/n)");
        String delAns = scanner.nextLine().trim();
        if (delAns.equalsIgnoreCase("y")) {
            cleanup();
            System.out.println("Clean up is complete.");
        }

        System.out.println("Press Enter to continue...");
        scanner.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Amazon S3 Object Locking Workflow is complete.");
        System.out.println(DASHES);
    }

    // Present the user with the demo action choices.
    public static void demoActionChoices() {
        String[] choices = {
            "List all files in buckets.",
            "Attempt to delete a file.",
            "Attempt to delete a file with retention period bypass.",
            "Attempt to overwrite a file.",
            "View the object and bucket retention settings for a file.",
            "View the legal hold settings for a file.",
            "Finish the workflow."
        };

        int choice = 0;
        while (true) {
            System.out.println(DASHES);
            choice = getChoiceResponse("Explore the S3 locking features by
selecting one of the following choices:", choices);
            System.out.println(DASHES);
            System.out.println("You selected "+choices[choice]);
            switch (choice) {
```

```
        case 0 -> {
            s3LockActions.listBucketsAndObjects(bucketNames, true);
        }

        case 1 -> {
            System.out.println("Enter the number of the object to
delete:");

            List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
            List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
            String[] fileKeysArray = fileKeys.toArray(new String[0]);
            int fileChoice = getChoiceResponse(null, fileKeysArray);
            String objectKey = fileKeys.get(fileChoice);
            String bucketName = allFiles.get(fileChoice).getBucketName();
            String version = allFiles.get(fileChoice).getVersion();
            s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
false, version);
        }

        case 2 -> {
            System.out.println("Enter the number of the object to
delete:");

            List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
            List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
            String[] fileKeysArray = fileKeys.toArray(new String[0]);
            int fileChoice = getChoiceResponse(null, fileKeysArray);
            String objectKey = fileKeys.get(fileChoice);
            String bucketName = allFiles.get(fileChoice).getBucketName();
            String version = allFiles.get(fileChoice).getVersion();
            s3LockActions.deleteObjectFromBucket(bucketName, objectKey,
true, version);
        }

        case 3 -> {
            System.out.println("Enter the number of the object to
overwrite:");

            List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
            List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
            String[] fileKeysArray = fileKeys.toArray(new String[0]);
```



```
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();

        // Attempt to overwrite the file.
        try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(objectKey))) {
            writer.write("This is a modified text.");

        } catch (IOException e) {
            e.printStackTrace();
        }
        s3LockActions.uploadFile(bucketName, objectKey, objectKey);
    }

    case 4 -> {
        System.out.println("Enter the number of the object to
overwrite:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectRetention(bucketName, objectKey);
    }

    case 5 -> {
        System.out.println("Enter the number of the object to
view:");

        List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, true);
        List<String> fileKeys = allFiles.stream().map(f ->
f.getKeyName()).collect(Collectors.toList());
        String[] fileKeysArray = fileKeys.toArray(new String[0]);
        int fileChoice = getChoiceResponse(null, fileKeysArray);
        String objectKey = fileKeys.get(fileChoice);
        String bucketName = allFiles.get(fileChoice).getBucketName();
        s3LockActions.getObjectLegalHold(bucketName, objectKey);
        s3LockActions.getBucketObjectLockConfiguration(bucketName);
    }
}
```

```

        case 6 -> {
            System.out.println("Exiting the workflow...");
            return;
        }

        default -> {
            System.out.println("Invalid choice. Please select again.");
        }
    }
}

// Clean up the resources from the scenario.
private static void cleanup() {
    List<S3InfoObject> allFiles =
s3LockActions.listBucketsAndObjects(bucketNames, false);
    for (S3InfoObject fileInfo : allFiles) {
        String bucketName = fileInfo.getBucketName();
        String key = fileInfo.getKeyName();
        String version = fileInfo.getVersion();
        if (bucketName.contains("lock-enabled") ||
(bucketName.contains("retention-after-creation"))) {
            ObjectLockLegalHold legalHold =
s3LockActions.getObjectLegalHold(bucketName, key);
            if (legalHold != null) {
                String holdStatus = legalHold.status().name();
                System.out.println(holdStatus);
                if (holdStatus.compareTo("ON") == 0) {
                    s3LockActions.modifyObjectLegalHold(bucketName, key,
false);
                }
            }
            // Check for a retention period.
            ObjectLockRetention retention =
s3LockActions.getObjectRetention(bucketName, key);
            boolean hasRetentionPeriod ;
            hasRetentionPeriod = retention != null;
            s3LockActions.deleteObjectFromBucket(bucketName,
key,hasRetentionPeriod, version);

        } else {
            System.out.println(bucketName +" objects do not have a legal
lock");

```

```
s3LockActions.deleteObjectFromBucket(bucketName, key, false,
version);
    }
}

// Delete the buckets.
System.out.println("Delete "+bucketName);
for (String bucket : bucketNames){
    s3LockActions.deleteBucketByName(bucket);
}
}

private static void setup() {
    Scanner scanner = new Scanner(System.in);
    System.out.println("""
        For this workflow, we will use the AWS SDK for Java to create
several S3
        buckets and files to demonstrate working with S3 locking
features.
        """);

    System.out.println("S3 buckets can be created either with or without
object lock enabled.");
    System.out.println("Press Enter to continue...");
    scanner.nextLine();

    // Create three S3 buckets.
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(0));
    s3LockActions.createBucketWithLockOptions(true, bucketNames.get(1));
    s3LockActions.createBucketWithLockOptions(false, bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Bucket "+bucketNames.get(2) +" will be configured to
use object locking with a default retention period.");
    s3LockActions.modifyBucketDefaultRetention(bucketNames.get(2));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();

    System.out.println("Object lock policies can also be added to existing
buckets. For this example, we will use "+bucketNames.get(1));
    s3LockActions.enableObjectLockOnBucket(bucketNames.get(1));
    System.out.println("Press Enter to continue.");
    scanner.nextLine();
}
```

```
// Upload some files to the buckets.
System.out.println("Now let's add some test files:");
String fileName = "exampleFile.txt";
int fileCount = 2;
try (BufferedWriter writer = new BufferedWriter(new
java.io.FileWriter(fileName))) {
    writer.write("This is a sample file for uploading to a bucket.");

} catch (IOException e) {
    e.printStackTrace();
}

for (String bucketName : bucketNames){
    for (int i = 0; i < fileCount; i++) {
        // Get the file name without extension.
        String fileNameWithoutExtension =
java.nio.file.Paths.get(fileName).getFileName().toString();
        int extensionIndex = fileNameWithoutExtension.lastIndexOf('.');
        if (extensionIndex > 0) {
            fileNameWithoutExtension =
fileNameWithoutExtension.substring(0, extensionIndex);
        }

        // Create the numbered file names.
        String numberedFileName = fileNameWithoutExtension + i +
getFileExtension(fileName);
        fileNames.add(numberedFileName);
        s3LockActions.uploadFile(bucketName, numberedFileName, fileName);
    }
}

String question = null;
System.out.print("Press Enter to continue...");
scanner.nextLine();
System.out.println("Now we can set some object lock policies on
individual files:");
for (String bucketName : bucketNames) {
    for (int i = 0; i < fileNames.size(); i++){

        // No modifications to the objects in the first bucket.
        if (!bucketName.equals(bucketNames.get(0))) {
            String exampleFileName = fileNames.get(i);
            switch (i) {
```

```

        case 0 -> {
            question = "Would you like to add a legal hold to " +
exampleFileName + " in " + bucketName + " (y/n)?";
            System.out.println(question);
            String ans = scanner.nextLine().trim();
            if (ans.equalsIgnoreCase("y")) {
                System.out.println("**** You have selected to put
a legal hold " + exampleFileName);

                // Set a legal hold.
                s3LockActions.modifyObjectLegalHold(bucketName,
exampleFileName, true);
            }
        }
        case 1 -> {
            """"
                Would you like to add a 1 day Governance
retention period to %s in %s (y/n)?
                Reminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.
                """".formatted(exampleFileName, bucketName);
                System.out.println(question);
                String ans2 = scanner.nextLine().trim();
                if (ans2.equalsIgnoreCase("y")) {

s3LockActions.modifyObjectRetentionPeriod(bucketName, exampleFileName);
                }
            }
        }
    }
}

// Get file extension.
private static String getFileExtension(String fileName) {
    int dotIndex = fileName.lastIndexOf('.');
    if (dotIndex > 0) {
        return fileName.substring(dotIndex);
    }
    return "";
}

```

```
public static void configurationSetup() {
    String noLockBucketName = bucketName + "-no-lock";
    String lockEnabledBucketName = bucketName + "-lock-enabled";
    String retentionAfterCreationBucketName = bucketName + "-retention-after-creation";
    bucketNames.add(noLockBucketName);
    bucketNames.add(lockEnabledBucketName);
    bucketNames.add(retentionAfterCreationBucketName);
}

public static int getChoiceResponse(String question, String[] choices) {
    Scanner scanner = new Scanner(System.in);
    if (question != null) {
        System.out.println(question);
        for (int i = 0; i < choices.length; i++) {
            System.out.println("\t" + (i + 1) + ". " + choices[i]);
        }
    }

    int choiceNumber = 0;
    while (choiceNumber < 1 || choiceNumber > choices.length) {
        String choice = scanner.nextLine();
        try {
            choiceNumber = Integer.parseInt(choice);
        } catch (NumberFormatException e) {
            System.out.println("Invalid choice. Please enter a valid number.");
        }
    }

    return choiceNumber - 1;
}
```

Une classe encapsuleur pour les fonctions S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
```

```
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
    software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {
```

```
private static S3Client getClient() {
    return S3Client.builder()
        .region(Region.US_EAST_1)
        .build();
}

// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time
    zone.
    ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);

    // Print the formatted date string.
    System.out.println("Formatted Date: " + humanReadableDate);
    ObjectLockRetention retention = ObjectLockRetention.builder()
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .retainUntilDate(futureInstant)
        .build();

    PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .retention(retention)
        .build();

    getClient().putObjectRetention(retentionRequest);
    System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}
```



```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
    }

    return null;
}

// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}
```

```

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
                })))
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the
updated value.
                if (interactive) {
                    System.out.println(i + ": " + s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();
    }

```

```
PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
    .bucket(bucketName)
    .versioningConfiguration(versioningConfiguration)
    .build();

getClient().putBucketVersioning(versioningRequest);
DefaultRetention rention = DefaultRetention.builder()
    .days(1)
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .build();

ObjectLockRule lockRule = ObjectLockRule.builder()
    .defaultRetention(rention)
    .build();

ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
    .objectLockEnabled(ObjectLockEnabled.ENABLED)
    .rule(lockRule)
    .build();

PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(objectLockConfiguration)
    .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
System.out.println("Added a default retention to bucket "+bucketName
+ ".");
}

// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();
```

```
        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        // Enable versioning on the bucket.
        getClient().putBucketVersioning(putBucketVersioningRequest);
        PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
                .build())
            .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on
"+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
    }
}

public void uploadFile(String bucketName, String objectName, String filePath)
{
    Path file = Paths.get(filePath);
    PutObjectRequest request = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectName)
        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
        .build();

    PutObjectResponse response = getClient().putObject(request, file);
    if (response != null) {
        System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
    } else {
        System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
    }
}
```

```
// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
}

// Delete an object from a specific bucket.
public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
    try {
        DeleteObjectRequest objectRequest;
        if (hasRetention) {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
                .bypassGovernanceRetention(true)
                .build();
        } else {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
```

```
        .build();
    }

    getClient().deleteObject(objectRequest) ;
    System.out.println("The object was successfully deleted");

} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
}
}

// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

public void deleteBucketByName(String bucketName) {
    try {
        DeleteBucketRequest request = DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        getClient().deleteBucket(request);
        System.out.println(bucketName +" was deleted.");

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
    }
}
```

```
    }  
}  
  
// Get the object lock configuration details for an S3 bucket.  
public void getBucketObjectLockConfiguration(String bucketName) {  
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =  
    GetObjectLockConfigurationRequest.builder()  
        .bucket(bucketName)  
        .build();  
  
    GetObjectLockConfigurationResponse response =  
    getClient().getObjectLockConfiguration(objectLockConfigurationRequest);  
    System.out.println("Bucket object lock config for "+bucketName+": ");  
    System.out.println("\tEnabled:  
"+response.getObjectLockConfiguration().objectLockEnabled());  
    System.out.println("\tRule: "+  
    response.getObjectLockConfiguration().rule().defaultRetention());  
}  
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Point d'entrée pour le scénario (index.js). Cela orchestre toutes les étapes. Consultez GitHub les détails de mise en œuvre de Scenario ScenarioInput, ScenarioOutput, et ScenarioAction.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "../welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  confirmSetLegalHoldFileEnabled,
  confirmSetLegalHoldFileRetention,
  confirmSetRetentionPeriodFileEnabled,
  confirmSetRetentionPeriodFileRetention,
  confirmUpdateLockPolicy,
  confirmUpdateRetention,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
  setLegalHoldFileEnabledAction,
  setLegalHoldFileRetentionAction,
  setRetentionPeriodFileEnabledAction,
  setRetentionPeriodFileRetentionAction,
  updateLockPolicy,
  updateLockPolicyAction,
  updateRetention,
  updateRetentionAction,
} from "../setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
```



```
deploy: new scenarios.Scenario(
    "S3 Object Locking - Deploy",
    [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        updateRetention(scenarios),
        confirmUpdateRetention(scenarios),
        exitOnFalse(scenarios, "confirmUpdateRetention"),
        updateRetentionAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
    ],
    initialState,
),
demo: new scenarios.Scenario(
    "S3 Object Locking - Demo",
    [loadState, replAction(scenarios, client)],
    initialState,
),
clean: new scenarios.Scenario(
    "S3 Object Locking - Destroy",
    [
        loadState,
```

```

        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
    ],
    initialState,
),
};
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
    const objectLockingScenarios = getWorkflowStages(Scenarios);
    Scenarios.parseScenarioArgs(objectLockingScenarios, {
        name: "Amazon S3 object locking workflow",
        description:
            "Work with Amazon Simple Storage Service (Amazon S3) object locking features.",
        synopsis:
            "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
    });
}

```

Envoyez des messages de bienvenue à la console (welcome.steps.js).

```

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
    new scenarios.ScenarioOutput(
        "welcome",

```

```

    "Welcome to the Amazon Simple Storage Service (S3) Object Locking Feature
    Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
    several S3 buckets and files to demonstrate working with S3 locking features.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };

```

Déployez des compartiments, des objets et des paramètres de fichier (setup.steps.js).

```

import {
  BucketVersioningStatus,
  ChecksumAlgorithm,
  CreateBucketCommand,
  MFADeleteStatus,
  PutBucketVersioningCommand,
  PutObjectCommand,
  PutObjectLockConfigurationCommand,
  PutObjectLegalHoldCommand,
  PutObjectRetentionCommand,
  ObjectLockLegalHoldStatus,
  ObjectLockRetentionMode,
  GetBucketVersioningCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

/**

```

```
* @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
*/

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(
    "bucketPrefix",
    "Provide a prefix that will be used for bucket creation.",
    { type: "input", default: "amzn-s3-demo-bucket" },
  );

/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "createBuckets",
    (state) => `The following buckets will be created:
      ${state.bucketPrefix}-no-lock with object lock False.
      ${state.bucketPrefix}-lock-enabled with object lock True.
      ${state.bucketPrefix}-retention-after-creation with object lock False.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
  new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("createBucketsAction", async (state) => {
```

```

const noLockBucketName = `${state.bucketPrefix}-no-lock`;
const lockEnabledBucketName = `${state.bucketPrefix}-lock-enabled`;
const retentionBucketName = `${state.bucketPrefix}-retention-after-creation`;

try {
  await client.send(new CreateBucketCommand({ Bucket: noLockBucketName }));
  await waitUntilBucketExists({ client }, { Bucket: noLockBucketName });
  await client.send(
    new CreateBucketCommand({
      Bucket: lockEnabledBucketName,
      ObjectLockEnabledForBucket: true,
    }),
  );
  await waitUntilBucketExists(
    { client },
    { Bucket: lockEnabledBucketName },
  );
  await client.send(
    new CreateBucketCommand({ Bucket: retentionBucketName }),
  );
  await waitUntilBucketExists({ client }, { Bucket: retentionBucketName });

  state.noLockBucketName = noLockBucketName;
  state.lockEnabledBucketName = lockEnabledBucketName;
  state.retentionBucketName = retentionBucketName;
} catch (caught) {
  if (
    caught instanceof BucketAlreadyExists ||
    caught instanceof BucketAlreadyOwnedByYou
  ) {
    console.error(`${caught.name}: ${caught.message}`);
    state.earlyExit = true;
  } else {
    throw caught;
  }
}
});

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",

```

```
(state) => `The following test files will be created:
    file0.txt in ${state.bucketPrefix}-no-lock.
    file1.txt in ${state.bucketPrefix}-no-lock.
    file0.txt in ${state.bucketPrefix}-lock-enabled.
    file1.txt in ${state.bucketPrefix}-lock-enabled.
    file0.txt in ${state.bucketPrefix}-retention-after-creation.
    file1.txt in ${state.bucketPrefix}-retention-after-creation.` ,
{ preformatted: true },
);

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file0.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
        new PutObjectCommand({
          Bucket: state.noLockBucketName,
          Key: "file1.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
      await client.send(
```

```
        new PutObjectCommand({
            Bucket: state.lockEnabledBucketName,
            Key: "file0.txt",
            Body: "Content",
            ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
    );
    await client.send(
        new PutObjectCommand({
            Bucket: state.lockEnabledBucketName,
            Key: "file1.txt",
            Body: "Content",
            ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
    );
    await client.send(
        new PutObjectCommand({
            Bucket: state.retentionBucketName,
            Key: "file0.txt",
            Body: "Content",
            ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
    );
    await client.send(
        new PutObjectCommand({
            Bucket: state.retentionBucketName,
            Key: "file1.txt",
            Body: "Content",
            ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
    );
} catch (caught) {
    if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while uploading object.  ${caught.name}:
${caught.message}`,
        );
    } else {
        throw caught;
    }
}
});

/**
```

```

    * @param {Scenarios} scenarios
    */
const updateRetention = (scenarios) =>
  new scenarios.ScenarioOutput(
    "updateRetention",
    (state) => `A bucket can be configured to use object locking with a default
  retention period.
  A default retention period will be configured for ${state.bucketPrefix}-
  retention-after-creation.` ,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmUpdateRetention",
    "Configure default retention period?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const updateRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateRetentionAction", async (state) => {
    await client.send(
      new PutBucketVersioningCommand({
        Bucket: state.retentionBucketName,
        VersioningConfiguration: {
          MFADelete: MFADeleteStatus.Disabled,
          Status: BucketVersioningStatus.Enabled,
        },
      }),
    );

    const getBucketVersioning = new GetBucketVersioningCommand({
      Bucket: state.retentionBucketName,
    });

    await retry({ intervalInMs: 500, maxRetries: 10 }, async () => {
      const { Status } = await client.send(getBucketVersioning);
    });
  });

```



```

        if (Status !== "Enabled") {
            throw new Error("Bucket versioning is not enabled.");
        }
    });

    await client.send(
        new PutObjectLockConfigurationCommand({
            Bucket: state.retentionBucketName,
            ObjectLockConfiguration: {
                ObjectLockEnabled: "Enabled",
                Rule: {
                    DefaultRetention: {
                        Mode: "GOVERNANCE",
                        Years: 1,
                    },
                },
            },
        }),
    );
});

/**
 * @param {Scenarios} scenarios
 */
const updateLockPolicy = (scenarios) =>
    new scenarios.ScenarioOutput(
        "updateLockPolicy",
        (state) => `Object lock policies can also be added to existing buckets.
An object lock policy will be added to ${state.bucketPrefix}-lock-enabled.`,
        { preformatted: true },
    );

/**
 * @param {Scenarios} scenarios
 */
const confirmUpdateLockPolicy = (scenarios) =>
    new scenarios.ScenarioInput(
        "confirmUpdateLockPolicy",
        "Add object lock policy?",
        { type: "confirm" },
    );

/**
 * @param {Scenarios} scenarios

```

```

    * @param {S3Client} client
    */
const updateLockPolicyAction = (scenarios, client) =>
  new scenarios.ScenarioAction("updateLockPolicyAction", async (state) => {
    await client.send(
      new PutObjectLockConfigurationCommand({
        Bucket: state.lockEnabledBucketName,
        ObjectLockConfiguration: {
          ObjectLockEnabled: "Enabled",
        },
      }),
    );
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileEnabled",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
${state.lockEnabledBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileEnabledAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        },
      );
    },
  );

```

```

    }},
  );
  console.log(
    `Modified legal hold for file0.txt in ${state.lockEnabledBucketName}.`,
  );
},
{ skipWhen: (state) => !state.confirmSetLegalHoldFileEnabled },
);

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetRetentionPeriodFileEnabled = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileEnabled",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
      ${state.lockEnabledBucketName}?
      Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
      able to delete this file or its bucket until the retention period has expired.`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileEnabledAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileEnabledAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.lockEnabledBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
        })
      );
    },
  );

```

```

    })),
  );
  console.log(
    `Set retention for file1.txt in ${state.lockEnabledBucketName} until
    ${retentionDate.toISOString().split("T")[0]}.`,
  );
},
{ skipWhen: (state) => !state.confirmSetRetentionPeriodFileEnabled },
);

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const confirmSetLegalHoldFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetLegalHoldFileRetention",
    (state) =>
      `Would you like to add a legal hold to file0.txt in
      ${state.retentionBucketName}?`,
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setLegalHoldFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setLegalHoldFileRetentionAction",
    async (state) => {
      await client.send(
        new PutObjectLegalHoldCommand({
          Bucket: state.retentionBucketName,
          Key: "file0.txt",
          LegalHold: {
            Status: ObjectLockLegalHoldStatus.ON,
          },
        }),
      );
      console.log(
        `Modified legal hold for file0.txt in ${state.retentionBucketName}.`,
      );
    },
  );

```

```

    );
  },
  { skipWhen: (state) => !state.confirmSetLegalHoldFileRetention },
);

/**
 * @param {Scenarios} scenarios
 */
const confirmSetRetentionPeriodFileRetention = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmSetRetentionPeriodFileRetention",
    (state) =>
      `Would you like to add a 1 day Governance retention period to file1.txt in
      ${state.retentionBucketName}?
      Reminder: Only a user with the s3:BypassGovernanceRetention permission will be
      able to delete this file or its bucket until the retention period has expired.`
    {
      type: "confirm",
    },
  );

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const setRetentionPeriodFileRetentionAction = (scenarios, client) =>
  new scenarios.ScenarioAction(
    "setRetentionPeriodFileRetentionAction",
    async (state) => {
      const retentionDate = new Date();
      retentionDate.setDate(retentionDate.getDate() + 1);
      await client.send(
        new PutObjectRetentionCommand({
          Bucket: state.retentionBucketName,
          Key: "file1.txt",
          Retention: {
            Mode: ObjectLockRetentionMode.GOVERNANCE,
            RetainUntilDate: retentionDate,
          },
          BypassGovernanceRetention: true,
        })
      );
      console.log(

```

```
        `Set retention for file1.txt in ${state.retentionBucketName} until  
        ${retentionDate.toISOString().split("T")[0]}.`,  
        );  
    },  
    { skipWhen: (state) => !state.confirmSetRetentionPeriodFileRetention },  
    );  
  
export {  
    getBucketPrefix,  
    createBuckets,  
    confirmCreateBuckets,  
    createBucketsAction,  
    populateBuckets,  
    confirmPopulateBuckets,  
    populateBucketsAction,  
    updateRetention,  
    confirmUpdateRetention,  
    updateRetentionAction,  
    updateLockPolicy,  
    confirmUpdateLockPolicy,  
    updateLockPolicyAction,  
    confirmSetLegalHoldFileEnabled,  
    setLegalHoldFileEnabledAction,  
    confirmSetRetentionPeriodFileEnabled,  
    setRetentionPeriodFileEnabledAction,  
    confirmSetLegalHoldFileRetention,  
    setLegalHoldFileRetentionAction,  
    confirmSetRetentionPeriodFileRetention,  
    setRetentionPeriodFileRetentionAction,  
};
```

Affichez et supprimez des fichiers dans les compartiments (repl.steps.js).

```
import {  
    ChecksumAlgorithm,  
    DeleteObjectCommand,  
    GetObjectLegalHoldCommand,  
    GetObjectLockConfigurationCommand,  
    GetObjectRetentionCommand,  
    ListObjectVersionsCommand,  
    PutObjectCommand,  
} from "@aws-sdk/client-s3";
```

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  DELETE_FILE: 2,
  DELETE_FILE_WITH_RETENTION: 3,
  OVERWRITE_FILE: 4,
  VIEW_RETENTION_SETTINGS: 5,
  VIEW_LEGAL_HOLD_SETTINGS: 6,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new scenarios.ScenarioInput(
    "replChoice",
    "Explore the S3 locking features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "List all files in buckets", value: choices.LIST_ALL_FILES },
        { name: "Attempt to delete a file.", value: choices.DELETE_FILE },
        {
          name: "Attempt to delete a file with retention period bypass.",
          value: choices.DELETE_FILE_WITH_RETENTION,
        },
        { name: "Attempt to overwrite a file.", value: choices.OVERWRITE_FILE },
        {
          name: "View the object and bucket retention settings for a file.",
          value: choices.VIEW_RETENTION_SETTINGS,
        },
        {
          name: "View the legal hold settings for a file.",
          value: choices.VIEW_LEGAL_HOLD_SETTINGS,
        },
      ],
    },
  ),
```

```

        { name: "Finish the workflow.", value: choices.EXIT },
    ],
},
);

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
    /** @type {{bucket: string, key: string, version: string}[]} */
    const files = [];
    for (const bucket of buckets) {
        const objectsResponse = await client.send(
            new ListObjectVersionsCommand({ Bucket: bucket }),
        );
        for (const version of objectsResponse.Versions || []) {
            const { Key, VersionId } = version;
            files.push({ bucket, key: Key, version: VersionId });
        }
    }

    return files;
};

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const replAction = (scenarios, client) =>
    new scenarios.ScenarioAction(
        "replAction",
        async (state) => {
            const files = await getAllFiles(client, [
                state.noLockBucketName,
                state.lockEnabledBucketName,
                state.retentionBucketName,
            ]);

            const fileInput = new scenarios.ScenarioInput(
                "selectedFile",
                "Select a file:",
                {
                    type: "select",

```



```

        choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${
                file.version
            })`,
            value: index,
        })),
    },
);

const { replChoice } = state;

switch (replChoice) {
    case choices.LIST_ALL_FILES: {
        const files = await getAllFiles(client, [
            state.noLockBucketName,
            state.lockEnabledBucketName,
            state.retentionBucketName,
        ]);
        state.replOutput = files
            .map(
                (file) =>
                    `${file.bucket}: ${file.key} (version: ${file.version})`,
            )
            .join("\n");
        break;
    }
    case choices.DELETE_FILE: {
        /** @type {number} */
        const fileToDelete = await fileInput.handle(state);
        const selectedFile = files[fileToDelete];
        try {
            await client.send(
                new DeleteObjectCommand({
                    Bucket: selectedFile.bucket,
                    Key: selectedFile.key,
                    VersionId: selectedFile.version,
                }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
        } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
        }
    }
}

```

```
        break;
    }
    case choices.DELETE_FILE_WITH_RETENTION: {
        /** @type {number} */
        const fileToDelete = await fileInput.handle(state);
        const selectedFile = files[fileToDelete];
        try {
            await client.send(
                new DeleteObjectCommand({
                    Bucket: selectedFile.bucket,
                    Key: selectedFile.key,
                    VersionId: selectedFile.version,
                    BypassGovernanceRetention: true,
                }),
            );
            state.replOutput = `Deleted ${selectedFile.key} in  
${selectedFile.bucket}.`;
        } catch (err) {
            state.replOutput = `Unable to delete object ${selectedFile.key} in  
bucket ${selectedFile.bucket}: ${err.message}`;
        }
        break;
    }
    case choices.OVERWRITE_FILE: {
        /** @type {number} */
        const fileToOverwrite = await fileInput.handle(state);
        const selectedFile = files[fileToOverwrite];
        try {
            await client.send(
                new PutObjectCommand({
                    Bucket: selectedFile.bucket,
                    Key: selectedFile.key,
                    Body: "New content",
                    ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
                }),
            );
            state.replOutput = `Overwrote ${selectedFile.key} in  
${selectedFile.bucket}.`;
        } catch (err) {
            state.replOutput = `Unable to overwrite object ${selectedFile.key} in  
bucket ${selectedFile.bucket}: ${err.message}`;
        }
        break;
    }
}
```

```

    case choices.VIEW_RETENTION_SETTINGS: {
      /** @type {number} */
      const fileToView = await fileInput.handle(state);
      const selectedFile = files[fileToView];
      try {
        const retention = await client.send(
          new GetObjectRetentionCommand({
            Bucket: selectedFile.bucket,
            Key: selectedFile.key,
            VersionId: selectedFile.version,
          }),
        );
        const bucketConfig = await client.send(
          new GetObjectLockConfigurationCommand({
            Bucket: selectedFile.bucket,
          }),
        );
        state.replOutput = `Object retention for ${selectedFile.key}
in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
      } catch (err) {
        state.replOutput = `Unable to fetch object lock retention:
'${err.message}'`;
      }
      break;
    }
    case choices.VIEW_LEGAL_HOLD_SETTINGS: {
      /** @type {number} */
      const fileToView = await fileInput.handle(state);
      const selectedFile = files[fileToView];
      try {
        const legalHold = await client.send(
          new GetObjectLegalHoldCommand({
            Bucket: selectedFile.bucket,
            Key: selectedFile.key,
            VersionId: selectedFile.version,
          }),
        );
        state.replOutput = `Object legal hold for ${selectedFile.key} in
${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
      }
    }
  }

```

```

        } catch (err) {
            state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
        }
        break;
    }
    default:
        throw new Error(`Invalid replChoice: ${replChoice}`);
    }
},
{
    whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
            "REPL output",
            (state) => state.replOutput,
            { preformatted: true },
        ),
    },
},
);

export { replInput, replAction, choices };

```

Détruisez toutes les ressources créées (clean.steps.js).

```

import {
    DeleteObjectCommand,
    DeleteBucketCommand,
    ListObjectVersionsCommand,
    GetObjectLegalHoldCommand,
    GetObjectRetentionCommand,
    PutObjectLegalHoldCommand,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

```

```
/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { noLockBucketName, lockEnabledBucketName, retentionBucketName } =
      state;

    const buckets = [
      noLockBucketName,
      lockEnabledBucketName,
      retentionBucketName,
    ];

    for (const bucket of buckets) {
      /** @type {import("@aws-sdk/client-s3").ListObjectVersionsCommandOutput} */
      let objectsResponse;

      try {
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
      } catch (e) {
        if (e instanceof Error && e.name === "NoSuchBucket") {
          console.log("Object's bucket has already been deleted.");
          continue;
        }
        throw e;
      }

      for (const version of objectsResponse.Versions || []) {
        const { Key, VersionId } = version;
      }
    }
  });
```

```
try {
  const legalHold = await client.send(
    new GetObjectLegalHoldCommand({
      Bucket: bucket,
      Key,
      VersionId,
    }),
  );

  if (legalHold.LegalHold?.Status === "ON") {
    await client.send(
      new PutObjectLegalHoldCommand({
        Bucket: bucket,
        Key,
        VersionId,
        LegalHold: {
          Status: "OFF",
        },
      }),
    );
  }
} catch (err) {
  console.log(
    `Unable to fetch legal hold for ${Key} in ${bucket}:` +
    `${err.message}`,
  );
}

try {
  const retention = await client.send(
    new GetObjectRetentionCommand({
      Bucket: bucket,
      Key,
      VersionId,
    }),
  );

  if (retention.Retention?.Mode === "GOVERNANCE") {
    await client.send(
      new DeleteObjectCommand({
        Bucket: bucket,
        Key,
        VersionId,
```

```
        BypassGovernanceRetention: true,
      })),
    );
  }
} catch (err) {
  console.log(
    `Unable to fetch object lock retention for ${Key} in ${bucket}:
'${err.message}'`,
  );
}

await client.send(
  new DeleteObjectCommand({
    Bucket: bucket,
    Key,
    VersionId,
  })),
);
}

await client.send(new DeleteBucketCommand({ Bucket: bucket }));
console.log(`Delete for ${bucket} complete.`);
}
});

export { confirmCleanup, cleanupAction };
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour JavaScript .
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Effectuez des demandes conditionnelles Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment ajouter des conditions préalables aux demandes Amazon S3.

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif illustrant les fonctionnalités de demande conditionnelle d'Amazon S3.

```
using Amazon.S3;
using Microsoft.Extensions.Configuration;
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;
using Microsoft.Extensions.Logging.Console;
using Microsoft.Extensions.Logging.Debug;

namespace S3ConditionalRequestsScenario;

public static class S3ConditionalRequestsScenario
{
    /*
        Before running this .NET code example, set up your development environment,
        including your credentials.

        This example demonstrates the use of conditional requests for S3 operations.
        You can use conditional requests to add preconditions to S3 read requests to
        return or copy
        an object based on its Entity tag (ETag), or last modified date.
        You can use a conditional write requests to prevent overwrites by ensuring
        there is no existing object with the same key.
    */
}
```



```
public static S3ActionsWrapper _s3ActionsWrapper = null!;  
public static IConfiguration _configuration = null!;  
public static string _resourcePrefix = null!;  
public static string _sourceBucketName = null!;  
public static string _destinationBucketName = null!;  
public static string _sampleObjectKey = null!;  
public static string _sampleObjectEtag = null!;  
public static bool _interactive = true;  
  
public static async Task Main(string[] args)  
{  
    // Set up dependency injection for the Amazon service.  
    using var host = Host.CreateDefaultBuilder(args)  
        .ConfigureLogging(logging =>  
            logging.AddFilter("System", LogLevel.Debug)  
                .AddFilter<DebugLoggerProvider>("Microsoft",  
LogLevel.Information)  
                .AddFilter<ConsoleLoggerProvider>("Microsoft",  
LogLevel.Trace))  
        .ConfigureServices((_, services) =>  
            services.AddAWSService<IAmazonS3>()  
                .AddTransient<S3ActionsWrapper>()  
        )  
        .Build();  
  
    _configuration = new ConfigurationBuilder()  
        .SetBasePath(Directory.GetCurrentDirectory())  
        .AddJsonFile("settings.json") // Load settings from .json file.  
        .AddJsonFile("settings.local.json",  
            true) // Optionally, load local settings.  
        .Build();  
  
    ServicesSetup(host);  
  
    try  
    {  
        Console.WriteLine(new string('-', 80));  
        Console.WriteLine("Welcome to the Amazon Simple Storage Service (S3)  
Conditional Requests Feature Scenario.");  
        Console.WriteLine(new string('-', 80));  
        ConfigurationSetup();  
    }  
}
```

```

        _sampleObjectEtag = await Setup(_sourceBucketName,
        _destinationBucketName, _sampleObjectKey);

        await DisplayDemoChoices(_sourceBucketName, _destinationBucketName,
        _sampleObjectKey, _sampleObjectEtag, 0);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cleaning up resources.");
        Console.WriteLine(new string('-', 80));
        await Cleanup(true);

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Amazon S3 Conditional Requests Feature Scenario is
complete.");
        Console.WriteLine(new string('-', 80));
    }
    catch (Exception ex)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"There was a problem: {ex.Message}");
        await CleanupScenario(_sourceBucketName, _destinationBucketName);
        Console.WriteLine(new string('-', 80));
    }
}

/// <summary>
/// Populate the services for use within the console application.
/// </summary>
/// <param name="host">The services host.</param>
private static void ServicesSetup(IHost host)
{
    _s3ActionsWrapper = host.Services.GetRequiredService<S3ActionsWrapper>();
}

/// <summary>
/// Any setup operations needed.
/// </summary>
public static void ConfigurationSetup()
{
    _resourcePrefix = _configuration["resourcePrefix"] ?? "dotnet-example";

    _sourceBucketName = _resourcePrefix + "-source";
    _destinationBucketName = _resourcePrefix + "-dest";
    _sampleObjectKey = _resourcePrefix + "-sample-object.txt";
}

```

```

    }

    /// <summary>
    /// Sets up the scenario by creating a source and destination bucket, and
    uploading a test file to the source bucket.
    /// </summary>
    /// <param name="sourceBucket">The name of the source bucket.</param>
    /// <param name="destBucket">The name of the destination bucket.</param>
    /// <param name="objectKey">The name of the test file to add to the source
    bucket.</param>
    /// <returns>The ETag of the uploaded test file.</returns>
    public static async Task<string> Setup(string sourceBucket, string
    destBucket, string objectKey)
    {
        Console.WriteLine(
            "\nFor this scenario, we will use the AWS SDK for .NET to create
            several S3\n" +
            "buckets and files to demonstrate working with S3 conditional
            requests.\n" +
            "This example demonstrates the use of conditional requests for S3
            operations.\r\n" +
            "You can use conditional requests to add preconditions to S3 read
            requests to return or copy\r\n" +
            "an object based on its Entity tag (ETag), or last modified date. \r
            \n" +
            "You can use a conditional write requests to prevent overwrites by
            ensuring \r\n" +
            "there is no existing object with the same key. \r\n\r\n" +
            "This example will allow you to perform conditional reads\r\n" +
            "and writes that will succeed or fail based on your selected options.
            \r\n\r\n" +
            "Sample buckets and a sample object will be created as part of the
            example.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you are ready to start.");
        if (_interactive)
            Console.ReadLine();

        await _s3ActionsWrapper.CreateBucketWithName(sourceBucket);
        await _s3ActionsWrapper.CreateBucketWithName(destBucket);

        var eTag = await _s3ActionsWrapper.PutObjectConditional(objectKey,
        sourceBucket,

```

```
        "Test file content.");

    return eTag;
}

/// <summary>
/// Cleans up the scenario by deleting the source and destination buckets.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
public static async Task CleanupScenario(string sourceBucket, string
destBucket)
{
    await _s3ActionsWrapper.CleanupBucketByName(sourceBucket);
    await _s3ActionsWrapper.CleanupBucketByName(destBucket);
}

/// <summary>
/// Displays a list of the objects in the test buckets.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
public static async Task DisplayBuckets(string sourceBucket, string
destBucket)
{
    await _s3ActionsWrapper.ListBucketContentsByName(sourceBucket);
    await _s3ActionsWrapper.ListBucketContentsByName(destBucket);
}

/// <summary>
/// Displays the menu of conditional request options for the user.
/// </summary>
/// <param name="sourceBucket">The name of the source bucket.</param>
/// <param name="destBucket">The name of the destination bucket.</param>
/// <param name="objectKey">The key of the test object in the source
bucket.</param>
/// <param name="etag">The ETag of the test object in the source bucket.</
param>
public static async Task DisplayDemoChoices(string sourceBucket, string
destBucket, string objectKey, string etag, int defaultChoice)
{
    var actions = new[]
    {
        "Print a list of bucket items.",
```

```

        "Perform a conditional read.",
        "Perform a conditional copy.",
        "Perform a conditional write.",
        "Clean up and exit."
    };

    var conditions = new[]
    {
        "If-Match: using the object's ETag. This condition should succeed.",
        "If-None-Match: using the object's ETag. This condition should
fail.",
        "If-Modified-Since: using yesterday's date. This condition should
succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
fail."
    };

    var conditionTypes = new[]
    {
        S3ConditionType.IfMatch,
        S3ConditionType.IfNoneMatch,
        S3ConditionType.IfModifiedSince,
        S3ConditionType.IfUnmodifiedSince,
    };

    var yesterdayDate = DateTime.UtcNow.AddDays(-1);

    int choice;
    while ((choice = GetChoiceResponse("\nExplore the S3 conditional request
features by selecting one of the following choices:", actions, defaultChoice)) != 4)
    {
        switch (choice)
        {
            case 0:
                Console.WriteLine("Listing the objects and buckets.");
                await DisplayBuckets(sourceBucket, destBucket);
                break;
            case 1:
                int conditionTypeIndex = GetChoiceResponse("Perform a
conditional read:", conditions, 1);
                if (conditionTypeIndex == 0 || conditionTypeIndex == 1)
                {

```

```

        await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], null, _sampleObjectEtag);
    }
    else if (conditionTypeIndex == 2 || conditionTypeIndex == 3)
    {
        await _s3ActionsWrapper.GetObjectConditional(objectKey,
sourceBucket, conditionTypes[conditionTypeIndex], yesterdayDate);
    }
    break;
case 2:
    int copyConditionTypeIndex = GetChoiceResponse("Perform a
conditional copy:", conditions, 1);
    string destKey = GetStringResponse("Enter an object key:",
"sampleObjectKey");
    if (copyConditionTypeIndex == 0 || copyConditionTypeIndex ==
1)
    {
        await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex], null,
etag);
    }
    else if (copyConditionTypeIndex == 2 ||
copyConditionTypeIndex == 3)
    {
        await _s3ActionsWrapper.CopyObjectConditional(objectKey,
destKey, sourceBucket, destBucket, conditionTypes[copyConditionTypeIndex],
yesterdayDate);
    }
    break;
case 3:
    Console.WriteLine("Perform a conditional write using
IfNoneMatch condition on the object key.");
    Console.WriteLine("If the key is a duplicate, the write will
fail.");
    string newObjectKey = GetStringResponse("Enter an object
key:", "newObjectKey");
    await _s3ActionsWrapper.PutObjectConditional(newObjectKey,
sourceBucket, "Conditional write example data.");
    break;
}

if (!_interactive)
{
    break;
}

```

```

        }
    }

    Console.WriteLine("Proceeding to cleanup.");
}

// <summary>
/// Clean up the resources from the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Cleanup(bool interactive)
{
    Console.WriteLine(new string('-', 80));

    if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
    {
        await _s3ActionsWrapper.CleanUpBucketByName(_sourceBucketName);
        await _s3ActionsWrapper.CleanUpBucketByName(_destinationBucketName);
    }
    else
    {
        Console.WriteLine(
            "Ok, we'll leave the resources intact.\n" +
            "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
        );
    }

    Console.WriteLine(new string('-', 80));
    return true;
}

/// <summary>
/// Helper method to get a yes or no response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <returns>True if the user responds with a yes.</returns>
private static bool GetYesNoResponse(string question)
{
    Console.WriteLine(question);

```

```
        var ynResponse = Console.ReadLine();
        var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
        return response;
    }

    /// <summary>
    /// Helper method to get a choice response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices, int
defaultChoice)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"{i + 1}. {choices[i]}");
            }
        }

        if (!_interactive)
            return defaultChoice;

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }

    /// <summary>
    /// Get a string response from the user.
    /// </summary>
    /// <param name="question">The question to print.</param>
```



```
    /// <param name="defaultAnswer">A default answer to use when not
    interactive.</param>
    /// <returns>The string response.</returns>
    public static string GetStringResponse(string? question, string
    defaultAnswer)
    {
        string? answer = "";
        if (_interactive)
        {
            do
            {
                Console.WriteLine(question);
                answer = Console.ReadLine();
            } while (string.IsNullOrEmpty(answer));
        }
        else
        {
            answer = defaultAnswer;
        }

        return answer;
    }
}
```

Une classe d'encapsuleur pour les fonctions S3.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Logging;

namespace S3ConditionalRequestsScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;
    private readonly ILogger<S3ActionsWrapper> _logger;
```

```
/// <summary>
/// Constructor for the S3ActionsWrapper.
/// </summary>
/// <param name="amazonS3">The injected S3 client.</param>
/// <param name="logger">The class logger.</param>
public S3ActionsWrapper(IAmazonS3 amazonS3, ILogger<S3ActionsWrapper> logger)
{
    _amazonS3 = amazonS3;
    _logger = logger;
}

/// <summary>
/// Retrieves an object from Amazon S3 with a conditional request.
/// </summary>
/// <param name="objectKey">The key of the object to retrieve.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="conditionType">The type of condition: 'IfMatch',
'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional read is successful, False otherwise.</
returns>
public async Task<bool> GetObjectConditional(string objectKey, string
sourceBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
        var getObjectRequest = new GetObjectRequest
        {
            BucketName = sourceBucket,
            Key = objectKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                getObjectRequest.EtagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                getObjectRequest.EtagToNotMatch = etagConditionalValue;
```

```
        break;
        case S3ConditionType.IfModifiedSince:
            getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
            break;
        case S3ConditionType.IfUnmodifiedSince:
            getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
            break;
        default:
            throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
    }

    var response = await _amazonS3.GetObjectAsync(getObjectRequest);
    var sampleBytes = new byte[20];
    await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
    _logger.LogInformation($"Conditional read
successful. Here are the first 20 bytes of the object:
\n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
    return true;
}
catch (AmazonS3Exception e)
{
    if (e.ErrorCode == "PreconditionFailed")
    {
        _logger.LogError("Conditional read failed: Precondition failed");
    }
    else if (e.ErrorCode == "NotModified")
    {
        _logger.LogError("Conditional read failed: Object not modified");
    }
    else
    {
        _logger.LogError($"Unexpected error: {e.ErrorCode}");
        throw;
    }
    return false;
}
}

/// <summary>
/// Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite using an IfNoneMatch condition for the object key.
```

```
/// </summary>
/// <param name="objectKey">The key of the object to upload.</param>
/// <param name="bucket">The source bucket of the object.</param>
/// <param name="content">The content to upload as a string.</param>
/// <returns>The ETag if the conditional write is successful, empty
otherwise.</returns>
public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
{
    try
    {
        var putObjectRequest = new PutObjectRequest
        {
            BucketName = bucket,
            Key = objectKey,
            ContentBody = content,
            IfNoneMatch = "*"
        };

        var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
        _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
        return putResult.ETag;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional write failed: Precondition
failed");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return string.Empty;
    }
}

/// <summary>
/// Copies an object from one Amazon S3 bucket to another with a conditional
request.
/// </summary>
```

```

    /// <param name="sourceKey">The key of the source object to copy.</param>
    /// <param name="destKey">The key of the destination object.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="destBucket">The destination bucket of the object.</param>
    /// <param name="conditionType">The type of condition to apply, e.g.
    'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
    'CopySourceIfUnmodifiedSince'.</param>
    /// <param name="conditionDateValue">The value to use for the condition for
    dates.</param>
    /// <param name="etagConditionalValue">The value to use for the condition for
    etags.</param>
    /// <returns>True if the conditional copy is successful, False otherwise.</
returns>
    public async Task<bool> CopyObjectConditional(string sourceKey, string
destKey, string sourceBucket, string destBucket,
        S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
    {
        try
        {
            var copyObjectRequest = new CopyObjectRequest
            {
                DestinationBucket = destBucket,
                DestinationKey = destKey,
                SourceBucket = sourceBucket,
                SourceKey = sourceKey
            };

            switch (conditionType)
            {
                case S3ConditionType.IfMatch:
                    copyObjectRequest.ETagToMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfNoneMatch:
                    copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                    break;
                case S3ConditionType.IfModifiedSince:
                    copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
                case S3ConditionType.IfUnmodifiedSince:
                    copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                    break;
            }
        }
    }

```

```

        default:
            throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
        }

        await _amazonS3.CopyObjectAsync(copyObjectRequest);
        _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
        return true;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional copy failed: Precondition failed");
        }
        else if (e.ErrorCode == "304")
        {
            _logger.LogError("Conditional copy failed: Object not modified");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

/// <summary>
/// Create a new Amazon S3 bucket with a specified name and check that the
bucket is ready.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateBucketWithName(string bucketName)
{
    Console.WriteLine($"\\tCreating bucket {bucketName}.");
    try
    {
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true
            }
        }
    }
}

```

```

    };

    await _amazonS3.PutBucketAsync(request);
    var bucketReady = false;
    var retries = 5;
    while (!bucketReady && retries > 0)
    {
        Thread.Sleep(5000);
        bucketReady = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_amazonS3, bucketName);
        retries--;
    }

    return bucketReady;
}
catch (BucketAlreadyExistsException ex)
{
    Console.WriteLine($"Bucket already exists: '{ex.Message}'");
    return true;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error creating bucket: '{ex.Message}'");
    return false;
}
}

/// <summary>
/// Cleans up objects and deletes the bucket by name.
/// </summary>
/// <param name="bucketName">The name of the bucket.</param>
/// <returns>Async task.</returns>
public async Task CleanupBucketByName(string bucketName)
{
    try
    {
        var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
        foreach (var obj in listObjectsResponse.S3Objects)
        {
            await _amazonS3.DeleteObjectAsync(new DeleteObjectRequest
{ BucketName = bucketName, Key = obj.Key });
        }
    }
}

```

```
        await _amazonS3.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        Console.WriteLine($"Cleaned up bucket: {bucketName}.");
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "NoSuchBucket")
        {
            Console.WriteLine($"Bucket {bucketName} does not exist, skipping
cleanup.");
        }
        else
        {
            Console.WriteLine($"Error deleting bucket: {e.ErrorCode}");
            throw;
        }
    }
}

/// <summary>
/// List the contents of the bucket with their ETag.
/// </summary>
/// <param name="bucketName">The name of the bucket.</param>
/// <returns>Async task.</returns>
public async Task<List<S3Object>> ListBucketContentsByName(string bucketName)
{
    var results = new List<S3Object>();
    try
    {
        Console.WriteLine($"\\t Items in bucket {bucketName}");
        var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
        if (listObjectsResponse.S3Objects.Count == 0)
        {
            Console.WriteLine("\\t\\tNo objects found.");
        }
        else
        {
            foreach (var obj in listObjectsResponse.S3Objects)
            {
                Console.WriteLine($"\\t\\t object: {obj.Key} ETag {obj.ETag}");
            }
        }
        results = listObjectsResponse.S3Objects;
    }
```



```
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "NoSuchBucket")
        {
            _logger.LogError($"Bucket {bucketName} does not exist.");
        }
        else
        {
            _logger.LogError($"Error listing bucket and objects:
{e.ErrorCode}");
            throw;
        }
    }

    return results;
}

/// <summary>
/// Delete an object from a specific bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey)
{
    try
    {
        var request = new DeleteObjectRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };
        await _amazonS3.DeleteObjectAsync(request);
        Console.WriteLine($"Deleted {objectKey} in {bucketName}.");
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
        return false;
    }
}
```

```

    }
}

/// <summary>
/// Delete a specific bucket by deleting the objects and then the bucket
itself.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <param name="versionId">Optional versionId.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CleanUpBucketByName(string bucketName)
{
    try
    {
        var allFiles = await ListBucketContentsByName(bucketName);

        foreach (var fileInfo in allFiles)
        {
            await DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key);
        }

        var request = new DeleteBucketRequest() { BucketName = bucketName, };
        var response = await _amazonS3.DeleteBucketAsync(request);
        Console.WriteLine($"\\tDelete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
ex.Message);
        return false;
    }
}
}

```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour .NET .
 - [CopyObject](#)

- [GetObject](#)
- [PutObject](#)

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Point d'entrée pour le flux de travail (index.js). Cela orchestre toutes les étapes. Consultez GitHub les détails de mise en œuvre de Scenario ScenarioInput, ScenarioOutput, et ScenarioAction.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "../welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
} from "../setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});
```

```
return {
  deploy: new scenarios.Scenario(
    "S3 Conditional Requests - Deploy",
    [
      welcome(scenarios),
      welcomeContinue(scenarios),
      exitOnFalse(scenarios, "welcomeContinue"),
      getBucketPrefix(scenarios),
      createBuckets(scenarios),
      confirmCreateBuckets(scenarios),
      exitOnFalse(scenarios, "confirmCreateBuckets"),
      createBucketsAction(scenarios, client),
      populateBuckets(scenarios),
      confirmPopulateBuckets(scenarios),
      exitOnFalse(scenarios, "confirmPopulateBuckets"),
      populateBucketsAction(scenarios, client),
      saveState,
    ],
    initialState,
  ),
  demo: new scenarios.Scenario(
    "S3 Conditional Requests - Demo",
    [loadState, welcome(scenarios), replAction(scenarios, client)],
    initialState,
  ),
  clean: new scenarios.Scenario(
    "S3 Conditional Requests - Destroy",
    [
      loadState,
      confirmCleanup(scenarios),
      exitOnFalse(scenarios, "confirmCleanup"),
      cleanupAction(scenarios, client),
    ],
    initialState,
  ),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";
```

```

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  const objectLockingScenarios = getWorkflowStages(Scenarios);
  Scenarios.parseScenarioArgs(objectLockingScenarios, {
    name: "Amazon S3 object locking workflow",
    description:
      "Work with Amazon Simple Storage Service (Amazon S3) object locking
features.",
    synopsis:
      "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
  });
}

```

Envoyez des messages de bienvenue à la console (welcome.steps.js).

```

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "This example demonstrates the use of conditional requests for S3
operations." +
    " You can use conditional requests to add preconditions to S3 read requests
to return " +
    "or copy an object based on its Entity tag (ETag), or last modified
date.You can use " +
    "a conditional write requests to prevent overwrites by ensuring there is no
existing " +
    "object with the same key.\n" +
    "This example will enable you to perform conditional reads and writes that
will succeed " +
    "or fail based on your selected options.\n" +
    "Sample buckets and a sample object will be created as part of the example.
\n" +
    "Some steps require a key name prefix to be defined by the user. Before you
begin, you can " +

```

```

    "optionally edit this prefix in ./object_name.json. If you do so, please
    reload the scenario before you begin.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };

```

Déployez des compartiments et des objets (setup.steps.js).

```

import {
  ChecksumAlgorithm,
  CreateBucketCommand,
  PutObjectCommand,
  BucketAlreadyExists,
  BucketAlreadyOwnedByYou,
  S3ServiceException,
  waitUntilBucketExists,
} from "@aws-sdk/client-s3";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const getBucketPrefix = (scenarios) =>
  new scenarios.ScenarioInput(

```

```

        "bucketPrefix",
        "Provide a prefix that will be used for bucket creation.",
        { type: "input", default: "amzn-s3-demo-bucket" },
    );
/**
 * @param {Scenarios} scenarios
 */
const createBuckets = (scenarios) =>
    new scenarios.ScenarioOutput(
        "createBuckets",
        (state) => `The following buckets will be created:
            ${state.bucketPrefix}-source-bucket.
            ${state.bucketPrefix}-destination-bucket.` ,
        { preformatted: true },
    );

/**
 * @param {Scenarios} scenarios
 */
const confirmCreateBuckets = (scenarios) =>
    new scenarios.ScenarioInput("confirmCreateBuckets", "Create the buckets?", {
        type: "confirm",
    });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const createBucketsAction = (scenarios, client) =>
    new scenarios.ScenarioAction("createBucketsAction", async (state) => {
        const sourceBucketName = `${state.bucketPrefix}-source-bucket`;
        const destinationBucketName = `${state.bucketPrefix}-destination-bucket`;

        try {
            await client.send(
                new CreateBucketCommand({
                    Bucket: sourceBucketName,
                }),
            );
            await waitUntilBucketExists({ client }, { Bucket: sourceBucketName });
            await client.send(
                new CreateBucketCommand({
                    Bucket: destinationBucketName,
                }),
            );

```

```

    );
    await waitUntilBucketExists(
      { client },
      { Bucket: destinationBucketName },
    );

    state.sourceBucketName = sourceBucketName;
    state.destinationBucketName = destinationBucketName;
  } catch (caught) {
    if (
      caught instanceof BucketAlreadyExists ||
      caught instanceof BucketAlreadyOwnedByYou
    ) {
      console.error(`${caught.name}: ${caught.message}`);
      state.earlyExit = true;
    } else {
      throw caught;
    }
  }
});

/**
 * @param {Scenarios} scenarios
 */
const populateBuckets = (scenarios) =>
  new scenarios.ScenarioOutput(
    "populateBuckets",
    (state) => `The following test files will be created:
      file01.txt in ${state.bucketPrefix}-source-bucket.`,
    { preformatted: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const confirmPopulateBuckets = (scenarios) =>
  new scenarios.ScenarioInput(
    "confirmPopulateBuckets",
    "Populate the buckets?",
    { type: "confirm" },
  );

/**
 * @param {Scenarios} scenarios

```



```
* @param {S3Client} client
*/
const populateBucketsAction = (scenarios, client) =>
  new scenarios.ScenarioAction("populateBucketsAction", async (state) => {
    try {
      await client.send(
        new PutObjectCommand({
          Bucket: state.sourceBucketName,
          Key: "file01.txt",
          Body: "Content",
          ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
        }),
      );
    } catch (caught) {
      if (caught instanceof S3ServiceException) {
        console.error(
          `Error from S3 while uploading object.  ${caught.name}:
${caught.message}`,
        );
      } else {
        throw caught;
      }
    }
  });

export {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
};
```

Obtenez, copiez et placez des objets à l'aide des requêtes conditionnelles S3 (repl.steps.js).

```
import path from "node:path";
import { fileURLToPath } from "node:url";
import { dirname } from "node:path";
```

```
import {
  ListObjectVersionsCommand,
  GetObjectCommand,
  CopyObjectCommand,
  PutObjectCommand,
} from "@aws-sdk/client-s3";
import data from "../object_name.json" assert { type: "json" };
import { readFile } from "node:fs/promises";
import {
  ScenarioInput,
  Scenario,
  ScenarioAction,
  ScenarioOutput,
} from "../../libs/scenario/index.js";

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

const choices = {
  EXIT: 0,
  LIST_ALL_FILES: 1,
  CONDITIONAL_READ: 2,
  CONDITIONAL_COPY: 3,
  CONDITIONAL_WRITE: 4,
};

/**
 * @param {Scenarios} scenarios
 */
const replInput = (scenarios) =>
  new ScenarioInput(
    "replChoice",
    "Explore the S3 conditional request features by selecting one of the following choices",
    {
      type: "select",
      choices: [
        { name: "Print list of bucket items.", value: choices.LIST_ALL_FILES },
        {
```

```

        name: "Perform a conditional read.",
        value: choices.CONDITIONAL_READ,
    },
    {
        name: "Perform a conditional copy. These examples use the key name
prefix defined in ./object_name.json.",
        value: choices.CONDITIONAL_COPY,
    },
    {
        name: "Perform a conditional write. This example use the sample file ./
text02.txt.",
        value: choices.CONDITIONAL_WRITE,
    },
    { name: "Finish the workflow.", value: choices.EXIT },
],
},
);

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */
const getAllFiles = async (client, buckets) => {
    /** @type {{bucket: string, key: string, version: string}[]} */
    const files = [];
    for (const bucket of buckets) {
        const objectsResponse = await client.send(
            new ListObjectVersionsCommand({ Bucket: bucket })),
        );
        for (const version of objectsResponse.Versions || []) {
            const { Key } = version;
            files.push({ bucket, key: Key });
        }
    }
    return files;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 * @param {string} key
 */
const getEtag = async (client, bucket, key) => {
    const objectsResponse = await client.send(

```

```

    new GetObjectCommand({
      Bucket: bucket,
      Key: key,
    }),
  );
  return objectsResponse.ETag;
};

/**
 * @param {S3Client} client
 * @param {string[]} buckets
 */

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
export const replAction = (scenarios, client) =>
  new ScenarioAction(
    "replAction",
    async (state) => {
      const files = await getAllFiles(client, [
        state.sourceBucketName,
        state.destinationBucketName,
      ]);

      const fileInput = new scenarios.ScenarioInput(
        "selectedFile",
        "Select a file to use:",
        {
          type: "select",
          choices: files.map((file, index) => ({
            name: `${index + 1}: ${file.bucket}: ${file.key} (Etag: ${
              file.version
            })`,
            value: index,
          })),
        },
      );

      const condReadOptions = new scenarios.ScenarioInput(
        "selectOption",
        "Which conditional read action would you like to take?",
        {
          type: "select",

```

```

        choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
        ],
    },
);
const condCopyOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional copy action would you like to take?",
    {
        type: "select",
        choices: [
            "If-Match: using the object's ETag. This condition should succeed.",
            "If-None-Match: using the object's ETag. This condition should
fail.",
            "If-Modified-Since: using yesterday's date. This condition should
succeed.",
            "If-Unmodified-Since: using yesterday's date. This condition should
fail.",
        ],
    },
);
const condWriteOptions = new scenarios.ScenarioInput(
    "selectOption",
    "Which conditional write action would you like to take?",
    {
        type: "select",
        choices: [
            "IfNoneMatch condition on the object key: If the key is a duplicate,
the write will fail.",
        ],
    },
);

const { replChoice } = state;

switch (replChoice) {
    case choices.LIST_ALL_FILES: {
        const files = await getAllFiles(client, [

```

```

        state.sourceBucketName,
        state.destinationBucketName,
    ]);
    state.replOutput = files
        .map(
            (file) => `Items in bucket ${file.bucket}: object: ${file.key} `,
        )
        .join("\n");
    break;
}
case choices.CONDITIONAL_READ:
{
    const selectedCondRead = await condReadOptions.handle(state);
    if (
        selectedCondRead ===
        "If-Match: using the object's ETag. This condition should succeed."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";
        const ETag = await getEtag(client, bucket, key);

        try {
            await client.send(
                new GetObjectCommand({
                    Bucket: bucket,
                    Key: key,
                    IfMatch: ETag,
                }),
            );
            state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because ETag provided matches the object's ETag.`;
        } catch (err) {
            state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
        }
        break;
    }
    if (
        selectedCondRead ===
        "If-None-Match: using the object's ETag. This condition should
fail."
    ) {
        const bucket = state.sourceBucketName;
        const key = "file01.txt";

```

```

    const ETag = await getEtag(client, bucket, key);

    try {
      await client.send(
        new GetObjectCommand({
          Bucket: bucket,
          Key: key,
          IfNoneMatch: ETag,
        })),
    );
    state.replOutput = `${key} in ${state.sourceBucketName} was
returned.`;
  } catch (err) {
    state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
  }
  break;
}
if (
  selectedCondRead ===
  "If-Modified-Since: using yesterday's date. This condition should
succeed."
) {
  const date = new Date();
  date.setDate(date.getDate() - 1);

  const bucket = state.sourceBucketName;
  const key = "file01.txt";
  try {
    await client.send(
      new GetObjectCommand({
        Bucket: bucket,
        Key: key,
        IfModifiedSince: date,
      })),
    );
    state.replOutput = `${key} in bucket ${state.sourceBucketName}
read because it has been created or modified in the last 24 hours.`;
  } catch (err) {
    state.replOutput = `Unable to read object ${key} in bucket
${state.sourceBucketName}: ${err.message}`;
  }
  break;
}
}

```

```

        if (
            selectedCondRead ===
            "If-Unmodified-Since: using yesterday's date. This condition should
fail."
        ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";

            const date = new Date();
            date.setDate(date.getDate() - 1);
            try {
                await client.send(
                    new GetObjectCommand({
                        Bucket: bucket,
                        Key: key,
                        IfUnmodifiedSince: date,
                    }),
                );
                state.replOutput = `${key} in ${state.sourceBucketName} was
read.`;
            } catch (err) {
                state.replOutput = `${key} in ${state.sourceBucketName} was not
read: ${err.message}`;
            }
            break;
        }
        break;
    case choices.CONDITIONAL_COPY: {
        const selectedCondCopy = await condCopyOptions.handle(state);
        if (
            selectedCondCopy ===
            "If-Match: using the object's ETag. This condition should succeed."
        ) {
            const bucket = state.sourceBucketName;
            const key = "file01.txt";
            const ETag = await getEtag(client, bucket, key);

            const copySource = `${bucket}/${key}`;
            // Optionally edit the default key name prefix of the copied object
            in ./object_name.json.
            const name = data.name;
            const copiedKey = `${name}${key}`;
            try {

```



```
        await client.send(
            new CopyObjectCommand({
                CopySource: copySource,
                Bucket: state.destinationBucketName,
                Key: copiedKey,
                CopySourceIfMatch: ETag,
            }),
        );
        state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because ETag provided matches the object's
ETag.`;
    } catch (err) {
        state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName}: ${err.message}`;
    }
    break;
}
if (
    selectedCondCopy ===
    "If-None-Match: using the object's ETag. This condition should fail."
) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    const ETag = await getEtag(client, bucket, key);
    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
    in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;

    try {
        await client.send(
            new CopyObjectCommand({
                CopySource: copySource,
                Bucket: state.destinationBucketName,
                Key: copiedKey,
                CopySourceIfNoneMatch: ETag,
            }),
        );
        state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName}`;
    } catch (err) {
        state.replOutput = `Unable to copy object as ${key} as as
${copiedKey} to bucket ${state.destinationBucketName}: ${err.message}`;
```

```
    }
    break;
  }
  if (
    selectedCondCopy ===
    "If-Modified-Since: using yesterday's date. This condition should
succeed."
  ) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;

    const date = new Date();
    date.setDate(date.getDate() - 1);

    try {
      await client.send(
        new CopyObjectCommand({
          CopySource: copySource,
          Bucket: state.destinationBucketName,
          Key: copiedKey,
          CopySourceIfModifiedSince: date,
        }),
      );
      state.replOutput = `${key} copied as ${copiedKey} to bucket
${state.destinationBucketName} because it has been created or modified in the
last 24 hours.`;
    } catch (err) {
      state.replOutput = `Unable to copy object ${key} as ${copiedKey} to
bucket ${state.destinationBucketName} : ${err.message}`;
    }
    break;
  }
  if (
    selectedCondCopy ===
    "If-Unmodified-Since: using yesterday's date. This condition should
fail."
  ) {
    const bucket = state.sourceBucketName;
    const key = "file01.txt";
```

```

    const copySource = `${bucket}/${key}`;
    // Optionally edit the default key name prefix of the copied object
    in ./object_name.json.
    const name = data.name;
    const copiedKey = `${name}${key}`;

    const date = new Date();
    date.setDate(date.getDate() - 1);

    try {
      await client.send(
        new CopyObjectCommand({
          CopySource: copySource,
          Bucket: state.destinationBucketName,
          Key: copiedKey,
          CopySourceIfUnmodifiedSince: date,
        }),
      );
      state.replOutput = `${copiedKey} copied to bucket
${state.destinationBucketName} because it has not been created or modified in
the last 24 hours.`;
    } catch (err) {
      state.replOutput = `Unable to copy object ${key} to bucket
${state.destinationBucketName}: ${err.message}`;
    }
    break;
  }
  case choices.CONDITIONAL_WRITE:
    {
      const selectedCondWrite = await condWriteOptions.handle(state);
      if (
        selectedCondWrite ===
        "IfNoneMatch condition on the object key: If the key is a
duplicate, the write will fail."
      ) {
        // Optionally edit the default key name prefix of the copied object
        in ./object_name.json.
        const key = "text02.txt";
        const __filename = fileURLToPath(import.meta.url);
        const __dirname = dirname(__filename);
        const filePath = path.join(__dirname, "text02.txt");
        try {
          await client.send(

```

```

        new PutObjectCommand({
            Bucket: `${state.destinationBucketName}`,
            Key: `${key}`,
            Body: await readFile(filePath),
            IfNoneMatch: "*",
        }),
    );
    state.replOutput = `${key} uploaded to bucket
${state.destinationBucketName} because the key is not a duplicate.`;
    } catch (err) {
        state.replOutput = `Unable to upload object to bucket
${state.destinationBucketName}:${err.message}`;
    }
    break;
}
}
break;

default:
    throw new Error(`Invalid replChoice: ${replChoice}`);
}
},
{
    whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new ScenarioOutput("REPL output", (state) => state.replOutput, {
            preformatted: true,
        }),
    },
},
);

export { replInput, choices };

```

Détruisez toutes les ressources créées (clean.steps.js).

```

import {
    DeleteObjectCommand,
    DeleteBucketCommand,
    ListObjectVersionsCommand,
} from "@aws-sdk/client-s3";

```

```
/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @typedef {import("@aws-sdk/client-s3").S3Client} S3Client
 */

/**
 * @param {Scenarios} scenarios
 */
const confirmCleanup = (scenarios) =>
  new scenarios.ScenarioInput("confirmCleanup", "Clean up resources?", {
    type: "confirm",
  });

/**
 * @param {Scenarios} scenarios
 * @param {S3Client} client
 */
const cleanupAction = (scenarios, client) =>
  new scenarios.ScenarioAction("cleanupAction", async (state) => {
    const { sourceBucketName, destinationBucketName } = state;
    const buckets = [sourceBucketName, destinationBucketName].filter((b) => b);

    for (const bucket of buckets) {
      try {
        let objectsResponse;
        objectsResponse = await client.send(
          new ListObjectVersionsCommand({
            Bucket: bucket,
          }),
        );
        for (const version of objectsResponse.Versions || []) {
          const { Key, VersionId } = version;
          try {
            await client.send(
              new DeleteObjectCommand({
                Bucket: bucket,
                Key,
                VersionId,
              }),
            );
          }
        }
      }
    }
  });
```

```

        } catch (err) {
            console.log(`An error occurred: ${err.message} `);
        }
    }
} catch (e) {
    if (e instanceof Error && e.name === "NoSuchBucket") {
        console.log("Objects and buckets have already been deleted.");
        continue;
    }
    throw e;
}

await client.send(new DeleteBucketCommand({ Bucket: bucket }));
console.log(`Delete for ${bucket} complete.`);
}
});

export { confirmCleanup, cleanupAction };

```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour JavaScript .
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif illustrant les demandes conditionnelles d'Amazon S3.

```

"""
Purpose

```

Shows how to use AWS SDK for Python (Boto3) to get started using conditional requests for Amazon Simple Storage Service (Amazon S3).

```
"""

import logging
import random
import sys
import datetime

import boto3
from botocore.exceptions import ClientError

from s3_conditional_requests import S3ConditionalRequests

# Add relative path to include demo_tools in this code example without need for
# setup.
sys.path.append(".././.././../")
import demo_tools.question as q # noqa

# Constants
FILE_CONTENT = "This is a test file for S3 conditional requests."
RANDOM_SUFFIX = str(random.randint(100, 999))

logger = logging.getLogger(__name__)

class ConditionalRequestsScenario:
    """Runs a scenario that shows how to use S3 Conditional Requests."""

    def __init__(self, conditional_requests, s3_client):
        """
        :param conditional_requests: An object that wraps S3 conditional request
        actions.
        :param s3_client: A Boto3 S3 client for setup and cleanup operations.
        """
        self.conditional_requests = conditional_requests
        self.s3_client = s3_client

    def setup_scenario(self, source_bucket: str, dest_bucket: str, object_key:
str):
        """
```

```

Sets up the scenario by creating a source and destination bucket.
Prompts the user to provide a bucket name prefix.

:param source_bucket: The name of the source bucket.
:param dest_bucket: The name of the destination bucket.
:param object_key: The name of a test file to add to the source bucket.
"""

# Create the buckets.
try:
    self.s3_client.create_bucket(Bucket=source_bucket)
    self.s3_client.create_bucket(Bucket=dest_bucket)
    print(
        f"Created source bucket: {source_bucket} and destination bucket:
{dest_bucket}"
    )
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    logger.error(f"Error creating buckets: {error_code}")
    raise

# Upload test file into the source bucket.
try:
    print(f"Uploading file {object_key} to bucket {source_bucket}")
    response = self.s3_client.put_object(
        Bucket=source_bucket, Key=object_key, Body=FILE_CONTENT
    )
    object_etag = response["ETag"]
    return object_etag

except Exception as e:
    logger.error(
        f"Failed to upload file {object_key} to bucket {source_bucket}:
{e}"
    )

def cleanup_scenario(self, source_bucket: str, dest_bucket: str):
    """
    Cleans up the scenario by deleting the source and destination buckets.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    """

```



```

        self.cleanup_bucket(source_bucket)
        self.cleanup_bucket(dest_bucket)

def cleanup_bucket(self, bucket_name: str):
    """
    Cleans up the bucket by deleting all objects and then the bucket itself.

    :param bucket_name: The name of the bucket.
    """
    try:
        # Get list of all objects in the bucket.
        list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
        objs = list_response.get("Contents", [])
        for obj in objs:
            key = obj["Key"]
            self.s3_client.delete_object(Bucket=bucket_name, Key=key)
        self.s3_client.delete_bucket(Bucket=bucket_name)
        print(f"Cleaned up bucket: {bucket_name}.")
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "NoSuchBucket":
            logger.info(f"Bucket {bucket_name} does not exist, skipping
cleanup.")
        else:
            logger.error(f"Error deleting bucket: {error_code}")
            raise

def display_buckets(self, source_bucket: str, dest_bucket: str):
    """
    Display a list of the objects in the test buckets.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    """
    self.list_bucket_contents(source_bucket)
    self.list_bucket_contents(dest_bucket)

def list_bucket_contents(self, bucket_name):
    """
    Display a list of the objects in the bucket.

    :param bucket_name: The name of the bucket.
    """

```

```
try:
    # Get list of all objects in the bucket.
    print(f"\t Items in bucket {bucket_name}")
    list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
    objs = list_response.get("Contents", [])
    if not objs:
        print("\t\tNo objects found.")
    for obj in objs:
        key = obj["Key"]
        print(f"\t\t object: {key} ETag {obj['ETag']}")
    return objs
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    if error_code == "NoSuchBucket":
        logger.info(f"Bucket {bucket_name} does not exist.")
    else:
        logger.error(f"Error listing bucket and objects: {error_code}")
        raise

def display_menu(
    self, source_bucket: str, dest_bucket: str, object_key: str, etag: str
):
    """
    Displays the menu of conditional request options for the user.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    :param object_key: The key of the test object in the source bucket.
    :param etag: The etag of the test object in the source bucket.
    """

    actions = [
        "Print list of bucket items.",
        "Perform a conditional read.",
        "Perform a conditional copy.",
        "Perform a conditional write.",
        "Clean up and exit.",
    ]

    conditions = [
        "If-Match: using the object's ETag. This condition should succeed.",
        "If-None-Match: using the object's ETag. This condition should
fail.",
```

```

        "If-Modified-Since: using yesterday's date. This condition should
        succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should
        fail.",
    ]

    condition_types = [
        "IfMatch",
        "IfNoneMatch",
        "IfModifiedSince",
        "IfUnmodifiedSince",
    ]
    copy_condition_types = [
        "CopySourceIfMatch",
        "CopySourceIfNoneMatch",
        "CopySourceIfModifiedSince",
        "CopySourceIfUnmodifiedSince",
    ]

    yesterday_date = datetime.datetime.utcnow() - datetime.timedelta(days=1)

    choice = 0
    while choice != 4:
        print("-" * 88)
        print("Choose an action to explore some example conditional
requests.")
        choice = q.choose("Which action would you like to take? ", actions)
        if choice == 0:
            print("Listing the objects and buckets.")
            self.display_buckets(source_bucket, dest_bucket)
        elif choice == 1:
            print("Perform a conditional read.")
            condition_type = q.choose("Enter the condition type : ",
conditions)
            if condition_type == 0 or condition_type == 1:
                self.conditional_requests.get_object_conditional(
                    object_key, source_bucket,
condition_types[condition_type], etag
                )
            elif condition_type == 2 or condition_type == 3:
                self.conditional_requests.get_object_conditional(
                    object_key,
                    source_bucket,
                    condition_types[condition_type],

```

```

        yesterday_date,
    )
    elif choice == 2:
        print("Perform a conditional copy.")
        condition_type = q.choose("Enter the condition type : ",
conditions)
        dest_key = q.ask("Enter an object key: ", q.non_empty)
        if condition_type == 0 or condition_type == 1:
            self.conditional_requests.copy_object_conditional(
                object_key,
                dest_key,
                source_bucket,
                dest_bucket,
                copy_condition_types[condition_type],
                etag,
            )
        elif condition_type == 2 or condition_type == 3:
            self.conditional_requests.copy_object_conditional(
                object_key,
                dest_key,
                copy_condition_types[condition_type],
                yesterday_date,
            )
    elif choice == 3:
        print(
            "Perform a conditional write using IfNoneMatch condition on
the object key."
        )
        print("If the key is a duplicate, the write will fail.")
        object_key = q.ask("Enter an object key: ", q.non_empty)
        self.conditional_requests.put_object_conditional(
            object_key, source_bucket, b"Conditional write example data."
        )
    elif choice == 4:
        print("Proceeding to cleanup.")

def run_scenario(self):
    """
    Runs the interactive scenario.
    """
    print("-" * 88)
    print("Welcome to the Amazon S3 conditional requests example.")
    print("-" * 88)

```

```

    print(
        f"""\
    This example demonstrates the use of conditional requests for S3
operations.
    You can use conditional requests to add preconditions to S3 read requests
to return or copy
    an object based on its Entity tag (ETag), or last modified date.
    You can use a conditional write requests to prevent overwrites by
ensuring
    there is no existing object with the same key.

    This example will allow you to perform conditional reads
and writes that will succeed or fail based on your selected options.

    Sample buckets and a sample object will be created as part of the
example.
    """
    )

    bucket_prefix = q.ask("Enter a bucket name prefix: ", q.non_empty)
    source_bucket_name = f"{bucket_prefix}-source-{RANDOM_SUFFIX}"
    dest_bucket_name = f"{bucket_prefix}-dest-{RANDOM_SUFFIX}"
    object_key = "test-upload-file.txt"

    try:
        etag = self.setup_scenario(source_bucket_name, dest_bucket_name,
object_key)
        self.display_menu(source_bucket_name, dest_bucket_name, object_key,
etag)
    finally:
        self.cleanup_scenario(source_bucket_name, dest_bucket_name)

    print("-" * 88)
    print("Thanks for watching.")
    print("-" * 88)

if __name__ == "__main__":
    scenario = ConditionalRequestsScenario(
        S3ConditionalRequests.from_client(), boto3.client("s3")
    )
    scenario.run_scenario()

```

Une classe d'encapsuleur qui définit les opérations de demande conditionnelle.

```
import boto3
import logging

from botocore.exceptions import ClientError

# Configure logging
logger = logging.getLogger(__name__)

class S3ConditionalRequests:
    """Encapsulates S3 conditional request operations."""

    def __init__(self, s3_client):
        self.s3 = s3_client

    @classmethod
    def from_client(cls):
        """
        Instantiates this class from a Boto3 client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def get_object_conditional(
        self,
        object_key: str,
        source_bucket: str,
        condition_type: str,
        condition_value: str,
    ):
        """
        Retrieves an object from Amazon S3 with a conditional request.

        :param object_key: The key of the object to retrieve.
        :param source_bucket: The source bucket of the object.

```

```

        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
        'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
    try:
        response = self.s3.get_object(
            Bucket=source_bucket,
            Key=object_key,
            **{condition_type: condition_value},
        )
        sample_bytes = response["Body"].read(20)
        print(
            f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
        )
        print(f"\t{sample_bytes}")
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional read failed: Precondition failed")
        elif error_code == "304": # Not modified error code.
            print("\tConditional read failed: Object not modified")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise

    def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
    try:
        self.s3.put_object(
            Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch=""
        )
    print(

```

```

        f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
    )
except ClientError as e:
    error_code = e.response["Error"]["Code"]
    if error_code == "PreconditionFailed":
        print("\tConditional write failed: Precondition failed")
    else:
        logger.error(f"Unexpected error: {error_code}")
        raise

def copy_object_conditional(
    self,
    source_key: str,
    dest_key: str,
    source_bucket: str,
    dest_bucket: str,
    condition_type: str,
    condition_value: str,
):
    """
    Copies an object from one Amazon S3 bucket to another with a conditional
    request.

    :param source_key: The key of the source object to copy.
    :param dest_key: The key of the destination object.
    :param source_bucket: The source bucket of the object.
    :param dest_bucket: The destination bucket of the object.
    :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch',
        'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
    :param condition_value: The value to use for the condition.
    """
    try:
        self.s3.copy_object(
            Bucket=dest_bucket,
            Key=dest_key,
            CopySource={"Bucket": source_bucket, "Key": source_key},
            **{condition_type: condition_value},
        )
        print(
            f"\tConditional copy successful for key {dest_key} in bucket
{dest_bucket}."

```



```
)  
except ClientError as e:  
    error_code = e.response["Error"]["Code"]  
    if error_code == "PreconditionFailed":  
        print("\tConditional copy failed: Precondition failed")  
    elif error_code == "304": # Not modified error code.  
        print("\tConditional copy failed: Object not modified")  
    else:  
        logger.error(f"Unexpected error: {error_code}")  
        raise
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Python (Boto3).
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Gérez les listes de contrôle d'accès (ACL) pour les compartiments Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment gérer les listes de contrôle d'accès (ACL) pour les compartiments Amazon S3.

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
/// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
/// access.
/// </summary>
public class ManageACLs
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket1";
        string newBucketName = "amzn-s3-demo-bucket2";
        string keyName = "sample-object.txt";
        string emailAddress = "someone@example.com";

        // If the AWS Region where your bucket is located is different from
        // the Region defined for the default user, pass the Amazon S3
bucket's
        // name to the client constructor. It should look like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
        IAmazonS3 client = new AmazonS3Client();

        await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
    }

    /// <summary>
    /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
    /// information and then adds a new ACL to one of the objects in the
    /// Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// methods to create a bucket, get an ACL, and add a different ACL to
    /// one of the objects.</param>
    /// <param name="bucketName">A string representing the original Amazon S3
    /// bucket name.</param>

```

S3

be

```

    /// <param name="newBucketName">A string representing the name of the
    /// new bucket that will be created.</param>
    /// <param name="keyName">A string representing the key name of an Amazon
    /// object for which we will change the ACL.</param>
    /// <param name="emailAddress">A string representing the email address
    /// belonging to the person to whom access to the Amazon S3 bucket will
    /// be granted.</param>
    public static async Task TestBucketObjectACLsAsync(
        IAmazonS3 client,
        string bucketName,
        string newBucketName,
        string keyName,
        string emailAddress)
    {
        try
        {
            // Create a new Amazon S3 bucket and specify canned ACL.
            var success = await CreateBucketWithCannedACLAsync(client,
newBucketName);

            // Get the ACL on a bucket.
            await GetBucketACLAsync(client, bucketName);

            // Add (replace) the ACL on an object in a bucket.
            await AddACLToExistingObjectAsync(client, bucketName, keyName,
emailAddress);
        }
        catch (AmazonS3Exception amazonS3Exception)
        {
            Console.WriteLine($"Exception: {amazonS3Exception.Message}");
        }
    }

    /// <summary>
    /// Creates a new Amazon S3 bucket with a canned ACL attached.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="newBucketName">A string representing the name of the
    /// new Amazon S3 bucket.</param>
    /// <returns>Returns a boolean value indicating success or failure.</
returns>

```

```
public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
client, string newBucketName)
{
    var request = new PutBucketRequest()
    {
        BucketName = newBucketName,
        BucketRegion = S3Region.EUWest1,

        // Add a canned ACL.
        CannedACL = S3CannedACL.LogDeliveryWrite,
    };

    var response = await client.PutBucketAsync(request);
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}

/// <summary>
/// Retrieves the ACL associated with the Amazon S3 bucket name in the
/// bucketName parameter.
/// </summary>
/// <param name="client">The initialized client object used to call
/// PutBucketAsync.</param>
/// <param name="bucketName">The Amazon S3 bucket for which we want to
get the
/// ACL list.</param>
/// <returns>Returns an S3AccessControlList returned from the call to
/// GetACLAsync.</returns>
public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
client, string bucketName)
{
    GetACLResponse response = await client.GetACLAsync(new GetACLRequest
    {
        BucketName = bucketName,
    });

    return response.AccessControlList;
}

/// <summary>
/// Adds a new ACL to an existing object in the Amazon S3 bucket.
/// </summary>
```

```
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="bucketName">A string representing the name of the Amazon
S3    /// bucket containing the object to which we want to apply a new ACL.</
param>    /// <param name="keyName">A string representing the name of the object
    /// to which we want to apply the new ACL.</param>
    /// <param name="emailAddress">The email address of the person to whom
    /// we will be applying to whom access will be granted.</param>
    public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
string    string bucketName, string keyName, string emailAddress)
    {
        // Retrieve the ACL for an object.
        GetACLResponse aclResponse = await client.GetACLAsync(new
GetACLRequest    {
            BucketName = bucketName,
            Key = keyName,
        });

        S3AccessControlList acl = aclResponse.AccessControlList;

        // Retrieve the owner.
        Owner owner = acl.Owner;

        // Clear existing grants.
        acl.Grants.Clear();

        // Add a grant to reset the owner's full permission
        // (the previous clear statement removed all permissions).
        var fullControlGrant = new S3Grant
        {
            Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
        };
        acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

        // Specify email to identify grantee for granting permissions.
        var grantUsingEmail = new S3Grant
        {
            Grantee = new S3Grantee { EmailAddress = emailAddress },
            Permission = S3Permission.WRITE_ACP,
        };
    }
}
```

```
// Specify log delivery group as grantee.
var grantLogDeliveryGroup = new S3Grant
{
    Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
    Permission = S3Permission.WRITE,
};

// Create a new ACL.
var newAcl = new S3AccessControlList
{
    Grants = new List<S3Grant> { grantUsingEmail,
grantLogDeliveryGroup },
    Owner = owner,
};

// Set the new ACL. We're throwing away the response here.
_ = await client.PutACLAsync(new PutACLRequest
{
    BucketName = bucketName,
    Key = keyName,
    AccessControlList = newAcl,
});
}

}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour .NET .
 - [GetBucketAcl](#)
 - [GetObjectAcl](#)
 - [PutBucketAcl](#)
 - [PutObjectAcl](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Gérez les messages Amazon SQS volumineux à l'aide d'Amazon S3 avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser la bibliothèque client étendue Amazon SQS pour travailler avec des messages Amazon SQS volumineux.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.joda.time.DateTime;
import org.joda.time.format.DateTimeFormat;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketLifecycleConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.ExpirationStatus;
import software.amazon.awssdk.services.s3.model.LifecycleExpiration;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.LifecycleRuleFilter;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationRequest;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.CreateQueueResponse;
import software.amazon.awssdk.services.sqs.model.DeleteMessageRequest;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
```

```
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageResponse;
import software.amazon.awssdk.services.sqs.model.SendMessageRequest;

import java.util.Arrays;
import java.util.List;
import java.util.UUID;

/**
 * Example of using Amazon SQS Extended Client Library for Java 2.x.
 */
public class SqsExtendedClientExample {
    private static final Logger logger =
        LoggerFactory.getLogger(SqsExtendedClientExample.class);

    private String s3BucketName;
    private String queueUrl;
    private final String queueName;
    private final S3Client s3Client;
    private final SqsClient sqsExtendedClient;
    private final int messageSize;

    /**
     * Constructor with default clients and message size.
     */
    public SqsExtendedClientExample() {
        this(S3Client.create(), 300000);
    }

    /**
     * Constructor with custom S3 client and message size.
     *
     * @param s3Client The S3 client to use
     * @param messageSize The size of the test message to create
     */
    public SqsExtendedClientExample(S3Client s3Client, int messageSize) {
        this.s3Client = s3Client;
        this.messageSize = messageSize;

        // Generate a unique bucket name.
        this.s3BucketName = UUID.randomUUID() + "-" +
            DateTimeFormat.forPattern("yyMMdd-hhmmss").print(new DateTime());
    }
}
```



```
// Generate a unique queue name.
this.queueName = "MyQueue-" + UUID.randomUUID();

// Configure the SQS extended client.
final ExtendedClientConfiguration extendedClientConfig = new
ExtendedClientConfiguration()
    .withPayloadSupportEnabled(s3Client, s3BucketName);

this.sqsExtendedClient = new
AmazonSQSExtendedClient(SqsClient.builder().build(), extendedClientConfig);
}

public static void main(String[] args) {
    SqsExtendedClientExample example = new SqsExtendedClientExample();
    try {
        example.setup();
        example.sendAndReceiveMessage();
    } finally {
        example.cleanup();
    }
}

/**
 * Send a large message and receive it back.
 *
 * @return The received message
 */
public Message sendAndReceiveMessage() {
    try {
        // Create a large message.
        char[] chars = new char[messageSize];
        Arrays.fill(chars, 'x');
        String largeMessage = new String(chars);

        // Send the message.
        final SendMessageRequest sendMessageRequest =
SendMessageRequest.builder()
            .queueUrl(queueUrl)
            .messageBody(largeMessage)
            .build();

        sqsExtendedClient.sendMessage(sendMessageRequest);
        logger.info("Sent message of size: {}", largeMessage.length());
    }
}
```

```
// Receive and return the message.
final ReceiveMessageResponse receiveMessageResponse =
sqsExtendedClient.receiveMessage(
    ReceiveMessageRequest.builder().queueUrl(queueUrl).build());

List<Message> messages = receiveMessageResponse.messages();
if (messages.isEmpty()) {
    throw new RuntimeException("No messages received");
}

Message message = messages.getFirst();
logger.info("\nMessage received.");
logger.info("  ID: {}", message.messageId());
logger.info("  Receipt handle: {}", message.receiptHandle());
logger.info("  Message body size: {}", message.body().length());
logger.info("  Message body (first 5 characters): {}",
message.body().substring(0, 5));

    return message;
} catch (RuntimeException e) {
    logger.error("Error during message processing: {}", e.getMessage(),
e);
    throw e;
}
}
```

- Pour plus d'informations, consultez le [Guide du développeur AWS SDK for Java 2.x](#).
- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .
 - [CreateBucket](#)
 - [PutBucketLifecycleConfiguration](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Gérez des objets Amazon S3 versionnés par lots à l'aide d'une fonction Lambda à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment gérer des objets S3 soumis au contrôle de version par lots avec une fonction Lambda.

Python

Kit SDK for Python (Boto3)

Montre comment manipuler des objets versionnés par Amazon Simple Storage Service (Amazon S3) par lots en créant des tâches qui AWS Lambda appellent des fonctions pour effectuer le traitement. Cet exemple montre comment créer un compartiment compatible avec les versions et télécharge les strophes du poème Vous êtes vieux, Père Guillaume de Lewis Carroll. Il utilise également des tâches par lots Amazon S3 pour modifier le poème de différentes manières.

Découvrez comment :

- Créez des fonctions Lambda qui fonctionnent sur des objets soumis au contrôle de version.
- Créez un manifeste des objets à mettre à jour.
- Créez des tâches par lots qui appellent des fonctions Lambda pour mettre à jour les objets.
- Supprimez les fonctions Lambda.
- Videz et supprimez un compartiment soumis au contrôle de version.

Il est préférable de visionner cet exemple sur GitHub. Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Amazon S3

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Analysez les URI Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment analyser des URI Amazon S3 pour extraire des composants importants tels que le nom du compartiment et la clé d'objet.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Analysez un URI Amazon S3 à l'aide de la classe [S3Uri](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
import java.util.List;
import java.util.Map;

/**
 *
 * @param s3Client      - An S3Client through which you acquire an S3Uri
instance.
 * @param s3objectUrl - A complex URL (String) that is used to demonstrate
S3Uri
 *
                      capabilities.
 */
public static void parseS3UriExample(S3Client s3Client, String s3objectUrl) {
    logger.info(s3objectUrl);
    // Console output:
    // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

    // Create an S3Utilities object using the configuration of the s3Client.
    S3Utilities s3Utilities = s3Client.utilities();

    // From a String URL create a URI object to pass to the parseUri()
method.
    URI uri = URI.create(s3objectUrl);
```

```
S3Uri s3Uri = s3Utilities.parseUri(uri);

// If the URI contains no value for the Region, bucket or key, the SDK
returns
// an empty Optional.
// The SDK returns decoded URI values.

Region region = s3Uri.region().orElse(null);
log("region", region);
// Console output: 'region: us-west-1'.

String bucket = s3Uri.bucket().orElse(null);
log("bucket", bucket);
// Console output: 'bucket: myBucket'.

String key = s3Uri.key().orElse(null);
log("key", key);
// Console output: 'key: resources/doc.txt'.

Boolean isPathStyle = s3Uri.isPathStyle();
log("isPathStyle", isPathStyle);
// Console output: 'isPathStyle: true'.

// If the URI contains no query parameters, the SDK returns an empty map.
Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
log("rawQueryParameters", queryParams);
// Console output: 'rawQueryParameters: {versionId=[abc123],
partNumber=[77,
// 88]]'.

// Retrieve the first or all values for a query parameter as shown in the
// following code.
String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
log("firstMatchingRawQueryParameter-versionId", versionId);
// Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
log("firstMatchingRawQueryParameter-partNumber", partNumber);
// Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
```

```

        log("firstMatchingRawQueryParameter", partNumbers);
        // Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

        /**
         * Object keys and query parameters with reserved or unsafe characters,
        must be
         * URL-encoded.
         * For example replace whitespace " " with "%20".
         * Valid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
        %5Bbrackets%5D"
         * Invalid:
         * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
        query=[brackets]"
         *
         * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
        the dot
         * must not be URL-encoded.
         * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
         * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
         */
    }

    private static void log(String s3UriElement, Object element) {
        if (element == null) {
            logger.info("{}: {}", s3UriElement, "null");
        } else {
            logger.info("{}: {}", s3UriElement, element);
        }
    }
}

```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Effectuez une copie en plusieurs parties d'un objet Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment effectuer une copie en plusieurs parties d'un objet Amazon S3.

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to perform a multi-part copy from one Amazon
/// Simple Storage Service (Amazon S3) bucket to another.
/// </summary>
public class MPUApiCopyObj
{
    private const string SourceBucket = "amzn-s3-demo-bucket1";
    private const string TargetBucket = "amzn-s3-demo-bucket2";
    private const string SourceObjectKey = "example.mov";
    private const string TargetObjectKey = "copied_video_file.mov";

    /// <summary>
    /// This method starts the multi-part upload.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        Console.WriteLine("Copying object...");
        await MPUCopyObjectAsync(s3Client);
    }

    /// <summary>
    /// This method uses the passed client object to perform a multipart
    /// copy operation.
    /// </summary>
    /// <param name="client">An Amazon S3 client object that will be used
```

```
/// to perform the copy.</param>
public static async Task MPUCopyObjectAsync(AmazonS3Client client)
{
    // Create a list to store the copy part responses.
    var copyResponses = new List<CopyPartResponse>();

    // Setup information required to initiate the multipart upload.
    var initiateRequest = new InitiateMultipartUploadRequest
    {
        BucketName = TargetBucket,
        Key = TargetObjectKey,
    };

    // Initiate the upload.
    InitiateMultipartUploadResponse initResponse =
        await client.InitiateMultipartUploadAsync(initiateRequest);

    // Save the upload ID.
    string uploadId = initResponse.UploadId;

    try
    {
        // Get the size of the object.
        var metadataRequest = new GetObjectMetadataRequest
        {
            BucketName = SourceBucket,
            Key = SourceObjectKey,
        };

        GetObjectMetadataResponse metadataResponse =
            await client.GetObjectMetadataAsync(metadataRequest);
        var objectSize = metadataResponse.ContentLength; // Length in
bytes.

        // Copy the parts.
        var partSize = 5 * (long)Math.Pow(2, 20); // Part size is 5 MB.

        long bytePosition = 0;
        for (int i = 1; bytePosition < objectSize; i++)
        {
            var copyRequest = new CopyPartRequest
            {
                DestinationBucket = TargetBucket,
                DestinationKey = TargetObjectKey,
```



```
        SourceBucket = SourceBucket,
        SourceKey = SourceObjectKey,
        UploadId = uploadId,
        FirstByte = bytePosition,
        LastByte = bytePosition + partSize - 1 >= objectSize ?
objectSize - 1 : bytePosition + partSize - 1,
        PartNumber = i,
    };

    copyResponses.Add(await client.CopyPartAsync(copyRequest));

    bytePosition += partSize;
}

// Set up to complete the copy.
var completeRequest = new CompleteMultipartUploadRequest
{
    BucketName = TargetBucket,
    Key = TargetObjectKey,
    UploadId = initResponse.UploadId,
};
completeRequest.AddPartETags(copyResponses);

// Complete the copy.
CompleteMultipartUploadResponse completeUploadResponse =
    await client.CompleteMultipartUploadAsync(completeRequest);
}
catch (AmazonS3Exception e)
{
    Console.WriteLine($"Error encountered on server.
Message: '{e.Message}' when writing an object");
}
catch (Exception e)
{
    Console.WriteLine($"Unknown encountered on server.
Message: '{e.Message}' when writing an object");
}
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour .NET .
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [GetObjectMetadata](#)
 - [UploadPartCopy](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Recevez et traitez les notifications d'événements Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment travailler avec les notifications d'événements S3 d'une manière orientée objet.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Cet exemple montre comment traiter un événement de notification S3 à l'aide d'Amazon SQS.

```
/**
 * This method receives S3 event notifications by using an SqsAsyncClient.
 * After the client receives the messages it deserializes the JSON payload
and logs them. It uses
 * the S3EventNotification class (part of the S3 event notification API for
Java) to deserialize
 * the JSON payload and access the messages in an object-oriented way.
 *
 * @param queueUrl The URL of the AWS SQS queue that receives the S3 event
notifications.
```

```

    * @see <a href="https://sdk.amazonaws.com/java/api/latest/software/amazon/
    awssdk/eventnotifications/s3/model/package-summary.html">S3EventNotification
    API</a>.
    * <p>
    * To use S3 event notification serialization/deserialization to objects, add
    the following
    * dependency to your Maven pom.xml file.
    * <dependency>
    * <groupId>software.amazon.awssdk</groupId>
    * <artifactId>s3-event-notifications</artifactId>
    * <version><LATEST></version>
    * </dependency>
    * <p>
    * The S3 event notification API became available with version 2.25.11 of the
    Java SDK.
    * <p>
    * This example shows the use of the API with AWS SQS, but it can be used to
    process S3 event notifications
    * in AWS SNS or AWS Lambda as well.
    * <p>
    * Note: The S3EventNotification class does not work with messages routed
    through AWS EventBridge.
    */
    static void processS3Events(String bucketName, String queueUrl, String
    queueArn) {
        try {
            // Configure the bucket to send Object Created and Object Tagging
            notifications to an existing SQS queue.
            s3Client.putBucketNotificationConfiguration(b -> b
                .notificationConfiguration(ncb -> ncb
                    .queueConfigurations(qcb -> qcb
                        .events(Event.S3_OBJECT_CREATED,
Event.S3_OBJECT_TAGGING)
                            .queueArn(queueArn)))
                    .bucket(bucketName)
                ).join();

            triggerS3EventNotifications(bucketName);
            // Wait for event notifications to propagate.
            Thread.sleep(Duration.ofSeconds(5).toMillis());

            boolean didReceiveMessages = true;
            while (didReceiveMessages) {

```

```

        // Display the number of messages that are available in the
queue.
        sqsClient.getQueueAttributes(b -> b
            .queueUrl(queueUrl)

            .attributeNames(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)
            ).thenAccept(attributeResponse ->
                logger.info("Approximate number of messages in
the queue: {}"),
            attributeResponse.attributes().get(QueueAttributeName.APPROXIMATE_NUMBER_OF_MESSAGES)))
            .join();

        // Receive the messages.
        ReceiveMessageResponse response = sqsClient.receiveMessage(b -> b
            .queueUrl(queueUrl)
            ).get();
        logger.info("Count of received messages: {}",
response.messages().size());
        didReceiveMessages = !response.messages().isEmpty();

        // Create a collection to hold the received message for deletion
// after we log the messages.
        HashSet<DeleteMessageBatchRequestEntry> messagesToDelete = new
HashSet<>();

        // Process each message.
        response.messages().forEach(message -> {
            logger.info("Message id: {}", message.messageId());
            // Deserialize JSON message body to a S3EventNotification
object

            // to access messages in an object-oriented way.
            S3EventNotification event =
S3EventNotification.fromJson(message.body());

            // Log the S3 event notification record details.
            if (event.getRecords() != null) {
                event.getRecords().forEach(record -> {
                    String eventName = record.getEventName();
                    String key = record.getS3().getObject().getKey();
                    logger.info(record.toString());
                    logger.info("Event name is {} and key is {}",
eventName, key);
                });
            }
        });
    }

```

```
        // Add logged messages to collection for batch deletion.
        messagesToDelete.add(DeleteMessageBatchRequestEntry.builder()
            .id(message.messageId())
            .receiptHandle(message.receiptHandle())
            .build());
    });
    // Delete messages.
    if (!messagesToDelete.isEmpty()) {
        sqsClient.deleteMessageBatch(DeleteMessageBatchRequest.builder()
            .queueUrl(queueUrl)
            .entries(messagesToDelete)
            .build()
        ).join();
    }
    } // End of while block.
} catch (InterruptedException | ExecutionException e) {
    throw new RuntimeException(e);
}
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .
 - [DeleteMessageBatch](#)
 - [GetQueueAttributes](#)
 - [PutBucketNotificationConfiguration](#)
 - [ReceiveMessage](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Enregistrez les informations EXIF et autres informations d'image à l'aide d'un AWS Kit SDK

L'exemple de code suivant illustre comment :

- Obtenir des informations EXIF à partir d'un fichier JPG, JPEG ou PNG.

- Charger le fichier image sur un compartiment Amazon S3.
- Utiliser Amazon Rekognition pour identifier les trois principaux attributs (étiquettes) dans le fichier.
- Ajouter les informations EXIF et les étiquettes à un tableau Amazon DynamoDB dans la région.

Rust

SDK pour Rust

Obtenez les informations EXIF à partir d'un fichier JPG, JPEG ou PNG, chargez le fichier image dans un compartiment Amazon S3, utilisez Amazon Rekognition pour identifier les trois principaux attributs (étiquettes dans Amazon Rekognition) du fichier et ajoutez les informations EXIF et d'étiquettes à un tableau Amazon DynamoDB dans la région.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- DynamoDB
- Amazon Rekognition
- Amazon S3

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Envoyez des notifications d'événements S3 à Amazon à EventBridge l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment activer un compartiment pour envoyer des notifications d'événements S3 EventBridge et les acheminer vers une rubrique Amazon SNS et une file d'attente Amazon SQS.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/** This method configures a bucket to send events to AWS EventBridge and
    creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    * An AWS CloudFormation stack sets up the bucket, queue, topic before the
    method runs.
    */
public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
    try {
        // Enable bucket to emit S3 Event notifications to EventBridge.
        s3Client.putBucketNotificationConfiguration(b -> b
            .bucket(bucketName)
            .notificationConfiguration(b1 -> b1
                .eventBridgeConfiguration(
                    SdkBuilder::build)
            ).build()).join();

        // Create an EventBridge rule to route Object Created notifications.
        PutRuleRequest putRuleRequest = PutRuleRequest.builder()
            .name(RULE_NAME)
            .eventPattern("""
                {
                    "source": ["aws.s3"],
                    "detail-type": ["Object Created"],
                    "detail": {
                        "bucket": {
                            "name": ["%s"]
                        }
                    }
                }
            """)
            .build();
    } catch (Exception e) {
        // Handle exception
    }
}
```

```

        }
    }
}
"".formatted(bucketName))
.build();

// Add the rule to the default event bus.
PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
    .whenComplete((r, t) -> {
        if (t != null) {
            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                throw new RuntimeException(t.getCause().getMessage(),
t);
        }
        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
    }).join();

// Add the existing SNS topic and SQS queue as targets to the rule.
eventBridgeClient.putTargets(b -> b
    .eventBusName("default")
    .rule(RULE_NAME)
    .targets(List.of (
        Target.builder()
            .arn(queueArn)
            .id("Queue")
            .build(),
        Target.builder()
            .arn(topicArn)
            .id("Topic")
            .build()
    )
    ).join();
    return putRuleResponse.ruleArn();
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}

```


- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .
 - [PutBucketNotificationConfiguration](#)
 - [PutRule](#)
 - [PutTargets](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Suivez le chargement ou le téléchargement d'un objet Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment suivre un chargement ou un téléchargement partitionné vers un objet Amazon S3.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Suivez la progression du chargement d'un fichier.

```
public void trackUploadFile(S3TransferManager transferManager, String
bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
```

```

        fileUpload.completionFuture().join();
    /*
        The SDK provides a LoggingTransferListener implementation of the
        TransferListener interface.
        You can also implement the interface to provide your own logic.

        Configure log4J2 with settings such as the following.
        <Configuration status="WARN">
            <Appenders>
                <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                    <PatternLayout pattern="%m%n"/>
                </Console>
            </Appenders>

            <Loggers>
                <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                    <AppenderRef ref="AlignedConsoleAppender"/>
                </logger>
            </Loggers>
        </Configuration>

        Log4J2 logs the progress. The following is example output for a 21.3
        MB file upload.
        Transfer initiated...
        |                               | 0.0%
        |====                         | 21.1%
        |=====                      | 60.5%
        |=====                      | 100.0%
        Transfer complete!

    */
}

```

Suivez la progression du téléchargement d'un fichier.

```

public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                                String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()

```

```

        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.

        .destination(Paths.get(downloadedFilePath))
        .build();

```

```

FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

```

```

CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
/*

```

The SDK provides a `LoggingTransferListener` implementation of the `TransferListener` interface.

You can also implement the interface to provide your own logic.

Configure `log4j2` with settings such as the following.

```

<Configuration status="WARN">
  <Appenders>
    <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
      <PatternLayout pattern="%m%n"/>
    </Console>
  </Appenders>

  <Loggers>
    <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
      <AppenderRef ref="AlignedConsoleAppender"/>
    </logger>
  </Loggers>
</Configuration>

```

`Log4j2` logs the progress. The following is example output for a 21.3 MB file download.

```

Transfer initiated...
|=====| 39.4%
|=====| 78.8%
|=====| 100.0%
Transfer complete!

```

```

*/
}

```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .
 - [GetObject](#)
 - [PutObject](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Transformation des données pour votre application avec S3 Object Lambda

L'exemple de code suivant illustre comment transformer des données pour votre application avec S3 Object Lambda.

.NET

SDK pour .NET

Explique comment ajouter du code personnalisé aux requêtes GET S3 standard afin de modifier l'objet demandé extrait de S3 pour qu'il réponde aux besoins du client ou de l'application qui le demande.

Pour obtenir le code source complet et les instructions de configuration et d'exécution, consultez l'exemple complet sur [GitHub](#).

Les services utilisés dans cet exemple

- Lambda
- Amazon S3

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Exemples d'approches pour les tests unitaires et d'intégration avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser des exemples de meilleures pratiques lors de l'écriture de tests unitaires et d'intégration à l'aide d'un AWS SDK.

Rust

SDK pour Rust

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Cargo.toml pour tester des exemples.

```
[package]
name = "testing-examples"
version = "0.1.0"
authors = [
  "John Disanti <jdisanti@amazon.com>",
  "Doug Schwartz <dougsch@amazon.com>",
]
edition = "2021"

[dependencies]
async-trait = "0.1.51"
aws-config = { version = "1.0.1", features = ["behavior-version-latest"] }
aws-credential-types = { version = "1.0.1", features = [ "hardcoded-credentials", ] }
aws-sdk-s3 = { version = "1.4.0" }
aws-smithy-types = { version = "1.0.1" }
aws-smithy-runtime = { version = "1.0.1", features = ["test-util"] }
aws-smithy-runtime-api = { version = "1.0.1", features = ["test-util"] }
aws-types = { version = "1.0.1" }
clap = { version = "4.4", features = ["derive"] }
http = "0.2.9"
mockall = "0.11.4"
serde_json = "1"
tokio = { version = "1.20.1", features = ["full"] }
tracing-subscriber = { version = "0.3.15", features = ["env-filter"] }

[[bin]]
name = "main"
path = "src/main.rs"
```

Exemple de test unitaire à l'aide d'une simulation automatique et d'un encapsuleur de service.

```

use aws_sdk_s3 as s3;
#[allow(unused_imports)]
use mockall::automock;

use s3::operation::list_objects_v2::{ListObjectsV2Error, ListObjectsV2Output};

#[cfg(test)]
pub use MockS3Impl as S3;
#[cfg(not(test))]
pub use S3Impl as S3;

#[allow(dead_code)]
pub struct S3Impl {
    inner: s3::Client,
}

#[cfg_attr(test, automock)]
impl S3Impl {
    #[allow(dead_code)]
    pub fn new(inner: s3::Client) -> Self {
        Self { inner }
    }

    #[allow(dead_code)]
    pub async fn list_objects(
        &self,
        bucket: &str,
        prefix: &str,
        continuation_token: Option<String>,
    ) -> Result<ListObjectsV2Output, s3::error::SdkError<ListObjectsV2Error>> {
        self.inner
            .list_objects_v2()
            .bucket(bucket)
            .prefix(prefix)
            .set_continuation_token(continuation_token)
            .send()
            .await
    }
}

#[allow(dead_code)]

```

```

pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3_list: S3,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3_list
            .list_objects(bucket, prefix, next_token.take())
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {
            break;
        }
    }
    Ok(total_size_bytes)
}

#[cfg(test)]
mod test {
    use super::*;
    use mockall::predicate::eq;

    #[tokio::test]
    async fn test_single_page() {
        let mut mock = MockS3Impl::default();
        mock.expect_list_objects()
            .with(eq("test-bucket"), eq("test-prefix"), eq(None))
            .return_once(|_, _, _| {
                Ok(ListObjectsV2Output::builder()
                    .set_contents(Some(vec![
                        // Mock content for ListObjectsV2 response
                        s3::types::Object::builder().size(5).build(),
                        s3::types::Object::builder().size(2).build(),
                    ]))
            })
    }
}

```

```

        ]))
        .build())
    });

    // Run the code we want to test with it
    let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
        .await
        .unwrap();

    // Verify we got the correct total size back
    assert_eq!(7, size);
}

#[tokio::test]
async fn test_multiple_pages() {
    // Create the Mock instance with two pages of objects now
    let mut mock = MockS3Impl::default();
    mock.expect_list_objects()
        .with(eq("test-bucket"), eq("test-prefix"), eq(None))
        .return_once(|_, _, _| {
            Ok(ListObjectsV2Output::builder()
                .set_contents(Some(vec![
                    // Mock content for ListObjectsV2 response
                    s3::types::Object::builder().size(5).build(),
                    s3::types::Object::builder().size(2).build(),
                ]))
                .set_next_continuation_token(Some("next".to_string()))
                .build())
        });
    mock.expect_list_objects()
        .with(
            eq("test-bucket"),
            eq("test-prefix"),
            eq(Some("next".to_string())),
        )
        .return_once(|_, _, _| {
            Ok(ListObjectsV2Output::builder()
                .set_contents(Some(vec![
                    // Mock content for ListObjectsV2 response
                    s3::types::Object::builder().size(3).build(),
                    s3::types::Object::builder().size(9).build(),
                ]))
                .build())
        });
}

```



```

        // Run the code we want to test with it
        let size = determine_prefix_file_size(mock, "test-bucket", "test-prefix")
            .await
            .unwrap();

        assert_eq!(19, size);
    }
}

```

Exemple de test d'intégration utilisant StaticReplayClient.

```

use aws_sdk_s3 as s3;

#[allow(dead_code)]
pub async fn determine_prefix_file_size(
    // Now we take a reference to our trait object instead of the S3 client
    // s3_list: ListObjectsService,
    s3: s3::Client,
    bucket: &str,
    prefix: &str,
) -> Result<usize, s3::Error> {
    let mut next_token: Option<String> = None;
    let mut total_size_bytes = 0;
    loop {
        let result = s3
            .list_objects_v2()
            .prefix(prefix)
            .bucket(bucket)
            .set_continuation_token(next_token.take())
            .send()
            .await?;

        // Add up the file sizes we got back
        for object in result.contents() {
            total_size_bytes += object.size().unwrap_or(0) as usize;
        }

        // Handle pagination, and break the loop if there are no more pages
        next_token = result.next_continuation_token.clone();
        if next_token.is_none() {

```

```

        break;
    }
}
Ok(total_size_bytes)
}

#[allow(dead_code)]
fn make_s3_test_credentials() -> s3::config::Credentials {
    s3::config::Credentials::new(
        "ATESTCLIENT",
        "astestsecretkey",
        Some("atestsessiontoken".to_string()),
        None,
        "",
    )
}

#[cfg(test)]
mod test {
    use super::*;
    use aws_config::BehaviorVersion;
    use aws_sdk_s3 as s3;
    use aws_smithy_runtime::client::http::test_util::{ReplayEvent,
StaticReplayClient};
    use aws_smithy_types::body::SdkBody;

    #[tokio::test]
    async fn test_single_page() {
        let page_1 = ReplayEvent::new(
            http::Request::builder()
                .method("GET")
                .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
                .body(SdkBody::empty())
                .unwrap(),
            http::Response::builder()
                .status(200)
                .body(SdkBody::from(include_str!("./testing/
response_1.xml")))
                .unwrap(),
        );
        let replay_client = StaticReplayClient::new(vec![page_1]);
        let client: s3::Client = s3::Client::from_conf(
            s3::Config::builder()

```

```

        .behavior_version(BehaviorVersion::latest())
        .credentials_provider(make_s3_test_credentials())
        .region(s3::config::Region::new("us-east-1"))
        .http_client(replay_client.clone())
        .build(),
    );

    // Run the code we want to test with it
    let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
        .await
        .unwrap();

    // Verify we got the correct total size back
    assert_eq!(7, size);
    replay_client.assert_requests_match(&[]);
}

#[tokio::test]
async fn test_multiple_pages() {
    let page_1 = ReplayEvent::new(
        http::Request::builder()
            .method("GET")
            .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix")
            .body(SdkBody::empty())
            .unwrap(),
        http::Response::builder()
            .status(200)
            .body(SdkBody::from(include_str!("./testing/
response_multi_1.xml")))
            .unwrap(),
    );
    let page_2 = ReplayEvent::new(
        http::Request::builder()
            .method("GET")
            .uri("https://test-bucket.s3.us-east-1.amazonaws.com/?list-
type=2&prefix=test-prefix&continuation-token=next")
            .body(SdkBody::empty())
            .unwrap(),
        http::Response::builder()
            .status(200)
            .body(SdkBody::from(include_str!("./testing/
response_multi_2.xml")))

```

```

        .unwrap(),
    );
    let replay_client = StaticReplayClient::new(vec![page_1, page_2]);
    let client: s3::Client = s3::Client::from_conf(
        s3::Config::builder()
            .behavior_version(BehaviorVersion::latest())
            .credentials_provider(make_s3_test_credentials())
            .region(s3::config::Region::new("us-east-1"))
            .http_client(replay_client.clone())
            .build(),
    );

    // Run the code we want to test with it
    let size = determine_prefix_file_size(client, "test-bucket", "test-
prefix")
        .await
        .unwrap();

    assert_eq!(19, size);

    replay_client.assert_requests_match(&[]);
}
}

```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Charger récursivement un répertoire local dans un compartiment Amazon Simple Storage Service (Amazon S3)

L'exemple de code suivant montre comment charger un répertoire local de manière récursive dans un compartiment Amazon Simple Storage Service (Amazon S3).

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez un [S3TransferManager](#) pour [télécharger un répertoire local](#). Consultez le [fichier complet](#) et le [test](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedDirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.DirectoryUpload;
import software.amazon.awssdk.transfer.s3.model.UploadDirectoryRequest;

import java.net.URI;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Paths;

    public Integer uploadDirectory(S3TransferManager transferManager,
        URI sourceDirectory, String bucketName) {
        DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());

        CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
        completedDirectoryUpload.failedTransfers()
            .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
```

- Pour plus de détails sur l'API, reportez-vous [UploadDirectory](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Chargez ou téléchargez des fichiers volumineux vers et depuis Amazon S3 à l'aide d'un AWS Kit SDK

Les exemples de code suivants montrent comment charger ou télécharger des fichiers volumineux vers et depuis Amazon S3.

Pour plus d'informations, consultez la rubrique [Uploading an object using multipart upload](#) (Chargement d'un objet à l'aide du chargement partitionné).

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Appelez des fonctions qui transfèrent des fichiers vers et depuis un compartiment S3 à l'aide d'Amazon S3 TransferUtility.

```
global using System.Text;
global using Amazon.S3;
global using Amazon.S3.Model;
global using Amazon.S3.Transfer;
global using TransferUtilityBasics;

// This Amazon S3 client uses the default user credentials
// defined for this computer.
using Microsoft.Extensions.Configuration;
```

```
IAmazonS3 client = new AmazonS3Client();
var transferUtil = new TransferUtility(client);
IConfiguration _configuration;

_configuration = new ConfigurationBuilder()
    .SetBasePath(Directory.GetCurrentDirectory())
    .AddJsonFile("settings.json") // Load test settings from JSON file.
    .AddJsonFile("settings.local.json",
        true) // Optionally load local settings.
    .Build();

// Edit the values in settings.json to use an S3 bucket and files that
// exist on your AWS account and on the local computer where you
// run this scenario.
var bucketName = _configuration["BucketName"];
var localPath =
    $"{Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData)}\
\TransferFolder";

DisplayInstructions();

PressEnter();

Console.WriteLine();

// Upload a single file to an S3 bucket.
DisplayTitle("Upload a single file");

var fileToUpload = _configuration["FileToUpload"];
Console.WriteLine($"Uploading {fileToUpload} to the S3 bucket, {bucketName}.");

var success = await TransferMethods.UploadSingleFileAsync(transferUtil,
    bucketName, fileToUpload, localPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the file, {fileToUpload} to
    {bucketName}.");
}

PressEnter();

// Upload a local directory to an S3 bucket.
DisplayTitle("Upload all files from a local directory");
Console.WriteLine("Upload all the files in a local folder to an S3 bucket.");
```

```
const string keyPrefix = "UploadFolder";
var uploadPath = $"{localPath}\\UploadFolder";

Console.WriteLine($"Uploading the files in {uploadPath} to {bucketName}");
DisplayTitle($"{uploadPath} files");
DisplayLocalFiles(uploadPath);
Console.WriteLine();

PressEnter();

success = await TransferMethods.UploadFullDirectoryAsync(transferUtil,
    bucketName, keyPrefix, uploadPath);
if (success)
{
    Console.WriteLine($"Successfully uploaded the files in {uploadPath} to
{bucketName}.");
    Console.WriteLine($"{bucketName} currently contains the following files:");
    await DisplayBucketFiles(client, bucketName, keyPrefix);
    Console.WriteLine();
}

PressEnter();

// Download a single file from an S3 bucket.
DisplayTitle("Download a single file");
Console.WriteLine("Now we will download a single file from an S3 bucket.");

var keyName = _configuration["FileToDownload"];

Console.WriteLine($"Downloading {keyName} from {bucketName}.");

success = await TransferMethods.DownloadSingleFileAsync(transferUtil, bucketName,
    keyName, localPath);
if (success)
{
    Console.WriteLine($"Successfully downloaded the file, {keyName} from
{bucketName}.");
}

PressEnter();

// Download the contents of a directory from an S3 bucket.
DisplayTitle("Download the contents of an S3 bucket");
var s3Path = _configuration["S3Path"];
```



```
var downloadPath = $"{localPath}\\{s3Path}";

Console.WriteLine($"Downloading the contents of {bucketName}\\{s3Path}");
Console.WriteLine($"{bucketName}\\{s3Path} contains the following files:");
await DisplayBucketFiles(client, bucketName, s3Path);
Console.WriteLine();

success = await TransferMethods.DownloadS3DirectoryAsync(transferUtil,
    bucketName, s3Path, downloadPath);
if (success)
{
    Console.WriteLine($"Downloaded the files in {bucketName} to
{downloadPath}.");
    Console.WriteLine($"{downloadPath} now contains the following files:");
    DisplayLocalFiles(downloadPath);
}

Console.WriteLine("\nThe TransferUtility Basics application has completed.");
PressEnter();

// Displays the title for a section of the scenario.
static void DisplayTitle(string titleText)
{
    var sepBar = new string('-', Console.WindowWidth);

    Console.WriteLine(sepBar);
    Console.WriteLine(CenterText(titleText));
    Console.WriteLine(sepBar);
}

// Displays a description of the actions to be performed by the scenario.
static void DisplayInstructions()
{
    var sepBar = new string('-', Console.WindowWidth);

    DisplayTitle("Amazon S3 Transfer Utility Basics");
    Console.WriteLine("This program shows how to use the Amazon S3 Transfer
Utility.");
    Console.WriteLine("It performs the following actions:");
    Console.WriteLine("\t1. Upload a single object to an S3 bucket.");
    Console.WriteLine("\t2. Upload an entire directory from the local computer to
an\t S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
}
```

```
        Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
        directory.");
        Console.WriteLine($"\\n{sepBar}");
    }

    // Pauses the scenario.
    static void PressEnter()
    {
        Console.WriteLine("Press <Enter> to continue.");
        _ = Console.ReadLine();
        Console.WriteLine("\\n");
    }

    // Returns the string textToCenter, padded on the left with spaces
    // that center the text on the console display.
    static string CenterText(string textToCenter)
    {
        var centeredText = new StringBuilder();
        var screenWidth = Console.WindowWidth;
        centeredText.Append(new string(' ', (int)(screenWidth -
        textToCenter.Length) / 2));
        centeredText.Append(textToCenter);
        return centeredText.ToString();
    }

    // Displays a list of file names included in the specified path.
    static void DisplayLocalFiles(string localPath)
    {
        var fileList = Directory.GetFiles(localPath);
        if (fileList.Length > 0)
        {
            foreach (var fileName in fileList)
            {
                Console.WriteLine(fileName);
            }
        }
    }

    // Displays a list of the files in the specified S3 bucket and prefix.
    static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
    s3Path)
    {
        ListObjectsV2Request request = new()
        {
```

```

        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
    };

    var response = new ListObjectsV2Response();

    do
    {
        response = await client.ListObjectsV2Async(request);

        response.S3Objects
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));

        // If the response is truncated, set the request ContinuationToken
        // from the NextContinuationToken property of the response.
        request.ContinuationToken = response.NextContinuationToken;
    } while (response.IsTruncated);
}

```

Chargez un seul fichier.

```

    /// <summary>
    /// Uploads a single file from the local computer to an S3 bucket.
    /// </summary>
    /// <param name="transferUtil">The transfer initialized TransferUtility
    /// object.</param>
    /// <param name="bucketName">The name of the S3 bucket where the file
    /// will be stored.</param>
    /// <param name="fileName">The name of the file to upload.</param>
    /// <param name="localPath">The local path where the file is stored.</
param>
    /// <returns>A boolean value indicating the success of the action.</
returns>
    public static async Task<bool> UploadSingleFileAsync(
        TransferUtility transferUtil,
        string bucketName,
        string fileName,
        string localPath)
    {

```

```
        if (File.Exists($"{localPath}\\{fileName}"))
        {
            try
            {
                await transferUtil.UploadAsync(new
TransferUtilityUploadRequest
                {
                    BucketName = bucketName,
                    Key = fileName,
                    FilePath = $"{localPath}\\{fileName}",
                });

                return true;
            }
            catch (AmazonS3Exception s3Ex)
            {
                Console.WriteLine($"Could not upload {fileName} from
{localPath} because:");
                Console.WriteLine(s3Ex.Message);
                return false;
            }
        }
        else
        {
            Console.WriteLine($"{fileName} does not exist in {localPath}");
            return false;
        }
    }
}
```

Chargez un répertoire local complet.

```
/// <summary>
/// Uploads all the files in a local directory to a directory in an S3
/// bucket.
/// </summary>
/// <param name="transferUtil">The transfer initialized TransferUtility
/// object.</param>
/// <param name="bucketName">The name of the S3 bucket where the files
/// will be stored.</param>
/// <param name="keyPrefix">The key prefix is the S3 directory where
/// the files will be stored.</param>
```

```
/// <param name="localPath">The local directory that contains the files
/// to be uploaded.</param>
/// <returns>A Boolean value representing the success of the action.</
returns>
public static async Task<bool> UploadFullDirectoryAsync(
    TransferUtility transferUtil,
    string bucketName,
    string keyPrefix,
    string localPath)
{
    if (Directory.Exists(localPath))
    {
        try
        {
            await transferUtil.UploadDirectoryAsync(new
TransferUtilityUploadDirectoryRequest
            {
                BucketName = bucketName,
                KeyPrefix = keyPrefix,
                Directory = localPath,
            });

            return true;
        }
        catch (AmazonS3Exception s3Ex)
        {
            Console.WriteLine($"Can't upload the contents of {localPath}
because:");

            Console.WriteLine(s3Ex?.Message);
            return false;
        }
    }
    else
    {
        Console.WriteLine($"The directory {localPath} does not exist.");
        return false;
    }
}
```

Téléchargez un seul fichier.

```

    /// <summary>
    /// Download a single file from an S3 bucket to the local computer.
    /// </summary>
    /// <param name="transferUtil">The transfer initialized TransferUtility
    /// object.</param>
    /// <param name="bucketName">The name of the S3 bucket containing the
    /// file to download.</param>
    /// <param name="keyName">The name of the file to download.</param>
    /// <param name="localPath">The path on the local computer where the
    /// downloaded file will be saved.</param>
    /// <returns>A Boolean value indicating the results of the action.</
returns>
    public static async Task<bool> DownloadSingleFileAsync(
        TransferUtility transferUtil,
        string bucketName,
        string keyName,
        string localPath)
    {
        await transferUtil.DownloadAsync(new TransferUtilityDownloadRequest
        {
            BucketName = bucketName,
            Key = keyName,
            FilePath = $"{localPath}\\{keyName}",
        });

        return (File.Exists($"{localPath}\\{keyName}"));
    }

```

Téléchargez le contenu d'un compartiment S3.

```

    /// <summary>
    /// Downloads the contents of a directory in an S3 bucket to a
    /// directory on the local computer.
    /// </summary>
    /// <param name="transferUtil">The transfer initialized TransferUtility
    /// object.</param>
    /// <param name="bucketName">The bucket containing the files to
download.</param>

```

```
    /// <param name="s3Path">The S3 directory where the files are located.</  
param>  
    /// <param name="localPath">The local path to which the files will be  
    /// saved.</param>  
    /// <returns>A Boolean value representing the success of the action.</  
returns>  
    public static async Task<bool> DownloadS3DirectoryAsync(  
        TransferUtility transferUtil,  
        string bucketName,  
        string s3Path,  
        string localPath)  
    {  
        int fileCount = 0;  
  
        // If the directory doesn't exist, it will be created.  
        if (Directory.Exists(s3Path))  
        {  
            var files = Directory.GetFiles(localPath);  
            fileCount = files.Length;  
        }  
  
        await transferUtil.DownloadDirectoryAsync(new  
TransferUtilityDownloadDirectoryRequest  
        {  
            BucketName = bucketName,  
            LocalDirectory = localPath,  
            S3Directory = s3Path,  
        });  
  
        if (Directory.Exists(localPath))  
        {  
            var files = Directory.GetFiles(localPath);  
            if (files.Length > fileCount)  
            {  
                return true;  
            }  
  
            // No change in the number of files. Assume  
            // the download failed.  
            return false;  
        }  
  
        // The local directory doesn't exist. No files  
        // were downloaded.
```

```
        return false;
    }
```

Suivez la progression d'un téléchargement à l'aide du `TransferUtility`.

```
using System;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Transfer;

/// <summary>
/// This example shows how to track the progress of a multipart upload
/// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
/// upload to an Amazon S3 bucket.
/// </summary>
public class TrackMPUUsingHighLevelAPI
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "sample_pic.png";
        string path = "filepath/directory/";
        string filePath = $"{path}{keyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USSouth1 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        await TrackMPUAsync(client, bucketName, filePath, keyName);
    }

    /// <summary>
    /// Starts an Amazon S3 multipart upload and assigns an event handler to
    /// track the progress of the upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
    /// perform the multipart upload.</param>
    /// <param name="bucketName">The name of the bucket to which to upload
    /// the file.</param>
```



```
/// <param name="filePath">The path, including the file name of the
/// file to be uploaded to the Amazon S3 bucket.</param>
/// <param name="keyName">The file name to be used in the
/// destination Amazon S3 bucket.</param>
public static async Task TrackMPUAsync(
    IAmazonS3 client,
    string bucketName,
    string filePath,
    string keyName)
{
    try
    {
        var fileTransferUtility = new TransferUtility(client);

        // Use TransferUtilityUploadRequest to configure options.
        // In this example we subscribe to an event.
        var uploadRequest =
            new TransferUtilityUploadRequest
            {
                BucketName = bucketName,
                FilePath = filePath,
                Key = keyName,
            };

        uploadRequest.UploadProgressEvent +=
            new EventHandler<UploadProgressArgs>(
                UploadRequest_UploadPartProgressEvent);

        await fileTransferUtility.UploadAsync(uploadRequest);
        Console.WriteLine("Upload completed");
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error:: {ex.Message}");
    }
}

/// <summary>
/// Event handler to check the progress of the multipart upload.
/// </summary>
/// <param name="sender">The object that raised the event.</param>
/// <param name="e">The object that contains multipart upload
/// information.</param>
```

```
public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
{
    // Process event.
    Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
}
}
```

Chargez un objet avec chiffrement.

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// Uses the Amazon Simple Storage Service (Amazon S3) low level API to
/// perform a multipart upload to an Amazon S3 bucket.
/// </summary>
public class SSECLowLevelMPUcopyObject
{
    public static async Task Main()
    {
        string existingBucketName = "amzn-s3-demo-bucket";
        string sourceKeyName = "sample_file.txt";
        string targetKeyName = "sample_file_copy.txt";
        string filePath = $"sample\\{targetKeyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USEast1.
        IAmazonS3 client = new AmazonS3Client();

        // Create the encryption key.
        var base64Key = CreateEncryptionKey();

        await CreateSampleObjUsingClientEncryptionKeyAsync(
```

```

        client,
        existingBucketName,
        sourceKeyName,
        filePath,
        base64Key);
    }

    /// <summary>
    /// Creates the encryption key to use with the multipart upload.
    /// </summary>
    /// <returns>A string containing the base64-encoded key for encrypting
    /// the multipart upload.</returns>
    public static string CreateEncryptionKey()
    {
        Aes aesEncryption = Aes.Create();
        aesEncryption.KeySize = 256;
        aesEncryption.GenerateKey();
        string base64Key = Convert.ToBase64String(aesEncryption.Key);
        return base64Key;
    }

    /// <summary>
    /// Creates and uploads an object using a multipart upload.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 object used to
    /// initialize and perform the multipart upload.</param>
    /// <param name="existingBucketName">The name of the bucket to which
    /// the object will be uploaded.</param>
    /// <param name="sourceKeyName">The source object name.</param>
    /// <param name="filePath">The location of the source object.</param>
    /// <param name="base64Key">The encryption key to use with the upload.</
param>
    public static async Task CreateSampleObjUsingClientEncryptionKeyAsync(
        IAmazonS3 client,
        string existingBucketName,
        string sourceKeyName,
        string filePath,
        string base64Key)
    {
        List<UploadPartResponse> uploadResponses = new
List<UploadPartResponse>();

        InitiateMultipartUploadRequest initiateRequest = new
InitiateMultipartUploadRequest

```

```
{
    BucketName = existingBucketName,
    Key = sourceKeyName,
    ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
    ServerSideEncryptionCustomerProvidedKey = base64Key,
};

InitiateMultipartUploadResponse initResponse =
    await client.InitiateMultipartUploadAsync(initiateRequest);

long contentLength = new FileInfo(filePath).Length;
long partSize = 5 * (long)Math.Pow(2, 20); // 5 MB

try
{
    long filePosition = 0;
    for (int i = 1; filePosition < contentLength; i++)
    {
        UploadPartRequest uploadRequest = new UploadPartRequest
        {
            BucketName = existingBucketName,
            Key = sourceKeyName,
            UploadId = initResponse.UploadId,
            PartNumber = i,
            PartSize = partSize,
            FilePosition = filePosition,
            FilePath = filePath,
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };

        // Upload part and add response to our list.
        uploadResponses.Add(await
client.UploadPartAsync(uploadRequest));

        filePosition += partSize;
    }

    CompleteMultipartUploadRequest completeRequest = new
CompleteMultipartUploadRequest
    {
        BucketName = existingBucketName,
```

```

        Key = sourceKeyName,
        UploadId = initResponse.UploadId,
    };
    completeRequest.AddPartETags(uploadResponses);

    CompleteMultipartUploadResponse completeUploadResponse =
        await client.CompleteMultipartUploadAsync(completeRequest);
    }
    catch (Exception exception)
    {
        Console.WriteLine($"Exception occurred: {exception.Message}");

        // If there was an error, abort the multipart upload.
        AbortMultipartUploadRequest abortMPURequest = new
AbortMultipartUploadRequest
    {
        BucketName = existingBucketName,
        Key = sourceKeyName,
        UploadId = initResponse.UploadId,
    };

        await client.AbortMultipartUploadAsync(abortMPURequest);
    }
}
}

```

Go

Kit SDK pour Go V2

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez des fonctions qui utilisent des gestionnaires de chargement et de téléchargement pour diviser les données en plusieurs parties et les transférer simultanément.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)  
// actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform  
// bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}  
  
// UploadLargeObject uses an upload manager to upload data to an object in a  
// bucket.  
// The upload manager breaks large data into parts and uploads the parts  
// concurrently.  
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName  
    string, objectKey string, largeObject []byte) error {  
    largeBuffer := bytes.NewReader(largeObject)  
    var partMiBs int64 = 10  
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {  
        u.PartSize = partMiBs * 1024 * 1024  
    })  
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{  
        Bucket: aws.String(bucketName),
```

```

    Key:    aws.String(objectKey),
    Body:   largeBuffer,
})
if err != nil {
    var apiErr smithy.APIError
    if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
        log.Printf("Error while uploading object to %s. The object is too large.\n"+
            "The maximum size for a multipart upload is 5TB.", bucketName)
    } else {
        log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
} else {
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
}

return err
}

// DownloadLargeObject uses a download manager to download an object from a
// bucket.
// The download manager gets the data in parts and writes them to a buffer until
// all of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10
    downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
{
        d.PartSize = partMiBs * 1024 * 1024
    })
    buffer := manager.NewWriteAtBuffer([]byte{})
    _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {

```

```
    log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
        bucketName, objectKey, err)
}
return buffer.Bytes(), err
}
```

Exécutez un scénario interactif qui vous montre comment utiliser les gestionnaires de chargement et de téléchargement dans leur contexte.

```
import (
    "context"
    "crypto/rand"
    "log"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunLargeObjectScenario is an interactive example that shows you how to use
// Amazon
// Simple Storage Service (Amazon S3) to upload and download large objects.
//
// 1. Create a bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 5. Download a large object by using a download manager.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../..demotools folder of this repo.
func RunLargeObjectScenario(ctx context.Context, sdkConfig aws.Config, questioner
    demotools.IQuestioner) {
```



```

defer func() {
    if r := recover(); r != nil {
        log.Println("Something went wrong with the demo.")
        _, isMock := questioner.(*demotools.MockQuestioner)
        if isMock || questioner.AskBool("Do you want to see the full error message (y/n)?", "y") {
            log.Println(r)
        }
    }
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 large object demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your bucket:",
    demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
_, _ = rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(ctx, bucketName, largeKey, largeBytes)

```

```
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(ctx, bucketName,
largeKey)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
"contents? (y/n)", "y") {
    log.Println("Deleting object.")
    err = bucketBasics.DeleteObjects(ctx, bucketName, []string{largeKey})
    if err != nil {
        panic(err)
    }
    log.Println("Deleting bucket.")
    err = bucketBasics.DeleteBucket(ctx, bucketName)
    if err != nil {
        panic(err)
    }
} else {
    log.Println("Okay. Don't forget to delete objects from your bucket to avoid
charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Appelez des fonctions qui transfèrent des fichiers depuis et vers un compartiment S3 à l'aide du `S3TransferManager`.

```
public Integer downloadObjectsToDirectory(S3TransferManager transferManager,
    URI destinationPathURI, String bucketName) {
    DirectoryDownload directoryDownload =
transferManager.downloadDirectory(DownloadDirectoryRequest.builder()
        .destination(Paths.get(destinationPathURI))
        .bucket(bucketName)
        .build());
    CompletedDirectoryDownload completedDirectoryDownload =
directoryDownload.completionFuture().join();

    completedDirectoryDownload.failedTransfers()
        .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
    return completedDirectoryDownload.failedTransfers().size();
}
```

Chargez un répertoire local complet.

```
public Integer uploadDirectory(S3TransferManager transferManager,
    URI sourceDirectory, String bucketName) {
    DirectoryUpload directoryUpload =
transferManager.uploadDirectory(UploadDirectoryRequest.builder()
        .source(Paths.get(sourceDirectory))
        .bucket(bucketName)
        .build());
    CompletedDirectoryUpload completedDirectoryUpload =
directoryUpload.completionFuture().join();
}
```

```
        completedDirectoryUpload.failedTransfers()
            .forEach(fail -> logger.warn("Object [{}] failed to transfer",
fail.toString()));
        return completedDirectoryUpload.failedTransfers().size();
    }
}
```

Chargez un seul fichier.

```
public String uploadFile(S3TransferManager transferManager, String
bucketName,
                        String key, URI filePathURI) {
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b.bucket(bucketName).key(key))
        .source(Paths.get(filePathURI))
        .build();

    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);

    CompletedFileUpload uploadResult = fileUpload.completionFuture().join();
    return uploadResult.response().eTag();
}
```

Les exemples de code utilisent les importations suivantes.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;
```

```
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.util.ArrayList;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
```

Utilisez le [gestionnaire de transfert S3](#) situé au-dessus du [client AWS CRT-based S3](#) pour effectuer un téléchargement partitionné de manière transparente lorsque la taille du contenu dépasse un seuil. La taille par défaut est de 8 Mo.

```
/**
 * Uploads a file to an Amazon S3 bucket using the S3TransferManager.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the file path of the file to be uploaded
 */
public void multipartUploadWithTransferManager(String bucketName, String key,
String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

Utilisez l'[API S3Client](#) ou pour effectuer un chargement partitionné.

```
/**
```

```
* Performs a multipart upload to Amazon S3 using the provided S3 client.
*
* @param bucketName the name of the S3 bucket
* @param key         the object key
* @param filePath    the path to the file to be uploaded
*/
public void multipartUploadWithS3Client(String bucketName, String key, String
filePath) {

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
                RequestBody.fromByteBuffer(bb));

            CompletedPart part = CompletedPart.builder()
                .partNumber(partNumber)
                .eTag(partResponse.eTag())
```

```

        .build();
        completedParts.add(part);

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (IOException e) {
    logger.error(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}

```

Utilisez l'[S3AsyncClient API](#) avec le support multipartie activé pour effectuer un téléchargement partitionné.

```

/**
 * Uploads a file to an S3 bucket using the S3AsyncClient and enabling
 * multipart support.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 * @param filePath    the local file path of the file to be uploaded
 */
public void multipartUploadWithS3AsyncClient(String bucketName, String key,
String filePath) {
    // Enable multipart support.
    S3AsyncClient s3AsyncClient = S3AsyncClient.builder()
        .multipartEnabled(true)
        .build();

    CompletableFuture<PutObjectResponse> response = s3AsyncClient.putObject(b
-> b
        .bucket(bucketName)

```

```
        .key(key),
        Paths.get(filePath));

    response.join();
    logger.info("File uploaded in multiple 8 MiB parts using
S3AsyncClient.");
}
```

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Chargez un fichier volumineux.

```
import { S3Client } from "@aws-sdk/client-s3";
import { Upload } from "@aws-sdk/lib-storage";

import {
  ProgressBar,
  logger,
} from "@aws-doc-sdk-examples/lib/utils/util-log.js";

const twentyFiveMB = 25 * 1024 * 1024;

export const createString = (size = twentyFiveMB) => {
  return "x".repeat(size);
};

/**
 * Create a 25MB file and upload it in parts to the specified
 * Amazon S3 bucket.
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  const str = createString();
```



```

const buffer = Buffer.from(str, "utf8");
const progressBar = new ProgressBar({
  description: `Uploading "${key}" to "${bucketName}"`,
  barLength: 30,
});

try {
  const upload = new Upload({
    client: new S3Client({}),
    params: {
      Bucket: bucketName,
      Key: key,
      Body: buffer,
    },
  });

  upload.on("httpUploadProgress", ({ loaded, total }) => {
    progressBar.update({ current: loaded, total });
  });

  await upload.done();
} catch (caught) {
  if (caught instanceof Error && caught.name === "AbortError") {
    logger.error(`Multipart upload was aborted. ${caught.message}`);
  } else {
    throw caught;
  }
}
};

```

Téléchargez un fichier volumineux.

```

import { fileURLToPath } from "node:url";
import { GetObjectCommand, NoSuchKey, S3Client } from "@aws-sdk/client-s3";
import { createWriteStream, rmSync } from "node:fs";

const s3Client = new S3Client({});
const oneMB = 1024 * 1024;

export const getObjectRange = ({ bucket, key, start, end }) => {
  const command = new GetObjectCommand({

```

```

    Bucket: bucket,
    Key: key,
    Range: `bytes=${start}-${end}`,
  });

  return s3Client.send(command);
};

/**
 * @param {string | undefined} contentRange
 */
export const getRangeAndLength = (contentRange) => {
  const [range, length] = contentRange.split("/");
  const [start, end] = range.split("-");
  return {
    start: Number.parseInt(start),
    end: Number.parseInt(end),
    length: Number.parseInt(length),
  };
};

export const isComplete = ({ end, length }) => end === length - 1;

const downloadInChunks = async ({ bucket, key }) => {
  const writeStream = createWriteStream(
    fileURLToPath(new URL(`./${key}`, import.meta.url)),
  ).on("error", (err) => console.error(err));

  let rangeAndLength = { start: -1, end: -1, length: -1 };

  while (!isComplete(rangeAndLength)) {
    const { end } = rangeAndLength;
    const nextRange = { start: end + 1, end: end + oneMB };

    const { ContentRange, Body } = await getObjectRange({
      bucket,
      key,
      ...nextRange,
    });
    console.log(`Downloaded bytes ${nextRange.start} to ${nextRange.end}`);

    writeStream.write(await Body.transformToByteArray());
    rangeAndLength = getRangeAndLength(ContentRange);
  }
}

```

```
};

/**
 * Download a large object from an Amazon S3 bucket.
 *
 * When downloading a large file, you might want to break it down into
 * smaller pieces. Amazon S3 accepts a Range header to specify the start
 * and end of the byte range to be downloaded.
 *
 * @param {{ bucketName: string, key: string }}
 */
export const main = async ({ bucketName, key }) => {
  try {
    await downloadInChunks({
      bucket: bucketName,
      key: key,
    });
  } catch (caught) {
    if (caught instanceof NoSuchKey) {
      console.error(`Failed to download object. No such key "${key}".`);
      rmSync(key);
    }
  }
};
```

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez des fonctions qui transfèrent des fichiers en utilisant plusieurs des paramètres disponibles du gestionnaire de transfert. Utilisez une classe de rappel pour écrire la progression des rappels pendant le transfert de fichiers.

```
import sys
```

```
import threading

import boto3
from boto3.s3.transfer import TransferConfig

MB = 1024 * 1024
s3 = boto3.resource("s3")

class TransferCallback:
    """
    Handle callbacks from the transfer manager.

    The transfer manager periodically calls the __call__ method throughout
    the upload and download process so that it can take action, such as
    displaying progress to the user and collecting data about the transfer.
    """

    def __init__(self, target_size):
        self._target_size = target_size
        self._total_transferred = 0
        self._lock = threading.Lock()
        self.thread_info = {}

    def __call__(self, bytes_transferred):
        """
        The callback method that is called by the transfer manager.

        Display progress during file transfer and collect per-thread transfer
        data. This method can be called by multiple threads, so shared instance
        data is protected by a thread lock.
        """
        thread = threading.current_thread()
        with self._lock:
            self._total_transferred += bytes_transferred
            if thread.ident not in self.thread_info.keys():
                self.thread_info[thread.ident] = bytes_transferred
            else:
                self.thread_info[thread.ident] += bytes_transferred

        target = self._target_size * MB
        sys.stdout.write(
            f"\r{self._total_transferred} of {target} transferred "
```

```

        f"({(self._total_transferred / target) * 100:.2f}%)."
    )
    sys.stdout.flush()

def upload_with_default_configuration(
    local_file_path, bucket_name, object_key, file_size_mb
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, using the default
    configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def upload_with_chunksize_and_meta(
    local_file_path, bucket_name, object_key, file_size_mb, metadata=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart chunk size and adding metadata to the Amazon S3 object.

    The multipart chunk size controls the size of the chunks of data that are
    sent in the request. A smaller chunk size typically results in the transfer
    manager using more threads for the upload.

    The metadata is a set of key-value pairs that are stored with the object
    in Amazon S3.
    """
    transfer_callback = TransferCallback(file_size_mb)

    config = TransferConfig(multipart_chunksize=1 * MB)
    extra_args = {"Metadata": metadata} if metadata else None
    s3.Bucket(bucket_name).upload_file(
        local_file_path,
        object_key,
        Config=config,
        ExtraArgs=extra_args,
        Callback=transfer_callback,
    )

```

```
    return transfer_callback.thread_info

def upload_with_high_threshold(local_file_path, bucket_name, object_key,
    file_size_mb):
    """
    Upload a file from a local folder to an Amazon S3 bucket, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard upload instead of
    a multipart upload.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def upload_with_sse(
    local_file_path, bucket_name, object_key, file_size_mb, sse_key=None
):
    """
    Upload a file from a local folder to an Amazon S3 bucket, adding server-side
    encryption with customer-provided encryption keys to the object.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)
    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).upload_file(
        local_file_path, object_key, ExtraArgs=extra_args,
Callback=transfer_callback
    )
    return transfer_callback.thread_info
```

```
def download_with_default_configuration(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using the
    default configuration.
    """
    transfer_callback = TransferCallback(file_size_mb)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def download_with_single_thread(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, using a
    single thread.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(use_threads=False)
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def download_with_high_threshold(
    bucket_name, object_key, download_file_path, file_size_mb
):
    """
    Download a file from an Amazon S3 bucket to a local folder, setting a
    multipart threshold larger than the size of the file.

    Setting a multipart threshold larger than the size of the file results
    in the transfer manager sending the file as a standard download instead
    of a multipart download.
    """
    transfer_callback = TransferCallback(file_size_mb)
    config = TransferConfig(multipart_threshold=file_size_mb * 2 * MB)
    s3.Bucket(bucket_name).Object(object_key).download_file(
```

```

        download_file_path, Config=config, Callback=transfer_callback
    )
    return transfer_callback.thread_info

def download_with_sse(
    bucket_name, object_key, download_file_path, file_size_mb, sse_key
):
    """
    Download a file from an Amazon S3 bucket to a local folder, adding a
    customer-provided encryption key to the request.

    When this kind of encryption is specified, Amazon S3 encrypts the object
    at rest and allows downloads only when the expected encryption key is
    provided in the download request.
    """
    transfer_callback = TransferCallback(file_size_mb)

    if sse_key:
        extra_args = {"SSECustomerAlgorithm": "AES256", "SSECustomerKey":
sse_key}
    else:
        extra_args = None
    s3.Bucket(bucket_name).Object(object_key).download_file(
        download_file_path, ExtraArgs=extra_args, Callback=transfer_callback
    )
    return transfer_callback.thread_info

```

Faites la démonstration des fonctions du gestionnaire de transfert et établissez un rapport des résultats.

```

import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError

```



```
from botocore.exceptions import ParamValidationError
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")

class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self.s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
files? "
```

```

    )
    if not os.path.isdir(self.demo_folder):
        print(f"{self.demo_folder} isn't a folder!")
        self.demo_folder = None

while not self.demo_bucket:
    self.demo_bucket = input(
        "Which Amazon S3 bucket do you want to use to store "
        "demonstration files? "
    )
    try:
        self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
    except ParamValidationError as err:
        print(err)
        self.demo_bucket = None
    except ClientError as err:
        print(err)
        print(
            f"Either {self.demo_bucket} doesn't exist or you don't "
            f"have access to it."
        )
        self.demo_bucket = None

def demo(
    self, question, upload_func, download_func, upload_args=None,
    download_args=None
):
    """Run a demonstration.

    Ask the user if they want to run this specific demonstration.
    If they say yes, create a file on the local path, upload it
    using the specified upload function, then download it using the
    specified download function.
    """
    if download_args is None:
        download_args = {}
    if upload_args is None:
        upload_args = {}
    question = question.format(self.file_size_mb)
    answer = input(f"{question} (y/n)")
    if answer.lower() == "y":
        local_file_path, object_key, download_file_path =
self._create_demo_file()

```

```
        file_transfer.TransferConfig = self._config_wrapper(
            TransferConfig, CONFIG_ATTRS
        )
        self._report_transfer_params(
            "Uploading", local_file_path, object_key, **upload_args
        )
        start_time = time.perf_counter()
        thread_info = upload_func(
            local_file_path,
            self.demo_bucket,
            object_key,
            self.file_size_mb,
            **upload_args,
        )
        end_time = time.perf_counter()
        self._report_transfer_result(thread_info, end_time - start_time)

        file_transfer.TransferConfig = self._config_wrapper(
            TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
        )
        self._report_transfer_params(
            "Downloading", object_key, download_file_path, **download_args
        )
        start_time = time.perf_counter()
        thread_info = download_func(
            self.demo_bucket,
            object_key,
            download_file_path,
            self.file_size_mb,
            **download_args,
        )
        end_time = time.perf_counter()
        self._report_transfer_result(thread_info, end_time - start_time)

    def last_name_set(self):
        """Get the name set used for the last demo."""
        return self._chore_list[-1]

    def cleanup(self):
        """
        Remove files from the demo folder, and uploaded objects from the
        Amazon S3 bucket.
        """
        print("-" * self._terminal_width)
```

```

        for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
            print(f"Removing {local_file_path}")
            try:
                os.remove(local_file_path)
            except FileNotFoundError as err:
                print(err)

            print(f"Removing {downloaded_file_path}")
            try:
                os.remove(downloaded_file_path)
            except FileNotFoundError as err:
                print(err)

            if self.demo_bucket:
                print(f"Removing {self.demo_bucket}:{s3_object_key}")
                try:

self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                    except ClientError as err:
                        print(err)

def _setup_platform_specific(self):
    """Set up platform-specific command used to create a large file."""
    if platform.system() == "Windows":
        self._create_file_cmd = "fsutil file createnew {} {}".format(
            self._size_multiplier, MB)
    elif platform.system() == "Linux" or platform.system() == "Darwin":
        self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={MB}
count={{}}"
        self._size_multiplier = 1
    else:
        raise EnvironmentError(
            f"Demo of platform {platform.system()} isn't supported."
        )

def _create_demo_file(self):
    """
    Create a file in the demo folder specified by the user. Store the local
    path, object name, and download path for later cleanup.

    Only the local file is created by this method. The Amazon S3 object and
    download file are created later during the demonstration.

```

```

Returns:
A tuple that contains the local file path, object name, and download
file path.
"""
file_name_template = "TestFile{}-{}.demo"
local_suffix = "local"
object_suffix = "s3object"
download_suffix = "downloaded"
file_tag = len(self._chore_list) + 1

local_file_path = os.path.join(
    self.demo_folder, file_name_template.format(file_tag, local_suffix)
)

s3_object_key = file_name_template.format(file_tag, object_suffix)

downloaded_file_path = os.path.join(
    self.demo_folder, file_name_template.format(file_tag,
download_suffix)
)

filled_cmd = self._create_file_cmd.format(
    local_file_path, self.file_size_mb * self._size_multiplier
)

print(
    f"Creating file of size {self.file_size_mb} MB "
    f"in {self.demo_folder} by running:"
)
print(f"{'':4}{filled_cmd}")
os.system(filled_cmd)

chore = (local_file_path, s3_object_key, downloaded_file_path)
self._chore_list.append(chore)
return chore

def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
    """Report configuration and extra arguments used for a file transfer."""
    print("-" * self._terminal_width)
    print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
    if kwargs:
        print("With extra args:")
        for arg, value in kwargs.items():
            print(f"{'':4}{arg:<20}: {value}')

```

```

@staticmethod
def ask_user(question):
    """
    Ask the user a yes or no question.

    Returns:
    True when the user answers 'y' or 'Y'; otherwise, False.
    """
    answer = input(f"{question} (y/n) ")
    return answer.lower() == "y"

@staticmethod
def _config_wrapper(func, config_attrs):
    def wrapper(*args, **kwargs):
        config = func(*args, **kwargs)
        print("With configuration:")
        for attr in config_attrs:
            print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
        return config

    return wrapper

@staticmethod
def _report_transfer_result(thread_info, elapsed):
    """Report the result of a transfer, including per-thread data."""
    print(f"\nUsed {len(thread_info)} threads.")
    for ident, byte_count in thread_info.items():
        print(f'{"":4}Thread {ident} copied {byte_count} bytes.')
    print(f"Your transfer took {elapsed:.2f} seconds.")

def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "

```

```
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

    # Upload and download with multipart_threshold set higher than the size of
    # the file. This causes the transfer manager to use standard transfers
    # instead of multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "
        "as a standard (not multipart) transfer?",
        file_transfer.upload_with_high_threshold,
        file_transfer.download_with_high_threshold,
    )

    # Upload with specific chunk size and additional metadata.
    # Download with a single thread.
    demo_manager.demo(
        "Do you want to upload a {} MB file with a smaller chunk size and "
        "then download the same file using a single thread?",
        file_transfer.upload_with_chunksize_and_meta,
        file_transfer.download_with_single_thread,
        upload_args={
            "metadata": {
                "upload_type": "chunky",
                "favorite_color": "aqua",
                "size": "medium",
            }
        },
    )

    # Upload using server-side encryption with customer-provided
    # encryption keys.
    # Generate a 256-bit key from a passphrase.
    sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
    demo_manager.demo(
        "Do you want to upload and download a {} MB file using "
        "server-side encryption?",
        file_transfer.upload_with_sse,
        file_transfer.download_with_sse,
        upload_args={"sse_key": sse_key},
        download_args={"sse_key": sse_key},
    )
```

```
# Download without specifying an encryption key to show that the
# encryption key must be included to download an encrypted object.
if demo_manager.ask_user(
    "Do you want to try to download the encrypted "
    "object without sending the required key?"
):
    try:
        _, object_key, download_file_path = demo_manager.last_name_set()
        file_transfer.download_with_default_configuration(
            demo_manager.demo_bucket,
            object_key,
            download_file_path,
            demo_manager.file_size_mb,
        )
    except ClientError as err:
        print(
            "Got expected error when trying to download an encrypted "
            "object without specifying encryption info:"
        )
        print(f"{'':4}{err}")

# Remove all created and downloaded files, remove all objects from
# S3 storage.
if demo_manager.ask_user(
    "Demonstration complete. Do you want to remove local files " "and S3
objects?"
):
    demo_manager.cleanup()

if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```


Rust

SDK pour Rust

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
use std::fs::File;
use std::io::prelude::*;
use std::path::Path;

use aws_config::meta::region::RegionProviderChain;
use aws_sdk_s3::error::DisplayErrorContext;
use aws_sdk_s3::operation::{
    create_multipart_upload::CreateMultipartUploadOutput,
    get_object::GetObjectOutput,
};
use aws_sdk_s3::types::{CompletedMultipartUpload, CompletedPart};
use aws_sdk_s3::{config::Region, Client as S3Client};
use aws_smithy_types::byte_stream::{ByteStream, Length};
use rand::distributions::Alphanumeric;
use rand::{thread_rng, Rng};
use s3_code_examples::error::S3ExampleError;
use std::process;
use uuid::Uuid;

//In bytes, minimum chunk size of 5MB. Increase CHUNK_SIZE to send larger chunks.
const CHUNK_SIZE: u64 = 1024 * 1024 * 5;
const MAX_CHUNKS: u64 = 10000;

#[tokio::main]
pub async fn main() {
    if let Err(err) = run_example().await {
        eprintln!("Error: {}", DisplayErrorContext(err));
        process::exit(1);
    }
}
```

```
async fn run_example() -> Result<(), S3ExampleError> {
    let shared_config = aws_config::load_from_env().await;
    let client = S3Client::new(&shared_config);

    let bucket_name = format!("amzn-s3-demo-bucket-{}", Uuid::new_v4());
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));
    let region = region_provider.region().await.unwrap();
    s3_code_examples::create_bucket(&client, &bucket_name, &region).await?;

    let key = "sample.txt".to_string();
    // Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
    // upload the file.
    let multipart_upload_res: CreateMultipartUploadOutput = client
        .create_multipart_upload()
        .bucket(&bucket_name)
        .key(&key)
        .send()
        .await?;

    let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
        "Missing upload_id after CreateMultipartUpload",
    ))?;

    //Create a file of random characters for the upload.
    let mut file = File::create(&key).expect("Could not create sample file.");
    // Loop until the file is 5 chunks.
    while file.metadata().unwrap().len() <= CHUNK_SIZE * 4 {
        let rand_string: String = thread_rng()
            .sample_iter(&Alphanumeric)
            .take(256)
            .map(char::from)
            .collect();
        let return_string: String = "\n".to_string();
        file.write_all(rand_string.as_ref())
            .expect("Error writing to file.");
        file.write_all(return_string.as_ref())
            .expect("Error writing to file.");
    }

    let path = Path::new(&key);
    let file_size = tokio::fs::metadata(path)
        .await
        .expect("it exists I swear")
    }
```

```
.len());

let mut chunk_count = (file_size / CHUNK_SIZE) + 1;
let mut size_of_last_chunk = file_size % CHUNK_SIZE;
if size_of_last_chunk == 0 {
    size_of_last_chunk = CHUNK_SIZE;
    chunk_count -= 1;
}

if file_size == 0 {
    return Err(S3ExampleError::new("Bad file size."));
}
if chunk_count > MAX_CHUNKS {
    return Err(S3ExampleError::new(
        "Too many chunks! Try increasing your chunk size.",
    ));
}

let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
```

```

        .await?;

    upload_parts.push(
        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
            .build(),
    );
}

// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;

let data: GetObjectOutput =
    s3_code_examples::download_object(&client, &bucket_name, &key).await?;
let data_length: u64 = data
    .content_length()
    .unwrap_or_default()
    .try_into()
    .unwrap();
if file.metadata().unwrap().len() == data_length {
    println!("Data lengths match.");
} else {
    println!("The data was not the same size!");
}

s3_code_examples::clear_bucket(&client, &bucket_name)
    .await
    .expect("Error emptying bucket.");
s3_code_examples::delete_bucket(&client, &bucket_name)
    .await
    .expect("Error deleting bucket.");

```

```
    ok(()  
}
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Chargez un flux de taille inconnue vers un objet Amazon S3 à l'aide d'un AWS Kit SDK

Les exemples de code suivants montrent comment charger un flux de taille inconnue dans un objet Amazon S3.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Utilisez le [client AWS CRT-based S3](#).

```
import com.example.s3.util.AsyncExampleUtils;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.core.async.AsyncRequestBody;  
import software.amazon.awssdk.core.exception.SdkException;  
import software.amazon.awssdk.services.s3.S3AsyncClient;  
import software.amazon.awssdk.services.s3.model.PutObjectResponse;  
  
import java.io.ByteArrayInputStream;  
import java.io.InputStream;  
import java.util.UUID;  
import java.util.concurrent.CompletableFuture;  
import java.util.concurrent.ExecutorService;  
import java.util.concurrent.Executors;
```

```

public class PutObjectFromStreamAsync {
    private static final Logger logger =
        LoggerFactory.getLogger(PutObjectFromStreamAsync.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.
        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            PutObjectFromStreamAsync example = new PutObjectFromStreamAsync();
            S3AsyncClient s3AsyncClientCrt = S3AsyncClient.crtCreate();
            PutObjectResponse putObjectResponse =
example.putObjectFromStreamCrt(s3AsyncClientCrt, bucketName, key);
            logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
            logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param s3CrtAsyncClient - To upload content from a stream of unknown
     size, use can the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
     Returns metadata pertaining to the put object operation.
     */
    public PutObjectResponse putObjectFromStreamCrt(S3AsyncClient
s3CrtAsyncClient, String bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
        characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
        InputStream inputStream = new
        ByteArrayInputStream(randomString.getBytes());

```

```
        // Executor required to handle reading from the InputStream on a separate
        thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();
        // Specify `null` for the content length when you don't know the content
        length.
        AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
        null, executor);

        CompletableFuture<PutObjectResponse> responseFuture =
            s3CrtAsyncClient.putObject(r -> r.bucket(bucketName).key(key),
        body);

        PutObjectResponse response = responseFuture.join(); // Wait for the
        response.
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        executor.shutdown();
        return response;
    }
}
```

Utilisez le [client S3 asynchrone standard avec le chargement partitionné](#) activé.

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.services.s3.S3AsyncClient;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class PutObjectFromStreamAsyncMp {
    private static final Logger logger =
        LoggerFactory.getLogger(PutObjectFromStreamAsyncMp.class);
```

```
public static void main(String[] args) {
    String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.
    String key = UUID.randomUUID().toString();

    AsyncExampleUtils.createBucket(bucketName);
    try {
        PutObjectFromStreamAsyncMp example = new
PutObjectFromStreamAsyncMp();
        S3AsyncClient s3AsyncClientMp =
S3AsyncClient.builder().multipartEnabled(true).build();
        PutObjectResponse putObjectResponse =
example.putObjectFromStreamMp(s3AsyncClientMp, bucketName, key);
        logger.info("Object {} etag: {}", key, putObjectResponse.eTag());
        logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
    } catch (SdkException e) {
        logger.error(e.getMessage(), e);
    } finally {
        AsyncExampleUtils.deleteObject(bucketName, key);
        AsyncExampleUtils.deleteBucket(bucketName);
    }
}

/**
 * @param s3AsyncClientMp - To upload content from a stream of unknown size,
use can the S3 asynchronous client with multipart enabled.
 * @param bucketName - The name of the bucket.
 * @param key - The name of the object.
 * @return software.amazon.awssdk.services.s3.model.PutObjectResponse -
Returns metadata pertaining to the put object operation.
 */
public PutObjectResponse putObjectFromStreamMp(S3AsyncClient s3AsyncClientMp,
String bucketName, String key) {

    // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
    String randomString = AsyncExampleUtils.randomString();
    logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
    InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());
```



```
// Executor required to handle reading from the InputStream on a separate
// thread so the main upload is not blocked.
ExecutorService executor = Executors.newSingleThreadExecutor();
// Specify 'null' for the content length when you don't know the content
// length.
AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
null, executor);

CompletableFuture<PutObjectResponse> responseFuture =
    s3AsyncClientMp.putObject(r -> r.bucket(bucketName).key(key),
body);

PutObjectResponse response = responseFuture.join(); // Wait for the
// response.
logger.info("Object {} uploaded to bucket {}.\"", key, bucketName);
executor.shutdown();
return response;
}
}
```

Utilisez le [Gestionnaire de transferts Amazon S3](#).

```
import com.example.s3.util.AsyncExampleUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.async.AsyncRequestBody;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedUpload;
import software.amazon.awssdk.transfer.s3.model.Upload;

import java.io.ByteArrayInputStream;
import java.io.InputStream;
import java.util.UUID;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class UploadStream {
    private static final Logger logger =
        LoggerFactory.getLogger(UploadStream.class);
```

```

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

        String key = UUID.randomUUID().toString();

        AsyncExampleUtils.createBucket(bucketName);
        try {
            UploadStream example = new UploadStream();
            CompletedUpload completedUpload =
example.uploadStream(S3TransferManager.create(), bucketName, key);
            logger.info("Object {} etag: {}", key,
completedUpload.response().eTag());
            logger.info("Object {} uploaded to bucket {}. ", key, bucketName);
        } catch (SdkException e) {
            logger.error(e.getMessage(), e);
        } finally {
            AsyncExampleUtils.deleteObject(bucketName, key);
            AsyncExampleUtils.deleteBucket(bucketName);
        }
    }

    /**
     * @param transferManager - To upload content from a stream of unknown size,
you can use the S3TransferManager based on the AWS CRT-based S3 client.
     * @param bucketName - The name of the bucket.
     * @param key - The name of the object.
     * @return - software.amazon.awssdk.transfer.s3.model.CompletedUpload - The
result of the completed upload.
     */
    public CompletedUpload uploadStream(S3TransferManager transferManager, String
bucketName, String key) {

        // AsyncExampleUtils.randomString() returns a random string up to 100
characters.
        String randomString = AsyncExampleUtils.randomString();
        logger.info("random string to upload: {}: length={}", randomString,
randomString.length());
        InputStream inputStream = new
ByteArrayInputStream(randomString.getBytes());

        // Executor required to handle reading from the InputStream on a separate
thread so the main upload is not blocked.
        ExecutorService executor = Executors.newSingleThreadExecutor();

```

```
// Specify `null` for the content length when you don't know the content
length.
AsyncRequestBody body = AsyncRequestBody.fromInputStream(inputStream,
null, executor);

Upload upload = transferManager.upload(builder -> builder
    .requestBody(body)
    .putObjectRequest(req -> req.bucket(bucketName).key(key))
    .build());

CompletedUpload completedUpload = upload.completionFuture().join();
executor.shutdown();
return completedUpload;
}
}
```

Swift

Kit SDK pour Swift

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import ArgumentParser
import AWSClientRuntime
import AWSS3
import Foundation
import Smithy
import SmithyHTTPAPI
import SmithyStreams

/// Upload a file to the specified bucket.
///
/// - Parameters:
///   - bucket: The Amazon S3 bucket name to store the file into.
///   - key: The name (or path) of the file to upload to in the `bucket`.
```

```
/// - sourcePath: The pathname on the local filesystem of the file to
///   upload.
func uploadFile(sourcePath: String, bucket: String, key: String?) async
throws {
    let fileURL: URL = URL(fileURLWithPath: sourcePath)
    let fileName: String

    // If no key was provided, use the last component of the filename.

    if key == nil {
        fileName = fileURL.lastPathComponent
    } else {
        fileName = key!
    }

    let s3Client = try await S3Client()

    // Create a FileHandle for the source file.

    let fileHandle = FileHandle(forReadingAtPath: sourcePath)
    guard let fileHandle = fileHandle else {
        throw TransferError.readError
    }

    // Create a byte stream to retrieve the file's contents. This uses the
    // Smithy FileStream and ByteStream types.

    let stream = FileStream(fileHandle: fileHandle)
    let body = ByteStream.stream(stream)

    // Create a `PutObjectInput` with the ByteStream as the body of the
    // request's data. The AWS SDK for Swift will handle sending the
    // entire file in chunks, regardless of its size.

    let putInput = PutObjectInput(
        body: body,
        bucket: bucket,
        key: fileName
    )

    do {
        _ = try await s3Client.putObject(input: putInput)
    } catch {
        throw TransferError.uploadError("Error uploading the file: \(error)")
    }
}
```

```
}  
  
    print("File uploaded to \"(fileURL.path).\"")  
}
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisez des checksums pour travailler avec un objet Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment utiliser des sommes de contrôle pour travailler avec un objet Amazon S3.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Les exemples de code utilisent un sous-ensemble des importations suivantes.

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.core.exception.SdkException;  
import software.amazon.awssdk.core.sync.RequestBody;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;  
import software.amazon.awssdk.services.s3.model.ChecksumMode;  
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;  
import software.amazon.awssdk.services.s3.model.CompletedPart;  
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;  
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
```

```
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.FileUpload;
import software.amazon.awssdk.transfer.s3.model.UploadFileRequest;

import java.io.FileInputStream;
import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.nio.file.Paths;
import java.security.DigestInputStream;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.Base64;
import java.util.List;
import java.util.Objects;
import java.util.UUID;
```

Spécifiez un algorithme de somme de contrôle pour la méthode `putObject` lorsque vous [créez l'élément `PutObjectRequest`](#).

```
public void putObjectWithChecksum(String bucketName, String key) {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```

Vérifiez la somme de contrôle de la `getObject` méthode lorsque vous [créez le `GetObjectRequest`](#).

```
public GetObjectResponse getObjectWithChecksum(String bucketName, String key)
{
    return s3Client.getObject(b -> b
```

```
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
    .response();
}
```

Pre-calculez une somme de contrôle pour la `putObject` méthode lorsque vous [créez le PutObjectRequest](#).

```
public void putObjectWithPrecalculatedChecksum(String bucketName, String key,
String filePath) {
    String checksum = calculateChecksum(filePath, "SHA-256");

    s3Client.putObject((b -> b
        .bucket(bucketName)
        .key(key)
        .checksumSHA256(checksum)),
        RequestBody.fromFile(Paths.get(filePath)));
}
```

Utilisez le [gestionnaire de transfert S3](#) situé au-dessus du [client AWS CRT-based S3](#) pour effectuer un téléchargement partitionné de manière transparente lorsque la taille du contenu dépasse un seuil. La taille par défaut est de 8 Mo.

Vous pouvez spécifier un algorithme de somme de contrôle que le kit SDK utilisera. Par défaut, le kit SDK utilise l'algorithme CRC32.

```
public void multipartUploadWithChecksumTm(String bucketName, String key,
String filePath) {
    S3TransferManager transferManager = S3TransferManager.create();
    UploadFileRequest uploadFileRequest = UploadFileRequest.builder()
        .putObjectRequest(b -> b
            .bucket(bucketName)
            .key(key)
            .checksumAlgorithm(ChecksumAlgorithm.SHA1))
        .source(Paths.get(filePath))
        .build();
    FileUpload fileUpload = transferManager.uploadFile(uploadFileRequest);
    fileUpload.completionFuture().join();
    transferManager.close();
}
```

```
}
```

Utilisez l'[API ou \(S3AsyncClient API\) S3Client](#) pour effectuer un téléchargement en plusieurs parties. Si vous spécifiez une somme de contrôle supplémentaire, vous devez spécifier l'algorithme à utiliser lors du lancement du chargement. Vous devez également spécifier l'algorithme pour chaque demande d'article et fournir la somme de contrôle calculée pour chaque article après son chargement.

```
public void multipartUploadWithChecksumS3Client(String bucketName, String
key, String filePath) {
    ChecksumAlgorithm algorithm = ChecksumAlgorithm.CRC32;

    // Initiate the multipart upload.
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(algorithm)); // Checksum specified on initiation.
    String uploadId = createMultipartUploadResponse.uploadId();

    // Upload the parts of the file.
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip(); // Swap position and limit before reading from the
buffer.

            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .checksumAlgorithm(algorithm) // Checksum specified on each
part.

                .partNumber(partNumber)
                .build();
```



```
UploadPartResponse partResponse = s3Client.uploadPart(
    uploadPartRequest,
    RequestBody.fromByteBuffer(bb));

CompletedPart part = CompletedPart.builder()
    .partNumber(partNumber)
    .checksumCRC32(partResponse.checksumCRC32()) // Provide the
calculated checksum.
    .eTag(partResponse.eTag())
    .build();
completedParts.add(part);

bb.clear();
position += read;
partNumber++;
}
} catch (IOException e) {
    System.err.println(e.getMessage());
}

// Complete the multipart upload.
s3Client.completeMultipartUpload(b -> b
    .bucket(bucketName)
    .key(key)
    .uploadId(uploadId)

.multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisez les fonctionnalités d'intégrité des objets d'Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment travailler avec les fonctionnalités d'intégrité des objets S3.

C++

Kit de développement logiciel (SDK) for C++

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif illustrant les fonctionnalités d'intégrité des objets Amazon S3.

```
//! Routine which runs the S3 object integrity workflow.
/*!
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
    const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
std::endl;
        return false;
    }

    Aws::String bucketName = TEST_BUCKET_PREFIX;
    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);
```

```

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
        std::cerr << "Workflow exiting because bucket creation failed." <<
std::endl;
        return false;
    }

    Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
    std::shared_ptr<Aws::S3::S3Client> client =
Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

    printAsterisksLine();
    std::cout << "Choose from one of the following checksum algorithms."
        << std::endl;

    for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
        std::cout << "  " << hashMethod << " - " <<
stringForHashMethod(hashMethod)
            << std::endl;
    }

    HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
DEFAULT,
                                                    SHA256);

    gUseCalculatedChecksum = !askYesNoQuestion(
        "Let the SDK calculate the checksum for you? (y/n) ");

    printAsterisksLine();

    std::cout << "The workflow will now upload a file using PutObject."
        << std::endl;
    std::cout << "Object integrity will be verified using the "
        << stringForHashMethod(chosenHashMethod) << " algorithm."
        << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout
            << "A checksum computed by this workflow will be used for object
integrity verification,"
            << std::endl;
        std::cout << "except for the TransferManager upload." << std::endl;
    } else {
        std::cout

```

```

        << "A checksum computed by the SDK will be used for object
integrity verification."
        << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();

    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                     TEST_FILE,
                                     std::ios_base::in |
                                     std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Hasher hasher;
    HASH_METHOD putObjectHashMethod = chosenHashMethod;
    if (putObjectHashMethod == DEFAULT) {
        putObjectHashMethod = MD5; // MD5 is the default hash method for
PutObject.

        std::cout << "The default checksum algorithm for PutObject is "
                   << stringForHashMethod(putObjectHashMethod)
                   << std::endl;
    }

    // Demonstrate in code how the hash is computed.
    if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
        std::cerr << "Error calculating hash for file " << TEST_FILE <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }
    Aws::String key = stringForHashMethod(putObjectHashMethod);
    key += "_";
    key += TEST_FILE_KEY;
    Aws::String localHash = hasher.getBase64HashString();

    // Upload the object with PutObject

```

```
    if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                          inputData, chosenHashMethod == DEFAULT,
                          *client)) {
        std::cerr << "Error putting file " << TEST_FILE << " to bucket "
                   << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    Aws::String retrievedHash;
    if (!retrieveObjectHash(bucketName, key,
                          putObjectHashMethod, retrievedHash,
                          nullptr, *client)) {
        std::cerr << "Error getting file " << TEST_FILE << " from bucket "
                   << bucketName << " with key " << key << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    explainPutObjectResults();
    verifyHashingResults(retrievedHash, hasher,
                        "PutObject upload", putObjectHashMethod);

    printAsterisksLine();
    pressEnterToContinue();

    key = "tr_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    introductoryTransferManagerUploadExplanations(key);

    HASH_METHOD transferManagerHashMethod = chosenHashMethod;
    if (transferManagerHashMethod == DEFAULT) {
        transferManagerHashMethod = CRC32; // The default hash method for the
        TransferManager is CRC32.

        std::cout << "The default checksum algorithm for TransferManager is "
                  << stringForHashMethod(transferManagerHashMethod)
                  << std::endl;
    }

    // Upload the large file using the transfer manager.
```

```
    if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
        chosenHashMethod == DEFAULT,
                                client)) {
        std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::vector<Aws::String> retrievedTransferManagerPartHashes;
    Aws::String retrievedTransferManagerFinalHash;

    // Retrieve all the hashes for the TransferManager upload.
    if (!retrieveObjectHash(bucketName, key,
                            transferManagerHashMethod,
                            retrievedTransferManagerFinalHash,
                            &retrievedTransferManagerPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    AwsDoc::S3::Hasher locallyCalculatedFinalHash;
    std::vector<Aws::String> locallyCalculatedPartHashes;

    // Calculate the hashes locally to demonstrate how TransferManager hashes are
    computed.
    if (!calculatePartHashesForFile(transferManagerHashMethod,
        MULTI_PART_TEST_FILE,
                                UPLOAD_BUFFER_SIZE,
                                locallyCalculatedFinalHash,
                                locallyCalculatedPartHashes)) {
        std::cerr << "Exiting because of an error in calculatePartHashesForFile."
<< std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedTransferManagerFinalHash,
                        locallyCalculatedFinalHash, "TransferManager upload",
                        transferManagerHashMethod,
                        retrievedTransferManagerPartHashes,
                        locallyCalculatedPartHashes);
```

```
printAsterisksLine();

key = "mp_";
key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

multiPartUploadExplanations(key, chosenHashMethod);

pressEnterToContinue();

std::shared_ptr<Aws::IOStream> largeFileInputData =
    Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                  MULTI_PART_TEST_FILE,
                                  std::ios_base::in |
                                  std::ios_base::binary);

if (!largeFileInputData->good()) {
    std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

HASH_METHOD multipartUploadHashMethod = chosenHashMethod;
if (multipartUploadHashMethod == DEFAULT) {
    multipartUploadHashMethod = MD5; // The default hash method for
    multipart uploads is MD5.

    std::cout << "The default checksum algorithm for multipart upload is "
                << stringForHashMethod(putObjectHashMethod)
                << std::endl;
}

AwsDoc::S3::Hasher hashData;
std::vector<Aws::String> partHashes;

if (!doMultipartUpload(bucketName, key,
                       multipartUploadHashMethod,
                       largeFileInputData, chosenHashMethod == DEFAULT,
                       hashData,
                       partHashes,
                       *client)) {
    std::cerr << "Exiting because of an error in doMultipartUpload." <<
    std::endl;
    cleanUp(bucketName, clientConfiguration);
}
```

```

        return false;
    }

    std::cout << "Finished multipart upload of with hash method " <<
        stringForHashMethod(multipartUploadHashMethod) << std::endl;

    std::cout << "Now we will retrieve the checksums from the server." <<
std::endl;

    retrievedHash.clear();
    std::vector<Aws::String> retrievedPartHashes;
    if (!retrieveObjectHash(bucketName, key,
        multipartUploadHashMethod,
        retrievedHash, &retrievedPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
multipart." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
        multipartUploadHashMethod,
        retrievedPartHashes, partHashes);

    printAsterisksLine();

    if (askYesNoQuestion("Would you like to delete the resources created in this
workflow? (y/n)")) {
        return cleanUp(bucketName, clientConfiguration);
    } else {
        std::cout << "The bucket " << bucketName << " was not deleted." <<
std::endl;
        return true;
    }
}

//! Routine which uploads an object to an S3 bucket with different object
integrity hashing methods.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashData: The hash value that will be associated with the uploaded
object.

```



```

    \param hashMethod: The hashing algorithm to use when calculating the hash
    value.
    \param body: The data content of the object being uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default hash
    method or the one specified in the hashMethod parameter.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
    &key,
                                   const Aws::String &hashData,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream> &body,
                                   bool useDefaultHashMethod,
                                   const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {
            request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                request.SetContentMD5(hashData);
                break;
            case AwsDoc::S3::SHA1:
                request.SetChecksumSHA1(hashData);
                break;
            case AwsDoc::S3::SHA256:
                request.SetChecksumSHA256(hashData);
                break;
            case AwsDoc::S3::CRC32:
                request.SetChecksumCRC32(hashData);
                break;
            case AwsDoc::S3::CRC32C:
                request.SetChecksumCRC32C(hashData);
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
        }
    }
}

```

```

        return false;
    }
}
request.SetBody(body);
Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
body->seekg(0, body->beg);
if (outcome.IsSuccess()) {
    std::cout << "Object successfully uploaded." << std::endl;
} else {
    std::cerr << "Error uploading object." <<
        outcome.GetError().GetMessage() << std::endl;
}
return outcome.IsSuccess();
}

// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of
the object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
&key,

                                AwsDoc::S3::HASH_METHOD hashMethod,
                                Aws::String &hashData,
                                std::vector<Aws::String> *partHashes,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(

```

```

        request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        hashData = result.GetETag();
    } else {
        std::cerr << "Error retrieving object etag attributes." <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} else { // hashMethod != MD5
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
            case AwsDoc::S3::MD5:
                break; // MD5 is not supported.
#pragma clang diagnostic pop
            case AwsDoc::S3::SHA1:
                hashData = result.GetChecksum().GetChecksumSHA1();
                break;
            case AwsDoc::S3::SHA256:
                hashData = result.GetChecksum().GetChecksumSHA256();
                break;
            case AwsDoc::S3::CRC32:
                hashData = result.GetChecksum().GetChecksumCRC32();
                break;
            case AwsDoc::S3::CRC32C:
                hashData = result.GetChecksum().GetChecksumCRC32C();
                break;
            default:
                std::cerr << "Unknown hash method." << std::endl;
                return false;
        }
    }
}

```

```

    }
} else {
    std::cerr << "Error retrieving object checksum attributes." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

if (nullptr != partHashes) {
    attributes.clear();
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
    request.SetObjectAttributes(attributes);
    outcome = client.GetObjectAttributes(request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
        for (const Aws::S3::Model::ObjectPart &part: parts) {
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                    break;
                case AwsDoc::S3::MD5: // MD5 is not supported.
                    break;
                case AwsDoc::S3::SHA1:
                    partHashes->push_back(part.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:
                    partHashes->push_back(part.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:
                    partHashes->push_back(part.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:
                    partHashes->push_back(part.GetChecksumCRC32C());
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        }
    } else {
        std::cerr << "Error retrieving object attributes for object
parts." <<

```

```

        outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

return true;
}

//! Verifies the hashing results between the retrieved and local hashes.
/*!
\param retrievedHash The hash value retrieved from the remote source.
\param localHash The hash value calculated locally.
\param uploadtype The type of upload (e.g., "multipart", "single-part").
\param hashMethod The hashing method used (e.g., MD5, SHA-256).
\param retrievedPartHashes (Optional) The list of hashes for the individual
parts retrieved from the remote source.
\param localPartHashes (Optional) The list of hashes for the individual parts
calculated locally.
*/
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                     const Hasher &localHash,
                                     const Aws::String &uploadtype,
                                     HASH_METHOD hashMethod,
                                     const std::vector<Aws::String>
&retrievedPartHashes,
                                     const std::vector<Aws::String>
&localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash
<< std::endl;
    if (!retrievedPartHashes.empty()) {
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
retrieved."
<< std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << " Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
    if (hashMethod == MD5) {
        hashString = localHash.getHexHashString();
        if (!localPartHashes.empty()) {
            hashString += "-" + std::to_string(localPartHashes.size());
        }
    }
}

```

```

    } else {
        hashString = localHash.getBase64HashString();
    }

    bool allMatch = true;
    if (hashString != retrievedHash) {
        std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
std::endl;
        std::cerr << "Local hash- '" << hashString << "'" << std::endl;
        std::cerr << "Remote hash - '" << retrievedHash << "'" << std::endl;
        allMatch = false;
    }

    if (hashMethod != MD5) {
        if (localPartHashes.size() != retrievedPartHashes.size()) {
            std::cerr << "For " << uploadtype << ", the number of part hashes do
not match" << std::endl;
            std::cerr << "Local number of hashes- '" << localPartHashes.size() <<
""
                << std::endl;
            std::cerr << "Remote number of hashes - '"
                << retrievedPartHashes.size()
                << "'" << std::endl;
        }

        for (int i = 0; i < localPartHashes.size(); ++i) {
            if (localPartHashes[i] != retrievedPartHashes[i]) {
                std::cerr << "For " << uploadtype << ", the part hashes do not
match for part " << i + 1
                    << "." << std::endl;
                std::cerr << "Local hash- '" << localPartHashes[i] << "'"
                    << std::endl;
                std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                    << std::endl;
                allMatch = false;
            }
        }
    }

    if (allMatch) {
        std::cout << "For " << uploadtype << ", locally and remotely calculated
hashes all match!" << std::endl;
    }
}

```

```

}

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                        const std::shared_ptr<const
                                        Aws::Transfer::TransferHandle> &,
                                        const
                                        Aws::Client::AWSError<Aws::S3::S3Errors> &err) {
    std::cerr << "Error during transfer: '" << err.GetMessage() << "'" <<
    std::endl;
}

static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                        const std::shared_ptr<const
                                        Aws::Transfer::TransferHandle> &handle) {
    if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
        std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
        std::endl;
    }
}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's
    Transfer Manager.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash
    value.
    \param useDefaultHashMethod: A flag indicating whether to use the default hash
    method or the one specified in the hashMethod parameter.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool
AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
&key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   bool useDefaultHashMethod,
                                   const std::shared_ptr<Aws::S3::S3Client>
&client) {
    std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
    Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
        "executor", 25);
    Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
    transfer_config.s3Client = client;

```

```

    transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
    if (!useDefaultHashMethod) {
        if (hashMethod == MD5) {
            transfer_config.computeContentMD5 = true;
        } else {
            transfer_config.checksumAlgorithm =
getChecksumAlgorithmForHashMethod(
                hashMethod);
        }
    }
    transfer_config.errorCallback = transferManagerErrorCallback;
    transfer_config.transferStatusUpdatedCallback =
transferManagerStatusCallback;

    std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
Aws::Transfer::TransferManager::Create(
        transfer_config);

    std::cout << "Uploading the file..." << std::endl;
    std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle =
transfer_manager->UploadFile(MULTI_PART_TEST_FILE,

        bucket, key,

        "text/plain",

        Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success =
        uploadHandle->GetStatus() ==
Aws::Transfer::TransferStatus::COMPLETED;
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle-
>GetLastError();
        std::cerr << "File upload failed:  " << err.GetMessage() << std::endl;
    }

    return success;
}

//! Routine which calculates the hash values for each part of a file being
    uploaded to an S3 bucket.
/*!

```



```

    \param hashMethod: The hashing algorithm to use when calculating the hash
    values.
    \param fileName: The path to the file for which the part hashes will be
    calculated.
    \param bufferSize: The size of the buffer to use when reading the file.
    \param[out] hashDataResult: The Hasher object that will store the concatenated
    hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
    for each part of the file.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                             const Aws::String &fileName,
                                             size_t bufferSize,
                                             AwsDoc::S3::Hasher &hashDataResult,
                                             std::vector<Aws::String> &partHashes)
{
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
    size_t uploadedBytes = 0;

    while (uploadedBytes < objectSize) {
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(), objectSize -
uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
            Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                           &preallocatedStreamBuf);

        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
        Aws::String base64HashString = hasher.getBase64HashString();
    }
}

```

```

        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getBytesBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
            hashBuffer.GetUnderlyingData(),
            hashBuffer.GetUnderlyingData() +
            hashBuffer.GetLength());

        uploadedBytes += bytesToRead;
    }

    return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String
    &key,
                                Aws::S3::Model::ChecksumAlgorithm
checksumAlgorithm,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
        client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {

```

```

        std::cerr << "Error creating multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the
checksum algorithm,
                                ignored when it is an empty string.
    \param body: An shared_ptr IOStream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String
&bucket,
                                                    const Aws::String &key,
                                                    const Aws::String
&uploadID,
                                                    int partNumber,
                                                    Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                    const Aws::String
&calculatedHash,
                                                    const
std::shared_ptr<Aws::IOStream> &body,
                                                    const Aws::S3::S3Client
&client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
}

```

```

    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
                                       const Aws::String &uploadID,
                                       const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =

```

```

        client.AbortMultipartUpload(request);

        if (outcome.IsSuccess()) {
            std::cout << "Multipart upload aborted." << std::endl;
        } else {
            std::cerr << "Error aborting multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
        }

        return outcome.IsSuccess();
    }

    //! Complete a multipart upload to an S3 bucket.
    /*!
        \param bucket: The name of the S3 bucket where the object will be uploaded.
        \param key: The unique identifier (key) for the object within the S3 bucket.
        \param uploadID: An upload ID string.
        \param parts: A vector of CompleteParts.
        \param client: The S3 client instance used to perform the upload operation.
        \return CompleteMultipartUploadOutcome: The request outcome.
    */
    Aws::S3::Model::CompleteMultipartUploadOutcome
    AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

        const Aws::String &key,

        const Aws::String &uploadID,

        const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

        const Aws::S3::S3Client &client) {
        Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
        completedMultipartUpload.SetParts(parts);

        Aws::S3::Model::CompleteMultipartUploadRequest request;
        request.SetBucket(bucket);
        request.SetKey(key);
        request.SetUploadId(uploadID);
        request.SetMultipartUpload(completedMultipartUpload);

        Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
            client.CompleteMultipartUpload(request);

        if (!outcome.IsSuccess()) {

```

```

        std::cerr << "Error completing multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

//! Routine which performs a multi-part upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash
value.
    \param ioStream: An IOStream for the data to be uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default
hash method or the one specified in the hashMethod parameter.
    \param[out] hashDataResult: The Hasher object that will store the
concatenated hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
for each part of the file.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                   const Aws::String &key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream>
&ioStream,
                                   bool useDefaultHashMethod,
                                   AwsDoc::S3::Hasher &hashDataResult,
                                   std::vector<Aws::String> &partHashes,
                                   const Aws::S3::S3Client &client) {

    // Get object size.
    ioStream->seekg(0, ioStream->end);
    size_t objectSize = ioStream->tellg();
    ioStream->seekg(0, ioStream->beg);

    Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {
            checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
        }
    }
}

```

```

    Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
client);
    if (uploadID.empty()) {
        return false;
    }

    std::vector<unsigned char> totalHashBuffer;
    bool uploadSucceeded = true;
    std::streamsize uploadedBytes = 0;
    int partNumber = 1;
    Aws::Vector<Aws::S3::Model::CompletedPart> parts;
    while (uploadedBytes < objectSize) {
        std::cout << "Uploading part " << partNumber << "." << std::endl;

        std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
        std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),
objectSize - uploadedBytes));
        ioStream->read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),
bytesToRead);
        std::shared_ptr<Aws::IOStream> body =
            Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                &preallocatedStreamBuf);

        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            uploadSucceeded = false;
            break;
        }

        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
                                hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

```

```

    Aws::String calculatedHash;
    if (gUseCalculatedChecksum) {
        calculatedHash = base64HashString;
    }
    Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,

checksumAlgorithm, base64HashString, body,

                                                                    client);

    if (uploadPartOutcome.IsSuccess()) {
        const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
        Aws::S3::Model::CompletedPart completedPart;
        completedPart.SetETag(uploadPartResult.GetETag());
        completedPart.SetPartNumber(partNumber);
        switch (hashMethod) {
            case AwsDoc::S3::MD5:
                break; // Do nothing.
            case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                break;
            case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                break;
            case AwsDoc::S3::CRC32:

completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                break;
            case AwsDoc::S3::CRC32C:

completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                break;
            default:
                std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                break;
        }

        parts.push_back(completedPart);
    } else {
        std::cerr << "Error uploading part. " <<

```



```

        uploadPartOutcome.GetError().GetMessage() << std::endl;
        uploadSucceeded = false;
        break;
    }

    uploadedBytes += bytesToRead;
    partNumber++;
}

if (!uploadSucceeded) {
    abortMultipartUpload(bucket, key, uploadID, client);
    return false;
} else {

    Aws::S3::Model::CompleteMultipartUploadOutcome
completeMultipartUploadOutcome = completeMultipartUpload(bucket,

                                key,

                                uploadID,

                                parts,

                                client);

    if (completeMultipartUploadOutcome.IsSuccess()) {
        std::cout << "Multipart upload completed." << std::endl;
        if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod))
    {
        std::cerr << "Error calculating hash." << std::endl;
        return false;
    }
    } else {
        std::cerr << "Error completing multipart upload." <<
            completeMultipartUploadOutcome.GetError().GetMessage()
            << std::endl;
    }

    return completeMultipartUploadOutcome.IsSuccess();
}
}

//! Routine which retrieves the string for a HASH_METHOD constant.
/*!
```

```

    \param: hashMethod: A HASH_METHOD constant.
    \return: String: A string description of the hash method.
*/
Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            return "Default";
        case AwsDoc::S3::MD5:
            return "MD5";
        case AwsDoc::S3::SHA1:
            return "SHA1";
        case AwsDoc::S3::SHA256:
            return "SHA256";
        case AwsDoc::S3::CRC32:
            return "CRC32";
        case AwsDoc::S3::CRC32C:
            return "CRC32C";
        default:
            return "Unknown";
    }
}

//! Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
/*!
    \param: hashMethod: A HASH_METHOD constant.
    \return: ChecksumAlgorithm: The ChecksumAlgorithm enum.
*/
Aws::S3::Model::ChecksumAlgorithm
AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod)
{
    Aws::S3::Model::ChecksumAlgorithm result =
    Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    switch (hashMethod) {
        case AwsDoc::S3::DEFAULT:
            std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not
valid." << std::endl;
            break; // Default is not supported.
        case AwsDoc::S3::MD5:
            break; // Ignore MD5.
        case AwsDoc::S3::SHA1:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA1;
            break;
        case AwsDoc::S3::SHA256:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA256;

```

```

        break;
    case AwsDoc::S3::CRC32:
        result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
        break;
    case AwsDoc::S3::CRC32C:
        result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
        break;
    default:
        std::cerr << "Unknown hash method." << std::endl;
        break;
}

return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                        const Aws::S3::S3ClientConfiguration
                        &clientConfiguration) {

    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                                clientConfiguration);
        }
    } else {
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!
    \param bucketName: The name of the S3 bucket to use.
*/

```

```
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout
        << "Welcome to the Amazon Simple Storage Service (Amazon S3) object
integrity workflow."
        << std::endl;
    printAsterisksLine();
    std::cout
        << "This workflow demonstrates how Amazon S3 uses checksum values to
verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
        << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
        << "By default it calculates a checksum that is uploaded with an
object.\n"
        << "The default checksum algorithm for PutObject and MultiPart upload
is an MD5 hash.\n"
        << "The default checksum algorithm for TransferManager uploads is a
CRC32 checksum."
        << std::endl;
    std::cout
        << "You can override the default behavior, requiring one of the
following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting
the SDK calculate the value."
        << std::endl;
    std::cout
        << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
        << std::endl;

    std::cout
        << "This workflow will locally compute checksums for files uploaded
to an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
        << "This is done to provide demonstration code for how the checksums
are calculated."
        << std::endl;
    std::cout << "A bucket named '" << bucketName << "' will be created for the
object uploads."
        << std::endl;
```

```
}

//! Console interaction which explains the PutObject results.
/*!
*/
void AwsDoc::S3::explainPutObjectResults() {

    std::cout << "The upload was successful.\n";
    std::cout << "If the checksums had not matched, the upload would have failed."
               << std::endl;
    std::cout
               << "The checksums calculated by the server have been retrieved using the GetObjectAttributes."
               << std::endl;
    std::cout
               << "The locally calculated checksums have been verified against the retrieved checksums."
               << std::endl;
}

//! Console interaction explaining transfer manager uploads.
/*!
\param objectKey: The key for the object being uploaded.
*/
void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
    const Aws::String &objectKey) {
    std::cout
               << "Now the workflow will demonstrate object integrity for TransferManager multi-part uploads."
               << std::endl;
    std::cout
               << "The AWS C++ SDK has a TransferManager class which simplifies multipart uploads."
               << std::endl;
    std::cout
               << "The following code lets the TransferManager handle much of the checksum configuration."
               << std::endl;

    std::cout << "An object with the key '" << objectKey
               << " will be uploaded by the TransferManager using a "
               << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
    if (gUseCalculatedChecksum) {
```

```

        std::cout << "For TransferManager uploads, this demo always lets the SDK
calculate the hash value."
                << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();
}

//! Console interaction explaining multi-part uploads.
/*!
\param objectKey: The key for the object being uploaded.
\param chosenHashMethod: The hash method selected by the user.
*/
void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                              HASH_METHOD chosenHashMethod) {

    std::cout
        << "Now we will provide an in-depth demonstration of multi-part
uploading by calling the multi-part upload APIs directly."
        << std::endl;
    std::cout << "These are the same APIs used by the TransferManager when
uploading large files."
        << std::endl;
    std::cout
        << "In the following code, the checksums are also calculated locally
and then compared."
        << std::endl;
    std::cout
        << "For multi-part uploads, a checksum is uploaded with each part.
The final checksum is a concatenation of"
        << std::endl;
    std::cout << "the checksums for each part." << std::endl;
    std::cout
        << "This is explained in the user guide, https://docs.aws.amazon.com/
AmazonS3/latest/userguide/checking-object-integrity.html,"
        << " in the section \"Using part-level checksums for multipart
uploads\"." << std::endl;

    std::cout << "Starting multipart upload of with hash method " <<
        stringForHashMethod(chosenHashMethod) << " uploading to with object
key\n"
        << "'" << objectKey << "'," << std::endl;
}

```

```
//! Create a large file for doing multi-part uploads.
/*!
*/
bool AwsDoc::S3::createLargeFileIfNotExists() {
    // Generate a large file by writing this source file multiple times to a new
    file.
    if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
        return true;
    }

    std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out

                                | std::ios::binary);

    if (!newFile) {
        std::cerr << "createLargeFileIfNotExists- Error creating file " <<
MULTI_PART_TEST_FILE <<
        std::endl;
        return false;
    }

    std::ifstream input(TEST_FILE, std::ios::in

                                | std::ios::binary);

    if (!input) {
        std::cerr << "Error opening file " << TEST_FILE <<
        std::endl;
        return false;
    }
    std::stringstream buffer;
    buffer << input.rdbuf();

    input.close();

    while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
        buffer.seekg(std::stringstream::beg);
        newFile << buffer.rdbuf();
    }

    newFile.close();

    return true;
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour C++ .
 - [AbortMultipartUpload](#)
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [DeleteObject](#)
 - [GetObjectAttributes](#)
 - [PutObject](#)
 - [UploadPart](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Travaillez avec des objets versionnés Amazon S3 à l'aide d'un AWS Kit SDK

L'exemple de code suivant illustre comment :

- Créez un compartiment S3 versionné.
- Obtenez toutes les versions d'un objet.
- Rétablissez un objet à une version précédente.
- Supprimez et restaurez un objet versionné.
- Supprimez définitivement toutes les versions d'un objet.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez des fonctions qui enveloppent les actions S3.

```
def create_versioned_bucket(bucket_name, prefix):
    """
    Creates an Amazon S3 bucket, enables it for versioning, and configures a
    lifecycle
    that expires noncurrent object versions after 7 days.

    Adding a lifecycle configuration to a versioned bucket is a best practice.
    It helps prevent objects in the bucket from accumulating a large number of
    noncurrent versions, which can slow down request performance.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket_name: The name of the bucket to create.
    :param prefix: Identifies which objects are automatically expired under the
                    configured lifecycle rules.
    :return: The newly created bucket.
    """
    try:
        bucket = s3.create_bucket(
            Bucket=bucket_name,
            CreateBucketConfiguration={
                "LocationConstraint": s3.meta.client.meta.region_name
            },
        )
        logger.info("Created bucket %s.", bucket.name)
    except ClientError as error:
        if error.response["Error"]["Code"] == "BucketAlreadyOwnedByYou":
            logger.warning("Bucket %s already exists! Using it.", bucket_name)
            bucket = s3.Bucket(bucket_name)
        else:
            logger.exception("Couldn't create bucket %s.", bucket_name)
            raise

    try:
        bucket.Versioning().enable()
        logger.info("Enabled versioning on bucket %s.", bucket.name)
    except ClientError:
        logger.exception("Couldn't enable versioning on bucket %s.", bucket.name)
        raise

    try:
```

```

        expiration = 7
        bucket.LifecycleConfiguration().put(
            LifecycleConfiguration={
                "Rules": [
                    {
                        "Status": "Enabled",
                        "Prefix": prefix,
                        "NoncurrentVersionExpiration": {"NoncurrentDays":
expiration},
                    }
                ]
            }
        )
        logger.info(
            "Configured lifecycle to expire noncurrent versions after %s days "
            "on bucket %s.",
            expiration,
            bucket.name,
        )
    except ClientError as error:
        logger.warning(
            "Couldn't configure lifecycle on bucket %s because %s. "
            "Continuing anyway.",
            bucket.name,
            error,
        )

    return bucket

def rollback_object(bucket, object_key, version_id):
    """
    Rolls back an object to an earlier version by deleting all versions that
    occurred after the specified rollback version.

    Usage is shown in the usage_demo_single_object function at the end of this
    module.

    :param bucket: The bucket that holds the object to roll back.
    :param object_key: The object to roll back.
    :param version_id: The version ID to roll back to.
    """
    # Versions must be sorted by last_modified date because delete markers are

```

```

# at the end of the list even when they are interspersed in time.
versions = sorted(
    bucket.object_versions.filter(Prefix=object_key),
    key=attrgetter("last_modified"),
    reverse=True,
)

logger.debug(
    "Got versions:\n%s",
    "\n".join(
        [
            f"\t{version.version_id}, last modified {version.last_modified}"
            for version in versions
        ]
    ),
)

if version_id in [ver.version_id for ver in versions]:
    print(f"Rolling back to version {version_id}")
    for version in versions:
        if version.version_id != version_id:
            version.delete()
            print(f"Deleted version {version.version_id}")
        else:
            break

    print(f"Active version is now {bucket.Object(object_key).version_id}")
else:
    raise KeyError(
        f"{version_id} was not found in the list of versions for "
        f"{object_key}."
    )

def revive_object(bucket, object_key):
    """
    Revives a versioned object that was deleted by removing the object's active
    delete marker.
    A versioned object presents as deleted when its latest version is a delete
    marker.
    By removing the delete marker, we make the previous version the latest
    version
    and the object then presents as not deleted.
    """

```

Usage is shown in the `usage_demo_single_object` function at the end of this module.

```

:param bucket: The bucket that contains the object.
:param object_key: The object to revive.
"""
# Get the latest version for the object.
response = s3.meta.client.list_object_versions(
    Bucket=bucket.name, Prefix=object_key, MaxKeys=1
)

if "DeleteMarkers" in response:
    latest_version = response["DeleteMarkers"][0]
    if latest_version["IsLatest"]:
        logger.info(
            "Object %s was indeed deleted on %s. Let's revive it.",
            object_key,
            latest_version["LastModified"],
        )
        obj = bucket.Object(object_key)
        obj.Version(latest_version["VersionId"]).delete()
        logger.info(
            "Revived %s, active version is now %s with body '%s'",
            object_key,
            obj.version_id,
            obj.get()["Body"].read(),
        )
    else:
        logger.warning(
            "Delete marker is not the latest version for %s!", object_key
        )
elif "Versions" in response:
    logger.warning("Got an active version for %s, nothing to do.",
object_key)
else:
    logger.error("Couldn't get any version info for %s.", object_key)

def permanently_delete_object(bucket, object_key):
    """
    Permanently deletes a versioned object by deleting all of its versions.

```

Usage is shown in the `usage_demo_single_object` function at the end of this module.

```
:param bucket: The bucket that contains the object.
:param object_key: The object to delete.
"""
try:
    bucket.object_versions.filter(Prefix=object_key).delete()
    logger.info("Permanently deleted all versions of object %s.", object_key)
except ClientError:
    logger.exception("Couldn't delete all versions of %s.", object_key)
    raise
```

Chargez la strophe d'un poème dans un objet versionné et effectuez une série d'actions sur celle-ci.

```
def usage_demo_single_object(obj_prefix="demo-versioning/"):
    """
    Demonstrates usage of versioned object functions. This demo uploads a stanza
    of a poem and performs a series of revisions, deletions, and revivals on it.

    :param obj_prefix: The prefix to assign to objects created by this demo.
    """
    with open("father_william.txt") as file:
        stanzas = file.read().split("\n\n")

    width = get_terminal_size((80, 20))[0]
    print("-" * width)
    print("Welcome to the usage demonstration of Amazon S3 versioning.")
    print(
        "This demonstration uploads a single stanza of a poem to an Amazon "
        "S3 bucket and then applies various revisions to it."
    )
    print("-" * width)
    print("Creating a version-enabled bucket for the demo...")
    bucket = create_versioned_bucket("bucket-" + str(uuid.uuid1()), obj_prefix)

    print("\nThe initial version of our stanza:")
    print(stanzas[0])
```

```

# Add the first stanza and revise it a few times.
print("\nApplying some revisions to the stanza...")
obj_stanza_1 = bucket.Object(f"{obj_prefix}stanza-1")
obj_stanza_1.put(Body=bytes(stanzas[0], "utf-8"))
obj_stanza_1.put(Body=bytes(stanzas[0].upper(), "utf-8"))
obj_stanza_1.put(Body=bytes(stanzas[0].lower(), "utf-8"))
obj_stanza_1.put(Body=bytes(stanzas[0][::-1], "utf-8"))
print(
    "The latest version of the stanza is now:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Versions are returned in order, most recent first.
obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
print(
    "The version data of the stanza revisions:",
    *[
        f"    {version.version_id}, last modified {version.last_modified}"
        for version in obj_stanza_1_versions
    ],
    sep="\n",
)

# Rollback two versions.
print("\nRolling back two versions...")
rollback_object(bucket, obj_stanza_1.key, list(obj_stanza_1_versions)
[2].version_id)
print(
    "The latest version of the stanza:",
    obj_stanza_1.get()["Body"].read().decode("utf-8"),
    sep="\n",
)

# Delete the stanza
print("\nDeleting the stanza...")
obj_stanza_1.delete()
try:
    obj_stanza_1.get()
except ClientError as error:
    if error.response["Error"]["Code"] == "NoSuchKey":
        print("The stanza is now deleted (as expected).")
    else:

```

```
        raise

    # Revive the stanza
    print("\nRestoring the stanza...")
    revive_object(bucket, obj_stanza_1.key)
    print(
        "The stanza is restored! The latest version is again:",
        obj_stanza_1.get()["Body"].read().decode("utf-8"),
        sep="\n",
    )

    # Permanently delete all versions of the object. This cannot be undone!
    print("\nPermanently deleting all versions of the stanza...")
    permanently_delete_object(bucket, obj_stanza_1.key)
    obj_stanza_1_versions =
bucket.object_versions.filter(Prefix=obj_stanza_1.key)
    if len(list(obj_stanza_1_versions)) == 0:
        print("The stanza has been permanently deleted and now has no versions.")
    else:
        print("Something went wrong. The stanza still exists!")

    print(f"\nRemoving {bucket.name}...")
    bucket.delete()
    print(f"{bucket.name} deleted.")
    print("Demo done!")
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Python (Boto3).
 - [CreateBucket](#)
 - [DeleteObject](#)
 - [ListObjectVersions](#)
 - [PutBucketLifecycleConfiguration](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Exemples sans serveur pour Amazon S3

Les exemples de code suivants montrent comment utiliser Amazon S3 avec des AWS SDK.

Exemples

- [Invoquer une fonction lambda à partir d'un déclencheur Amazon S3](#)

Invoquer une fonction lambda à partir d'un déclencheur Amazon S3

Les exemples de code suivants montrent comment implémenter une fonction Lambda qui reçoit un événement déclenché par le chargement d'un objet vers un compartiment S3. La fonction extrait le nom du compartiment S3 et la clé de l'objet à partir du paramètre d'événement et appelle l'API Amazon S3 pour récupérer et consigner le type de contenu de l'objet.

.NET

SDK pour .NET

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Utilisation d'un événement S3 avec Lambda en utilisant .NET.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using System.Threading.Tasks;
using Amazon.Lambda.Core;
using Amazon.S3;
using System;
using Amazon.Lambda.S3Events;
using System.Web;

// Assembly attribute to enable the Lambda function's JSON input to be converted
// into a .NET class.
[assembly:
    LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSerializer))]

namespace S3Integration
```



```
{
    public class Function
    {
        private static AmazonS3Client _s3Client;
        public Function() : this(null)
        {
        }

        internal Function(AmazonS3Client s3Client)
        {
            _s3Client = s3Client ?? new AmazonS3Client();
        }

        public async Task<string> Handler(S3Event evt, ILambdaContext context)
        {
            try
            {
                if (evt.Records.Count <= 0)
                {
                    context.Logger.LogLine("Empty S3 Event received");
                    return string.Empty;
                }

                var bucket = evt.Records[0].S3.Bucket.Name;
                var key = HttpUtility.UrlDecode(evt.Records[0].S3.Object.Key);

                context.Logger.LogLine($"Request is for {bucket} and {key}");

                var objectResult = await _s3Client.GetObjectAsync(bucket, key);

                context.Logger.LogLine($"Returning {objectResult.Key}");

                return objectResult.Key;
            }
            catch (Exception e)
            {
                context.Logger.LogLine($"Error processing request -
{e.Message}");

                return string.Empty;
            }
        }
    }
}
```

```
}
```

Go

Kit SDK pour Go V2

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Utilisation d'un événement S3 avec Lambda en utilisant Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "log"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Printf("failed to load default config: %s", err)
        return err
    }
    s3Client := s3.NewFromConfig(sdkConfig)

    for _, record := range s3Event.Records {
        bucket := record.S3.Bucket.Name
        key := record.S3.Object.URLDecodedKey
        headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
            Bucket: &bucket,
            Key:    &key,
        })
    }
}
```

```

    })
    if err != nil {
        log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
        return err
    }
    log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
*headOutput.ContentType)
}

return nil
}

func main() {
    lambda.Start(handler)
}

```

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Utilisation d'un événement S3 avec Lambda en utilisant Go.

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.S3Client;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.S3Event;
import
    com.amazonaws.services.lambda.runtime.events.models.s3.S3EventNotification.S3EventNotifi

```

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

public class Handler implements RequestHandler<S3Event, String> {
    private static final Logger logger = LoggerFactory.getLogger(Handler.class);
    @Override
    public String handleRequest(S3Event s3event, Context context) {
        try {
            S3EventNotificationRecord record = s3event.getRecords().get(0);
            String srcBucket = record.getS3().getBucket().getName();
            String srcKey = record.getS3().getObject().getUrlDecodedKey();

            S3Client s3Client = S3Client.builder().build();
            HeadObjectResponse headObject = getHeadObject(s3Client, srcBucket,
srcKey);

            logger.info("Successfully retrieved " + srcBucket + "/" + srcKey + " of
type " + headObject.contentType());

            return "Ok";
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

    private HeadObjectResponse getHeadObject(S3Client s3Client, String bucket,
String key) {
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucket)
            .key(key)
            .build();
        return s3Client.headObject(headObjectRequest);
    }
}
```

JavaScript

SDK pour JavaScript (v3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Consommation d'un événement S3 avec Lambda en utilisant. JavaScript

```
import { S3Client, HeadObjectCommand } from "@aws-sdk/client-s3";

const client = new S3Client();

export const handler = async (event, context) => {

    // Get the object from the event and show its content type
    const bucket = event.Records[0].s3.bucket.name;
    const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));

    try {
        const { ContentType } = await client.send(new HeadObjectCommand({
            Bucket: bucket,
            Key: key,
        }));

        console.log('CONTENT TYPE:', ContentType);
        return ContentType;

    } catch (err) {
        console.log(err);
        const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your bucket is in the same region as this function.`;
        console.log(message);
        throw new Error(message);
    }
};
```

Consommation d'un événement S3 avec Lambda en utilisant TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { S3Event } from 'aws-lambda';
import { S3Client, HeadObjectCommand } from '@aws-sdk/client-s3';

const s3 = new S3Client({ region: process.env.AWS_REGION });

export const handler = async (event: S3Event): Promise<string | undefined> => {
  // Get the object from the event and show its content type
  const bucket = event.Records[0].s3.bucket.name;
  const key = decodeURIComponent(event.Records[0].s3.object.key.replace(/\+/g, ' '));
  const params = {
    Bucket: bucket,
    Key: key,
  };
  try {
    const { ContentType } = await s3.send(new HeadObjectCommand(params));
    console.log('CONTENT TYPE:', ContentType);
    return ContentType;
  } catch (err) {
    console.log(err);
    const message = `Error getting object ${key} from bucket ${bucket}. Make sure they exist and your bucket is in the same region as this function.`;
    console.log(message);
    throw new Error(message);
  }
};
```

PHP

Kit SDK pour PHP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Consommation d'un événement S3 avec Lambda à l'aide de PHP.

```
<?php

use Bref\Context\Context;
use Bref\Event\S3\S3Event;
use Bref\Event\S3\S3Handler;
use Bref\Logger\StderrLogger;

require __DIR__ . '/vendor/autoload.php';

class Handler extends S3Handler
{
    private StderrLogger $logger;
    public function __construct(StderrLogger $logger)
    {
        $this->logger = $logger;
    }

    public function handleS3(S3Event $event, Context $context) : void
    {
        $this->logger->info("Processing S3 records");

        // Get the object from the event and show its content type
        $records = $event->getRecords();

        foreach ($records as $record)
        {
            $bucket = $record->getBucket()->getName();
            $key = urldecode($record->getObject()->getKey());

            try {
                $fileSize = urldecode($record->getObject()->getSize());
                echo "File Size: " . $fileSize . "\n";
                // TODO: Implement your custom processing logic here
            } catch (Exception $e) {
                echo $e->getMessage() . "\n";
                echo 'Error getting object ' . $key . ' from bucket ' .
                $bucket . '. Make sure they exist and your bucket is in the same region as this
                function.' . "\n";
                throw $e;
            }
        }
    }
}
```

```
    }  
}  
  
$logger = new StderrLogger();  
return new Handler($logger);
```

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Utilisation d'un événement S3 avec Lambda en utilisant Python.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.  
# SPDX-License-Identifier: Apache-2.0  
import json  
import urllib.parse  
import boto3  
  
print('Loading function')  
  
s3 = boto3.client('s3')  
  
def lambda_handler(event, context):  
    #print("Received event: " + json.dumps(event, indent=2))  
  
    # Get the object from the event and show its content type  
    bucket = event['Records'][0]['s3']['bucket']['name']  
    key = urllib.parse.unquote_plus(event['Records'][0]['s3']['object']['key'],  
    encoding='utf-8')  
    try:  
        response = s3.get_object(Bucket=bucket, Key=key)  
        print("CONTENT TYPE: " + response['ContentType'])  
        return response['ContentType']  
    except Exception as e:
```



```
print(e)
print('Error getting object {} from bucket {}. Make sure they exist and
your bucket is in the same region as this function.'.format(key, bucket))
raise e
```

Ruby

Kit SDK pour Ruby

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Consommation d'un événement S3 avec Lambda à l'aide de Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
  Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
and your bucket is in the same region as this function."
    raise e
  end
end
```

```
end
```

Rust

SDK pour Rust

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le référentiel d'[exemples sans serveur](#).

Utilisation d'un événement S3 avec Lambda en utilisant Rust.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::s3::S3Event;
use aws_sdk_s3::{Client};
use lambda_runtime::{run, service_fn, Error, LambdaEvent};

/// Main function
#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    // Initialize the AWS SDK for Rust
    let config = aws_config::load_from_env().await;
    let s3_client = Client::new(&config);

    let res = run(service_fn(|request: LambdaEvent<S3Event>| {
        function_handler(&s3_client, request)
    })).await;

    res
}
```

```
async fn function_handler(
    s3_client: &Client,
    evt: LambdaEvent<S3Event>
) -> Result<(), Error> {
    tracing::info!(records = ?evt.payload.records.len(), "Received request from
    SQS");

    if evt.payload.records.len() == 0 {
        tracing::info!("Empty S3 event received");
    }

    let bucket = evt.payload.records[0].s3.bucket.name.as_ref().expect("Bucket
    name to exist");
    let key = evt.payload.records[0].s3.object.key.as_ref().expect("Object key to
    exist");

    tracing::info!("Request is for {} and object {}", bucket, key);

    let s3_get_object_result = s3_client
        .get_object()
        .bucket(bucket)
        .key(key)
        .send()
        .await;

    match s3_get_object_result {
        Ok(_) => tracing::info!("S3 Get Object success, the s3GetObjectResult
        contains a 'body' property of type ByteStream"),
        Err(_) => tracing::info!("Failure with S3 Get Object request")
    }

    Ok(())
}
```

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Exemples de code pour Amazon S3 Control à l'aide de AWS Kits SDK

Les exemples de code suivants montrent comment utiliser Amazon S3 Control avec un kit de développement AWS logiciel (SDK).

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Amazon S3 Control

- [Exemples de base pour l'utilisation d'Amazon S3 Control AWS Kits SDK](#)
 - [Bonjour Amazon S3 Control](#)
 - [Découvrez les principes de base d'Amazon S3 Control grâce à AWS Kit SDK](#)
 - [Actions pour Amazon S3 Control à l'aide de AWS Kits SDK](#)
 - [À utiliser CreateJob avec un AWS SDK ou CLI](#)
 - [À utiliser DeleteJobTagging avec un AWS Kit SDK](#)
 - [À utiliser DescribeJob avec un AWS SDK ou CLI](#)
 - [À utiliser GetJobTagging avec un AWS Kit SDK](#)
 - [À utiliser PutJobTagging avec un AWS Kit SDK](#)
 - [À utiliser UpdateJobPriority avec un AWS SDK ou CLI](#)
 - [À utiliser UpdateJobStatus avec un AWS SDK ou CLI](#)

Exemples de base pour l'utilisation d'Amazon S3 Control AWS Kits SDK

Les exemples de code suivants montrent les bases de l'utilisation d'Amazon S3 Control avec des kits AWS SDK.

Exemples

- [Bonjour Amazon S3 Control](#)
- [Découvrez les principes de base d'Amazon S3 Control grâce à AWS Kit SDK](#)
- [Actions pour Amazon S3 Control à l'aide de AWS Kits SDK](#)
 - [À utiliser CreateJobavec un AWS SDK ou CLI](#)
 - [À utiliser DeleteJobTaggingavec un AWS Kit SDK](#)
 - [À utiliser DescribeJobavec un AWS SDK ou CLI](#)
 - [À utiliser GetJobTaggingavec un AWS Kit SDK](#)
 - [À utiliser PutJobTaggingavec un AWS Kit SDK](#)
 - [À utiliser UpdateJobPriorityavec un AWS SDK ou CLI](#)
 - [À utiliser UpdateJobStatusavec un AWS SDK ou CLI](#)

Bonjour Amazon S3 Control

Les exemples de code suivants montrent comment commencer à utiliser Amazon S3 Control.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
import
    software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
import software.amazon.awssdk.core.client.config.ClientOverrideConfiguration;
import software.amazon.awssdk.core.retry.RetryMode;
import software.amazon.awssdk.core.retry.RetryPolicy;
import software.amazon.awssdk.http.async.SdkAsyncHttpClient;
import software.amazon.awssdk.http.nio.netty.NettyNioAsyncHttpClient;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3control.S3ControlAsyncClient;
import software.amazon.awssdk.services.s3control.model.JobListDescriptor;
import software.amazon.awssdk.services.s3control.model.JobStatus;
import software.amazon.awssdk.services.s3control.model.ListJobsRequest;
import software.amazon.awssdk.services.s3control.paginators.ListJobsPublisher;
```

```
import java.time.Duration;
import java.util.List;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.CompletionException;

/**
 * Before running this example:
 * <p/>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
 * authentication for SDKs and tools, see https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs and Tools Reference Guide.
 * <p/>
 * You must have a runtime environment configured with the Java SDK.
 * See https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html
 * in the Developer Guide if this is not set up.
 */
public class HelloS3Batch {
    private static S3ControlAsyncClient asyncClient;

    public static void main(String[] args) {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        try {
            listBatchJobsAsync(accountId)
                .exceptionally(ex -> {
                    System.err.println("List batch jobs failed: " +
ex.getMessage());
                    return null;
                })
                .join();

        } catch (CompletionException ex) {
            System.err.println("Failed to list batch jobs: " + ex.getMessage());
        }
    }

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
     configuration:
    */
}
```

```

* <ul>
*   <li>Maximum concurrency: 100</li>
*   <li>Connection timeout: 60 seconds</li>
*   <li>Read timeout: 60 seconds</li>
*   <li>Write timeout: 60 seconds</li>
*   <li>API call timeout: 2 minutes</li>
*   <li>API call attempt timeout: 90 seconds</li>
*   <li>Retry policy: 3 retries</li>
*   <li>Region: US_EAST_1</li>
*   <li>Credentials provider: {@link
EnvironmentVariableCredentialsProvider}</li>
* </ul>
*
* @return the asynchronous S3 Control client instance
*/
private static S3ControlAsyncClient getAsyncClient() {
    if (asyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryStrategy(RetryMode.STANDARD)
            .build();

        asyncClient = S3ControlAsyncClient.builder()
            .region(Region.US_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return asyncClient;
}

/**
 * Asynchronously lists batch jobs that have completed for the specified
 * account.
 *

```

```
* @param accountId the ID of the account to list jobs for
* @return a CompletableFuture that completes when the job listing operation
is finished
*/
public static CompletableFuture<Void> listBatchJobsAsync(String accountId) {
    ListJobsRequest jobsRequest = ListJobsRequest.builder()
        .jobStatuses(JobStatus.COMPLETE)
        .accountId(accountId)
        .maxResults(10)
        .build();

    ListJobsPublisher publisher =
getAsyncClient().listJobsPaginator(jobsRequest);
    return publisher.subscribe(response -> {
        List<JobListDescriptor> jobs = response.jobs();
        for (JobListDescriptor job : jobs) {
            System.out.println("The job id is " + job.jobId());
            System.out.println("The job priority is " + job.priority());
        }
    }).thenAccept(response -> {
        System.out.println("Listing batch jobs completed");
    }).exceptionally(ex -> {
        System.err.println("Failed to list batch jobs: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
}
```

- Pour plus de détails sur l'API, reportez-vous [ListJobs](#) à la section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```
def list_jobs(self, account_id: str) -> None:
    """
    List all batch jobs for the account.

    Args:
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.list_jobs(
            AccountId=account_id,
            JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
                        'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
        )
        jobs = response.get('Jobs', [])
        for job in jobs:
            print(f"The job id is {job['JobId']}")
            print(f"The job priority is {job['Priority']}")
    except ClientError as e:
        print(f"Error listing jobs: {e}")
        raise
```

- Pour plus de détails sur l'API, consultez [ListJobs](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Découvrez les principes de base d'Amazon S3 Control grâce à AWS Kit SDK

Les exemples de code suivants montrent comment apprendre les opérations de base pour Amazon S3 Control.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Découvrez les opérations de base.

```
package com.example.s3.batch;

import software.amazon.awssdk.services.s3.model.S3Exception;
import java.io.IOException;
import java.util.Map;
import java.util.Scanner;
import java.util.UUID;
import java.util.concurrent.CompletionException;

public class S3BatchScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
"-");
    private static final String STACK_NAME = "MyS3Stack";
    public static void main(String[] args) throws IOException {
        S3BatchActions actions = new S3BatchActions();
        String accountId = actions.getAccountId();
        String uuid = java.util.UUID.randomUUID().toString();
        Scanner scanner = new Scanner(System.in);

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon S3 Batch basics scenario.");
        System.out.println("""
            S3 Batch operations enables efficient and cost-effective processing
of large-scale
            data stored in Amazon S3. It automatically scales resources to handle
varying workloads
            without the need for manual intervention.

            One of the key features of S3 Batch is its ability to perform tagging
operations on objects stored in
```

S3 buckets. Users can leverage S3 Batch to apply, update, or remove tags on thousands or millions of objects in a single operation, streamlining the management and organization of their data.

This can be particularly useful for tasks such as cost allocation, lifecycle management, or metadata-driven workflows, where consistent and accurate tagging is essential.

S3 Batch's scalability and serverless nature make it an ideal solution for organizations with growing data volumes and complex data management requirements.

This Java program walks you through Amazon S3 Batch operations.

Let's get started...

```
        "");
        waitForInputToContinue(scanner);
        // Use CloudFormation to stand up the resource required for this
        scenario.
        System.out.println("Use CloudFormation to stand up the resource required
        for this scenario.");
        CloudFormationHelper.deployCloudFormationStack(STACK_NAME);

        Map<String, String> stackOutputs =
        CloudFormationHelper.getStackOutputs(STACK_NAME);
        String iamRoleArn = stackOutputs.get("S3BatchRoleArn");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Setup the required bucket for this scenario.");
        waitForInputToContinue(scanner);
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.
        actions.createBucket(bucketName);
        String reportBucketName = "arn:aws:s3:::"+bucketName;
        String manifestLocation = "arn:aws:s3:::"+bucketName+"/job-manifest.csv";
        System.out.println("Populate the bucket with the required files.");
        String[] fileNames = {"job-manifest.csv", "object-key-1.txt", "object-
        key-2.txt", "object-key-3.txt", "object-key-4.txt"};
        actions.uploadFilesToBucket(bucketName, fileNames, actions);
        waitForInputToContinue(scanner);
        System.out.println(DASHES);
```

```

        System.out.println(DASHES);
        System.out.println("1. Create a S3 Batch Job");
        System.out.println("This job tags all objects listed in the manifest file
with tags");
        waitForInputToContinue(scanner);
        String jobId ;
        try {
            jobId = actions.createS3JobAsync(accountId, iamRoleArn,
manifestLocation, reportBucketName, uuid).join();
            System.out.println("The Job id is " + jobId);

        } catch (S3Exception e) {
            System.err.println("SSM error: " + e.getMessage());
            return;
        } catch (RuntimeException e) {
            System.err.println("Unexpected error: " + e.getMessage());
            return;
        }

        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Update an existing S3 Batch Operations job's
priority");
        System.out.println("""
            In this step, we modify the job priority value. The higher the
number, the higher the priority.
            So, a job with a priority of `30` would have a higher priority than
a job with
            a priority of `20`. This is a common way to represent the priority
of a task
            or job, with higher numbers indicating a higher priority.

            Ensure that the job status allows for priority updates. Jobs in
certain
            states (e.g., Cancelled, Failed, or Completed) cannot have their
priorities
            updated. Only jobs in the Active or Suspended state typically allow
priority
            updates.
            """);

```

```
        try {
            actions.updateJobPriorityAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Update job priority failed: " +
ex.getMessage());
                    return null;
                })
                .join();
        } catch (CompletionException ex) {
            System.err.println("Failed to update job priority: " +
ex.getMessage());
        }
        waitForInputToContinue(scanner);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Cancel the S3 Batch job");
        System.out.print("Do you want to cancel the Batch job? (y/n): ");
        String cancelAns = scanner.nextLine();
        if (cancelAns != null && cancelAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.cancelJobAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Cancel job failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to cancel job: " + ex.getMessage());
            }
        } else {
            System.out.println("Job " + jobId + " was not canceled.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Describe the job that was just created");
        waitForInputToContinue(scanner);
        try {
            actions.describeJobAsync(jobId, accountId)
                .exceptionally(ex -> {
                    System.err.println("Describe job failed: " +
ex.getMessage());
```

```
        return null;
    })
    .join();
} catch (CompletionException ex) {
    System.err.println("Failed to describe job: " + ex.getMessage());
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("5. Describe the tags associated with the job");
waitForInputToContinue(scanner);
try {
    actions.getJobTagsAsync(jobId, accountId)
        .exceptionally(ex -> {
            System.err.println("Get job tags failed: " +
ex.getMessage());
            return null;
        })
        .join();
} catch (CompletionException ex) {
    System.err.println("Failed to get job tags: " + ex.getMessage());
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("6. Update Batch Job Tags");
waitForInputToContinue(scanner);
try {
    actions.putJobTaggingAsync(jobId, accountId)
        .exceptionally(ex -> {
            System.err.println("Put job tagging failed: " +
ex.getMessage());
            return null;
        })
        .join();
} catch (CompletionException ex) {
    System.err.println("Failed to put job tagging: " + ex.getMessage());
}
System.out.println(DASHES);

System.out.println(DASHES);
System.out.println("7. Delete the Amazon S3 Batch job tagging.");
System.out.print("Do you want to delete Batch job tagging? (y/n)");
String delAns = scanner.nextLine();
```

```
        if (delAns != null && delAns.trim().equalsIgnoreCase("y")) {
            try {
                actions.deleteBatchJobTagsAsync(jobId, accountId)
                    .exceptionally(ex -> {
                        System.err.println("Delete batch job tags failed: " +
ex.getMessage());
                        return null;
                    })
                    .join();
            } catch (CompletionException ex) {
                System.err.println("Failed to delete batch job tags: " +
ex.getMessage());
            }
        } else {
            System.out.println("Tagging was not deleted.");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.print("Do you want to delete the AWS resources used in this
scenario? (y/n)");
        String delResAns = scanner.nextLine();
        if (delResAns != null && delResAns.trim().equalsIgnoreCase("y")) {
            actions.deleteFilesFromBucket(bucketName, fileNames, actions);
            actions.deleteBucketFolderAsync(bucketName);
            actions.deleteBucket(bucketName)
                .thenRun(() -> System.out.println("Bucket deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
            CloudFormationHelper.destroyCloudFormationStack(STACK_NAME);
        } else {
            System.out.println("The AWS resources were not deleted.");
        }
        System.out.println("The Amazon S3 Batch scenario has successfully
completed.");
        System.out.println(DASHES);
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            System.out.println();
            System.out.println("Enter 'c' followed by <ENTER> to continue:");
        }
    }
}
```

```

        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            System.out.println("Continuing with the program...");
            System.out.println();
            break;
        } else {
            // Handle invalid input.
            System.out.println("Invalid input. Please try again.");
        }
    }
}
}

```

Une classe d'action qui encapsule les opérations.

```

public class S3BatchActions {

    private static S3ControlAsyncClient asyncClient;

    private static S3AsyncClient s3AsyncClient ;

    /**
     * Retrieves the asynchronous S3 Control client instance.
     * <p>
     * This method creates and returns a singleton instance of the {@link
     S3ControlAsyncClient}. If the instance
     * has not been created yet, it will be initialized with the following
     configuration:
     * <ul>
     *   <li>Maximum concurrency: 100</li>
     *   <li>Connection timeout: 60 seconds</li>
     *   <li>Read timeout: 60 seconds</li>
     *   <li>Write timeout: 60 seconds</li>
     *   <li>API call timeout: 2 minutes</li>
     *   <li>API call attempt timeout: 90 seconds</li>
     *   <li>Retry policy: 3 retries</li>
     *   <li>Region: US_EAST_1</li>
     *   <li>Credentials provider: {@link
     EnvironmentVariableCredentialsProvider}</li>
     * </ul>

```



```
*
* @return the asynchronous S3 Control client instance
*/
private static S3ControlAsyncClient getAsyncClient() {
    if (asyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryPolicy(RetryPolicy.builder()
                .numRetries(3)
                .build())
            .build();

        asyncClient = S3ControlAsyncClient.builder()
            .region(Region.US_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return asyncClient;
}

private static S3AsyncClient getS3AsyncClient() {
    if (asyncClient == null) {
        SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()
            .maxConcurrency(100)
            .connectionTimeout(Duration.ofSeconds(60))
            .readTimeout(Duration.ofSeconds(60))
            .writeTimeout(Duration.ofSeconds(60))
            .build();

        ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
            .apiCallTimeout(Duration.ofMinutes(2))
            .apiCallAttemptTimeout(Duration.ofSeconds(90))
            .retryStrategy(RetryMode.STANDARD)
```

```

        .build();

        s3AsyncClient = S3AsyncClient.builder()
            .region(Region.US_EAST_1)
            .httpClient(httpClient)
            .overrideConfiguration(overrideConfig)
            .build();
    }
    return s3AsyncClient;
}

/**
 * Cancels a job asynchronously.
 *
 * @param jobId The ID of the job to be canceled.
 * @param accountId The ID of the account associated with the job.
 * @return A {@link CompletableFuture} that completes when the job status has
 *         been updated to "CANCELLED".
 *         If an error occurs during the update, the returned future will
 *         complete exceptionally.
 */
public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
{
    UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
    .accountId(accountId)
    .jobId(jobId)
    .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
    .build();

    return asyncClient.updateJobStatus(updateJobStatusRequest)
        .thenAccept(updateJobStatusResponse -> {
            System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
        })
        .exceptionally(ex -> {
            System.err.println("Failed to cancel job: " + ex.getMessage());
            throw new RuntimeException(ex); // Propagate the exception
        });
}

/**
 * Updates the priority of a job asynchronously.

```

```

*
* @param jobId      the ID of the job to update
* @param accountId  the ID of the account associated with the job
* @return a {@link CompletableFuture} that represents the asynchronous
operation, which completes when the job priority has been updated or an error
has occurred
*/
public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
    UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .priority(60)
        .build();

    CompletableFuture<Void> future = new CompletableFuture<>();
    getAsyncClient().updateJobPriority(priorityRequest)
        .thenAccept(response -> {
            System.out.println("The job priority was updated");
            future.complete(null); // Complete the CompletableFuture on
successful execution
        })
        .exceptionally(ex -> {
            System.err.println("Failed to update job priority: " +
ex.getMessage());
            future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
            return null; // Return null to handle the exception
        });

    return future;
}

/**
* Asynchronously retrieves the tags associated with a specific job in an AWS
account.
*
* @param jobId      the ID of the job for which to retrieve the tags
* @param accountId  the ID of the AWS account associated with the job
* @return a {@link CompletableFuture} that completes when the job tags have
been retrieved, or with an exception if the operation fails
* @throws RuntimeException if an error occurs while retrieving the job tags
*/

```

```
public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
    GetJobTaggingRequest request = GetJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return asyncClient.getJobTagging(request)
        .thenAccept(response -> {
            List<S3Tag> tags = response.tags();
            if (tags.isEmpty()) {
                System.out.println("No tags found for job ID: " + jobId);
            } else {
                for (S3Tag tag : tags) {
                    System.out.println("Tag key is: " + tag.key());
                    System.out.println("Tag value is: " + tag.value());
                }
            }
        })
        .exceptionally(ex -> {
            System.err.println("Failed to get job tags: " + ex.getMessage());
            throw new RuntimeException(ex); // Propagate the exception
        });
}

/**
 * Asynchronously deletes the tags associated with a specific batch job.
 *
 * @param jobId      The ID of the batch job whose tags should be deleted.
 * @param accountId  The ID of the account associated with the batch job.
 * @return A CompletableFuture that completes when the job tags have been
 * successfully deleted, or an exception is thrown if the deletion fails.
 */
public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
    DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .build();

    return asyncClient.deleteJobTagging(jobTaggingRequest)
        .thenAccept(response -> {
```

```

        System.out.println("You have successfully deleted " + jobId + "
tagging.");
    })
    .exceptionally(ex -> {
        System.err.println("Failed to delete job tags: " +
ex.getMessage());
        throw new RuntimeException(ex);
    });
}

/**
 * Asynchronously describes the specified job.
 *
 * @param jobId      the ID of the job to describe
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job
description is available
 * @throws RuntimeException if an error occurs while describing the job
 */
public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {
    DescribeJobRequest jobRequest = DescribeJobRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return getAsyncClient().describeJob(jobRequest)
        .thenAccept(response -> {
            System.out.println("Job ID: " + response.job().jobId());
            System.out.println("Description: " +
response.job().description());
            System.out.println("Status: " + response.job().statusAsString());
            System.out.println("Role ARN: " + response.job().roleArn());
            System.out.println("Priority: " + response.job().priority());
            System.out.println("Progress Summary: " +
response.job().progressSummary());

            // Print out details about the job manifest.
            JobManifest manifest = response.job().manifest();
            System.out.println("Manifest Location: " +
manifest.location().objectArn());
            System.out.println("Manifest ETag: " +
manifest.location().eTag());

```

```

        // Print out details about the job operation.
        JobOperation operation = response.job().operation();
        if (operation.s3PutObjectTagging() != null) {
            System.out.println("Operation: S3 Put Object Tagging");
            System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
        }

        // Print out details about the job report.
        JobReport report = response.job().report();
        System.out.println("Report Bucket: " + report.bucket());
        System.out.println("Report Prefix: " + report.prefix());
        System.out.println("Report Format: " + report.format());
        System.out.println("Report Enabled: " + report.enabled());
        System.out.println("Report Scope: " +
report.reportScopeAsString());
    })
    .exceptionally(ex -> {
        System.err.println("Failed to describe job: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
}

/**
 * Creates an asynchronous S3 job using the AWS Java SDK.
 *
 * @param accountId      the AWS account ID associated with the job
 * @param iamRoleArn      the ARN of the IAM role to be used for the job
 * @param manifestLocation the location of the job manifest file in S3
 * @param reportBucketName the name of the S3 bucket to store the job report
 * @param uuid            a unique identifier for the job
 * @return a CompletableFuture that represents the asynchronous creation of
the S3 job.
 *      The CompletableFuture will return the job ID if the job is created
successfully,
 *      or throw an exception if there is an error.
 */
public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
                                                String manifestLocation,
String reportBucketName, String uuid) {

    String[] bucketName = new String[]{" "};
    String[] parts = reportBucketName.split("::");

```

```
        if (parts.length > 1) {
            bucketName[0] = parts[1];
        } else {
            System.out.println("The input string does not contain the expected
format.");
        }

        return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
            .thenCompose(eTag -> {
                ArrayList<S3Tag> tagSet = new ArrayList<>();
                S3Tag s3Tag = S3Tag.builder()
                    .key("keyOne")
                    .value("ValueOne")
                    .build();
                S3Tag s3Tag2 = S3Tag.builder()
                    .key("keyTwo")
                    .value("ValueTwo")
                    .build();
                tagSet.add(s3Tag);
                tagSet.add(s3Tag2);

                S3SetObjectTaggingOperation objectTaggingOperation =
S3SetObjectTaggingOperation.builder()
                    .tagSet(tagSet)
                    .build();

                JobOperation jobOperation = JobOperation.builder()
                    .s3PutObjectTagging(objectTaggingOperation)
                    .build();

                JobManifestLocation jobManifestLocation =
JobManifestLocation.builder()
                    .objectArn(manifestLocation)
                    .eTag(eTag)
                    .build();

                JobManifestSpec manifestSpec = JobManifestSpec.builder()
                    .fieldsWithStrings("Bucket", "Key")
                    .format("S3BatchOperations_CSV_20180820")
                    .build();

                JobManifest jobManifest = JobManifest.builder()
                    .spec(manifestSpec)
```

```

        .location(jobManifestLocation)
        .build();

    JobReport jobReport = JobReport.builder()
        .bucket(reportBucketName)
        .prefix("reports")
        .format("Report_CSV_20180820")
        .enabled(true)
        .reportScope("AllTasks")
        .build();

    CreateJobRequest jobRequest = CreateJobRequest.builder()
        .accountId(accountId)
        .description("Job created using the AWS Java SDK")
        .manifest(jobManifest)
        .operation(jobOperation)
        .report(jobReport)
        .priority(42)
        .roleArn(iamRoleArn)
        .clientRequestToken(uuid)
        .confirmationRequired(false)
        .build();

    // Create the job asynchronously.
    return getAsyncClient().createJob(jobRequest)
        .thenApply(CreateJobResponse::jobId);
    })
    .handle((jobId, ex) -> {
        if (ex != null) {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof S3ControlException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }
        return jobId;
    });
}

/**
 * Retrieves the ETag (Entity Tag) for an object stored in an Amazon S3
 * bucket.

```



```

    *
    * @param bucketName the name of the Amazon S3 bucket where the object is
stored
    * @param key the key (file name) of the object in the Amazon S3 bucket
    * @return the ETag of the object
    */
    public String getETag(String bucketName, String key) {
        S3Client s3Client = S3Client.builder()
            .region(Region.US_EAST_1)
            .build();

        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
        return headObjectResponse.eTag();
    }

    /**
    * Asynchronously adds tags to a job in the system.
    *
    * @param jobId      the ID of the job to add tags to
    * @param accountId the account ID associated with the job
    * @return a CompletableFuture that completes when the tagging operation is
finished
    */
    public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
        S3Tag departmentTag = S3Tag.builder()
            .key("department")
            .value("Marketing")
            .build();

        S3Tag fiscalYearTag = S3Tag.builder()
            .key("FiscalYear")
            .value("2020")
            .build();

        PutJobTaggingRequest putJobTaggingRequest =
PutJobTaggingRequest.builder()
            .jobId(jobId)

```

```
        .accountId(accountId)
        .tags(departmentTag, fiscalYearTag)
        .build();

return asyncClient.putJobTagging(putJobTaggingRequest)
    .thenRun(() -> {
        System.out.println("Additional Tags were added to job " + jobId);
    })
    .exceptionally(ex -> {
        System.err.println("Failed to add tags to job: " +
ex.getMessage());
        throw new RuntimeException(ex); // Propagate the exception
    });
}

// Setup the S3 bucket required for this scenario.
/**
 * Creates an Amazon S3 bucket with the specified name.
 *
 * @param bucketName the name of the S3 bucket to create
 * @throws S3Exception if there is an error creating the bucket
 */
public void createBucket(String bucketName) {
    try {
        S3Client s3Client = S3Client.builder()
            .region(Region.US_EAST_1)
            .build();

        S3Waiter s3Waiter = s3Client.waiter();
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        s3Client.createBucket(bucketRequest);
        HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
            .bucket(bucketName)
            .build();

        // Wait until the bucket is created and print out the response.
        WaiterResponse<HeadBucketResponse> waiterResponse =
s3Waiter.waitUntilBucketExists(bucketRequestWait);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println(bucketName + " is ready");
    }
}
```

```
        } catch (S3Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    /**
     * Uploads a file to an Amazon S3 bucket asynchronously.
     *
     * @param bucketName the name of the S3 bucket to upload the file to
     * @param fileName the name of the file to be uploaded
     * @throws RuntimeException if an error occurs during the file upload
     */
    public void populateBucket(String bucketName, String fileName) {
        // Define the path to the directory.
        Path filePath = Paths.get("src/main/resources/batch/",
            fileName).toAbsolutePath();
        PutObjectRequest putOb = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(fileName)
            .build();

        CompletableFuture<PutObjectResponse> future =
            getS3AsyncClient().putObject(putOb, AsyncRequestBody.fromFile(filePath));
        future.whenComplete((result, ex) -> {
            if (ex != null) {
                System.err.println("Error uploading file: " + ex.getMessage());
            } else {
                System.out.println("Successfully placed " + fileName + " into
                    bucket " + bucketName);
            }
        }).join();
    }

    // Update the bucketName in CSV.
    public void updateCSV(String newValue) {
        Path csvFilePath = Paths.get("src/main/resources/batch/job-
            manifest.csv").toAbsolutePath();
        try {
            // Read all lines from the CSV file.
            List<String> lines = Files.readAllLines(csvFilePath);
        }
    }
}
```

```
// Update the first value in each line.
List<String> updatedLines = lines.stream()
    .map(line -> {
        String[] parts = line.split(",");
        parts[0] = newValue;
        return String.join(",", parts);
    })
    .collect(Collectors.toList());

// Write the updated lines back to the CSV file
Files.write(csvFilePath, updatedLines);
System.out.println("CSV file updated successfully.");
} catch (Exception e) {
    e.printStackTrace();
}
}

/**
 * Deletes an object from an Amazon S3 bucket asynchronously.
 *
 * @param bucketName The name of the S3 bucket where the object is stored.
 * @param objectName The name of the object to be deleted.
 * @return A {@link CompletableFuture} that completes when the object has
 * been deleted,
 *         or throws a {@link RuntimeException} if an error occurs during the
 * deletion.
 */
public CompletableFuture<Void> deleteBucketObjects(String bucketName, String
objectName) {
    ArrayList<ObjectIdentifier> toDelete = new ArrayList<>();
    toDelete.add(ObjectIdentifier.builder()
        .key(objectName)
        .build());

    DeleteObjectsRequest dor = DeleteObjectsRequest.builder()
        .bucket(bucketName)
        .delete(Delete.builder()
            .objects(toDelete).build())
        .build();

    return getS3AsyncClient().deleteObjects(dor)
        .thenAccept(result -> {
            System.out.println("The object was deleted!");
        })
}
```

```

        .exceptionally(ex -> {
            throw new RuntimeException("Error deleting object: " +
ex.getMessage(), ex);
        });
    }

    /**
     * Deletes a folder and all its contents asynchronously from an Amazon S3
     bucket.
     *
     * @param bucketName the name of the S3 bucket containing the folder to be
     deleted
     * @return a {@link CompletableFuture} that completes when the folder and its
     contents have been deleted
     * @throws RuntimeException if any error occurs during the deletion process
     */
    public void deleteBucketFolderAsync(String bucketName) {
        String folderName = "reports/";
        ListObjectsV2Request request = ListObjectsV2Request.builder()
            .bucket(bucketName)
            .prefix(folderName)
            .build();

        CompletableFuture<ListObjectsV2Response> listObjectsFuture =
getS3AsyncClient().listObjectsV2(request);
        listObjectsFuture.thenCompose(response -> {
            List<CompletableFuture<DeleteObjectResponse>> deleteFutures =
response.contents().stream()
                .map(obj -> {
                    DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                        .bucket(bucketName)
                        .key(obj.key())
                        .build();
                    return getS3AsyncClient().deleteObject(deleteRequest)
                        .thenApply(deleteResponse -> {
                            System.out.println("Deleted object: " + obj.key());
                            return deleteResponse;
                        });
                })
            .collect(Collectors.toList());

            return CompletableFuture.allOf(deleteFutures.toArray(new
CompletableFuture[0]))

```

```

        .thenCompose(v -> {
            // Delete the folder.
            DeleteObjectRequest deleteRequest =
DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(folderName)
                .build();
            return getS3AsyncClient().deleteObject(deleteRequest)
                .thenApply(deleteResponse -> {
                    System.out.println("Deleted folder: " + folderName);
                    return deleteResponse;
                });
        });
    }).join();
}

/**
 * Deletes an Amazon S3 bucket.
 *
 * @param bucketName the name of the bucket to delete
 * @return a {@link CompletableFuture} that completes when the bucket has
been deleted, or exceptionally if there is an error
 * @throws RuntimeException if there is an error deleting the bucket
 */
public CompletableFuture<Void> deleteBucket(String bucketName) {
    S3AsyncClient s3Client = getS3AsyncClient();
    return s3Client.deleteBucket(DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build())
        .thenAccept(deleteBucketResponse -> {
            System.out.println(bucketName + " was deleted");
        })
        .exceptionally(ex -> {
            // Handle the exception or rethrow it.
            throw new RuntimeException("Failed to delete bucket: " +
bucketName, ex);
        });
}

/**
 * Uploads a set of files to an Amazon S3 bucket.
 *
 * @param bucketName the name of the S3 bucket to upload the files to
 * @param fileNames an array of file names to be uploaded

```

```

    * @param actions an instance of {@link S3BatchActions} that provides the
    implementation for the necessary S3 operations
    * @throws IOException if there's an error creating the text files or
    uploading the files to the S3 bucket
    */
    public static void uploadFilesToBucket(String bucketName, String[] fileNames,
    S3BatchActions actions) throws IOException {
        actions.updateCSV(bucketName);
        createTextFiles(fileNames);
        for (String fileName : fileNames) {
            actions.populateBucket(bucketName, fileName);
        }
        System.out.println("All files are placed in the S3 bucket " +
    bucketName);
    }

    /**
    * Deletes the specified files from the given S3 bucket.
    *
    * @param bucketName the name of the S3 bucket
    * @param fileNames an array of file names to be deleted from the bucket
    * @param actions the S3BatchActions instance to be used for the file
    deletion
    * @throws IOException if an I/O error occurs during the file deletion
    */
    public void deleteFilesFromBucket(String bucketName, String[] fileNames,
    S3BatchActions actions) throws IOException {
        for (String fileName : fileNames) {
            actions.deleteBucketObjects(bucketName, fileName)
                .thenRun(() -> System.out.println("Object deletion completed"))
                .exceptionally(ex -> {
                    System.err.println("Error occurred: " + ex.getMessage());
                    return null;
                });
        }
        System.out.println("All files have been deleted from the bucket " +
    bucketName);
    }

    public static void createTextFiles(String[] fileNames) {
        String currentDirectory = System.getProperty("user.dir");
        String directoryPath = currentDirectory + "\\src\\main\\resources\\
    \\batch";
        Path path = Paths.get(directoryPath);
    
```

```
try {
    // Create the directory if it doesn't exist.
    if (Files.notExists(path)) {
        Files.createDirectories(path);
        System.out.println("Created directory: " + path.toString());
    } else {
        System.out.println("Directory already exists: " +
path.toString());
    }

    for (String fileName : fileNames) {
        // Check if the file is a .txt file.
        if (fileName.endsWith(".txt")) {
            // Define the path for the new file.
            Path filePath = path.resolve(fileName);
            System.out.println("Attempting to create file: " +
filePath.toString());

            // Create and write content to the new file.
            Files.write(filePath, "This is a test".getBytes());

            // Verify the file was created.
            if (Files.exists(filePath)) {
                System.out.println("Successfully created file: " +
filePath.toString());
            } else {
                System.out.println("Failed to create file: " +
filePath.toString());
            }
        }
    }

} catch (IOException e) {
    System.err.println("An error occurred: " + e.getMessage());
    e.printStackTrace();
}

}

public String getAccountId() {
    StsClient stsClient = StsClient.builder()
        .region(Region.US_EAST_1)
        .build();
}
```



```
        GetCallerIdentityResponse callerIdentityResponse =
stsClient.getCallerIdentity();
        return callerIdentityResponse.account();
    }
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .
 - [CreateJob](#)
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)
 - [UpdateJobStatus](#)

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Découvrez le scénario de base de S3 Batch.

```
class S3BatchWrapper:
    """Wrapper class for managing S3 Batch Operations."""

    def __init__(self, s3_client: Any, s3control_client: Any, sts_client: Any) ->
None:
        """
        Initializes the S3BatchWrapper with AWS service clients.
```

```

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                                access to AWS S3 services.
        :param s3control_client: A Boto3 Amazon S3 Control client. This client
provides
                                low-level access to AWS S3 Control services.
        :param sts_client: A Boto3 AWS STS client. This client provides low-level
                                access to AWS STS services.
    """
    self.s3_client = s3_client
    self.s3control_client = s3control_client
    self.sts_client = sts_client
    # Get region from the client for bucket creation logic
    self.region_name = self.s3_client.meta.region_name

def get_account_id(self) -> str:
    """
    Get AWS account ID.

    Returns:
        str: AWS account ID
    """
    return self.sts_client.get_caller_identity()["Account"]

def create_bucket(self, bucket_name: str) -> None:
    """
    Create an S3 bucket.

    Args:
        bucket_name (str): Name of the bucket to create

    Raises:
        ClientError: If bucket creation fails
    """
    try:
        if self.region_name and self.region_name != 'us-east-1':
            self.s3_client.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration={
                    'LocationConstraint': self.region_name
                }
            )
        else:
            self.s3_client.create_bucket(Bucket=bucket_name)

```

```
        print(f"Created bucket: {bucket_name}")
    except ClientError as e:
        print(f"Error creating bucket: {e}")
        raise

def upload_files_to_bucket(self, bucket_name: str, file_names: List[str]) ->
str:
    """
    Upload files to S3 bucket including manifest file.

    Args:
        bucket_name (str): Target bucket name
        file_names (list): List of file names to upload

    Returns:
        str: ETag of the manifest file

    Raises:
        ClientError: If file upload fails
    """
    try:
        for file_name in file_names:
            if file_name != "job-manifest.csv":
                content = f"Content for {file_name}"
                self.s3_client.put_object(
                    Bucket=bucket_name,
                    Key=file_name,
                    Body=content.encode('utf-8')
                )
                print(f"Uploaded {file_name} to {bucket_name}")

        manifest_content = ""
        for file_name in file_names:
            if file_name != "job-manifest.csv":
                manifest_content += f"{bucket_name},{file_name}\n"

        manifest_response = self.s3_client.put_object(
            Bucket=bucket_name,
            Key="job-manifest.csv",
            Body=manifest_content.encode('utf-8')
        )
        print(f"Uploaded manifest file to {bucket_name}")
        print(f"Manifest content:\n{manifest_content}")
        return manifest_response['ETag'].strip('\"')
```

```

except ClientError as e:
    print(f"Error uploading files: {e}")
    raise

def create_s3_batch_job(self, account_id: str, role_arn: str,
manifest_location: str,
                        report_bucket_name: str) -> str:
    """
    Create an S3 batch operation job.

    Args:
        account_id (str): AWS account ID
        role_arn (str): IAM role ARN for batch operations
        manifest_location (str): Location of the manifest file
        report_bucket_name (str): Bucket for job reports

    Returns:
        str: Job ID

    Raises:
        ClientError: If job creation fails
    """
    try:
        bucket_name = manifest_location.split(':::')[1].split('/')[0]
        manifest_key = 'job-manifest.csv'
        manifest_obj = self.s3_client.head_object(
            Bucket=bucket_name,
            Key=manifest_key
        )
        etag = manifest_obj['ETag'].strip('""')

        response = self.s3control_client.create_job(
            AccountId=account_id,
            Operation={
                'S3PutObjectTagging': {
                    'TagSet': [
                        {
                            'Key': 'BatchTag',
                            'Value': 'BatchValue'
                        },
                    ],
                }
            },

```

```

        Report={
            'Bucket': report_bucket_name,
            'Format': 'Report_CSV_20180820',
            'Enabled': True,
            'Prefix': 'batch-op-reports',
            'ReportScope': 'AllTasks'
        },
        Manifest={
            'Spec': {
                'Format': 'S3BatchOperations_CSV_20180820',
                'Fields': ['Bucket', 'Key']
            },
            'Location': {
                'ObjectArn': manifest_location,
                'ETag': etag
            }
        },
        Priority=10,
        RoleArn=role_arn,
        Description='Batch job for tagging objects',
        ConfirmationRequired=True
    )
    job_id = response['JobId']
    print(f"The Job id is {job_id}")
    return job_id
except ClientError as e:
    print(f"Error creating batch job: {e}")
    if 'Message' in str(e):
        print(f"Detailed error message: {e.response['Message']}")
    raise

def check_job_failure_reasons(self, job_id: str, account_id: str) ->
List[Dict[str, Any]]:
    """
    Check for any failure reasons of a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID

    Returns:
        list: List of failure reasons

    Raises:

```

```

        ClientError: If checking job failure reasons fails
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        if 'FailureReasons' in response['Job']:
            for reason in response['Job']['FailureReasons']:
                print(f"- {reason}")
        return response['Job'].get('FailureReasons', [])
    except ClientError as e:
        print(f"Error checking job failure reasons: {e}")
        raise

def wait_for_job_ready(self, job_id: str, account_id: str, desired_status:
str = 'Ready') -> bool:
    """
    Wait for a job to reach the desired status.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
        desired_status (str): Target status to wait for

    Returns:
        bool: True if desired status is reached, False otherwise

    Raises:
        ClientError: If checking job status fails
    """
    print(f"Waiting for job to become {desired_status}...")
    max_attempts = 60
    attempt = 0
    while attempt < max_attempts:
        try:
            response = self.s3control_client.describe_job(
                AccountId=account_id,
                JobId=job_id
            )
            current_status = response['Job']['Status']
            print(f"Current job status: {current_status}")
            if current_status == desired_status:
                return True

```

```

        if current_status == 'Suspended':
            print("Job is in Suspended state, can proceed with
activation")

            return True
        if current_status in ['Active', 'Failed', 'Cancelled',
'Complete']:
            print(f"Job is in {current_status} state, cannot reach
{desired_status} status")
            if 'FailureReasons' in response['Job']:
                print("Failure reasons:")
                for reason in response['Job']['FailureReasons']:
                    print(f"- {reason}")
            return False

        time.sleep(20)
        attempt += 1
    except ClientError as e:
        print(f"Error checking job status: {e}")
        raise
print(f"Timeout waiting for job to become {desired_status}")
return False

def update_job_priority(self, job_id: str, account_id: str) -> None:
    """
    Update the priority of a batch job and start it.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended']:
            self.s3control_client.update_job_priority(
                AccountId=account_id,
                JobId=job_id,
                Priority=60
            )

```

```

        print("The job priority was updated")

        try:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,
                RequestedJobStatus='Ready'
            )
            print("Job activated successfully")
        except ClientError as activation_error:
            print(f"Note: Could not activate job automatically:
{activation_error}")
            print("Job priority was updated successfully. Job may need
manual activation in the console.")
            elif current_status in ['Active', 'Completing', 'Complete']:
                print(f"Job is in '{current_status}' state - priority cannot be
updated")

                if current_status == 'Completing':
                    print("Job is finishing up and will complete soon.")
                elif current_status == 'Complete':
                    print("Job has already completed successfully.")
                else:
                    print("Job is currently running.")
            else:
                print(f"Job is in '{current_status}' state - priority update not
allowed")

        except ClientError as e:
            print(f"Error updating job priority: {e}")
            print("Continuing with the scenario...")
            return

def cancel_job(self, job_id: str, account_id: str) -> None:
    """
    Cancel an S3 batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id

```



```

    )
    current_status = response['Job']['Status']
    print(f"Current job status: {current_status}")

    if current_status in ['Ready', 'Suspended', 'Active']:
        self.s3control_client.update_job_status(
            AccountId=account_id,
            JobId=job_id,
            RequestedJobStatus='Cancelled'
        )
        print(f"Job {job_id} was successfully canceled.")
    elif current_status in ['Completing', 'Complete']:
        print(f"Job is in '{current_status}' state - cannot be
cancelled")
        if current_status == 'Completing':
            print("Job is finishing up and will complete soon.")
        elif current_status == 'Complete':
            print("Job has already completed successfully.")
    else:
        print(f"Job is in '{current_status}' state - cancel not allowed")
except ClientError as e:
    print(f"Error canceling job: {e}")
    raise

def describe_job_details(self, job_id: str, account_id: str) -> None:
    """
    Describe detailed information about a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        job = response['Job']
        print(f"Job ID: {job['JobId']}")
        print(f"Description: {job.get('Description', 'N/A')}")
        print(f"Status: {job['Status']}")
        print(f"Role ARN: {job['RoleArn']}")
        print(f"Priority: {job['Priority']}")
        if 'ProgressSummary' in job:

```

```

        progress = job['ProgressSummary']
        print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
              f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
              f"Failed={progress.get('NumberOfTasksFailed', 0)}")
    except ClientError as e:
        print(f"Error describing job: {e}")
        raise

def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.get_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        tags = response.get('Tags', [])
        if tags:
            print(f"Tags for job {job_id}:")
            for tag in tags:
                print(f"  {tag['Key']}: {tag['Value']}")
        else:
            print(f"No tags found for job ID: {job_id}")
    except ClientError as e:
        print(f"Error getting job tags: {e}")
        raise

def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Add tags to a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.put_job_tagging(
            AccountId=account_id,

```

```

        JobId=job_id,
        Tags=[
            {'Key': 'Environment', 'Value': 'Development'},
            {'Key': 'Team', 'Value': 'DataProcessing'}
        ]
    )
    print(f"Additional tags were added to job {job_id}")
except ClientError as e:
    print(f"Error adding job tags: {e}")
    raise

def list_jobs(self, account_id: str) -> None:
    """
    List all batch jobs for the account.

    Args:
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.list_jobs(
            AccountId=account_id,
            JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
        )
        jobs = response.get('Jobs', [])
        for job in jobs:
            print(f"The job id is {job['JobId']}")
            print(f"The job priority is {job['Priority']}")
    except ClientError as e:
        print(f"Error listing jobs: {e}")
        raise

def delete_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Delete all tags from a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.delete_job_tagging(
            AccountId=account_id,
            JobId=job_id

```

```

    )
    print(f"You have successfully deleted {job_id} tagging.")
except ClientError as e:
    print(f"Error deleting job tags: {e}")
    raise

def cleanup_resources(self, bucket_name: str, file_names: List[str]) -> None:
    """
    Clean up all resources created during the scenario.

    Args:
        bucket_name (str): Name of the bucket to clean up
        file_names (list): List of files to delete

    Raises:
        ClientError: If cleanup fails
    """
    try:
        for file_name in file_names:
            self.s3_client.delete_object(Bucket=bucket_name, Key=file_name)
            print(f"Deleted {file_name}")

        response = self.s3_client.list_objects_v2(
            Bucket=bucket_name,
            Prefix='batch-op-reports/'
        )
        if 'Contents' in response:
            for obj in response['Contents']:
                self.s3_client.delete_object(
                    Bucket=bucket_name,
                    Key=obj['Key']
                )
                print(f"Deleted {obj['Key']}")

            self.s3_client.delete_bucket(Bucket=bucket_name)
            print(f"Deleted bucket {bucket_name}")
    except ClientError as e:
        print(f"Error in cleanup: {e}")
        raise

```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Python (Boto3).

- [CreateJob](#)
- [DeleteJobTagging](#)
- [DescribeJob](#)
- [GetJobTagging](#)
- [ListJobs](#)
- [PutJobTagging](#)
- [UpdateJobPriority](#)
- [UpdateJobStatus](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Actions pour Amazon S3 Control à l'aide de AWS Kits SDK

Les exemples de code suivants montrent comment effectuer des actions Amazon S3 Control individuelles avec des AWS SDK. Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions pour configurer et exécuter le code.

Les exemples suivants incluent uniquement les actions les plus couramment utilisées. Pour obtenir la liste complète, consultez la [Référence des API Amazon S3 Control](#).

Exemples

- [À utiliser CreateJob avec un AWS SDK ou CLI](#)
- [À utiliser DeleteJobTagging avec un AWS Kit SDK](#)
- [À utiliser DescribeJob avec un AWS SDK ou CLI](#)
- [À utiliser GetJobTagging avec un AWS Kit SDK](#)
- [À utiliser PutJobTagging avec un AWS Kit SDK](#)
- [À utiliser UpdateJobPriority avec un AWS SDK ou CLI](#)
- [À utiliser UpdateJobStatus avec un AWS SDK ou CLI](#)

À utiliser **CreateJob** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser CreateJob.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

CLI

AWS CLI

Pour créer une tâche d'opérations par lot Amazon S3

L'exemple `create-job` suivant crée une tâche d'opérations par lot Amazon S3 pour baliser les objets comme `confidential` in the bucket `employee-records`.

```
aws s3control create-job \
  --account-id 123456789012 \
  --operation '{"S3PutObjectTagging": { "TagSet": [{"Key":"confidential",
    "Value":"true"}] }}' \
  --report '{"Bucket":"arn:aws:s3:::employee-records-logs", "Prefix":"batch-op-
create-job",
    "Format":"Report_CSV_20180820", "Enabled":true, "ReportScope":"AllTasks"}' \
  --manifest '{"Spec":{"Format":"S3BatchOperations_CSV_20180820", "Fields":
["Bucket", "Key"]}, "Location":{"ObjectArn":"arn:aws:s3:::employee-records-logs/
inv-report/7a6a9be4-072c-407e-85a2-
ec3e982f773e.csv", "ETag":"69f52a4e9f797e987155d9c8f5880897"}}' \
  --priority 42 \
  --role-arn arn:aws:iam::123456789012:role/S3BatchJobRole
```

Sortie :

```
{
  "JobId": "93735294-df46-44d5-8638-6356f335324e"
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateJob](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez une tâche S3 asynchrone.

```
/**
 * Creates an asynchronous S3 job using the AWS Java SDK.
 *
 * @param accountId      the AWS account ID associated with the job
 * @param iamRoleArn      the ARN of the IAM role to be used for the job
 * @param manifestLocation the location of the job manifest file in S3
 * @param reportBucketName the name of the S3 bucket to store the job report
 * @param uuid            a unique identifier for the job
 * @return a CompletableFuture that represents the asynchronous creation of
 the S3 job.
 *      The CompletableFuture will return the job ID if the job is created
 successfully,
 *      or throw an exception if there is an error.
 */
public CompletableFuture<String> createS3JobAsync(String accountId, String
iamRoleArn,
                                                String manifestLocation,
String reportBucketName, String uuid) {

    String[] bucketName = new String[]{" "};
    String[] parts = reportBucketName.split("::");
    if (parts.length > 1) {
        bucketName[0] = parts[1];
    } else {
        System.out.println("The input string does not contain the expected
format.");
    }

    return CompletableFuture.supplyAsync(() -> getETag(bucketName[0], "job-
manifest.csv"))
        .thenCompose(eTag -> {
```

```
        ArrayList<S3Tag> tagSet = new ArrayList<>();
        S3Tag s3Tag = S3Tag.builder()
            .key("keyOne")
            .value("ValueOne")
            .build();
        S3Tag s3Tag2 = S3Tag.builder()
            .key("keyTwo")
            .value("ValueTwo")
            .build();
        tagSet.add(s3Tag);
        tagSet.add(s3Tag2);

        S3SetObjectTaggingOperation objectTaggingOperation =
        S3SetObjectTaggingOperation.builder()
            .tagSet(tagSet)
            .build();

        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectTagging(objectTaggingOperation)
            .build();

        JobManifestLocation jobManifestLocation =
        JobManifestLocation.builder()
            .objectArn(manifestLocation)
            .eTag(eTag)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest jobManifest = JobManifest.builder()
            .spec(manifestSpec)
            .location(jobManifestLocation)
            .build();

        JobReport jobReport = JobReport.builder()
            .bucket(reportBucketName)
            .prefix("reports")
            .format("Report_CSV_20180820")
            .enabled(true)
            .reportScope("AllTasks")
            .build();
```



```

        CreateJobRequest jobRequest = CreateJobRequest.builder()
            .accountId(accountId)
            .description("Job created using the AWS Java SDK")
            .manifest(jobManifest)
            .operation(jobOperation)
            .report(jobReport)
            .priority(42)
            .roleArn(iamRoleArn)
            .clientRequestToken(uuid)
            .confirmationRequired(false)
            .build();

        // Create the job asynchronously.
        return getAsyncClient().createJob(jobRequest)
            .thenApply(CreateJobResponse::jobId);
    })
    .handle((jobId, ex) -> {
        if (ex != null) {
            Throwable cause = (ex instanceof CompletionException) ?
ex.getCause() : ex;
            if (cause instanceof S3ControlException) {
                throw new CompletionException(cause);
            } else {
                throw new RuntimeException(cause);
            }
        }
        return jobId;
    });
}

```

Créez une tâche de conservation de la conformité.

```

/**
 * Creates a compliance retention job in Amazon S3 Control.
 * <p>
 * A compliance retention job in Amazon S3 Control is a feature that allows
you to
 * set a retention period for objects stored in an S3 bucket.
 * This feature is particularly useful for organizations that need to comply
with

```

```

    * regulatory requirements or internal policies that mandate the retention of
    data for
    * a specific duration.
    *
    * @param s3ControlClient The S3ControlClient instance to use for the API
    call.
    * @return The job ID of the created compliance retention job.
    */
    public static String createComplianceRetentionJob(final S3ControlClient
    s3ControlClient, String roleArn, String bucketName, String accountId) {
        final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
    bucket/compliance-objects-manifest.csv";
        final String manifestObjectVersionId = "your-object-version-Id";

        Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
        JobOperation jobOperation = JobOperation.builder()
            .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
                .retention(S3Retention.builder()
                    .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                    .retainUntilDate(jan2025)
                    .build())
                .build())
            .build();

        JobManifestLocation manifestLocation = JobManifestLocation.builder()
            .objectArn(manifestObjectArn)
            .eTag(manifestObjectVersionId)
            .build();

        JobManifestSpec manifestSpec = JobManifestSpec.builder()
            .fieldsWithStrings("Bucket", "Key")
            .format("S3BatchOperations_CSV_20180820")
            .build();

        JobManifest manifestToPublicApi = JobManifest.builder()
            .location(manifestLocation)
            .spec(manifestSpec)
            .build();

        // Report details.
        final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
        final String jobReportPrefix = "reports/compliance-objects-bops";

        JobReport jobReport = JobReport.builder()

```

```

        .enabled(true)
        .reportScope(JobReportScope.ALL_TASKS)
        .bucket(jobReportBucketArn)
        .prefix(jobReportPrefix)
        .format(JobReportFormat.REPORT_CSV_20180820)
        .build();

final Boolean requiresConfirmation = true;
final int priority = 10;
CreateJobRequest request = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Set compliance retain-until to 1 Jan 2025")
    .manifest(manifestToPublicApi)
    .operation(jobOperation)
    .priority(priority)
    .roleArn(roleArn)
    .report(jobReport)
    .confirmationRequired(requiresConfirmation)
    .build();

// Create the job and get the result.
CreateJobResponse result = s3ControlClient.createJob(request);
return result.jobId();
}

```

Créez une tâche de mise en suspens juridique.

```

/**
 * Creates a compliance retention job in Amazon S3 Control.
 * <p>
 * A compliance retention job in Amazon S3 Control is a feature that allows
you to
 * set a retention period for objects stored in an S3 bucket.
 * This feature is particularly useful for organizations that need to comply
with
 * regulatory requirements or internal policies that mandate the retention of
data for
 * a specific duration.
 *
 * @param s3ControlClient The S3ControlClient instance to use for the API
call.

```

```
* @return The job ID of the created compliance retention job.
*/
public static String createComplianceRetentionJob(final S3ControlClient
s3ControlClient, String roleArn, String bucketName, String accountId) {
    final String manifestObjectArn = "arn:aws:s3:::amzn-s3-demo-manifest-
bucket/compliance-objects-manifest.csv";
    final String manifestObjectVersionId = "your-object-version-Id";

    Instant jan2025 = Instant.parse("2025-01-01T00:00:00Z");
    JobOperation jobOperation = JobOperation.builder()
        .s3PutObjectRetention(S3SetObjectRetentionOperation.builder()
            .retention(S3Retention.builder()
                .mode(S3ObjectLockRetentionMode.COMPLIANCE)
                .retainUntilDate(jan2025)
                .build())
            .build())
        .build();

    JobManifestLocation manifestLocation = JobManifestLocation.builder()
        .objectArn(manifestObjectArn)
        .eTag(manifestObjectVersionId)
        .build();

    JobManifestSpec manifestSpec = JobManifestSpec.builder()
        .fieldsWithStrings("Bucket", "Key")
        .format("S3BatchOperations_CSV_20180820")
        .build();

    JobManifest manifestToPublicApi = JobManifest.builder()
        .location(manifestLocation)
        .spec(manifestSpec)
        .build();

    // Report details.
    final String jobReportBucketArn = "arn:aws:s3:::" + bucketName;
    final String jobReportPrefix = "reports/compliance-objects-bops";

    JobReport jobReport = JobReport.builder()
        .enabled(true)
        .reportScope(JobReportScope.ALL_TASKS)
        .bucket(jobReportBucketArn)
        .prefix(jobReportPrefix)
        .format(JobReportFormat.REPORT_CSV_20180820)
        .build();
```

```
final Boolean requiresConfirmation = true;
final int priority = 10;
CreateJobRequest request = CreateJobRequest.builder()
    .accountId(accountId)
    .description("Set compliance retain-until to 1 Jan 2025")
    .manifest(manifestToPublicApi)
    .operation(jobOperation)
    .priority(priority)
    .roleArn(roleArn)
    .report(jobReport)
    .confirmationRequired(requiresConfirmation)
    .build();

// Create the job and get the result.
CreateJobResponse result = s3ControlClient.createJob(request);
return result.jobId();
}
```

Créez une nouvelle tâche de conservation de la gouvernance.

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */
public class CreateGovernanceRetentionJob {

    public static void main(String[] args) throws ParseException {
        final String usage = ""

            Usage:
                <manifestObjectArn> <jobReportBucketArn> <roleArn> <accountId>
                <manifestObjectVersionId>

            Where:
                manifestObjectArn - The Amazon Resource Name (ARN) of the S3
                object that contains the manifest file for the governance objects.\s
    }
```

```

        bucketName - The ARN of the S3 bucket where the job report will
be stored.
        roleArn - The ARN of the IAM role that will be used to perform
the governance retention operation.
        accountId - Your AWS account Id.
        manifestObjectVersionId = A unique value that is used as the
`eTag` property of the `JobManifestLocation` object.
        """;

    if (args.length != 4) {
        System.out.println(usage);
        return;
    }

    String manifestObjectArn = args[0];
    String jobReportBucketArn = args[1];
    String roleArn = args[2];
    String accountId = args[3];
    String manifestObjectVersionId = args[4];

    S3ControlClient s3ControlClient = S3ControlClient.create();
    createGovernanceRetentionJob(s3ControlClient, manifestObjectArn,
jobReportBucketArn, roleArn, accountId, manifestObjectVersionId);
}

public static String createGovernanceRetentionJob(final S3ControlClient
s3ControlClient, String manifestObjectArn, String jobReportBucketArn, String
roleArn, String accountId, String manifestObjectVersionId) throws ParseException
{
    final JobManifestLocation manifestLocation =
JobManifestLocation.builder()
        .objectArn(manifestObjectArn)
        .eTag(manifestObjectVersionId)
        .build();

    final JobManifestSpec manifestSpec = JobManifestSpec.builder()
        .format(JobManifestFormat.S3_BATCH_OPERATIONS_CSV_20180820)
        .fields(Arrays.asList(JobManifestFieldName.BUCKET,
JobManifestFieldName.KEY))
        .build();

    final JobManifest manifestToPublicApi = JobManifest.builder()
        .location(manifestLocation)
        .spec(manifestSpec)

```

```
        .build();

    final String jobReportPrefix = "reports/governance-objects";
    final JobReport jobReport = JobReport.builder()
        .enabled(true)
        .reportScope(JobReportScope.ALL_TASKS)
        .bucket(jobReportBucketArn)
        .prefix(jobReportPrefix)
        .format(JobReportFormat.REPORT_CSV_20180820)
        .build();

    final SimpleDateFormat format = new SimpleDateFormat("dd/MM/yyyy");
    final Date jan30th = format.parse("30/01/2025");

    final S3SetObjectRetentionOperation s3SetObjectRetentionOperation =
    S3SetObjectRetentionOperation.builder()
        .retention(S3Retention.builder()
            .mode(S3ObjectLockRetentionMode.GOVERNANCE)
            .retainUntilDate(jan30th.toInstant())
            .build())
        .build();

    final JobOperation jobOperation = JobOperation.builder()
        .s3PutObjectRetention(s3SetObjectRetentionOperation)
        .build();

    final Boolean requiresConfirmation = true;
    final int priority = 10;

    final CreateJobRequest request = CreateJobRequest.builder()
        .accountId(accountId)
        .description("Put governance retention")
        .manifest(manifestToPublicApi)
        .operation(jobOperation)
        .priority(priority)
        .roleArn(roleArn)
        .report(jobReport)
        .confirmationRequired(requiresConfirmation)
        .build();

    final CreateJobResponse result = s3ControlClient.createJob(request);
    return result.jobId();
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateJob](#) à la section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
def create_s3_batch_job(self, account_id: str, role_arn: str,
manifest_location: str,
                        report_bucket_name: str) -> str:
    """
    Create an S3 batch operation job.

    Args:
        account_id (str): AWS account ID
        role_arn (str): IAM role ARN for batch operations
        manifest_location (str): Location of the manifest file
        report_bucket_name (str): Bucket for job reports

    Returns:
        str: Job ID

    Raises:
        ClientError: If job creation fails
    """
    try:
        bucket_name = manifest_location.split('::')[1].split('/')[0]
        manifest_key = 'job-manifest.csv'
        manifest_obj = self.s3_client.head_object(
            Bucket=bucket_name,
            Key=manifest_key
        )
        etag = manifest_obj['ETag'].strip('')
```



```
response = self.s3control_client.create_job(
    AccountId=account_id,
    Operation={
        'S3PutObjectTagging': {
            'TagSet': [
                {
                    'Key': 'BatchTag',
                    'Value': 'BatchValue'
                },
            ],
        }
    },
    Report={
        'Bucket': report_bucket_name,
        'Format': 'Report_CSV_20180820',
        'Enabled': True,
        'Prefix': 'batch-op-reports',
        'ReportScope': 'AllTasks'
    },
    Manifest={
        'Spec': {
            'Format': 'S3BatchOperations_CSV_20180820',
            'Fields': ['Bucket', 'Key']
        },
        'Location': {
            'ObjectArn': manifest_location,
            'ETag': etag
        }
    },
    Priority=10,
    RoleArn=role_arn,
    Description='Batch job for tagging objects',
    ConfirmationRequired=True
)
job_id = response['JobId']
print(f"The Job id is {job_id}")
return job_id
except ClientError as e:
    print(f"Error creating batch job: {e}")
    if 'Message' in str(e):
        print(f"Detailed error message: {e.response['Message']}")
    raise
```

- Pour plus de détails sur l'API, consultez [CreateJob](#) AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.
  " iv_manifest_arn  = 'arn:aws:s3::my-bucket/job-manifest.csv'
  " iv_manifest_etag = 'abc123def456'
  " iv_report_bucket = 'arn:aws:s3::my-report-bucket'
  DATA(lo_result) = lo_s3c->createjob(
    iv_accountid      = iv_account_id
    iv_rolearn        = iv_role_arn
    iv_confirmationrequired = abap_true
    iv_priority        = 10
    iv_description     = 'Batch job for tagging objects'
    io_operation       = NEW /aws1/cl_s3cjoboperation(
      io_s3putobjecttagging = NEW /aws1/cl_s3cs3setobjecttagop(
        it_tagset = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
          ( NEW /aws1/cl_s3cs3tag(
            iv_key   = 'BatchTag'
            iv_value = 'BatchValue' ) )
        )
      )
    )
    io_manifest      = NEW /aws1/cl_s3cjobmanifest(
      io_spec        = NEW /aws1/cl_s3cjobmanifestspec(
        iv_format    = 'S3BatchOperations_CSV_20180820'
        it_fields    = VALUE /aws1/
      cl_s3cjobmanifestfield00=>tt_jobmanifestfieldlist(
        ( NEW /aws1/cl_s3cjobmanifestfield00( 'Bucket' ) )
        ( NEW /aws1/cl_s3cjobmanifestfield00( 'Key' ) )
      )
    )
  )
```

```

        )
    )
    io_location = NEW /aws1/cl_s3cjobmanifestloc(
        iv_objectarn = iv_manifest_arn
        iv_etag       = iv_manifest_etag
    )
)
io_report      = NEW /aws1/cl_s3cjobreport(
    iv_bucket    = iv_report_bucket
    iv_format     = 'Report_CSV_20180820'
    iv_enabled    = abap_true
    iv_prefix     = 'batch-op-reports'
    iv_reportscope = 'AllTasks'
)
).
ov_job_id = lo_result->get_jobid( ).
MESSAGE |S3 Batch Operations job created: { ov_job_id }| TYPE 'I'.
CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_bad.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- Pour plus de détails sur l'API, consultez [CreateJob](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteJobTagging** avec un AWS Kit SDK

Les exemples de code suivants illustrent comment utiliser DeleteJobTagging.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Asynchronously deletes the tags associated with a specific batch job.
 *
 * @param jobId      The ID of the batch job whose tags should be deleted.
 * @param accountId The ID of the account associated with the batch job.
 * @return A CompletableFuture that completes when the job tags have been
 * successfully deleted, or an exception is thrown if the deletion fails.
 */
public CompletableFuture<Void> deleteBatchJobTagsAsync(String jobId, String
accountId) {
    DeleteJobTaggingRequest jobTaggingRequest =
DeleteJobTaggingRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .build();

    return asyncClient.deleteJobTagging(jobTaggingRequest)
        .thenAccept(response -> {
            System.out.println("You have successfully deleted " + jobId + "
tagging.");
        })
        .exceptionally(ex -> {
            System.err.println("Failed to delete job tags: " +
ex.getMessage());
            throw new RuntimeException(ex);
        });
}
```

```
    });  
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteJobTagging](#) à la section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
def delete_job_tags(self, job_id: str, account_id: str) -> None:  
    """  
    Delete all tags from a batch job.  
  
    Args:  
        job_id (str): ID of the batch job  
        account_id (str): AWS account ID  
    """  
    try:  
        self.s3control_client.delete_job_tagging(  
            AccountId=account_id,  
            JobId=job_id  
        )  
        print(f"You have successfully deleted {job_id} tagging.")  
    except ClientError as e:  
        print(f"Error deleting job tags: {e}")  
        raise
```

- Pour plus de détails sur l'API, consultez [DeleteJobTagging](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
TRY.  
    lo_s3c->deletejobtagging(  
        iv_accountid = iv_account_id  
        iv_jobid      = iv_job_id  
    ).  
    MESSAGE |Tags deleted from job { iv_job_id }| TYPE 'I'.  
CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).  
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.  
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic  
        EXPORTING previous = lo_ex_nf.  
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).  
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.  
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic  
        EXPORTING previous = lo_ex_cli.  
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).  
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.  
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic  
        EXPORTING previous = lo_ex_srv.  
ENDTRY.
```

- Pour plus de détails sur l'API, consultez [DeleteJobTagging](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DescribeJob** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser DescribeJob.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

CLI

AWS CLI

Pour décrire une tâche d'opérations par lots Amazon S3

`describe-job` ci-dessous fournit les paramètres de configuration et l'état de la tâche d'opérations par lots spécifiée.

```
aws s3control describe-job \  
  --account-id 123456789012 \  
  --job-id 93735294-df46-44d5-8638-6356f335324e
```

Sortie :

```
{  
  "Job": {  
    "TerminationDate": "2019-10-03T21:49:53.944Z",  
    "JobId": "93735294-df46-44d5-8638-6356f335324e",  
    "FailureReasons": [],  
    "Manifest": {  
      "Spec": {  
        "Fields": [  
          "Bucket",  
          "Key"  
        ],  
        "Format": "S3BatchOperations_CSV_20180820"  
      },  
      "Location": {  
        "ETag": "69f52a4e9f797e987155d9c8f5880897",  
        "ObjectArn": "arn:aws:s3:::employee-records-logs/inv-report/7a6a9be4-072c-407e-85a2-ec3e982f773e.csv"  
      }  
    },  
    "Operation": {  
      "S3PutObjectTagging": {
```

```
        "TagSet": [
            {
                "Value": "true",
                "Key": "confidential"
            }
        ]
    },
    "RoleArn": "arn:aws:iam::123456789012:role/S3BatchJobRole",
    "ProgressSummary": {
        "TotalNumberOfTasks": 8,
        "NumberOfTasksFailed": 0,
        "NumberOfTasksSucceeded": 8
    },
    "Priority": 42,
    "Report": {
        "ReportScope": "AllTasks",
        "Format": "Report_CSV_20180820",
        "Enabled": true,
        "Prefix": "batch-op-create-job",
        "Bucket": "arn:aws:s3:::employee-records-logs"
    },
    "JobArn": "arn:aws:s3:us-west-2:123456789012:job/93735294-
df46-44d5-8638-6356f335324e",
    "CreationTime": "2019-10-03T21:48:48.048Z",
    "Status": "Complete"
}
}
```

- Pour plus de détails sur l'API, reportez-vous [DescribeJob](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).


```
/**
 * Asynchronously describes the specified job.
 *
 * @param jobId      the ID of the job to describe
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job
description is available
 * @throws RuntimeException if an error occurs while describing the job
 */
public CompletableFuture<Void> describeJobAsync(String jobId, String
accountId) {
    DescribeJobRequest jobRequest = DescribeJobRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return getAsyncClient().describeJob(jobRequest)
        .thenAccept(response -> {
            System.out.println("Job ID: " + response.job().jobId());
            System.out.println("Description: " +
response.job().description());
            System.out.println("Status: " + response.job().statusAsString());
            System.out.println("Role ARN: " + response.job().roleArn());
            System.out.println("Priority: " + response.job().priority());
            System.out.println("Progress Summary: " +
response.job().progressSummary());

            // Print out details about the job manifest.
            JobManifest manifest = response.job().manifest();
            System.out.println("Manifest Location: " +
manifest.location().objectArn());
            System.out.println("Manifest ETag: " +
manifest.location().eTag());

            // Print out details about the job operation.
            JobOperation operation = response.job().operation();
            if (operation.s3PutObjectTagging() != null) {
                System.out.println("Operation: S3 Put Object Tagging");
                System.out.println("Tag Set: " +
operation.s3PutObjectTagging().tagSet());
            }

            // Print out details about the job report.
```

```

        JobReport report = response.job().report();
        System.out.println("Report Bucket: " + report.bucket());
        System.out.println("Report Prefix: " + report.prefix());
        System.out.println("Report Format: " + report.format());
        System.out.println("Report Enabled: " + report.enabled());
        System.out.println("Report Scope: " +
report.reportScopeAsString());
    })
    .exceptionally(ex -> {
        System.err.println("Failed to describe job: " + ex.getMessage());
        throw new RuntimeException(ex);
    });
}

```

- Pour plus de détails sur l'API, reportez-vous [DescribeJob](#) à la section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

def describe_job_details(self, job_id: str, account_id: str) -> None:
    """
    Describe detailed information about a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
    
```

```

        job = response['Job']
        print(f"Job ID: {job['JobId']}")
        print(f"Description: {job.get('Description', 'N/A')}")
        print(f"Status: {job['Status']}")
        print(f"Role ARN: {job['RoleArn']}")
        print(f"Priority: {job['Priority']}")
        if 'ProgressSummary' in job:
            progress = job['ProgressSummary']
            print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                  f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                  f"Failed={progress.get('NumberOfTasksFailed', 0)}")
        except ClientError as e:
            print(f"Error describing job: {e}")
            raise

```

- Pour plus de détails sur l'API, consultez [DescribeJob](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_s3c->describejob(           " oo_result is returned for
testing purposes.
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    ).
    DATA(lo_job) = oo_result->get_job( ).
    DATA(lv_status) = lo_job->get_status( ).
    DATA(lv_priority) = lo_job->get_priority( ).
    DATA(lo_progress) = lo_job->get_progresssummary( ).
    IF lo_progress IS NOT INITIAL.

```

```

        MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }, | &&
        |total={ lo_progress->get_totalnumberoftasks( ) }, | &&
        |succeeded={ lo_progress->get_numberoftaskssucceeded( ) }, | &&
        |failed={ lo_progress->get_numberoftasksfailed( ) }| TYPE 'I'.

    ELSE.
        MESSAGE |Job { iv_job_id }: status={ lv_status },
priority={ lv_priority }| TYPE 'I'.
    ENDIF.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
        MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
            EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
        MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
            EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
        MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
        RAISE EXCEPTION TYPE /aws1/cx_rt_generic
            EXPORTING previous = lo_ex_srv.
    ENDTRY.

```

- Pour plus de détails sur l'API, consultez [DescribeJob](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetJobTagging** avec un AWS Kit SDK

Les exemples de code suivants illustrent comment utiliser GetJobTagging.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Asynchronously retrieves the tags associated with a specific job in an AWS
 * account.
 *
 * @param jobId      the ID of the job for which to retrieve the tags
 * @param accountId the ID of the AWS account associated with the job
 * @return a {@link CompletableFuture} that completes when the job tags have
 *         been retrieved, or with an exception if the operation fails
 * @throws RuntimeException if an error occurs while retrieving the job tags
 */
public CompletableFuture<Void> getJobTagsAsync(String jobId, String
accountId) {
    GetJobTaggingRequest request = GetJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .build();

    return asyncClient.getJobTagging(request)
        .thenAccept(response -> {
            List<S3Tag> tags = response.tags();
            if (tags.isEmpty()) {
                System.out.println("No tags found for job ID: " + jobId);
            } else {
                for (S3Tag tag : tags) {
                    System.out.println("Tag key is: " + tag.key());
                    System.out.println("Tag value is: " + tag.value());
                }
            }
        })
        .exceptionally(ex -> {
            System.err.println("Failed to get job tags: " + ex.getMessage());
            throw new RuntimeException(ex); // Propagate the exception
        });
}
```

```
    });  
}
```

- Pour plus de détails sur l'API, reportez-vous [GetJobTagging](#) à la section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.get_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        tags = response.get('Tags', [])
        if tags:
            print(f"Tags for job {job_id}:")
            for tag in tags:
                print(f"  {tag['Key']}: {tag['Value']}")
        else:
            print(f"No tags found for job ID: {job_id}")
    except ClientError as e:
        print(f"Error getting job tags: {e}")
        raise
```

- Pour plus de détails sur l'API, consultez [GetJobTagging](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_s3c->getjobtagging(          " oo_result is returned for
testing purposes.
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    ).
    DATA(lt_tags) = oo_result->get_tags( ).
    MESSAGE |Retrieved { lines( lt_tags ) } tag(s) for job { iv_job_id }|
TYPE 'I'.
CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_nf.
CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- Pour plus de détails sur l'API, consultez [GetJobTagging](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutJobTagging** avec un AWS Kit SDK

Les exemples de code suivants illustrent comment utiliser PutJobTagging.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Asynchronously adds tags to a job in the system.
 *
 * @param jobId      the ID of the job to add tags to
 * @param accountId the account ID associated with the job
 * @return a CompletableFuture that completes when the tagging operation is
 * finished
 */
public CompletableFuture<Void> putJobTaggingAsync(String jobId, String
accountId) {
    S3Tag departmentTag = S3Tag.builder()
        .key("department")
        .value("Marketing")
        .build();

    S3Tag fiscalYearTag = S3Tag.builder()
        .key("FiscalYear")
```



```

        .value("2020")
        .build();

    PutJobTaggingRequest putJobTaggingRequest =
    PutJobTaggingRequest.builder()
        .jobId(jobId)
        .accountId(accountId)
        .tags(departmentTag, fiscalYearTag)
        .build();

    return asyncClient.putJobTagging(putJobTaggingRequest)
        .thenRun(() -> {
            System.out.println("Additional Tags were added to job " + jobId);
        })
        .exceptionally(ex -> {
            System.err.println("Failed to add tags to job: " +
ex.getMessage());
            throw new RuntimeException(ex); // Propagate the exception
        });
}

```

- Pour plus de détails sur l'API, reportez-vous [PutJobTagging](#) à la section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Add tags to a batch job.

    Args:
        job_id (str): ID of the batch job

```

```

        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.put_job_tagging(
            AccountId=account_id,
            JobId=job_id,
            Tags=[
                {'Key': 'Environment', 'Value': 'Development'},
                {'Key': 'Team', 'Value': 'DataProcessing'}
            ]
        )
        print(f"Additional tags were added to job {job_id}")
    except ClientError as e:
        print(f"Error adding job tags: {e}")
        raise

```

- Pour plus de détails sur l'API, consultez [PutJobTagging](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
  lo_s3c->putjobtagging(
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    it_tags       = VALUE /aws1/cl_s3cs3tag=>tt_s3tagset(
      ( NEW /aws1/cl_s3cs3tag(
        iv_key   = 'Environment'
        iv_value = 'Development' ) )
      ( NEW /aws1/cl_s3cs3tag(
        iv_key   = 'Team'
        iv_value = 'DataProcessing' ) )
    )

```

```

    )
  ).
  MESSAGE |Tags added to job { iv_job_id }| TYPE 'I'.
  CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
  MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_nf.
  CATCH /aws1/cx_s3ctoomanytagsex INTO DATA(lo_ex_tags).
  MESSAGE lo_ex_tags->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_tags.
  CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
  MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
  CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
  MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
  RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- Pour plus de détails sur l'API, consultez [PutJobTagging](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **UpdateJobPriority** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser UpdateJobPriority.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

CLI

AWS CLI

Pour mettre à jour la priorité d'une tâche d'opérations par lot Amazon S3

L'exemple `update-job-priority` suivant met à jour la tâche spécifiée en fonction de la nouvelle priorité.

```
aws s3control update-job-priority \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
  --priority 52
```

Sortie :

```
{
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",
  "Priority": 52
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateJobPriority](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Updates the priority of a job asynchronously.
 *
 * @param jobId      the ID of the job to update
 * @param accountId the ID of the account associated with the job
```

```
    * @return a {@link CompletableFuture} that represents the asynchronous
    operation, which completes when the job priority has been updated or an error
    has occurred
    */
    public CompletableFuture<Void> updateJobPriorityAsync(String jobId, String
accountId) {
        UpdateJobPriorityRequest priorityRequest =
UpdateJobPriorityRequest.builder()
            .accountId(accountId)
            .jobId(jobId)
            .priority(60)
            .build();

        CompletableFuture<Void> future = new CompletableFuture<>();
        getAsyncClient().updateJobPriority(priorityRequest)
            .thenAccept(response -> {
                System.out.println("The job priority was updated");
                future.complete(null); // Complete the CompletableFuture on
successful execution
            })
            .exceptionally(ex -> {
                System.err.println("Failed to update job priority: " +
ex.getMessage());
                future.completeExceptionally(ex); // Complete the
CompletableFuture exceptionally on error
                return null; // Return null to handle the exception
            });

        return future;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateJobPriority](#) à la section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
def update_job_priority(self, job_id: str, account_id: str) -> None:
    """
    Update the priority of a batch job and start it.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended']:
            self.s3control_client.update_job_priority(
                AccountId=account_id,
                JobId=job_id,
                Priority=60
            )
            print("The job priority was updated")

        try:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,
                RequestedJobStatus='Ready'
            )
            print("Job activated successfully")
        except ClientError as activation_error:
```

```

        print(f"Note: Could not activate job automatically:
{activation_error}")
        print("Job priority was updated successfully. Job may need
manual activation in the console.")
        elif current_status in ['Active', 'Completing', 'Complete']:
            print(f"Job is in '{current_status}' state - priority cannot be
updated")
            if current_status == 'Completing':
                print("Job is finishing up and will complete soon.")
            elif current_status == 'Complete':
                print("Job has already completed successfully.")
            else:
                print("Job is currently running.")
        else:
            print(f"Job is in '{current_status}' state - priority update not
allowed")

    except ClientError as e:
        print(f"Error updating job priority: {e}")
        print("Continuing with the scenario...")
        return

```

- Pour plus de détails sur l'API, consultez [UpdateJobPriority](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    oo_result = lo_s3c->updatejobpriority(    " oo_result is returned for
testing purposes.
        iv_accountid = iv_account_id
        iv_jobid      = iv_job_id

```

```

        iv_priority = 60
    ).
    MESSAGE |Job { oo_result->get_jobid( ) } priority updated to { oo_result-
>get_priority( ) }| TYPE 'I'.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
    MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_bad.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
        EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- Pour plus de détails sur l'API, consultez [UpdateJobPriority](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **UpdateJobStatus** avec un AWS SDK ou CLI

Les exemples de code suivants illustrent comment utiliser UpdateJobStatus.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

CLI

AWS CLI

Pour mettre à jour le statut d'une tâche d'opérations par lot Amazon S3

L'exemple `update-job-status` suivant annule la tâche spécifiée en attente d'approbation.

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
  --requested-job-status Cancelled
```

Sortie :

```
{
  "Status": "Cancelled",
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"
}
```

L'exemple `update-job-status` suivant confirme et exécute la tâche spécifiée en attente d'approbation.

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
  --requested-job-status Ready
```

Output::

```
{
  "Status": "Ready",
  "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"
}
```

L'exemple `update-job-status` suivant annule la tâche spécifiée en cours d'exécution.

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 5782949f-3301-4fb3-be34-8d5bab54dbca \
```

```
--requested-job-status Cancelled
```

Output::

```
{
    "Status": "Cancelling",
    "JobId": "5782949f-3301-4fb3-be34-8d5bab54dbca"
}
```

- Pour plus de détails sur l'API, reportez-vous [UpdateJobStatus](#) à la section Référence des AWS CLI commandes.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
/**
 * Cancels a job asynchronously.
 *
 * @param jobId The ID of the job to be canceled.
 * @param accountId The ID of the account associated with the job.
 * @return A {@link CompletableFuture} that completes when the job status has
 *         been updated to "CANCELLED".
 *         If an error occurs during the update, the returned future will
 *         complete exceptionally.
 */
public CompletableFuture<Void> cancelJobAsync(String jobId, String accountId)
{
    UpdateJobStatusRequest updateJobStatusRequest =
UpdateJobStatusRequest.builder()
        .accountId(accountId)
        .jobId(jobId)
        .requestedJobStatus(String.valueOf(JobStatus.CANCELLED))
        .build();

    return asyncClient.updateJobStatus(updateJobStatusRequest)
```

```

        .thenAccept(updateJobStatusResponse -> {
            System.out.println("Job status updated to: " +
updateJobStatusResponse.status());
        })
        .exceptionally(ex -> {
            System.err.println("Failed to cancel job: " + ex.getMessage());
            throw new RuntimeException(ex); // Propagate the exception
        });
    }
}

```

- Pour plus de détails sur l'API, reportez-vous [UpdateJobStatus](#) à la section Référence des AWS SDK for Java 2.x API.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

def cancel_job(self, job_id: str, account_id: str) -> None:
    """
    Cancel an S3 batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended', 'Active']:

```

```

        self.s3control_client.update_job_status(
            AccountId=account_id,
            JobId=job_id,
            RequestedJobStatus='Cancelled'
        )
        print(f"Job {job_id} was successfully canceled.")
    elif current_status in ['Completing', 'Complete']:
        print(f"Job is in '{current_status}' state - cannot be
cancelled")
    if current_status == 'Completing':
        print("Job is finishing up and will complete soon.")
    elif current_status == 'Complete':
        print("Job has already completed successfully.")
    else:
        print(f"Job is in '{current_status}' state - cancel not allowed")
except ClientError as e:
    print(f"Error canceling job: {e}")
    raise

```

- Pour plus de détails sur l'API, consultez [UpdateJobStatus](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

SAP ABAP

Kit SDK pour SAP ABAP

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```

TRY.
    " iv_requested_status = 'Cancelled'
    oo_result = lo_s3c->updatejobstatus(      " oo_result is returned for
testing purposes.
    iv_accountid      = iv_account_id
    iv_jobid          = iv_job_id
    iv_requestedjobstatus = iv_requested_status
    ).

```

```

    MESSAGE |Job { oo_result->get_jobid( ) } status updated to { oo_result-
>get_status( ) }| TYPE 'I'.
    CATCH /aws1/cx_s3cjobstatusexception INTO DATA(lo_ex_js).
    MESSAGE lo_ex_js->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_js.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
ENDTRY.

```

- Pour plus de détails sur l'API, consultez [UpdateJobStatus](#) la section de référence du AWS SDK pour l'API SAP ABAP.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Exemples de code pour les compartiments de répertoire S3 utilisant AWS Kits SDK

Les exemples de code suivants montrent comment utiliser les compartiments de répertoire S3 avec un kit de développement AWS logiciel (SDK).

Les principes de base sont des exemples de code qui vous montrent comment effectuer les opérations essentielles au sein d'un service.

Les actions sont des extraits de code de programmes plus larges et doivent être exécutées dans leur contexte. Alors que les actions vous indiquent comment appeler des fonctions de service individuelles, vous pouvez les voir en contexte dans leurs scénarios associés.

Les scénarios sont des exemples de code qui vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions au sein d'un même service ou combinés à d'autres Services AWS.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Compartiments de répertoires S3

- [Exemples de base pour les compartiments de répertoire S3 utilisant AWS Kits SDK](#)
 - [Bonjour compartiments de répertoires Amazon S3](#)
 - [Apprenez les bases des compartiments d'annuaire S3 avec un AWS Kit SDK](#)
 - [Actions pour les compartiments de répertoire S3 à l'aide de AWS Kits SDK](#)
 - [À utiliser AbortMultipartUploadavec un AWS Kit SDK](#)
 - [À utiliser CompleteMultipartUploadavec un AWS Kit SDK](#)
 - [À utiliser CopyObjectavec un AWS Kit SDK](#)
 - [À utiliser CreateBucketavec un AWS Kit SDK](#)
 - [À utiliser CreateMultipartUploadavec un AWS Kit SDK](#)
 - [À utiliser CreateSessionavec un AWS Kit SDK](#)
 - [À utiliser DeleteBucketavec un AWS Kit SDK](#)
 - [À utiliser DeleteBucketEncryptionavec un AWS Kit SDK](#)
 - [À utiliser DeleteBucketPolicyavec un AWS Kit SDK](#)
 - [À utiliser DeleteObjectavec un AWS Kit SDK](#)
 - [À utiliser DeleteObjectsavec un AWS Kit SDK](#)
 - [À utiliser GetBucketEncryptionavec un AWS Kit SDK](#)
 - [À utiliser GetBucketPolicyavec un AWS Kit SDK](#)
 - [À utiliser GetObjectavec un AWS Kit SDK](#)
 - [À utiliser GetObjectAttributesavec un AWS Kit SDK](#)
 - [À utiliser HeadBucketavec un AWS Kit SDK](#)
 - [À utiliser HeadObjectavec un AWS Kit SDK](#)
 - [À utiliser ListDirectoryBucketsavec un AWS Kit SDK](#)
 - [À utiliser ListMultipartUploadsavec un AWS Kit SDK](#)

- [Utilisez la ListObjectsV2 avec un AWS Kit SDK](#)
- [À utiliser ListParts avec un AWS Kit SDK](#)
- [À utiliser PutBucketEncryption avec un AWS Kit SDK](#)
- [À utiliser PutBucketPolicy avec un AWS Kit SDK](#)
- [À utiliser PutObject avec un AWS Kit SDK](#)
- [À utiliser UploadPart avec un AWS Kit SDK](#)
- [À utiliser UploadPartCopy avec un AWS Kit SDK](#)
- [Scénarios pour les compartiments d'annuaire S3 utilisant AWS Kits SDK](#)
 - [Créez une URL présignée pour les compartiments d'annuaire Amazon S3 afin d'obtenir un objet à l'aide d'un AWS Kit SDK](#)

Exemples de base pour les compartiments de répertoire S3 utilisant AWS Kits SDK

Les exemples de code suivants montrent les bases de l'utilisation de compartiments de répertoires Amazon S3 avec des kits AWS SDK.

Exemples

- [Bonjour compartiments de répertoires Amazon S3](#)
- [Apprenez les bases des compartiments d'annuaire S3 avec un AWS Kit SDK](#)
- [Actions pour les compartiments de répertoire S3 à l'aide de AWS Kits SDK](#)
 - [À utiliser AbortMultipartUpload avec un AWS Kit SDK](#)
 - [À utiliser CompleteMultipartUpload avec un AWS Kit SDK](#)
 - [À utiliser CopyObject avec un AWS Kit SDK](#)
 - [À utiliser CreateBucket avec un AWS Kit SDK](#)
 - [À utiliser CreateMultipartUpload avec un AWS Kit SDK](#)
 - [À utiliser CreateSession avec un AWS Kit SDK](#)
 - [À utiliser DeleteBucket avec un AWS Kit SDK](#)
 - [À utiliser DeleteBucketEncryption avec un AWS Kit SDK](#)
 - [À utiliser DeleteBucketPolicy avec un AWS Kit SDK](#)
 - [À utiliser DeleteObject avec un AWS Kit SDK](#)
 - [À utiliser DeleteObjects avec un AWS Kit SDK](#)

- [À utiliser GetBucketEncryptionavec un AWS Kit SDK](#)
- [À utiliser GetBucketPolicyavec un AWS Kit SDK](#)
- [À utiliser GetObjectavec un AWS Kit SDK](#)
- [À utiliser GetObjectAttributesavec un AWS Kit SDK](#)
- [À utiliser HeadBucketavec un AWS Kit SDK](#)
- [À utiliser HeadObjectavec un AWS Kit SDK](#)
- [À utiliser ListDirectoryBucketsavec un AWS Kit SDK](#)
- [À utiliser ListMultipartUploadsavec un AWS Kit SDK](#)
- [Utilisez la ListObjectsV2 avec un AWS Kit SDK](#)
- [À utiliser ListPartsWith un AWS Kit SDK](#)
- [À utiliser PutBucketEncryptionavec un AWS Kit SDK](#)
- [À utiliser PutBucketPolicyavec un AWS Kit SDK](#)
- [À utiliser PutObjectavec un AWS Kit SDK](#)
- [À utiliser UploadPartavec un AWS Kit SDK](#)
- [À utiliser UploadPartCopyavec un AWS Kit SDK](#)

Bonjour compartiments de répertoires Amazon S3

L'exemple de code suivant montre comment démarrer avec des compartiments de répertoires Amazon S3.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
package com.example.s3.directorybucket;
```



```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Before running this example:
 * <p>
 * The SDK must be able to authenticate AWS requests on your behalf. If you have
 * not configured
 * authentication for SDKs and tools, see
 * https://docs.aws.amazon.com/sdkref/latest/guide/access.html in the AWS SDKs
 * and Tools Reference Guide.
 * <p>
 * You must have a runtime environment configured with the Java SDK.
 * See
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/setup.html in
 * the Developer Guide if this is not set up.
 * <p>
 * To use S3 directory buckets, configure a gateway VPC endpoint. This is the
 * recommended method to enable directory bucket traffic without
 * requiring an internet gateway or NAT device. For more information on
 * configuring VPC gateway endpoints, visit
 * https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-networking.html#s3-express-networking-vpc-gateway.
 * <p>
```

```
* Directory buckets are available in specific AWS Regions and Zones. For
* details on Regions and Zones supporting directory buckets, see
* https://docs.aws.amazon.com/AmazonS3/latest/userguide/s3-express-networking.html#s3-express-endpoints.
*/

public class HelloS3DirectoryBuckets {
    private static final Logger logger =
        LoggerFactory.getLogger(HelloS3DirectoryBuckets.class);

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.
        Region region = Region.US_WEST_2;
        String zone = "usw2-az1";
        S3Client s3Client = createS3Client(region);

        try {
            // Create the directory bucket
            createDirectoryBucket(s3Client, bucketName, zone);
            logger.info("Created bucket: {}", bucketName);

            // List all directory buckets
            List<String> bucketNames = listDirectoryBuckets(s3Client);
            bucketNames.forEach(name -> logger.info("Bucket Name: {}", name));
        } catch (S3Exception e) {
            logger.error("An error occurred during S3 operations: {} - Error
code: {}",
                e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        } finally {
            try {
                // Delete the created bucket
                deleteDirectoryBucket(s3Client, bucketName);
                logger.info("Deleted bucket: {}", bucketName);
            } catch (S3Exception e) {
                logger.error("Failed to delete the bucket due to S3 error: {} -
Error code: {}",
                    e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
            } catch (RuntimeException e) {
                logger.error("Failed to delete the bucket due to unexpected
error: {}", e.getMessage(), e);
            } finally {
```

```

        s3Client.close();
    }
}

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client    The S3 client used to create the bucket
 * @param bucketName  The name of the bucket to be created
 * @param zone        The region where the bucket will be created
 * @throws S3Exception if there's an error creating the bucket
 */
public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
    logger.info("Creating bucket: {}", bucketName);

    CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
        .location(LocationInfo.builder()
            .type(LocationType.AVAILABILITY_ZONE)
            .name(zone).build())
        .bucket(BucketInfo.builder()
            .type(BucketType.DIRECTORY)
            .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
            .build())
        .build();

    try {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration).build();
        CreateBucketResponse response = s3Client.createBucket(bucketRequest);
        logger.info("Bucket created successfully with location: {}",
response.location());
    } catch (S3Exception e) {
        logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

/**

```

```
* Lists all S3 directory buckets.
*
* @param s3Client The S3 client used to interact with S3
* @return A list of bucket names
*/
public static List<String> listDirectoryBuckets(S3Client s3Client) {
    logger.info("Listing all directory buckets");

    try {
        // Create a ListBucketsRequest
        ListDirectoryBucketsRequest listBucketsRequest =
ListDirectoryBucketsRequest.builder().build();

        // Retrieve the list of buckets
        ListDirectoryBucketsResponse response =
s3Client.listDirectoryBuckets(listBucketsRequest);

        // Extract bucket names
        List<String> bucketNames = response.buckets().stream()
            .map(Bucket::name)
            .collect(Collectors.toList());

        return bucketNames;
    } catch (S3Exception e) {
        logger.error("Failed to list buckets: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}

/**
 * Deletes the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the bucket to delete
 */
public static void deleteDirectoryBucket(S3Client s3Client, String
bucketName) {
    try {
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();
```

```
s3Client.deleteBucket(deleteBucketRequest);
} catch (S3Exception e) {
    logger.error("Failed to delete bucket: " + bucketName + " - Error
code: " + e.awsErrorDetails().errorCode(),
                e);
    throw e;
}
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .
 - [CreateBucket](#)
 - [ListDirectoryBuckets](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Apprenez les bases des compartiments d'annuaire S3 avec un AWS Kit SDK

Les exemples de code suivants montrent comment :

- configurer un VPC et un point de terminaison d'un VPC ;
- Configurez les politiques, les rôles et l'utilisateur pour qu'ils fonctionnent avec les compartiments de répertoires S3 et la classe de stockage S3 Express One Zone.
- Créez deux clients S3.
- Créez deux compartiments.
- créer un objet et le copier ;
- démontree la différence de performance ;
- remplir les compartiments pour montrer la différence lexicographique ;
- Invitez l'utilisateur à voir s'il souhaite nettoyer les ressources.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario interactif illustrant les fonctionnalités d'Amazon S3.

```
public class S3DirectoriesScenario {

    public static final String DASHES = new String(new char[80]).replace("\0",
    "-");

    private static final Logger logger =
    LoggerFactory.getLogger(S3DirectoriesScenario.class);
    static Scanner scanner = new Scanner(System.in);

    private static S3AsyncClient mS3RegularClient;
    private static S3AsyncClient mS3ExpressClient;

    private static String mdirectoryBucketName;
    private static String mregularBucketName;

    private static String stackName = "cfn-stack-s3-express-basics--" +
    UUID.randomUUID();

    private static String regularUser = "";
    private static String vpcId = "";
    private static String expressUser = "";

    private static String vpcEndpointId = "";

    private static final S3DirectoriesActions s3DirectoriesActions = new
    S3DirectoriesActions();

    public static void main(String[] args) {
        try {
            s3ExpressScenario();
        } catch (RuntimeException e) {
```

```
        logger.info(e.getMessage());
    }
}

// Runs the scenario.
private static void s3ExpressScenario() {
    logger.info(DASHES);
    logger.info("Welcome to the Amazon S3 Express Basics demo using AWS SDK
for Java V2.");
    logger.info("""
        Let's get started! First, please note that S3 Express One Zone works
        best when working within the AWS infrastructure,
        specifically when working in the same Availability Zone (AZ). To see
        the best results in this example and when you implement
        directory buckets into your infrastructure, it is best to put your
        compute resources in the same AZ as your directory
        bucket.
        """);
    waitForInputToContinue(scanner);
    logger.info(DASHES);

    // Create an optional VPC and create 2 IAM users.
    UserNames userNames = createVpcUsers();
    String expressUserName = userNames.getExpressUserName();
    String regularUserName = userNames.getRegularUserName();

    // Set up two S3 clients, one regular and one express,
    // and two buckets, one regular and one directory.
    setupClientsAndBuckets(expressUserName, regularUserName);

    // Create an S3 session for the express S3 client and add objects to the
    buckets.
    logger.info("Now let's add some objects to our buckets and demonstrate
how to work with S3 Sessions.");
    waitForInputToContinue(scanner);
    String bucketObject = createSessionAddObjects();

    // Demonstrate performance differences between regular and directory
    buckets.
    demonstratePerformance(bucketObject);

    // Populate the buckets to show the lexicographical difference between
    // regular and express buckets.
    showLexicographicalDifferences(bucketObject);
}
```

```

        logger.info(DASHES);
        logger.info("That's it for our tour of the basic operations for S3
Express One Zone.");
        logger.info("Would you like to cleanUp the AWS resources? (y/n): ");
        String response = scanner.next().trim().toLowerCase();
        if (response.equals("y")) {
            cleanUp(stackName);
        }
    }

    /*
     Delete resources created by this scenario.
    */
    public static void cleanUp(String stackName) {
        try {
            if (mdirectoryBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
                }
                logger.info("Deleted directory bucket " + mdirectoryBucketName);
                mdirectoryBucketName = null;
                if (mregularBucketName != null) {

s3DirectoriesActions.deleteBucketAndObjectsAsync(mS3RegularClient,
mregularBucketName).join();
                }
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof S3Exception) {
                    logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }

            logger.info("Deleted regular bucket " + mregularBucketName);
            mregularBucketName = null;
            CloudFormationHelper.destroyCloudFormationStack(stackName);
        }

        private static void showLexicographicalDifferences(String bucketObject) {

```



```
logger.info(DASHES);
logger.info("
    7. Populate the buckets to show the lexicographical (alphabetical)
difference
    when object names are listed. Now let's explore how directory buckets
store
    objects in a different manner to regular buckets. The key is in the
name
    "Directory". Where regular buckets store their key/value pairs in a
    flat manner, directory buckets use actual directories/folders.
    This allows for more rapid indexing, traversing, and therefore
    retrieval times!

    The more segmented your bucket is, with lots of
    directories, sub-directories, and objects, the more efficient it
becomes.
    This structural difference also causes `ListObject` operations to
behave
    differently, which can cause unexpected results. Let's add a few
more
    objects in sub-directories to see how the output of
    ListObjects changes.
    """);

waitForInputToContinue(scanner);

// Populate a few more files in each bucket so that we can use
// ListObjects and show the difference.
String otherObject = "other/" + bucketObject;
String altObject = "alt/" + bucketObject;
String otherAltObject = "other/alt/" + bucketObject;

try {
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();
}
```

```

        s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchBucketException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    try {
        // List objects in both S3 buckets.
        List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
        List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient,
mregularBucketName).join();

        logger.info("Directory bucket content");
        for (String obj : dirBucketObjects) {
            logger.info(obj);
        }

        logger.info("Regular bucket content");
        for (String obj : regBucketObjects) {
            logger.info(obj);
        }
    } catch (CompletionException e) {
        logger.error("Async operation failed: {} ",
e.getCause().getMessage());
        return;
    }

    logger.info("""
        Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
        because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually

```

```

        creates directories and uses the object "key" as a path to the
object.
        """);
        waitForInputToContinue(scanner);
    }

    /**
     * Demonstrates the performance difference between downloading an object from
a directory bucket and a regular bucket.
     *
     * <p>This method:
     * <ul>
     *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
     *     <li>Downloads the specified object from the directory bucket and
measures the total time.</li>
     *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
     *     <li>Compares the time differences and prints the results.</li>
     * </ul>
     *
     * <p>Note: The performance difference will be more pronounced if this
example is run on an EC2 instance
     * in the same Availability Zone as the buckets.
     *
     * @param bucketObject the name of the object to download
     */
    private static void demonstratePerformance(String bucketObject) {
        logger.info(DASHES);
        logger.info("6. Demonstrate the performance difference.");
        logger.info("
        Now, let's do a performance test. We'll download the same object from
each
        bucket repeatedly and compare the total time needed.

        Note: the performance difference will be much more pronounced if this
example is run in an EC2 instance in the same Availability Zone as
the bucket.
        """);
        waitForInputToContinue(scanner);

        int downloads = 1000; // Default value.
        logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");
    }

```

```
// Ask if the user wants to download a different number.
logger.info("Would you like to download the file a different number of
times? (y/n): ");
String response = scanner.next().trim().toLowerCase();
if (response.equals("y")) {
    int maxDownloads = 1_000_000;

    // Ask for a valid number of downloads.
    while (true) {
        logger.info("Enter a number between 1 and " + maxDownloads + "
for the number of downloads: ");
        if (scanner.hasNextInt()) {
            downloads = scanner.nextInt();
            if (downloads >= 1 && downloads <= maxDownloads) {
                break;
            } else {
                logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
            }
        } else {
            logger.info("Invalid input. Please enter a valid integer.");
            scanner.next();
        }
    }

    logger.info("You have chosen to download {} items.", downloads);
} else {
    logger.info("No changes made. Using default downloads: {}",
downloads);
}
// Simulating the download process for the directory bucket.
logger.info("Downloading from the directory bucket.");
long directoryTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        // Get the object from the directory bucket.
        s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
    } catch (CompletionException ce) {
```

```
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
}
long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

// Download from the regular bucket.
logger.info("Downloading from the regular bucket.");
long normalTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
}

long normalTimeDifference = System.nanoTime() - normalTimeStart;
logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
long difference = normalTimeDifference - directoryTimeDifference;
logger.info("That's a difference of " + difference + " nanoseconds, or");
logger.info(difference / 1_000_000_000.0 + " seconds.");
```

```

        if (difference < 0) {
            logger.info("The directory buckets were slower. This can happen if
you are not running on the cloud within a VPC.");
        }
        waitForInputToContinue(scanner);
    }

    private static String createSessionAdd0Objects() {
        logger.info(DASHES);
        logger.info("""
            5. Create an object and copy it.
            We'll create an object consisting of some text and upload it to the
            regular bucket.
            """);
        waitForInputToContinue(scanner);

        String bucketObject = "basic-text-object.txt";
        try {
            s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
            s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

            // Copy the object to the destination S3 bucket.
            s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof S3Exception) {
                logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
        logger.info("""
            It worked! This is because the S3Client that performed the copy
operation
            is the expressClient using the credentials for the user with
permission to
            work with directory buckets.
        """);
    }

```

```

        It's important to remember the user permissions when interacting
with
        directory buckets. Instead of validating permissions on every call
as
        regular buckets do, directory buckets utilize the user credentials
and session
        token to validate. This allows for much faster connection speeds on
every call.
        For single calls, this is low, but for many concurrent calls
        this adds up to a lot of time saved.
        """);
        waitForInputToContinue(scanner);
        return bucketObject;
    }

    /**
     * Creates VPC users for the S3 Express One Zone scenario.
     * <p>
     * This method performs the following steps:
     * <ol>
     *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
     *     <li>Creates two IAM users: one with S3 Express One Zone permissions
and one without.</li>
     * </ol>
     *
     * @return a {@link UserNames} object containing the names of the created IAM
users
     */
    public static UserNames createVpcUsers() {
        /*
         * Optionally create a VPC.
         * Create two IAM users, one with S3 Express One Zone permissions and one
without.
         */
        logger.info(DASHES);
        logger.info("""
            1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
            directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
            """);
    }

```

```

        logger.info("Do you want to setup a VPC Endpoint? (y/n)");
        String endpointAns = scanner.nextLine().trim();
        if (endpointAns.equalsIgnoreCase("y")) {
            logger.info("
                Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
                ");
            try {
                s3DirectoriesActions.setupVPCAsync().join();
            } catch (CompletionException ce) {
                Throwable cause = ce.getCause();
                if (cause instanceof Ec2Exception) {
                    logger.error("IamException occurred: {}", cause.getMessage(),
ce);
                } else {
                    logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                }
            }
            waitForInputToContinue(scanner);
        } else {
            logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
        }
        logger.info(DASHES);
        logger.info("
            2. Create a RegularUser and ExpressUser by using the AWS CDK.
            One IAM User, named RegularUser, will have permissions to work only
            with regular buckets and one IAM user, named ExpressUser, will have
            permissions to work only with directory buckets.
            ");
        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");

        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

```



```
/**
 * Creates users using AWS CloudFormation.
 *
 * @return a {@link Map} of String keys and String values representing the
stack outputs,
 * which may include user-related information such as user names and IDs.
 */
public static Map<String, String> createUsersUsingCDK(String stackName) {
    logger.info("We'll use an AWS CloudFormation template to create the IAM
users and policies.");
    CloudFormationHelper.deployCloudFormationStack(stackName);
    return CloudFormationHelper.getStackOutputsAsync(stackName).join();
}

/**
 * Sets up the necessary clients and buckets for the S3 Express service.
 *
 * @param expressUserName the username for the user with S3 Express
permissions
 * @param regularUserName the username for the user with regular S3
permissions
 */
public static void setupClientsAndBuckets(String expressUserName, String
regularUserName) {
    Scanner locscanner = new Scanner(System.in);
    String accessKeyIdforRegUser;
    String secretAccessforRegUser;
    try {
        CreateAccessKeyResponse keyResponse =
s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
        accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();
        secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof IamException) {
            logger.error("IamException occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
    }
    return;
}
```

```

        String accessKeyIdforExpressUser;
        String secretAccessforExpressUser;
        try {
            CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
            accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
            secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IamException) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
            return;
        }

        logger.info(DASHES);
        logger.info("""
            3. Create two S3Clients; one uses the ExpressUser's credentials and
one uses the RegularUser's credentials.
            The 2 S3Clients will use different credentials.
            """);
        waitForInputToContinue(locscanner);
        try {
            mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
            mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof IllegalArgumentException) {
                logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
    }

```

```

        return;
    }

    logger.info("""
        We can now use the ExpressUser client to make calls to S3 Express
        operations.
        """);
    waitForInputToContinue(locscanner);
    logger.info(DASHES);
    logger.info("""
        4. Create two buckets.
        Now we will create a directory bucket which is the linchpin of the S3
        Express One Zone service. Directory buckets
        behave differently from regular S3 buckets which we will explore
        here. We'll also create a regular bucket, put
        an object into the regular bucket, and copy it to the directory
        bucket.
        """);

    logger.info("""
        Now, let's choose an availability zone (AZ) for the directory
        bucket.
        We'll choose one that is supported.
        """);
    String zoneId;
    String regularBucketName;
    try {
        zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();
        regularBucketName = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof Ec2Exception) {
            logger.error("EC2Exception occurred: {}", cause.getMessage(),
            ce);
        } else {
            logger.error("An unexpected error occurred: {}",
            cause.getMessage(), ce);
        }
        return;
    }
    logger.info("""
        Now, let's create the actual directory bucket, as well as a regular
        bucket."

```

```

        "");

        String directoryBucketName = "amzn-s3-demo-bucket--" + zoneId + "--x-
s3"; // Replace with your directory bucket name.
        try {
            s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
            logger.info("Created directory bucket {}", directoryBucketName);
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                return;
            }
        }

        // Assign to the data member.
        mdirectoryBucketName = directoryBucketName;
        try {
            s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
            logger.info("Created regular bucket {} ", regularBucketName);
            mregularBucketName = regularBucketName;
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
                return;
            }
        }
        logger.info("Great! Both buckets were created.");
        waitForInputToContinue(locscanner);
    }

    /**

```

```

    * Creates an asynchronous S3 client with the specified access key and secret
    access key.
    *
    * @param accessKeyId      the AWS access key ID
    * @param secretAccessKey the AWS secret access key
    * @return a {@link CompletableFuture} that asynchronously creates the S3
    client
    * @throws IllegalArgumentException if the access key ID or secret access key
    is null
    */
    public static CompletableFuture<S3AsyncClient>
    createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
        return CompletableFuture.supplyAsync(() -> {
            // Validate input parameters
            if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey
    == null || secretAccessKey.isBlank()) {
                throw new IllegalArgumentException("Access Key ID and Secret
    Access Key must not be null or empty");
            }

            AwsBasicCredentials awsCredentials =
    AwsBasicCredentials.create(accessKeyId, secretAccessKey);
            return S3AsyncClient.builder()

    .credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
                .region(Region.US_WEST_2)
                .build();
        });
    }

    private static void waitForInputToContinue(Scanner scanner) {
        while (true) {
            logger.info("");
            logger.info("Enter 'c' followed by <ENTER> to continue:");
            String input = scanner.nextLine();

            if (input.trim().equalsIgnoreCase("c")) {
                logger.info("Continuing with the program...");
                logger.info("");
                break;
            } else {
                logger.info("Invalid input. Please try again.");
            }
        }
    }
}

```

```
}  
}
```

Une classe d'encapsuleur pour les méthodes du kit SDK Amazon S3.

```
public class S3DirectoriesActions {  
  
    private static IamAsyncClient iamAsyncClient;  
  
    private static Ec2AsyncClient ec2AsyncClient;  
    private static final Logger logger =  
        LoggerFactory.getLogger(S3DirectoriesActions.class);  
  
    private static IamAsyncClient getIAMAsyncClient() {  
        if (iamAsyncClient == null) {  
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()  
                .maxConcurrency(100)  
                .connectionTimeout(Duration.ofSeconds(60))  
                .readTimeout(Duration.ofSeconds(60))  
                .writeTimeout(Duration.ofSeconds(60))  
                .build();  
  
            ClientOverrideConfiguration overrideConfig =  
                ClientOverrideConfiguration.builder()  
                    .apiCallTimeout(Duration.ofMinutes(2))  
                    .apiCallAttemptTimeout(Duration.ofSeconds(90))  
                    .retryStrategy(RetryMode.STANDARD)  
                    .build();  
  
            iamAsyncClient = IamAsyncClient.builder()  
                .httpClient(httpClient)  
                .overrideConfiguration(overrideConfig)  
                .build();  
        }  
        return iamAsyncClient;  
    }  
  
    private static Ec2AsyncClient getEc2AsyncClient() {  
        if (ec2AsyncClient == null) {  
            SdkAsyncHttpClient httpClient = NettyNioAsyncHttpClient.builder()  
                .maxConcurrency(100)  
                .connectionTimeout(Duration.ofSeconds(60))
```

```

        .readTimeout(Duration.ofSeconds(60))
        .writeTimeout(Duration.ofSeconds(60))
        .build();

    ClientOverrideConfiguration overrideConfig =
ClientOverrideConfiguration.builder()
        .apiCallTimeout(Duration.ofMinutes(2))
        .apiCallAttemptTimeout(Duration.ofSeconds(90))
        .retryStrategy(RetryMode.STANDARD)
        .build();

    ec2AsyncClient = Ec2AsyncClient.builder()
        .httpClient(httpClient)
        .region(Region.US_WEST_2)
        .overrideConfiguration(overrideConfig)
        .build();
}
return ec2AsyncClient;
}

/**
 * Deletes the specified S3 bucket and all the objects within it
asynchronously.
 *
 * @param s3AsyncClient the S3 asynchronous client to use for the operations
 * @param bucketName the name of the S3 bucket to be deleted
 * @return a {@link CompletableFuture} that completes with a {@link
WaiterResponse} containing the
 *         {@link HeadBucketResponse} when the bucket has been successfully
deleted
 * @throws CompletionException if there was an error deleting the bucket or
its objects
 */
public CompletableFuture<WaiterResponse<HeadBucketResponse>>
deleteBucketAndObjectsAsync(S3AsyncClient s3AsyncClient, String bucketName) {
    ListObjectsV2Request listRequest = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3AsyncClient.listObjectsV2(listRequest)
        .thenCompose(listResponse -> {
            if (!listResponse.contents().isEmpty()) {
                List<ObjectIdentifier> objectIdentifiers =
listResponse.contents().stream()

```

```

        .map(s3object ->
ObjectIdentifier.builder().key(s3object.key()).build())
        .collect(Collectors.toList());

        DeleteObjectsRequest deleteRequest =
DeleteObjectsRequest.builder()
        .bucket(bucketName)

.delete(Delete.builder().objects(objectIdentifiers).build())
        .build();

        return s3AsyncClient.deleteObjects(deleteRequest)
            .thenAccept(deleteResponse -> {
                if (!deleteResponse.errors().isEmpty()) {
                    deleteResponse.errors().forEach(error ->
                        logger.error("Couldn't delete object " +
error.key() + ". Reason: " + error.message()));
                }
            });
    }
    return CompletableFuture.completedFuture(null);
})
.thenCompose(ignored -> {
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
        .bucket(bucketName)
        .build();
        return s3AsyncClient.deleteBucket(deleteBucketRequest);
    })
    .thenCompose(ignored -> {
        S3AsyncWaiter waiter = s3AsyncClient.waiter();
        HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
        return waiter.waitUntilBucketNotExists(headBucketRequest);
    })
    .whenComplete((ignored, exception) -> {
        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof S3Exception) {
                throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
            }
            throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
        }
    });
}

```



```

        }
        logger.info("Bucket deleted successfully: " + bucketName);
    });
}

/**
 * Lists the objects in an S3 bucket asynchronously.
 *
 * @param s3Client the S3 async client to use for the operation
 * @param bucketName the name of the S3 bucket containing the objects to list
 * @return a {@link CompletableFuture} that contains the list of object keys
 * in the specified bucket
 */
public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient
s3Client, String bucketName) {
    ListObjectsV2Request request = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3Client.listObjectsV2(request)
        .thenApply(response -> response.contents().stream()
            .map(S3Object::key)
            .toList())
        .whenComplete((result, exception) -> {
            if (exception != null) {
                throw new CompletionException("Couldn't list objects in
bucket: " + bucketName, exception);
            }
        });
}

/**
 * Retrieves an object from an Amazon S3 bucket asynchronously.
 *
 * @param s3Client the S3 async client to use for the operation
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName the unique identifier (key) of the object to retrieve
 * @return a {@link CompletableFuture} that, when completed, contains the
 * object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
 */
public CompletableFuture<ResponseBytes<GetObjectResponse>>
getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .key(keyName)

```

```

        .bucket(bucketName)
        .build();

    // Get the object asynchronously and transform it into a byte array
    return s3Client.getObject(objectRequest,
        AsyncResponseTransformer.toBytes())
        .exceptionally(exception -> {
            Throwable cause = exception.getCause();
            if (cause instanceof NoSuchKeyException) {
                throw new CompletionException("Failed to get the object.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
            }
            throw new CompletionException("Failed to get the object",
exception);
        });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param s3Client          the S3 async client to use for the copy
operation
     * @param sourceBucket      the name of the source bucket
     * @param sourceKey         the key of the object to be copied in the source
bucket
     * @param destinationBucket the name of the destination bucket
     * @param destinationKey    the key of the copied object in the destination
bucket
     * @return a {@link CompletableFuture} that completes when the copy operation
is finished
     */
    public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey)
    {
        CopyObjectRequest copyRequest = CopyObjectRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(sourceKey)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .build();

        return s3Client.copyObject(copyRequest)
            .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
    }

```

```

        .whenComplete((ignored, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof S3Exception) {
                    throw new CompletionException("Couldn't
copy object '" + sourceKey + "' from bucket '" + sourceBucket + "'
to bucket '" + destinationBucket + "'. Reason: " + ((S3Exception)
cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to copy object",
exception);
            }
        });
    }

    /**
     * Asynchronously creates a session for the specified S3 bucket.
     *
     * @param s3Client the S3 asynchronous client to use for creating the
session
     * @param bucketName the name of the S3 bucket for which to create the
session
     * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
     */
    public CompletableFuture<CreateSessionResponse>
createSessionAsync(S3AsyncClient s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createSession(request)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof S3Exception) {
                        throw new CompletionException("Couldn't create the
session. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(),
cause);
                    }
                    throw new CompletionException("Unexpected error occurred
while creating session", exception);
                }
                logger.info("Created session for bucket: " + bucketName);
            });
    }

```

```

    });

}

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client    The asynchronous S3 client used to create the bucket
 * @param bucketName The name of the bucket to be created
 * @param zone        The Availability Zone where the bucket will be created
 * @throws CompletionException if there's an error creating the bucket
 */
public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String
zone) {
    logger.info("Creating bucket: " + bucketName);

    CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
        .location(LocationInfo.builder()
            .type(LocationType.AVAILABILITY_ZONE)
            .name(zone)
            .build())
        .bucket(BucketInfo.builder()
            .type(BucketType.DIRECTORY)
            .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
            .build())
        .build();

    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .createBucketConfiguration(bucketConfiguration)
        .build();

    return s3Client.createBucket(bucketRequest)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof BucketAlreadyExistsException) {
                    throw new CompletionException("The bucket already exists:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
            }
        })
}

```

```

        throw new CompletionException("Unexpected error occurred
while creating bucket", exception);
    }
    logger.info("Bucket created successfully with location: " +
response.location());
    });
}

/**
 * Creates an S3 bucket asynchronously.
 *
 * @param s3Client    the S3 async client to use for the bucket creation
 * @param bucketName  the name of the S3 bucket to create
 * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
 *         when the bucket is successfully created
 * @throws CompletionException if there's an error creating the bucket
 */
public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .build();

    return s3Client.createBucket(bucketRequest)
        .thenCompose(response -> {
            S3AsyncWaiter s3Waiter = s3Client.waiter();
            HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                .bucket(bucketName)
                .build();
            return s3Waiter.waitUntilBucketExists(bucketRequestWait);
        })
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof BucketAlreadyExistsException) {
                    throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                }
            }
            logger.info(bucketName + " is ready");
        });
}

```

```

        });
    }

    /**
     * Uploads an object to an Amazon S3 bucket asynchronously.
     *
     * @param s3Client    the S3 async client to use for the upload
     * @param bucketName  the destination S3 bucket name
     * @param bucketObject the name of the object to be uploaded
     * @param text        the content to be uploaded as the object
     */
    public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
        PutObjectRequest objectRequest = PutObjectRequest.builder()
            .bucket(bucketName)
            .key(bucketObject)
            .build();

        return s3Client.putObject(objectRequest,
AsyncRequestBody.fromString(text))
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof NoSuchBucketException) {
                        throw new CompletionException("The S3 bucket does not
exist: " + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
                    }
                }
            });
    }

    /**
     * Creates an AWS IAM access key asynchronously for the specified user name.
     *
     * @param userName the name of the IAM user for whom to create the access key
     * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
     */
    public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
        CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()

```

```

        .userName(userName)
        .build();

return getIAMAsyncClient().createAccessKey(request)
    .whenComplete((response, exception) -> {
        if (response != null) {
            logger.info("Access Key Created.");
        } else {
            if (exception == null) {
                Throwable cause = exception.getCause();
                if (cause instanceof IamException) {
                    throw new CompletionException("IAM error while
creating access key: " + cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Failed to create
access key: " + exception.getMessage(), exception);
                }
            }
        }
    });
}

/**
 * Asynchronously selects an Availability Zone ID from the available EC2
zones.
 *
 * @return A {@link CompletableFuture} that resolves to the selected
Availability Zone ID.
 * @throws CompletionException if an error occurs during the request or
processing.
 */
public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
    DescribeAvailabilityZonesRequest zonesRequest =
DescribeAvailabilityZonesRequest.builder()
        .build();

    return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
        .thenCompose(response -> {
            List<AvailabilityZone> zonesList = response.availabilityZones();
            if (zonesList.isEmpty()) {
                logger.info("No availability zones found.");
                return CompletableFuture.completedFuture(null); // Return
null if no zones are found
            }
        })
}

```

```

        List<String> zoneIds = zonesList.stream()
            .map(AvailabilityZone::zoneId) // Get the zoneId (e.g.,
"usw2-az1")
            .toList();

        return CompletableFuture.supplyAsync(() ->
promptUserForZoneSelection(zonesList, zoneIds))
            .thenApply(selectedZone -> {
                // Return only the selected Zone ID (e.g., "usw2-az1").
                return selectedZone.zoneId();
            });
    })
    .whenComplete((result, exception) -> {
        if (exception == null) {
            if (result != null) {
                logger.info("Selected Availability Zone ID: " + result);
            } else {
                logger.info("No availability zone selected.");
            }
        } else {
            Throwable cause = exception.getCause();
            if (cause instanceof Ec2Exception) {
                throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
            }
            throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
        }
    });
}

/**
 * Prompts the user to select an Availability Zone from the given list.
 *
 * @param zonesList the list of Availability Zones
 * @param zoneIds the list of zone IDs
 * @return the selected Availability Zone
 */
private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String>
zoneIds) {
    Scanner scanner = new Scanner(System.in);
    int index = -1;

```



```

while (index < 0 || index >= zoneIds.size()) {
    logger.info("Select an availability zone:");
    IntStream.range(0, zoneIds.size()).forEach(i ->
        logger.info(i + ": " + zoneIds.get(i))
    );

    logger.info("Enter the number corresponding to your choice: ");
    if (scanner.hasNextInt()) {
        index = scanner.nextInt();
    } else {
        scanner.next();
    }
}

AvailabilityZone selectedZone = zonesList.get(index);
logger.info("You selected: " + selectedZone.zoneId());
return selectedZone;
}

/**
 * Asynchronously sets up a new VPC, including creating the VPC, finding the
 * associated route table, and
 * creating a VPC endpoint for the S3 service.
 *
 * @return a {@link CompletableFuture} that, when completed, contains a
 * AbstractMap with the
 *      VPC ID and VPC endpoint ID.
 */
public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
setupVPCAsync() {
    String cidr = "10.0.0.0/16";
    CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
        .cidrBlock(cidr)
        .build();

    return getEc2AsyncClient().createVpc(vpcRequest)
        .thenCompose(vpcResponse -> {
            String vpcId = vpcResponse.vpc().vpcId();
            logger.info("VPC Created: {}", vpcId);

            Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
            DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                .vpcIds(vpcId)

```

```
        .build();

        return waiter.waitForVpcAvailable(request)
            .thenApply(waiterResponse -> vpcId);
    })
    .thenCompose(vpcId -> {
        Filter filter = Filter.builder()
            .name("vpc-id")
            .values(vpcId)
            .build();

        DescribeRouteTablesRequest describeRouteTablesRequest =
DescribeRouteTablesRequest.builder()
            .filters(filter)
            .build();

        return
getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
            .thenApply(routeTablesResponse -> {
                if (routeTablesResponse.routeTables().isEmpty()) {
                    throw new CompletionException("No route tables found
for VPC: " + vpcId, null);
                }
                String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                logger.info("Route table found: {}", routeTableId);
                return new AbstractMap.SimpleEntry<>(vpcId,
routeTableId);
            });
    })
    .thenCompose(vpcAndRouteTable -> {
        String vpcId = vpcAndRouteTable.getKey();
        String routeTableId = vpcAndRouteTable.getValue();
        Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
        String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

        CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
            .vpcId(vpcId)
            .routeTableIds(routeTableId)
            .serviceName(serviceName)
            .build();
```

```

        return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
            .thenApply(vpcEndpointResponse -> {
                String vpcEndpointId =
vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                logger.info("VPC Endpoint created: {}", vpcEndpointId);
                return new AbstractMap.SimpleEntry<>(vpcId,
vpcEndpointId);
            });
    })
    .exceptionally(exception -> {
        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
        if (cause instanceof Ec2Exception) {
            logger.error("EC2 error during VPC setup: {}",
cause.getMessage(), cause);
            throw new CompletionException("EC2 error during VPC setup: "
+ cause.getMessage(), cause);
        }

        logger.error("VPC setup failed: {}", cause.getMessage(), cause);
        throw new CompletionException("VPC setup failed: " +
cause.getMessage(), cause);
    });
}
}
}

```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Java 2.x .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

PHP

Kit SDK pour PHP

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario illustrant les bases des compartiments de répertoires Amazon S3 et S3 Express One Zone.

```
echo "\n";
echo "-----\n";
echo "Welcome to the Amazon S3 Express Basics demo using PHP!\n";
echo "-----\n";

// Change these both of these values to use a different region/
availability zone.
$region = "us-west-2";
$az = "usw2-az1";

$this->s3Service = new S3Service(new S3Client(['region' => $region]));
$this->iamService = new IAMService(new IamClient(['region' => $region]));

$uuid = uniqid();

echo <<<INTRO
Let's get started! First, please note that S3 Express One Zone works best when
working within the AWS infrastructure,
specifically when working in the same Availability Zone. To see the best results
in this example, and when you implement
Directory buckets into your infrastructure, it is best to put your Compute
resources in the same AZ as your Directory
bucket.\n
INTRO;
    pressEnter();
    // 1. Configure a gateway VPC endpoint. This is the recommended method to
allow S3 Express One Zone traffic without
    // the need to pass through an internet gateway or NAT device.
echo "\n";
```

```

        echo "1. First, we'll set up a new VPC and VPC Endpoint if this program
        is running in an EC2 instance in the same AZ as your Directory buckets will be.
        \n";

        $ec2Choice = testable_readline("Are you running this in an EC2 instance
        located in the same AZ as your intended Directory buckets? Enter Y/y to setup a
        VPC Endpoint, or N/n/blank to skip this section.");
        if($ec2Choice == "Y" || $ec2Choice == "y") {
            echo "Great! Let's set up a VPC, retrieve the Route Table from it,
            and create a VPC Endpoint to connect the S3 Client to.\n";
            pressEnter();
            $this->ec2Service = new EC2Service(new Ec2Client(['region' =>
            $region]));
            $cidr = "10.0.0.0/16";
            $vpc = $this->ec2Service->createVpc($cidr);
            $this->resources['vpcId'] = $vpc['VpcId'];

            $this->ec2Service->waitForVpcAvailable($vpc['VpcId']);

            $routeTable = $this->ec2Service->describeRouteTables([], [
                [
                    'Name' => "vpc-id",
                    'Values' => [$vpc['VpcId']],
                ],
            ]);

            $serviceName = "com.amazonaws." . $this->ec2Service->getRegion() .
            ".s3express";
            $vpcEndpoint = $this->ec2Service->createVpcEndpoint($serviceName,
            $vpc['VpcId'], [$routeTable[0]]);
            $this->resources['vpcEndpointId'] = $vpcEndpoint['VpcEndpointId'];
        }else{
            echo "Skipping the VPC setup. Don't forget to use this in production!
            \n";
        }

        // 2. Policies, user, and roles with CDK.
        echo "\n";
        echo "2. Policies, users, and roles with CDK.\n";
        echo "Now, we'll set up some policies, roles, and a user. This user will
        only have permissions to do S3 Express One Zone actions.\n";
        pressEnter();

        $this->cloudFormationClient = new CloudFormationClient([]);
        $stackName = "cfn-stack-s3-express-basics-" . uniqid();

```

```

    $file = file_get_contents(__DIR__ . "/../../../../../resources/cfn/
s3_express_basics/s3_express_template.yml");
    $result = $this->cloudFormationClient->createStack([
        'StackName' => $stackName,
        'TemplateBody' => $file,
        'Capabilities' => ['CAPABILITY_IAM'],
    ]);
    $waiter = $this->cloudFormationClient->getWaiter("StackCreateComplete",
['StackName' => $stackName]);
    try {
        $waiter->promise()->wait();
    }catch(CloudFormationException $caught){
        echo "Error waiting for the CloudFormation stack to create: {$caught-
>getAwsErrorMessage()}\n";
        throw $caught;
    }
    $this->resources['stackName'] = $stackName;
    $stackInfo = $this->cloudFormationClient->describeStacks([
        'StackName' => $result['StackId'],
    ]);

    $expressUserName = "";
    $regularUserName = "";
    foreach($stackInfo['Stacks'][0]['Outputs'] as $output) {
        if ($output['OutputKey'] == "RegularUser") {
            $regularUserName = $output['OutputValue'];
        }
        if ($output['OutputKey'] == "ExpressUser") {
            $expressUserName = $output['OutputValue'];
        }
    }
    $regularKey = $this->iamService->createAccessKey($regularUserName);
    $regularCredentials = new Credentials($regularKey['AccessKeyId'],
    $regularKey['SecretAccessKey']);
    $expressKey = $this->iamService->createAccessKey($expressUserName);
    $expressCredentials = new Credentials($expressKey['AccessKeyId'],
    $expressKey['SecretAccessKey']);

    // 3. Create an additional client using the credentials with S3 Express
    permissions.
    echo "\n";
    echo "3. Create an additional client using the credentials with S3
    Express permissions.\n";

```

```

    echo "This client is created with the credentials associated with the
user account with the S3 Express policy attached, so it can perform S3 Express
operations.\n";
    pressEnter();
    $s3RegularClient = new S3Client([
        'Region' => $region,
        'Credentials' => $regularCredentials,
    ]);
    $s3RegularService = new S3Service($s3RegularClient);
    $s3ExpressClient = new S3Client([
        'Region' => $region,
        'Credentials' => $expressCredentials,
    ]);
    $s3ExpressService = new S3Service($s3ExpressClient);
    echo "All the roles and policies were created and attached to the user.
Then, a new S3 Client and Service were created using that user's credentials.
\n";

    echo "We can now use this client to make calls to S3 Express operations.
Keeping permissions in mind (and adhering to least-privilege) is crucial to S3
Express.\n";
    pressEnter();

    // 4. Create two buckets.
    echo "\n";
    echo "3. Create two buckets.\n";
    echo "Now we will create a Directory bucket, which is the linchpin of the
S3 Express One Zone service.\n";
    echo "Directory buckets behave in different ways from regular S3 buckets,
which we will explore here.\n";
    echo "We'll also create a normal bucket, put an object into the normal
bucket, and copy it over to the Directory bucket.\n";
    pressEnter();

    // Create a directory bucket. These are different from normal S3 buckets
in subtle ways.
    $directoryBucketName = "s3-express-demo-directory-bucket-$uuid--$az--x-
s3";
    echo "Now, let's create the actual Directory bucket, as well as a regular
bucket.\n";
    pressEnter();
    $s3ExpressService->createBucket($directoryBucketName, [
        'CreateBucketConfiguration' => [
            'Bucket' => [

```

```

        'Type' => "Directory", // This is what causes S3 to create a
Directory bucket as opposed to a normal bucket.
        'DataRedundancy' => "SingleAvailabilityZone",
    ],
    'Location' => [
        'Name' => $az,
        'Type' => "AvailabilityZone",
    ],
],
]);
$this->resources['directoryBucketName'] = $directoryBucketName;

// Create a normal bucket.
$normalBucketName = "normal-bucket-$uuid";
$s3RegularService->createBucket($normalBucketName);
$this->resources['normalBucketName'] = $normalBucketName;
echo "Great! Both buckets were created.\n";
pressEnter();

// 5. Create an object and copy it over.
echo "\n";
echo "5. Create an object and copy it over.\n";
echo "We'll create a basic object consisting of some text and upload it
to the normal bucket.\n";
echo "Next, we'll copy the object into the Directory bucket using the
regular client.\n";
echo "This works fine, because Copy operations are not restricted for
Directory buckets.\n";
pressEnter();

$objectKey = "basic-text-object";
$s3RegularService->putObject($normalBucketName, $objectKey, $args =
['Body' => "Look Ma, I'm a bucket!"]);
$this->resources['objectKey'] = $objectKey;

// Create a session to access the directory bucket. The SDK Client will
automatically refresh this as needed.
$s3ExpressService->createSession($directoryBucketName);
$s3ExpressService->copyObject($directoryBucketName, $objectKey,
"$normalBucketName/$objectKey");

echo "It worked! It's important to remember the user permissions when
interacting with Directory buckets.\n";

```



```
    echo "Instead of validating permissions on every call as normal buckets
do, Directory buckets utilize the user credentials and session token to
validate.\n";

    echo "This allows for much faster connection speeds on every call. For
single calls, this is low, but for many concurrent calls, this adds up to a lot
of time saved.\n";
    pressEnter();

    // 6. Demonstrate performance difference.
    echo "\n";
    echo "6. Demonstrate performance difference.\n";
    $downloads = 1000;
    echo "Now, let's do a performance test. We'll download the same object
from each bucket $downloads times and compare the total time needed. Note: the
performance difference will be much more pronounced if this example is run in an
EC2 instance in the same AZ as the bucket.\n";
    $downloadChoice = testable_readline("If you would like to download each
object $downloads times, press enter. Otherwise, enter a custom amount and press
enter.");
    if($downloadChoice && is_numeric($downloadChoice) && $downloadChoice <
1000000){ // A million is enough. I promise.
        $downloads = $downloadChoice;
    }

    // Download the object $downloads times from each bucket and time it to
demonstrate the speed difference.
    $directoryStartTime = hrtime(true);
    for($i = 0; $i < $downloads; ++$i){
        $s3ExpressService->getObject($directoryBucketName, $objectKey);
    }
    $directoryEndTime = hrtime(true);
    $directoryTimeDiff = $directoryEndTime - $directoryStartTime;

    $normalStartTime = hrtime(true);
    for($i = 0; $i < $downloads; ++$i){
        $s3RegularService->getObject($normalBucketName, $objectKey);
    }
    $normalEndTime = hrtime(true);
    $normalTimeDiff = $normalEndTime - $normalStartTime;

    echo "The directory bucket took $directoryTimeDiff nanoseconds, while the
normal bucket took $normalTimeDiff.\n";
```

```
    echo "That's a difference of " . ($normalTimeDiff - $directoryTimeDiff) .  
    " nanoseconds, or " . (($normalTimeDiff - $directoryTimeDiff)/1000000000) . "  
seconds.\n";  
    pressEnter();  
  
    // 7. Populate the buckets to show the lexicographical difference.  
    echo "\n";  
    echo "7. Populate the buckets to show the lexicographical difference.\n";  
    echo "Now let's explore how Directory buckets store objects in a  
different manner to regular buckets.\n";  
    echo "The key is in the name \"Directory!\".\n";  
    echo "Where regular buckets store their key/value pairs in a flat manner,  
Directory buckets use actual directories/folders.\n";  
    echo "This allows for more rapid indexing, traversing, and therefore  
retrieval times!\n";  
    echo "The more segmented your bucket is, with lots of directories, sub-  
directories, and objects, the more efficient it becomes.\n";  
    echo "This structural difference also causes ListObjects to behave  
differently, which can cause unexpected results.\n";  
    echo "Let's add a few more objects with layered directories as see how  
the output of ListObjects changes.\n";  
    pressEnter();  
  
    // Populate a few more files in each bucket so that we can use  
ListObjects and show the difference.  
    $otherObject = "other/$objectKey";  
    $altObject = "alt/$objectKey";  
    $otherAltObject = "other/alt/$objectKey";  
    $s3ExpressService->putObject($directoryBucketName, $otherObject);  
    $s3RegularService->putObject($normalBucketName, $otherObject);  
    $this->resources['otherObject'] = $otherObject;  
    $s3ExpressService->putObject($directoryBucketName, $altObject);  
    $s3RegularService->putObject($normalBucketName, $altObject);  
    $this->resources['altObject'] = $altObject;  
    $s3ExpressService->putObject($directoryBucketName, $otherAltObject);  
    $s3RegularService->putObject($normalBucketName, $otherAltObject);  
    $this->resources['otherAltObject'] = $otherAltObject;  
  
    $listDirectoryBucket = $s3ExpressService->  
>listObjects($directoryBucketName);  
    $listNormalBucket = $s3RegularService->listObjects($normalBucketName);  
  
    // Directory bucket content  
    echo "Directory bucket content\n";
```

```

        foreach($listDirectoryBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        // Normal bucket content
        echo "\nNormal bucket content\n";
        foreach($listNormalBucket['Contents'] as $result){
            echo $result['Key'] . "\n";
        }

        echo "Notice how the normal bucket lists objects in lexicographical
order, while the directory bucket does not. This is because the normal bucket
considers the whole \"key\" to be the object identifies, while the directory
bucket actually creates directories and uses the object \"key\" as a path to the
object.\n";
        pressEnter();

        echo "\n";
        echo "That's it for our tour of the basic operations for S3 Express One
Zone.\n";
        $cleanUp = testable_readline("Would you like to delete all the resources
created during this demo? Enter Y/y to delete all the resources.");
        if($cleanUp){
            $this->cleanUp();
        }
    }

namespace S3;

use Aws\CommandInterface;
use Aws\Exception\AwsException;
use Aws\Result;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use AwsUtilities\AWSServiceClass;
use DateTimeInterface;

class S3Service extends AWSServiceClass
{
    protected S3Client $client;
    protected bool $verbose;

    public function __construct(S3Client $client = null, $verbose = false)

```

```
{
    if ($client) {
        $this->client = $client;
    } else {
        $this->client = new S3Client([
            'version' => 'latest',
            'region' => 'us-west-2',
        ]);
    }
    $this->verbose = $verbose;
}

public function setVerbose($verbose)
{
    $this->verbose = $verbose;
}

public function isVerbose(): bool
{
    return $this->verbose;
}

public function getClient(): S3Client
{
    return $this->client;
}

public function setClient(S3Client $client)
{
    $this->client = $client;
}

public function emptyAndDeleteBucket($bucketName, array $args = [])
{
    try {
        $objects = $this->listAllObjects($bucketName, $args);
        $this->deleteObjects($bucketName, $objects, $args);
        if ($this->verbose) {
            echo "Deleted all objects and folders from $bucketName.\n";
        }
        $this->deleteBucket($bucketName, $args);
    } catch (AwsException $exception) {
        if ($this->verbose) {
```

```
        echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
        echo "\nPlease fix error with bucket deletion before continuing.
\n";
    }
    throw $exception;
}

public function createBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->createBucket($parameters);
        if ($this->verbose) {
            echo "Created the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to create $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket creation before continuing.";
        }
        throw $exception;
    }
}

public function putObject(string $bucketName, string $key, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
    $args);
    try {
        $this->client->putObject($parameters);
        if ($this->verbose) {
            echo "Uploaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
```

```

        echo "Failed to create $key in $bucketName with error:
{$exception->getMessage()}\n";
        echo "Please fix error with object uploading before continuing.";
    }
    throw $exception;
}
}

public function getObject(string $bucketName, string $key, array $args = []):
Result
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key],
$args);
    try {
        $object = $this->client->getObject($parameters);
        if ($this->verbose) {
            echo "Downloaded the object named: $key to the bucket named:
$bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to download $key from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object downloading before
continuing.";
        }
        throw $exception;
    }
    return $object;
}

public function copyObject($bucketName, $key, $copySource, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $key,
"CopySource" => $copySource], $args);
    try {
        $this->client->copyObject($parameters);
        if ($this->verbose) {
            echo "Copied the object from: $copySource in $bucketName to:
$key.\n";

```

```

    }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to copy $copySource in $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object copying before continuing.";
        }
        throw $exception;
    }
}

public function listObjects(string $bucketName, $start = 0, $max = 1000,
array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Marker' => $start,
"MaxKeys" => $max], $args);
    try {
        $objects = $this->client->listObjectsV2($parameters);
        if ($this->verbose) {
            echo "Retrieved the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to retrieve the objects from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with list objects before continuing.";
        }
        throw $exception;
    }
    return $objects;
}

public function listAllObjects($bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);

    $contents = [];
    $paginator = $this->client->getPaginator("ListObjectsV2", $parameters);

    foreach ($paginator as $result) {

```

```
        if($result['KeyCount'] == 0){
            break;
        }
        foreach ($result['Contents'] as $object) {
            $contents[] = $object;
        }
    }
    return $contents;
}

public function deleteObjects(string $bucketName, array $objects, array $args
= [])
{
    $listOfObjects = array_map(
        function ($object) {
            return ['Key' => $object];
        },
        array_column($objects, 'Key')
    );
    if(!$listOfObjects){
        return;
    }

    $parameters = array_merge(['Bucket' => $bucketName, 'Delete' =>
['Objects' => $listOfObjects]], $args);
    try {
        $this->client->deleteObjects($parameters);
        if ($this->verbose) {
            echo "Deleted the list of objects from: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete the list of objects from $bucketName with
error: {$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}
```



```
public function deleteBucket(string $bucketName, array $args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName], $args);
    try {
        $this->client->deleteBucket($parameters);
        if ($this->verbose) {
            echo "Deleted the bucket named: $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $bucketName with error: {$exception-
>getMessage()}\n";
            echo "Please fix error with bucket deletion before continuing.";
        }
        throw $exception;
    }
}

public function deleteObject(string $bucketName, string $fileName, array
$args = [])
{
    $parameters = array_merge(['Bucket' => $bucketName, 'Key' => $fileName],
$args);
    try {
        $this->client->deleteObject($parameters);
        if ($this->verbose) {
            echo "Deleted the object named: $fileName from $bucketName.\n";
        }
    } catch (AwsException $exception) {
        if ($this->verbose) {
            echo "Failed to delete $fileName from $bucketName with error:
{$exception->getMessage()}\n";
            echo "Please fix error with object deletion before continuing.";
        }
        throw $exception;
    }
}

public function listBuckets(array $args = [])
{

```

```

        try {
            $buckets = $this->client->listBuckets($args);
            if ($this->verbose) {
                echo "Retrieved all " . count($buckets) . "\n";
            }
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to retrieve bucket list with error: {$exception-
>getMessage()}\n";
                echo "Please fix error with bucket lists before continuing.";
            }
            throw $exception;
        }
        return $buckets;
    }

    public function preSignedUrl(CommandInterface $command, DateTimeInterface|
int|string $expires, array $options = [])
    {
        $request = $this->client->createPresignedRequest($command, $expires,
$options);
        try {
            $presignedUrl = (string)$request->getUri();
        } catch (AwsException $exception) {
            if ($this->verbose) {
                echo "Failed to create a presigned url: {$exception-
>getMessage()}\n";
                echo "Please fix error with presigned urls before continuing.";
            }
            throw $exception;
        }
        return $presignedUrl;
    }

    public function createSession(string $bucketName)
    {
        try{
            $result = $this->client->createSession([
                'Bucket' => $bucketName,
            ]);
        }
    }

```

```
        return $result;
    } catch (S3Exception $caught) {
        if ($caught->getAwsErrorType() == "NoSuchBucket") {
            echo "The specified bucket does not exist.";
        }
        throw $caught;
    }
}
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK pour PHP .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Exécutez un scénario illustrant les bases des compartiments de répertoires Amazon S3 et S3 Express One Zone.

```
class S3ExpressScenario:
```

```
"""Runs an interactive scenario that shows how to get started with S3
Express."""
```

```
def __init__(
    self,
    cloud_formation_resource: ServiceResource,
    ec2_client: client,
    iam_client: client,
):
    self.cloud_formation_resource = cloud_formation_resource
    self.ec2_client = ec2_client
    self.iam_client = iam_client
    self.region = ec2_client.meta.region_name
    self.stack = None
    self.vpc_id = None
    self.vpc_endpoint_id = None
    self.regular_bucket_name = None
    self.directory_bucket_name = None
    self.s3_express_wrapper = None
    self.s3_regular_wrapper = None
```

```
def s3_express_scenario(self):
    """
    Runs the scenario.
    """
    print("")
    print_dashes()
    print("Welcome to the Amazon S3 Express Basics demo using Python (Boto
3)!")
    print_dashes()
    print(
        """
```

Let's get started! First, please note that S3 Express One Zone works best when working within the AWS infrastructure, specifically when working in the same Availability Zone. To see the best results in this example and when you implement Directory buckets into your infrastructure, it is best to put your compute resources in the same AZ as your Directory bucket.

```
"""
    )
    press_enter_to_continue()

    # Create an optional VPC and create 2 IAM users.
```

```

    express_user_name, regular_user_name = self.create_vpc_and_users()

    # Set up two S3 clients, one regular and one express, and two buckets,
    one regular and one express.
    self.setup_clients_and_buckets(express_user_name, regular_user_name)

    # Create an S3 session for the express S3 client and add objects to the
    buckets.
    bucket_object = self.create_session_and_add_objects()

    # Demonstrate performance differences between regular and express
    buckets.
    self.demonstrate_performance(bucket_object)

    # Populate the buckets to show the lexicographical difference between
    regular and express buckets.
    self.show_lexicographical_differences(bucket_object)

    print("")
    print("That's it for our tour of the basic operations for S3 Express One
    Zone.")

    if q.ask(
        "Would you like to delete all the resources created during this demo
    (y/n)? ",
        q.is_yesno,
    ):
        self.cleanup()

    def create_vpc_and_users(self) -> None:
        """
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
        without.
        """
        # Configure a gateway VPC endpoint. This is the recommended method to
        allow S3 Express One Zone traffic without
        # the need to pass through an internet gateway or NAT device.
        print(
            """
1. First, we'll set up a new VPC and VPC Endpoint if this program is running in
    an EC2 instance in the same AZ as your
Directory buckets will be. Are you running this in an EC2 instance located in the
    same AZ as your intended Directory buckets?

```

```

"""
    )
    if q.ask("Do you want to setup a VPC Endpoint? (y/n) ", q.is_yesno):
        print(
            "Great! Let's set up a VPC, retrieve the Route Table from it, and
            create a VPC Endpoint to connect the S3 Client to."
        )
        self.setup_vpc()
        press_enter_to_continue()
    else:
        print("Skipping the VPC setup. Don't forget to use this in
        production!")
        print(
            """
2. Policies, users, and roles with CDK.
Now, we'll set up some policies, roles, and a user. This user will only have
permissions to do S3 Express One Zone actions.
"""
        )
        press_enter_to_continue()
        stack_name = f"cfn-stack-s3-express-basics--{uuid.uuid4()}"
        template_as_string = S3ExpressScenario.get_template_as_string()
        self.stack = self.deploy_cloudformation_stack(stack_name,
        template_as_string)
        regular_user_name = None
        express_user_name = None
        outputs = self.stack.outputs
        for output in outputs:
            if output.get("OutputKey") == "RegularUser":
                regular_user_name = output.get("OutputValue")
            elif output.get("OutputKey") == "ExpressUser":
                express_user_name = output.get("OutputValue")
        if not regular_user_name or not express_user_name:
            error_string = f"""
            Failed to retrieve required outputs from CloudFormation stack.
            'regular_user_name'={regular_user_name},
            'express_user_name'={express_user_name}
            """
            logger.error(error_string)
            raise ValueError(error_string)
        return express_user_name, regular_user_name

    def setup_clients_and_buckets(
        self, express_user_name: str, regular_user_name: str

```

```

) -> None:
    """
    Set up two S3 clients, one regular and one express, and two buckets, one
    regular and one express.
    :param express_user_name: The name of the user with S3 Express
    permissions.
    :param regular_user_name: The name of the user with regular S3
    permissions.
    """
    regular_credentials = self.create_access_key(regular_user_name)
    express_credentials = self.create_access_key(express_user_name)
    # 3. Create an additional client using the credentials with S3 Express
    permissions.
    print(
        """
3. Create an additional client using the credentials with S3 Express permissions.
This client is created with the
credentials associated with the user account with the S3 Express policy attached,
so it can perform S3 Express operations.
"""
    )
    press_enter_to_continue()
    s3_regular_client = self.create_s3_client_with_access_key_credentials(
        regular_credentials
    )
    self.s3_regular_wrapper = S3ExpressWrapper(s3_regular_client)
    s3_express_client = self.create_s3_client_with_access_key_credentials(
        express_credentials
    )
    self.s3_express_wrapper = S3ExpressWrapper(s3_express_client)
    print(
        """
All the roles and policies were created and attached to the user. Then a new S3
Client were created using
that user's credentials. We can now use this client to make calls to S3 Express
operations. Keeping permissions in mind
(and adhering to least-privilege) is crucial to S3 Express.
"""
    )
    press_enter_to_continue()
    # 4. Create two buckets.
    print(
        """
3. Create two buckets.

```

```

Now we will create a Directory bucket which is the linchpin of the S3 Express One
Zone service. Directory buckets
behave in different ways from regular S3 buckets which we will explore here.
We'll also create a normal bucket, put
an object into the normal bucket, and copy it over to the Directory bucket.
"""

    )

    # Create a directory bucket. These are different from normal S3 buckets
    in subtle ways.
    bucket_prefix = q.ask(
        "Enter a bucket name prefix that will be used for both buckets: ",
        q.re_match(r"[a-z0-9](?:[a-z0-9-\.]*)[a-z0-9]$"),
    )

    # Some availability zones are not supported for Directory buckets. We'll
    choose one that is supported.
    print(
        "Now, let's choose an availability zone for the Directory bucket.
    We'll choose one that is supported."
    )
    while True:
        availability_zone = self.select_availability_zone_id(self.region)
        # Construct the parts of a directory bucket name that is made unique
        with a UUID string.
        directory_bucket_suffix = f"--{availability_zone['ZoneId']}--x-s3"
        max_uuid_length = 63 - len(bucket_prefix) -
len(directory_bucket_suffix) - 1
        bucket_uuid = str(uuid.uuid4()).replace("-", "")[:max_uuid_length]
        directory_bucket_name = (
            f"{bucket_prefix}-{bucket_uuid}{directory_bucket_suffix}"
        )
        regular_bucket_name = f"{bucket_prefix}-regular-{bucket_uuid}"
        configuration = {
            "Bucket": {
                "Type": "Directory",
                "DataRedundancy": "SingleAvailabilityZone",
            },
            "Location": {
                "Name": availability_zone["ZoneId"],
                "Type": "AvailabilityZone",
            },
        }
    press_enter_to_continue()

```



```

        print(
            "Now, let's create the actual Directory bucket, as well as a
regular bucket."
        )
        press_enter_to_continue()
        try:
            self.s3_express_wrapper.create_bucket(
                directory_bucket_name, configuration
            )
            break
        except ClientError as client_error:
            if client_error.response["Error"]["Code"] == "InvalidBucketName":
                print(
                    f"Bucket '{directory_bucket_name}' is invalid. This may
be because of selected availability zone."
                )
                if q.ask(
                    "Would you like to select a different availability zone?",
                    q.is_yesno,
                ):
                    continue
                else:
                    raise
            else:
                raise
        print(f"Created directory bucket, '{directory_bucket_name}'")
        self.directory_bucket_name = directory_bucket_name

        self.s3_regular_wrapper.create_bucket(regular_bucket_name)
        print(f"Created regular bucket, '{regular_bucket_name}'")
        self.regular_bucket_name = regular_bucket_name
        print("Great! Both buckets were created.")
        press_enter_to_continue()

    def create_session_and_add_objects(self) -> None:
        """
        Create a session for the express S3 client and add objects to the
buckets.
        """
        print(
            """
5. Create an object and copy it over.

```

We'll create a basic object consisting of some text and upload it to the normal bucket. Next we'll copy the object into the Directory bucket using the regular client. This works fine because copy operations are not restricted for Directory buckets.

```

"""
)
press_enter_to_continue()
bucket_object = "basic-text-object"
self.s3_regular_wrapper.put_object(
    self.regular_bucket_name, bucket_object, "Look Ma, I'm a bucket!"
)
self.s3_express_wrapper.create_session(self.directory_bucket_name)
self.s3_express_wrapper.copy_object(
    self.regular_bucket_name,
    bucket_object,
    self.directory_bucket_name,
    bucket_object,
)
print(
    """

```

It worked! It's important to remember the user permissions when interacting with Directory buckets. Instead of validating permissions on every call as normal buckets do, Directory buckets utilize the user credentials and session token to validate. This allows for much faster connection speeds on every call. For single calls, this is low, but for many concurrent calls this adds up to a lot of time saved.

```

"""
)
press_enter_to_continue()
return bucket_object

def demonstrate_performance(self, bucket_object: str) -> None:
    """
    Demonstrate performance differences between regular and Directory
    buckets.
    :param bucket_object: The name of the object to download from each
    bucket.
    """
    print("")
    print("6. Demonstrate performance difference.")
    print(
        """

```

Now, let's do a performance test. We'll download the same object from each bucket 'downloads' times and compare the total time needed. Note: the performance difference will be much more pronounced if this example is run in an EC2 instance in the same Availability Zone as the bucket.

```

"""
    )
    downloads = 1000
    print(
        f"The number of downloads of the same object for this example is set
at {downloads}."
    )
    if q.ask("Would you like to download a different number? (y/n) ",
q.is_yesno):
        max_downloads = 1000000
        downloads = q.ask(
            f"Enter a number between 1 and {max_downloads} for the number of
downloads: ",
            q.is_int,
            q.in_range(1, max_downloads),
        )
        # Download the object 'downloads' times from each bucket and time it to
demonstrate the speed difference.
        print("Downloading from the Directory bucket.")
        directory_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")

                self.s3_express_wrapper.get_object(
                    self.directory_bucket_name, bucket_object
                )

        directory_time_difference = time.time_ns() - directory_time_start
        print("Downloading from the normal bucket.")
        normal_time_start = time.time_ns()

        for index in range(downloads):
            if index % 10 == 0:
                print(f"Download {index} of {downloads}")
                self.s3_regular_wrapper.get_object(self.regular_bucket_name,
bucket_object)

```

```

        normal_time_difference = time.time_ns() - normal_time_start
        print(
            f"The directory bucket took {directory_time_difference} nanoseconds,
while the normal bucket took {normal_time_difference}."
        )
        difference = normal_time_difference - directory_time_difference
        print(f"That's a difference of {difference} nanoseconds, or")
        print(f"{{(difference) / 1000000000}} seconds.")
        if difference < 0:
            print(
                "The directory buckets were slower. This can happen if you are
not running on the cloud within a vpc."
            )
        press_enter_to_continue()

def show_lexicographical_differences(self, bucket_object: str) -> None:
    """
    Show the lexicographical difference between Directory buckets and regular
    buckets.
    This is done by creating a few objects in each bucket and listing them to
    show the difference.
    :param bucket_object: The object to use for the listing operations.
    """
    print(
        """
7. Populate the buckets to show the lexicographical difference.
Now let's explore how Directory buckets store objects in a different manner to
regular buckets. The key is in the name
"Directory". Where regular buckets store their key/value pairs in a flat manner,
Directory buckets use actual
directories/folders. This allows for more rapid indexing, traversing, and
therefore retrieval times! The more segmented
your bucket is, with lots of directories, sub-directories, and objects, the more
efficient it becomes. This structural
difference also causes ListObjects to behave differently, which can cause
unexpected results. Let's add a few more
objects with layered directories to see how the output of ListObjects changes.
"""
    )
    press_enter_to_continue()
    # Populate a few more files in each bucket so that we can use ListObjects
    and show the difference.
    other_object = f"other/{bucket_object}"
    alt_object = f"alt/{bucket_object}"

```

```

        other_alt_object = f"other/alt/{bucket_object}"
        self.s3_regular_wrapper.put_object(self.regular_bucket_name,
other_object, "")
        self.s3_express_wrapper.put_object(self.directory_bucket_name,
other_object, "")
        self.s3_regular_wrapper.put_object(self.regular_bucket_name, alt_object,
"")
        self.s3_express_wrapper.put_object(self.directory_bucket_name,
alt_object, "")
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, other_alt_object, ""
        )
        self.s3_express_wrapper.put_object(
            self.directory_bucket_name, other_alt_object, ""
        )
        directory_bucket_objects = self.s3_express_wrapper.list_objects(
            self.directory_bucket_name
        )

        regular_bucket_objects = self.s3_regular_wrapper.list_objects(
            self.regular_bucket_name
        )

        print("Directory bucket content")
        for bucket_object in directory_bucket_objects:
            print(f"    {bucket_object['Key']}")
        print("Normal bucket content")
        for bucket_object in regular_bucket_objects:
            print(f"    {bucket_object['Key']}")
        print(
            """
Notice how the normal bucket lists objects in lexicographical order, while the
directory bucket does not. This is
because the normal bucket considers the whole "key" to be the object identifier,
while the directory bucket actually
creates directories and uses the object "key" as a path to the object.
            """
        )
        press_enter_to_continue()

    def cleanup(self) -> None:
        """
        Delete resources created by this scenario.
        """

```

```

        if self.directory_bucket_name is not None:
            self.s3_express_wrapper.delete_bucket_and_objects(
                self.directory_bucket_name
            )
            print(f"Deleted directory bucket, '{self.directory_bucket_name}'")
            self.directory_bucket_name = None

        if self.regular_bucket_name is not None:
            self.s3_regular_wrapper.delete_bucket_and_objects(self.regular_bucket_name)
            print(f"Deleted regular bucket, '{self.regular_bucket_name}'")
            self.regular_bucket_name = None

        if self.stack is not None:
            self.destroy_cloudformation_stack(self.stack)
            self.stack = None

        self.tear_done_vpc()

    def create_access_key(self, user_name: str) -> dict[str, any]:
        """
        Creates an access key for the user.
        :param user_name: The name of the user.
        :return: The access key for the user.
        """
        try:
            access_key = self.iam_client.create_access_key(UserName=user_name)
            return access_key["AccessKey"]
        except ClientError as client_error:
            logging.error(
                "Couldn't create the access key. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

    def create_s3_client_with_access_key_credentials(
        self, access_key: dict[str, any]
    ) -> client:
        """
        Creates an S3 client with access key credentials.
        :param access_key: The access key for the user.
        :return: The S3 Express One Zone client.
        """
        try:

```

```

        s3_express_client = boto3.client(
            "s3",
            aws_access_key_id=access_key["AccessKeyId"],
            aws_secret_access_key=access_key["SecretAccessKey"],
            region_name=self.region,
        )
        return s3_express_client
    except ClientError as client_error:
        logging.error(
            "Couldn't create the S3 Express One Zone client. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def select_availability_zone_id(self, region: str) -> dict[str, any]:
    """
    Selects an availability zone.
    :param region: The region to select the availability zone from.
    :return: The availability zone dictionary.
    """
    try:
        response = self.ec2_client.describe_availability_zones(
            Filters=[{"Name": "region-name", "Values": [region]}]
        )
        availability_zones = response["AvailabilityZones"]
        zone_names = [zone["ZoneName"] for zone in availability_zones]
        index = q.choose("Select an availability zone: ", zone_names)
        return availability_zones[index]
    except ClientError as client_error:
        logging.error(
            "Couldn't describe availability zones. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

def deploy_cloudformation_stack(
    self, stack_name: str, cfn_template: str
) -> ServiceResource:
    """
    Deploys prerequisite resources used by the scenario. The resources are
    defined in the associated `cfn_template.yaml` AWS CloudFormation script
    and are deployed
    as a CloudFormation stack, so they can be easily managed and destroyed.

```

```

:param stack_name: The name of the CloudFormation stack.
:param cfn_template: The CloudFormation template as a string.
:return: The CloudFormation stack resource.
"""
print(f"Deploying CloudFormation stack: {stack_name}.")
stack = self.cloud_formation_resource.create_stack(
    StackName=stack_name,
    TemplateBody=cfn_template,
    Capabilities=["CAPABILITY_NAMED_IAM"],
)
print(f"CloudFormation stack creation started: {stack_name}")
print("Waiting for CloudFormation stack creation to complete...")
waiter = self.cloud_formation_resource.meta.client.get_waiter(
    "stack_create_complete"
)
waiter.wait(StackName=stack.name)
stack.load()
print("CloudFormation stack creation complete.")

return stack

def destroy_cloudformation_stack(self, stack: ServiceResource) -> None:
    """
    Destroys the resources managed by the CloudFormation stack, and the
    CloudFormation
    stack itself.

    :param stack: The CloudFormation stack that manages the example
    resources.
    """
    try:
        print(
            f"CloudFormation stack '{stack.name}' is being deleted. This may
            take a few minutes."
        )
        stack.delete()
        waiter = self.cloud_formation_resource.meta.client.get_waiter(
            "stack_delete_complete"
        )
        waiter.wait(StackName=stack.name)
        print(f"CloudFormation stack '{stack.name}' has been deleted.")
    except ClientError as client_error:
        logging.error(
            "Couldn't delete the CloudFormation stack. Here's why: %s",

```



```

        client_error.response["Error"]["Message"],
    )

    @staticmethod
    def get_template_as_string() -> str:
        """
        Returns a string containing this scenario's CloudFormation template.
        """
        script_directory = os.path.dirname(os.path.abspath(__file__))
        template_file_path = os.path.join(script_directory,
            "s3_express_template.yaml")
        file = open(template_file_path, "r")
        return file.read()

    def setup_vpc(self):
        cidr = "10.0.0.0/16"
        try:
            response = self.ec2_client.create_vpc(CidrBlock=cidr)
            self.vpc_id = response["Vpc"]["VpcId"]

            waiter = self.ec2_client.get_waiter("vpc_available")
            waiter.wait(VpcIds=[self.vpc_id])
            print(f"Created vpc {self.vpc_id}")

        except ClientError as client_error:
            logging.error(
                "Couldn't create the vpc. Here's why: %s",
                client_error.response["Error"]["Message"],
            )
            raise

        try:
            response = self.ec2_client.describe_route_tables(
                Filters=[{"Name": "vpc-id", "Values": [self.vpc_id]}]
            )
            route_table_id = response["RouteTables"][0]["RouteTableId"]
            service_name = f"com.amazonaws.
{self.ec2_client.meta.region_name}.s3express"

            response = self.ec2_client.create_vpc_endpoint(
                VpcId=self.vpc_id,
                RouteTableIds=[route_table_id],
                ServiceName=service_name,
            )
            self.vpc_endpoint_id = response["VpcEndpoint"]["VpcEndpointId"]

```

```

        print(f"Created vpc endpoint {self.vpc_endpoint_id}")

    except ClientError as client_error:
        logging.error(
            "Couldn't create the vpc endpoint. Here's why: %s",
            client_error.response["Error"]["Message"],
        )
        raise

    def tear_down_vpc(self) -> None:
        if self.vpc_endpoint_id is not None:
            try:
                self.ec2_client.delete_vpc_endpoints(
                    VpcEndpointIds=[self.vpc_endpoint_id]
                )
                print(f"Deleted vpc endpoint {self.vpc_endpoint_id}.")
                self.vpc_endpoint_id = None
            except ClientError as client_error:
                logging.error(
                    "Couldn't delete the vpc endpoint %s. Here's why: %s",
                    self.vpc_endpoint_id,
                    client_error.response["Error"]["Message"],
                )
        if self.vpc_id is not None:
            try:
                self.ec2_client.delete_vpc(VpcId=self.vpc_id)
                print(f"Deleted vpc {self.vpc_id}")
                self.vpc_id = None
            except ClientError as client_error:
                logging.error(
                    "Couldn't delete the vpc %s. Here's why: %s",
                    self.vpc_id,
                    client_error.response["Error"]["Message"],
                )

```

Une classe d'encapsuleur pour les fonctions SDK Amazon S3 Express.

```

class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
    interface."""

```

```

def __init__(self, s3_client: Any) -> None:
    """
    Initializes the S3ExpressWrapper with an S3 client.

    :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                        access to AWS S3 services.
    """
    self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

def create_bucket(
    self, bucket_name: str, bucket_configuration: dict[str, any] = None
) -> None:
    """
    Creates a bucket.
    :param bucket_name: The name of the bucket.
    :param bucket_configuration: The optional configuration for the bucket.
    """
    try:
        params = {"Bucket": bucket_name}
        if bucket_configuration:
            params["CreateBucketConfiguration"] = bucket_configuration

        self.s3_client.create_bucket(**params)
    except ClientError as client_error:
        # Do not log InvalidBucketName error because it is logged elsewhere.
        if client_error.response["Error"]["Code"] != "InvalidBucketName":
            logging.error(
                "Couldn't create the bucket %s. Here's why: %s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )

```

```

        )
        raise

    def delete_bucket_and_objects(self, bucket_name: str) -> None:
        """
        Deletes a bucket and its objects.
        :param bucket_name: The name of the bucket.
        """
        try:
            # Delete the objects in the bucket first. This is required for a
            bucket to be deleted.
            paginator = self.s3_client.get_paginator("list_objects_v2")
            page_iterator = paginator.paginate(Bucket=bucket_name)
            for page in page_iterator:
                if "Contents" in page:
                    delete_keys = {
                        "Objects": [{"Key": obj["Key"]} for obj in
page["Contents"]]
                    }
                    response = self.s3_client.delete_objects(
                        Bucket=bucket_name, Delete=delete_keys
                    )
                    if "Errors" in response:
                        for error in response["Errors"]:
                            logging.error(
                                "Couldn't delete object %s. Here's why: %s",
                                error["Key"],
                                error["Message"],
                            )

            self.s3_client.delete_bucket(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't delete the bucket %s. Here's why: %s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )

    def put_object(self, bucket_name: str, object_key: str, content: str) ->
None:
        """
        Puts an object into a bucket.
        :param bucket_name: The name of the bucket.
        :param object_key: The key of the object.

```

```

        :param content: The content of the object.
        """
        try:
            self.s3_client.put_object(Body=content, Bucket=bucket_name,
Key=object_key)
        except ClientError as client_error:
            logging.error(
                "Couldn't put the object %s into bucket %s. Here's why: %s",
                object_key,
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise

def list_objects(self, bucket: str) -> list[str]:
    """
    Lists objects in a bucket.
    :param bucket: The name of the bucket.
    :return: The list of objects in the bucket.
    """
    try:
        response = self.s3_client.list_objects_v2(Bucket=bucket)
        return response.get("Contents", [])
    except ClientError as client_error:
        logging.error(
            "Couldn't list objects in bucket %s. Here's why: %s",
            bucket,
            client_error.response["Error"]["Message"],
        )
        raise

def copy_object(
    self,
    source_bucket: str,
    source_key: str,
    destination_bucket: str,
    destination_key: str,
) -> None:
    """
    Copies an object from one bucket to another.
    :param source_bucket: The source bucket.
    :param source_key: The source key.
    :param destination_bucket: The destination bucket.
    :param destination_key: The destination key.
    """

```

```

        :return: None
        """
        try:
            self.s3_client.copy_object(
                CopySource={"Bucket": source_bucket, "Key": source_key},
                Bucket=destination_bucket,
                Key=destination_key,
            )
        except ClientError as client_error:
            logging.error(
                "Couldn't copy object %s from bucket %s to bucket %s. Here's why:
%s",
                source_key,
                source_bucket,
                destination_bucket,
                client_error.response["Error"]["Message"],
            )
            raise

    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
        :param bucket_name: The name of the bucket.
        """
        try:
            self.s3_client.create_session(Bucket=bucket_name)
        except ClientError as client_error:
            logging.error(
                "Couldn't create the express session for bucket %s. Here's why:
%s",
                bucket_name,
                client_error.response["Error"]["Message"],
            )
            raise

    def get_object(self, bucket_name: str, object_key: str) -> None:
        """
        Gets an object from a bucket.
        :param bucket_name: The name of the bucket.
        :param object_key: The key of the object.
        """
        try:
            self.s3_client.get_object(Bucket=bucket_name, Key=object_key)

```

```
except ClientError as client_error:
    logging.error(
        "Couldn't get the object %s from bucket %s. Here's why: %s",
        object_key,
        bucket_name,
        client_error.response["Error"]["Message"],
    )
    raise
```

- Pour plus de détails sur l'API, consultez les rubriques suivantes dans la Référence des API du kit AWS SDK for Python (Boto3).
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Actions pour les compartiments de répertoire S3 à l'aide de AWS Kits SDK

Les exemples de code suivants montrent comment effectuer des actions individuelles sur les compartiments de répertoire S3 avec des AWS SDK. Chaque exemple inclut un lien vers GitHub, où vous pouvez trouver des instructions pour configurer et exécuter le code.

Ces extraits appellent l'API des compartiments de répertoires S3 et sont des extraits de code de programmes plus volumineux qui doivent être exécutés en contexte. Vous pouvez voir les actions dans leur contexte dans [Scénarios pour les compartiments d'annuaire S3 utilisant AWS Kits SDK](#).

Les exemples suivants incluent uniquement les actions les plus couramment utilisées. Pour obtenir la liste complète, consultez la [Référence des API des compartiments de répertoires Amazon S3](#).

Exemples

- [À utiliser AbortMultipartUploadavec un AWS Kit SDK](#)
- [À utiliser CompleteMultipartUploadavec un AWS Kit SDK](#)
- [À utiliser CopyObjectavec un AWS Kit SDK](#)
- [À utiliser CreateBucketavec un AWS Kit SDK](#)
- [À utiliser CreateMultipartUploadavec un AWS Kit SDK](#)
- [À utiliser CreateSessionavec un AWS Kit SDK](#)
- [À utiliser DeleteBucketavec un AWS Kit SDK](#)
- [À utiliser DeleteBucketEncryptionavec un AWS Kit SDK](#)
- [À utiliser DeleteBucketPolicyavec un AWS Kit SDK](#)
- [À utiliser DeleteObjectavec un AWS Kit SDK](#)
- [À utiliser DeleteObjectsavec un AWS Kit SDK](#)
- [À utiliser GetBucketEncryptionavec un AWS Kit SDK](#)
- [À utiliser GetBucketPolicyavec un AWS Kit SDK](#)
- [À utiliser GetObjectavec un AWS Kit SDK](#)
- [À utiliser GetObjectAttributesavec un AWS Kit SDK](#)
- [À utiliser HeadBucketavec un AWS Kit SDK](#)
- [À utiliser HeadObjectavec un AWS Kit SDK](#)
- [À utiliser ListDirectoryBucketsavec un AWS Kit SDK](#)
- [À utiliser ListMultipartUploadsavec un AWS Kit SDK](#)
- [Utilisez la ListObjectsV2 avec un AWS Kit SDK](#)
- [À utiliser ListPartsWith un AWS Kit SDK](#)
- [À utiliser PutBucketEncryptionavec un AWS Kit SDK](#)
- [À utiliser PutBucketPolicyavec un AWS Kit SDK](#)
- [À utiliser PutObjectavec un AWS Kit SDK](#)
- [À utiliser UploadPartavec un AWS Kit SDK](#)
- [À utiliser UploadPartCopyavec un AWS Kit SDK](#)

À utiliser **AbortMultipartUpload**avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliserAbortMultipartUpload.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Annulez un chargement partitionné dans un compartiment de répertoire.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Aborts a specific multipart upload for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be uploaded
 * @param uploadId The upload ID of the multipart upload to abort
 * @return True if the multipart upload is successfully aborted, false
 * otherwise
 */
public static boolean abortDirectoryBucketMultipartUpload(S3Client s3Client,
String bucketName,
    String objectKey, String uploadId) {
    logger.info("Aborting multipart upload: {} for bucket: {}", uploadId,
bucketName);
    try {
        // Abort the multipart upload
```

```
AbortMultipartUploadRequest abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .uploadId(uploadId)
    .build();

s3Client.abortMultipartUpload(abortMultipartUploadRequest);
logger.info("Aborted multipart upload: {} for object: {}", uploadId,
objectKey);
return true;
} catch (S3Exception e) {
    logger.error("Failed to abort multipart upload: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
    return false;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [AbortMultipartUpload](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **CompleteMultipartUpload** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `CompleteMultipartUpload`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Terminez un chargement partitionné dans un compartiment de répertoires.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CompleteMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * This method completes the multipart upload request by collating all the
 * upload parts.
 *
 * @param s3Client    The S3 client used to interact with S3
 * @param bucketName  The name of the directory bucket
 * @param objectKey   The key (name) of the object to be uploaded
 * @param uploadId    The upload ID used to track the multipart upload
 * @param uploadParts The list of completed parts
 * @return True if the multipart upload is successfully completed, false
 *         otherwise
 */
public static boolean completeDirectoryBucketMultipartUpload(S3Client
s3Client, String bucketName, String objectKey,
    String uploadId, List<CompletedPart> uploadParts) {
```

```
try {
    CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
    .parts(uploadParts)
    .build();

    CompleteMultipartUploadRequest completeMultipartUploadRequest =
CompletedMultipartUploadRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .uploadId(uploadId)
    .multipartUpload(completedMultipartUpload)
    .build();

    CompleteMultipartUploadResponse response =
s3Client.completeMultipartUpload(completeMultipartUploadRequest);
    logger.info("Multipart upload completed. ETag: {}", response.eTag());
    return true;
} catch (S3Exception e) {
    logger.error("Failed to complete multipart upload: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
    return false;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [CompleteMultipartUpload](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **CopyObject** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `CopyObject`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Copiez un objet d'un compartiment de répertoires vers un autre compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CopyObjectRequest;
import software.amazon.awssdk.services.s3.model.CopyObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Copies an object from one S3 general purpose bucket to one S3 directory
 * bucket.
 *
 * @param s3Client      The S3 client used to interact with S3
 * @param sourceBucket  The name of the source bucket
 * @param objectKey     The key (name) of the object to be copied
 * @param targetBucket  The name of the target bucket
 */
public static void copyDirectoryBucketObject(S3Client s3Client, String
sourceBucket, String objectKey,
```

```
String targetBucket) {
    logger.info("Copying object: {} from bucket: {} to bucket: {}",
objectKey, sourceBucket, targetBucket);

    try {
        // Create a CopyObjectRequest
        CopyObjectRequest copyReq = CopyObjectRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(objectKey)
            .destinationBucket(targetBucket)
            .destinationKey(objectKey)
            .build();

        // Copy the object
        CopyObjectResponse copyRes = s3Client.copyObject(copyReq);
        logger.info("Successfully copied {} from bucket {} into bucket {}.
CopyObjectResponse: {}",
            objectKey, sourceBucket, targetBucket,
copyRes.copyObjectResult().toString());

    } catch (S3Exception e) {
        logger.error("Failed to copy object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CopyObject](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **CreateBucket** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `CreateBucket`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez un compartiment de répertoires S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketInfo;
import software.amazon.awssdk.services.s3.model.BucketType;
import software.amazon.awssdk.services.s3.model.CreateBucketConfiguration;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.CreateBucketResponse;
import software.amazon.awssdk.services.s3.model.DataRedundancy;
import software.amazon.awssdk.services.s3.model.LocationInfo;
import software.amazon.awssdk.services.s3.model.LocationType;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client The S3 client used to create the bucket
```

```
* @param bucketName The name of the bucket to be created
* @param zone        The region where the bucket will be created
* @throws S3Exception if there's an error creating the bucket
*/
public static void createDirectoryBucket(S3Client s3Client, String
bucketName, String zone) throws S3Exception {
    logger.info("Creating bucket: {}", bucketName);

    CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
        .location(LocationInfo.builder()
            .type(LocationType.AVAILABILITY_ZONE)
            .name(zone).build())
        .bucket(BucketInfo.builder()
            .type(BucketType.DIRECTORY)
            .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
            .build())
        .build();

    try {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration).build();
        CreateBucketResponse response = s3Client.createBucket(bucketRequest);
        logger.info("Bucket created successfully with location: {}",
response.location());
    } catch (S3Exception e) {
        logger.error("Error creating bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateBucket](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **CreateMultipartUpload** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `CreateMultipartUpload`.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez un chargement partitionné dans un compartiment de répertoires.

```
import com.example.s3.util.S3DirectoryBucketUtils;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * This method creates a multipart upload request that generates a unique
 * upload
 * ID used to track
 * all the upload parts.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be uploaded
 * @return The upload ID used to track the multipart upload
 */
public static String createDirectoryBucketMultipartUpload(S3Client s3Client,
String bucketName, String objectKey) {
```

```
        logger.info("Creating multipart upload for object: {} in bucket: {}",
            objectKey, bucketName);

        try {
            // Create a CreateMultipartUploadRequest
            CreateMultipartUploadRequest createMultipartUploadRequest =
                CreateMultipartUploadRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .build();

            // Initiate the multipart upload
            CreateMultipartUploadResponse response =
                s3Client.createMultipartUpload(createMultipartUploadRequest);
            String uploadId = response.uploadId();
            logger.info("Multipart upload initiated. Upload ID: {}", uploadId);
            return uploadId;

        } catch (S3Exception e) {
            logger.error("Failed to create multipart upload: {} - Error code:
                {}", e.awsErrorDetails().errorMessage(),
                    e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [CreateMultipartUpload](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **CreateSession** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `CreateSession`.

Python

Kit SDK for Python (Boto3)

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

```
class S3ExpressWrapper:
    """Encapsulates Amazon S3 Express One Zone actions using the client
    interface."""

    def __init__(self, s3_client: Any) -> None:
        """
        Initializes the S3ExpressWrapper with an S3 client.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
level
                           access to AWS S3 services.
        """
        self.s3_client = s3_client

    @classmethod
    def from_client(cls) -> "S3ExpressWrapper":
        """
        Creates an S3ExpressWrapper instance with a default s3 client.

        :return: An instance of S3ExpressWrapper initialized with the default S3
client.
        """
        s3_client = boto3.client("s3")
        return cls(s3_client)

    def create_session(self, bucket_name: str) -> None:
        """
        Creates an express session.
        :param bucket_name: The name of the bucket.
        """
        try:
```

```
        self.s3_client.create_session(Bucket=bucket_name)
    except ClientError as client_error:
        logging.error(
            "Couldn't create the express session for bucket %s. Here's why:
%s",
            bucket_name,
            client_error.response["Error"]["Message"],
        )
        raise
```

- Pour plus de détails sur l'API, consultez [CreateSession](#) le AWS manuel de référence de l'API SDK for Python (Boto3).

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteBucket** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `DeleteBucket`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez un compartiment de répertoires S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;

/**
 * Deletes the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket to delete
 */
public static void deleteDirectoryBucket(S3Client s3Client, String
bucketName) {
    logger.info("Deleting bucket: {}", bucketName);

    try {
        // Create a DeleteBucketRequest
        DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                        .bucket(bucketName)
                        .build();

        // Delete the bucket
        s3Client.deleteBucket(deleteBucketRequest);
        logger.info("Successfully deleted bucket: {}", bucketName);

    } catch (S3Exception e) {
        logger.error("Failed to delete bucket: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                        e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucket](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteBucketEncryption** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `DeleteBucketEncryption`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez la configuration de chiffrement d'un compartiment de répertoires.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Deletes the encryption configuration from an S3 bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 */
public static void deleteDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
    DeleteBucketEncryptionRequest deleteRequest =
DeleteBucketEncryptionRequest.builder()
        .bucket(bucketName)
        .build();
```

```
try {
    s3Client.deleteBucketEncryption(deleteRequest);
    logger.info("Bucket encryption deleted for bucket: {}", bucketName);
} catch (S3Exception e) {
    logger.error("Failed to delete bucket encryption: {} - Error code:
{}", e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
    throw e;
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketEncryption](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteBucketPolicy** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `DeleteBucketPolicy`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez une stratégie de compartiment pour un compartiment de répertoires.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

/**
 * Deletes the bucket policy for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 */
public static void deleteDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
    logger.info("Deleting policy for bucket: {}", bucketName);

    try {
        // Create a DeleteBucketPolicyRequest
        DeleteBucketPolicyRequest deletePolicyReq =
DeleteBucketPolicyRequest.builder()
            .bucket(bucketName)
            .build();

        // Delete the bucket policy
        s3Client.deleteBucketPolicy(deletePolicyReq);
        logger.info("Successfully deleted bucket policy");

    } catch (S3Exception e) {
        logger.error("Failed to delete bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteBucketPolicy](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteObject** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `DeleteObject`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez un objet dans un compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;
```

```
/**
 * Deletes an object from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be deleted
 */
public static void deleteDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Deleting object: {} from bucket: {}", objectKey,
bucketName);

    try {
        // Create a DeleteObjectRequest
        DeleteObjectRequest deleteObjectRequest =
DeleteObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Delete the object
        s3Client.deleteObject(deleteObjectRequest);
        logger.info("Object {} has been deleted", objectKey);

    } catch (S3Exception e) {
        logger.error("Failed to delete object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObject](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **DeleteObjects** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `DeleteObjects`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Supprimez plusieurs objets dans un compartiment de répertoires.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Delete;
import software.amazon.awssdk.services.s3.model.DeleteObjectsRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectsResponse;
import software.amazon.awssdk.services.s3.model.ObjectIdentifier;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.net.URISyntaxException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.List;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;
```

```
/**
 * Deletes multiple objects from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKeys The list of keys (names) of the objects to be deleted
 */
public static void deleteDirectoryBucketObjects(S3Client s3Client, String
bucketName, List<String> objectKeys) {
    logger.info("Deleting objects from bucket: {}", bucketName);

    try {
        // Create a list of ObjectIdentifier.
        List<ObjectIdentifier> identifiers = objectKeys.stream()
            .map(key -> ObjectIdentifier.builder().key(key).build())
            .toList();

        Delete delete = Delete.builder()
            .objects(identifiers)
            .build();

        DeleteObjectsRequest deleteObjectsRequest =
DeleteObjectsRequest.builder()
            .bucket(bucketName)
            .delete(delete)
            .build();

        DeleteObjectsResponse deleteObjectsResponse =
s3Client.deleteObjects(deleteObjectsRequest);
        deleteObjectsResponse.deleted().forEach(deleted ->
logger.info("Deleted object: {}", deleted.key()));

    } catch (S3Exception e) {
        logger.error("Failed to delete objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [DeleteObjects](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetBucketEncryption** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `GetBucketEncryption`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez la configuration de chiffrement d'un compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.GetBucketEncryptionResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Retrieves the encryption configuration for an S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 */
```

```
* @param bucketName The name of the directory bucket
* @return The type of server-side encryption applied to the bucket (e.g.,
*         AES256, aws:kms)
*/
public static String getDirectoryBucketEncryption(S3Client s3Client, String
bucketName) {
    try {
        // Create a GetBucketEncryptionRequest
        GetBucketEncryptionRequest getRequest =
GetBucketEncryptionRequest.builder()
            .bucket(bucketName)
            .build();

        // Retrieve the bucket encryption configuration
        GetBucketEncryptionResponse response =
s3Client.getBucketEncryption(getRequest);
        ServerSideEncryptionRule rule =
response.serverSideEncryptionConfiguration().rules().get(0);

        String encryptionType =
rule.applyServerSideEncryptionByDefault().sseAlgorithmAsString();
        logger.info("Bucket encryption algorithm: {}", encryptionType);
        logger.info("KMS Customer Managed Key ID: {}",
rule.applyServerSideEncryptionByDefault().kmsMasterKeyID());
        logger.info("Bucket Key Enabled: {}", rule.bucketKeyEnabled());

        return encryptionType;
    } catch (S3Exception e) {
        logger.error("Failed to get bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketEncryption](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetBucketPolicy** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `GetBucketPolicy`.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez la stratégie d'un compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.GetBucketPolicyResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketPolicy;

/**
 * Retrieves the bucket policy for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return The bucket policy text
 */
public static String getDirectoryBucketPolicy(S3Client s3Client, String
bucketName) {
    logger.info("Getting policy for bucket: {}", bucketName);
```

```
try {
    // Create a GetBucketPolicyRequest
    GetBucketPolicyRequest policyReq = GetBucketPolicyRequest.builder()
        .bucket(bucketName)
        .build();

    // Retrieve the bucket policy
    GetBucketPolicyResponse response =
s3Client.getBucketPolicy(policyReq);

    // Print and return the policy text
    String policyText = response.policy();
    logger.info("Bucket policy: {}", policyText);
    return policyText;

} catch (S3Exception e) {
    logger.error("Failed to get bucket policy: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
    throw e;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [GetBucketPolicy](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetObject** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `GetObject`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action dans son contexte dans les exemples de code suivants :

- [Principes de base](#)
- [Création d'une URL présignée pour obtenir un objet](#)

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez un objet à partir d'un compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.ResponseBytes;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.charset.StandardCharsets;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves an object from the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be retrieved
 * @return The retrieved object as a ResponseInputStream
 */
```

```
public static boolean getDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Retrieving object: {} from bucket: {}", objectKey,
bucketName);

    try {
        // Create a GetObjectRequest
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(objectKey)
            .bucket(bucketName)
            .build();

        // Retrieve the object as bytes
        ResponseBytes<GetObjectResponse> objectBytes =
s3Client.getObjectAsBytes(objectRequest);
        byte[] data = objectBytes.asByteArray();

        // Print object contents to console
        String objectContent = new String(data, StandardCharsets.UTF_8);
        logger.info("Object contents: \n{}", objectContent);

        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObject](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **GetObjectAttributes** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `GetObjectAttributes`.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez les attributs d'un objet à partir d'un compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesRequest;
import software.amazon.awssdk.services.s3.model.GetObjectAttributesResponse;
import software.amazon.awssdk.services.s3.model.ObjectAttributes;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves attributes for an object in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to retrieve attributes for
 * @return True if the object attributes are successfully retrieved, false
 *         otherwise
 */
```

```
public static boolean getDirectoryBucketObjectAttributes(S3Client s3Client,
String bucketName, String objectKey) {
    logger.info("Retrieving attributes for object: {} from bucket: {}",
objectKey, bucketName);

    try {
        // Create a GetObjectAttributesRequest
        GetObjectAttributesRequest getObjectAttributesRequest =
GetObjectAttributesRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .objectAttributes(ObjectAttributes.E_TAG,
ObjectAttributes.STORAGE_CLASS,
                ObjectAttributes.OBJECT_SIZE)
            .build();

        // Retrieve the object attributes
        GetObjectAttributesResponse response =
s3Client.getObjectAttributes(getObjectAttributesRequest);
        logger.info("Attributes for object {}:", objectKey);
        logger.info("ETag: {}", response.eTag());
        logger.info("Storage Class: {}", response.storageClass());
        logger.info("Object Size: {}", response.objectSize());
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object attributes: {} - Error code:
{}",
            e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [GetObjectAttributes](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **HeadBucket** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `HeadBucket`.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Vérifie si le compartiment de répertoire S3 spécifié existe et est accessible.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Checks if the specified S3 directory bucket exists and is accessible.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket to check
 * @return True if the bucket exists and is accessible, false otherwise
 */
public static boolean headDirectoryBucket(S3Client s3Client, String
bucketName) {
    logger.info("Checking if bucket exists: {}", bucketName);

    try {
        // Create a HeadBucketRequest
        HeadBucketRequest headBucketRequest = HeadBucketRequest.builder()
            .bucket(bucketName)
```

```
        .build();
        // If the bucket doesn't exist, the following statement throws
        NoSuchBucketException,
        // which is a subclass of S3Exception.
        s3Client.headBucket(headBucketRequest);
        logger.info("Amazon S3 directory bucket: \"{}\" found.", bucketName);
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to access bucket: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        throw e;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [HeadBucket](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **HeadObject** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `HeadObject`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Obtenez les métadonnées d'un objet dans un compartiment de répertoires.

```
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Retrieves metadata for an object in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to retrieve metadata for
 * @return True if the object exists, false otherwise
 */
public static boolean headDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey) {
    logger.info("Retrieving metadata for object: {} from bucket: {}",
objectKey, bucketName);

    try {
        // Create a HeadObjectRequest
        HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        // Retrieve the object metadata
        HeadObjectResponse response = s3Client.headObject(headObjectRequest);
        logger.info("Amazon S3 object: \"{}\" found in bucket: \"{}\" with
ETag: \"{}\"", objectKey, bucketName,
            response.eTag());
        logger.info("Content-Type: {}", response.contentType());
    }
```

```
        logger.info("Content-Length: {}", response.contentLength());
        logger.info("Last Modified: {}", response.lastModified());
        return true;

    } catch (S3Exception e) {
        logger.error("Failed to retrieve object metadata: {} - Error code:
        {}", e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode(), e);
        return false;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [HeadObject](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **ListDirectoryBuckets** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `ListDirectoryBuckets`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Répertoriez tous les compartiments de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.Bucket;
```



```
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsRequest;
import software.amazon.awssdk.services.s3.model.ListDirectoryBucketsResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.util.List;
import java.util.UUID;
import java.util.stream.Collectors;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;

/**
 * Lists all S3 directory buckets and no general purpose buckets.
 *
 * @param s3Client The S3 client used to interact with S3
 * @return A list of bucket names
 */
public static List<String> listDirectoryBuckets(S3Client s3Client) {
    logger.info("Listing all directory buckets");

    try {
        // Create a ListBucketsRequest
        ListDirectoryBucketsRequest listDirectoryBucketsRequest =
            ListDirectoryBucketsRequest.builder().build();

        // Retrieve the list of buckets
        ListDirectoryBucketsResponse response =
            s3Client.listDirectoryBuckets(listDirectoryBucketsRequest);

        // Extract bucket names
        List<String> bucketNames = response.buckets().stream()
            .map(Bucket::name)
            .collect(Collectors.toList());

        return bucketNames;
    } catch (S3Exception e) {
        logger.error("Failed to list buckets: {} - Error code: {}",
            e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        throw e;
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListDirectoryBuckets](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **ListMultipartUploads** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `ListMultipartUploads`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Répertoriez les chargements partitionnés dans un compartiment de répertoire.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```

```
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Lists multipart uploads for the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return A list of MultipartUpload objects representing the multipart
uploads
 */
public static List<MultipartUpload>
listDirectoryBucketMultipartUploads(S3Client s3Client, String bucketName) {
    logger.info("Listing in-progress multipart uploads for bucket: {}",
bucketName);

    try {
        // Create a ListMultipartUploadsRequest
        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
            .bucket(bucketName)
            .build();

        // List the multipart uploads
        ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
        List<MultipartUpload> uploads = response.uploads();
        for (MultipartUpload upload : uploads) {
            logger.info("In-progress multipart upload: Upload ID: {}, Key:
{}", Initiated: {}", upload.uploadId(),
                upload.key(), upload.initiated());
        }
        return uploads;

    } catch (S3Exception e) {
        logger.error("Failed to list multipart uploads: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
    }
}
```

```
        return List.of(); // Return an empty list if an exception is thrown
    }
}
```

- Pour plus de détails sur l'API, reportez-vous [ListMultipartUploads](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Utilisez la **ListObjectsV2** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `ListObjectsV2`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Répertoriez les objets dans un compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Request;
import software.amazon.awssdk.services.s3.model.ListObjectsV2Response;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.S3Object;

import java.nio.file.Path;
import java.util.List;
import java.util.stream.Collectors;
```

```
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Lists objects in the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @return A list of object keys in the bucket
 */
public static List<String> listDirectoryBucketObjectsV2(S3Client s3Client,
String bucketName) {
    logger.info("Listing objects in bucket: {}", bucketName);

    try {
        // Create a ListObjectsV2Request
        ListObjectsV2Request listObjectsV2Request =
ListObjectsV2Request.builder()
            .bucket(bucketName)
            .build();

        // Retrieve the list of objects
        ListObjectsV2Response response =
s3Client.listObjectsV2(listObjectsV2Request);

        // Extract and return the object keys
        return response.contents().stream()
            .map(S3Object::key)
            .collect(Collectors.toList());

    } catch (S3Exception e) {
        logger.error("Failed to list objects: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
            e.awsErrorDetails().errorCode());
        throw e;
    }
}
```

- Pour plus de détails sur l'API, voir [ListObjectsV2](#) dans le manuel de référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **ListParts** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `ListParts`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Répertorier les parties d'un chargement partitionné dans un compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.ListPartsRequest;
import software.amazon.awssdk.services.s3.model.ListPartsResponse;
import software.amazon.awssdk.services.s3.model.Part;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.IOException;
import java.nio.file.Path;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
```

```
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Lists the parts of a multipart upload for the specified S3 directory
 * bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object being uploaded
 * @param uploadId The upload ID used to track the multipart upload
 * @return A list of Part representing the parts of the multipart upload
 */
public static List<Part> listDirectoryBucketMultipartUploadParts(S3Client
s3Client, String bucketName,
    String objectKey, String uploadId) {
    logger.info("Listing parts for object: {} in bucket: {}", objectKey,
bucketName);

    try {
        // Create a ListPartsRequest
        ListPartsRequest listPartsRequest = ListPartsRequest.builder()
            .bucket(bucketName)
            .uploadId(uploadId)
            .key(objectKey)
            .build();

        // List the parts of the multipart upload
        ListPartsResponse response = s3Client.listParts(listPartsRequest);
        List<Part> parts = response.parts();
        for (Part part : parts) {
            logger.info("Uploaded part: Part number = \"{}\", etag = {}",
part.partNumber(), part.eTag());
        }
        return parts;

    } catch (S3Exception e) {
        logger.error("Failed to list parts: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
```

```
        e.awsErrorDetails().errorCode());
    return List.of(); // Return an empty list if an exception is thrown
}
}
```

- Pour plus de détails sur l'API, reportez-vous [ListParts](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketEncryption** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `PutBucketEncryption`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Définissez le chiffrement de compartiment sur un compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.kms.KmsClient;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketEncryptionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.ServerSideEncryption;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionByDefault;
import
    software.amazon.awssdk.services.s3.model.ServerSideEncryptionConfiguration;
import software.amazon.awssdk.services.s3.model.ServerSideEncryptionRule;
```



```
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsClient;
import static com.example.s3.util.S3DirectoryBucketUtils.createKmsKey;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.scheduleKeyDeletion;

/**
 * Sets the default encryption configuration for an S3 bucket as SSE-KMS.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param kmsKeyId The ID of the customer-managed KMS key
 */
public static void putDirectoryBucketEncryption(S3Client s3Client, String
bucketName, String kmsKeyId) {
    // Define the default encryption configuration to use SSE-KMS. For
    directory
    // buckets, AWS managed KMS keys aren't supported. Only customer-managed
    keys
    // are supported.
    ServerSideEncryptionByDefault encryptionByDefault =
    ServerSideEncryptionByDefault.builder()
        .sseAlgorithm(ServerSideEncryption.AWS_KMS)
        .kmsMasterKeyID(kmsKeyId)
        .build();

    // Create a server-side encryption rule to apply the default encryption
    // configuration. For directory buckets, the bucketKeyEnabled field is
    enforced
    // to be true.
    ServerSideEncryptionRule rule = ServerSideEncryptionRule.builder()
        .bucketKeyEnabled(true)
        .applyServerSideEncryptionByDefault(encryptionByDefault)
        .build();

    // Create the server-side encryption configuration for the bucket
    ServerSideEncryptionConfiguration encryptionConfiguration =
    ServerSideEncryptionConfiguration.builder()
        .rules(rule)
        .build();

    // Create the PutBucketEncryption request
```

```
PutBucketEncryptionRequest putRequest =
PutBucketEncryptionRequest.builder()
    .bucket(bucketName)
    .serverSideEncryptionConfiguration(encryptionConfiguration)
    .build();

// Set the bucket encryption
try {
    s3Client.putBucketEncryption(putRequest);
    logger.info("SSE-KMS Bucket encryption configuration set for the
directory bucket: {}", bucketName);
} catch (S3Exception e) {
    logger.error("Failed to set bucket encryption: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode());
    throw e;
}
}
```

- Pour plus de détails sur l'API, reportez-vous [PutBucketEncryption](#) à la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutBucketPolicy** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `PutBucketPolicy`.

Java

SDK pour Java 2.x

 Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Appliquez une stratégie de compartiment à un compartiment de répertoires.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutBucketPolicyRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getAwsAccountId;

/**
 * Sets the following bucket policy for the specified S3 directory bucket.
 * <pre>
 * {
 *     "Version": "2012-10-17",
 *     "Statement": [
 *         {
 *             "Sid": "AdminPolicy",
 *             "Effect": "Allow",
 *             "Principal": {
 *                 "AWS": "arn:aws:iam::<ACCOUNT_ID>:root"
 *             },
 *             "Action": "s3express:*",
 *             "Resource": "arn:aws:s3express:us-west-2:<ACCOUNT_ID>:bucket/
<DIR_BUCKET_NAME>
 *         }
 *     ]
 * }
 * </pre>
 * This policy grants all S3 directory bucket actions to identities in the
 * same account as the bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param policyText The policy text to be applied
 */
public static void putDirectoryBucketPolicy(S3Client s3Client, String
bucketName, String policyText) {
    logger.info("Setting policy on bucket: {}", bucketName);
    logger.info("Policy: {}", policyText);

```

```
try {
    PutBucketPolicyRequest policyReq = PutBucketPolicyRequest.builder()
        .bucket(bucketName)
        .policy(policyText)
        .build();

    s3Client.putBucketPolicy(policyReq);
    logger.info("Bucket policy set successfully!");

} catch (S3Exception e) {
    logger.error("Failed to set bucket policy: {} - Error code: {}",
        e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
    throw e;
}
```

- Pour plus de détails sur l'API, voir [PutBucketPolicy](#) la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **PutObject** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `PutObject`.

Les exemples d'actions sont des extraits de code de programmes de plus grande envergure et doivent être exécutés en contexte. Vous pouvez voir cette action en contexte dans l'exemple de code suivant :

- [Principes de base](#)

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Copiez un objet vers un compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.awscore.exception.AwsErrorDetails;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;

import java.io.UncheckedIOException;
import java.nio.file.Path;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

/**
 * Puts an object into the specified S3 directory bucket.
 *
 * @param s3Client The S3 client used to interact with S3
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to be placed in the bucket
 * @param filePath The path of the file to be uploaded
 */
public static void putDirectoryBucketObject(S3Client s3Client, String
bucketName, String objectKey, Path filePath) {
    logger.info("Putting object: {} into bucket: {}", objectKey, bucketName);
```

```
try {
    // Create a PutObjectRequest
    PutObjectRequest putObj = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .build();

    // Upload the object
    s3Client.putObject(putObj, filePath);
    logger.info("Successfully placed {} into bucket {}", objectKey,
bucketName);

} catch (UncheckedIOException e) {
    throw S3Exception.builder().message("Failed to read the file: " +
e.getMessage()).cause(e)
        .awsErrorDetails(AwsErrorDetails.builder()
            .errorCode("ClientSideException:FailedToReadFile")
            .errorMessage(e.getMessage())
            .build())
        .build();
} catch (S3Exception e) {
    logger.error("Failed to put object: {}", e.getMessage(), e);
    throw e;
}
}
```

- Pour plus de détails sur l'API, voir [PutObject](#) la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **UploadPart** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `UploadPart`.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Chargez une partie d'un chargement partitionné pour un compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.nio.ByteBuffer;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;

/**
 * This method creates part requests and uploads individual parts to S3.
```

```

    * While it uses the UploadPart API to upload a single part, it does so
    * sequentially to handle multiple parts of a file, returning all the
    completed
    * parts.
    *
    * @param s3Client    The S3 client used to interact with S3
    * @param bucketName The name of the directory bucket
    * @param objectKey   The key (name) of the object to be uploaded
    * @param uploadId    The upload ID used to track the multipart upload
    * @param filePath    The path to the file to be uploaded
    * @return A list of uploaded parts
    * @throws IOException if an I/O error occurs
    */
    public static List<CompletedPart> multipartUploadForDirectoryBucket(S3Client
s3Client, String bucketName,
        String objectKey, String uploadId, Path filePath) throws IOException
    {
        logger.info("Uploading parts for object: {} in bucket: {}", objectKey,
bucketName);

        int partNumber = 1;
        List<CompletedPart> uploadedParts = new ArrayList<>();
        ByteBuffer bb = ByteBuffer.allocate(1024 * 1024 * 5); // 5 MB byte buffer

        // Read the local file, break down into chunks and process
        try (RandomAccessFile file = new RandomAccessFile(filePath.toFile(),
"r")) {
            long fileSize = file.length();
            int position = 0;

            // Sequentially upload parts of the file
            while (position < fileSize) {
                file.seek(position);
                int read = file.getChannel().read(bb);

                bb.flip(); // Swap position and limit before reading from the
buffer

                UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .uploadId(uploadId)
                    .partNumber(partNumber)
                    .build();

```



```

        UploadPartResponse partResponse = s3Client.uploadPart(
            uploadPartRequest,
            RequestBody.fromByteBuffer(bb));

        // Build the uploaded part
        CompletedPart uploadedPart = CompletedPart.builder()
            .partNumber(partNumber)
            .eTag(partResponse.eTag())
            .build();

        // Add the uploaded part to the list
        uploadedParts.add(uploadedPart);

        // Log to indicate the part upload is done
        logger.info("Uploaded part number: {} with ETag: {}", partNumber,
            partResponse.eTag());

        bb.clear();
        position += read;
        partNumber++;
    }
} catch (S3Exception e) {
    logger.error("Failed to list parts: {} - Error code: {}",
        e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode());
    throw e;
}
return uploadedParts;
}

```

- Pour plus de détails sur l'API, voir [UploadPart](#) la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

À utiliser **UploadPartCopy** avec un AWS Kit SDK

L'exemple de code suivant montre comment utiliser `UploadPartCopy`.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Créez des parties de copie en fonction de la taille de l'objet source et copiez des parties individuelles dans un compartiment de répertoires.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.HeadObjectRequest;
import software.amazon.awssdk.services.s3.model.HeadObjectResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartCopyRequest;
import software.amazon.awssdk.services.s3.model.UploadPartCopyResponse;

import java.io.IOException;
import java.nio.file.Path;
import java.util.ArrayList;
import java.util.List;

import static
    com.example.s3.util.S3DirectoryBucketUtils.abortDirectoryBucketMultipartUploads;
import static
    com.example.s3.util.S3DirectoryBucketUtils.completeDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static
    com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucketMultipartUpload;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
```

```
import static
com.example.s3.util.S3DirectoryBucketUtils.multipartUploadForDirectoryBucket;

/**
 * Creates copy parts based on source object size and copies over individual
 * parts.
 *
 * @param s3Client      The S3 client used to interact with S3
 * @param sourceBucket  The name of the source bucket
 * @param sourceKey     The key (name) of the source object
 * @param destinationBucket The name of the destination bucket
 * @param destinationKey The key (name) of the destination object
 * @param uploadId      The upload ID used to track the multipart upload
 * @return A list of completed parts
 */
public static List<CompletedPart>
multipartUploadCopyForDirectoryBucket(S3Client s3Client, String sourceBucket,
String sourceKey, String destinationBucket, String destinationKey,
String uploadId) {
    // Get the object size to track the end of the copy operation
    HeadObjectRequest headObjectRequest = HeadObjectRequest.builder()
        .bucket(sourceBucket)
        .key(sourceKey)
        .build();
    HeadObjectResponse headObjectResponse =
s3Client.headObject(headObjectRequest);
    long objectSize = headObjectResponse.contentLength();

    logger.info("Source Object size: {}", objectSize);

    // Copy the object using 20 MB parts
    long partSize = 20 * 1024 * 1024; // 20 MB
    long bytePosition = 0;
    int partNum = 1;
    List<CompletedPart> uploadedParts = new ArrayList<>();

    while (bytePosition < objectSize) {
        long lastByte = Math.min(bytePosition + partSize - 1, objectSize -
1);
        logger.info("Part Number: {}, Byte Position: {}, Last Byte: {}",
partNum, bytePosition, lastByte);

        try {
```

```

        UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
            .sourceBucket(sourceBucket)
            .sourceKey(sourceKey)
            .destinationBucket(destinationBucket)
            .destinationKey(destinationKey)
            .uploadId(uploadId)
            .copySourceRange("bytes=" + bytePosition + "-" +
lastByte)

            .partNumber(partNum)
            .build();
        UploadPartCopyResponse uploadPartCopyResponse =
s3Client.uploadPartCopy(uploadPartCopyRequest);

        CompletedPart part = CompletedPart.builder()
            .partNumber(partNum)
            .eTag(uploadPartCopyResponse.copyPartResult().eTag())
            .build();
        uploadedParts.add(part);

        bytePosition += partSize;
        partNum++;
    } catch (S3Exception e) {
        logger.error("Failed to copy part number {}: {} - Error code:
{}", partNum,
            e.awsErrorDetails().errorMessage(),
e.awsErrorDetails().errorCode());
        throw e;
    }
}

return uploadedParts;
}

```

- Pour plus de détails sur l'API, voir [UploadPartCopy](#) la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Scénarios pour les compartiments d'annuaire S3 utilisant AWS Kits SDK

Les exemples de code suivants vous montrent comment implémenter des scénarios courants dans des compartiments de répertoire S3 avec des AWS SDK. Ces scénarios vous montrent comment accomplir des tâches spécifiques en appelant plusieurs fonctions dans des compartiments de répertoires S3 ou en les combinant avec d'autres Services AWS. Chaque exemple inclut un lien vers le code source complet, où vous trouverez des instructions sur la configuration et l'exécution du code.

Les scénarios ciblent un niveau d'expérience intermédiaire pour vous aider à comprendre les actions de service dans leur contexte.

Exemples

- [Créez une URL présignée pour les compartiments d'annuaire Amazon S3 afin d'obtenir un objet à l'aide d'un AWS Kit SDK](#)

Créez une URL présignée pour les compartiments d'annuaire Amazon S3 afin d'obtenir un objet à l'aide d'un AWS Kit SDK

L'exemple de code suivant montre comment créer une URL présignée pour des compartiments de répertoires S3 et obtenir un objet.

Java

SDK pour Java 2.x

Note

Il y en a plus à ce sujet GitHub. Trouvez l'exemple complet et découvrez comment le configurer et l'exécuter dans le [référentiel d'exemples de code AWS](#).

Générez une URL GET présignée pour accéder à un objet dans un compartiment de répertoires S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
```

```
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;

import java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Generates a presigned URL for accessing an object in the specified S3
 * directory bucket.
 *
 * @param s3Presigner The S3 presigner client used to generate the presigned
URL
 * @param bucketName The name of the directory bucket
 * @param objectKey The key (name) of the object to access
 * @return A presigned URL for accessing the specified object
 */
public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
s3Presigner, String bucketName,
    String objectKey) {
    logger.info("Generating presigned URL for object: {} in bucket: {}",
objectKey, bucketName);

    try {
        // Create a GetObjectRequest
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();
```

```
// Create a GetObjectPresignRequest
GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
    .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
    valid for 10 minutes
    .getObjectRequest(getObjectRequest)
    .build();

// Generate the presigned URL
PresignedGetObjectRequest presignedGetObjectRequest =
s3Presigner.presignGetObject(getObjectPresignRequest);

// Get the presigned URL
String presignedURL = presignedGetObjectRequest.url().toString();
logger.info("Presigned URL: {}", presignedURL);
return presignedURL;

} catch (S3Exception e) {
    logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
        e.awsErrorDetails().errorCode(), e);
    throw e;
}
}
```

- Pour plus de détails sur l'API, voir [GetObject](#) la section Référence des AWS SDK for Java 2.x API.

Pour obtenir la liste complète des guides de développement du AWS SDK et des exemples de code, consultez [Développement avec Amazon S3 à l'aide du AWS Kits SDK](#). Cette rubrique comprend également des informations sur le démarrage et sur les versions précédentes du kit SDK.

Demandes d'authentification (AWS Signature (version 4))

Rubriques

- [Méthodes d'authentification](#)
- [Présentation des demandes de signature](#)
- [Authentification des demandes : utilisation de l'en-tête d'autorisation \(AWS Signature \(version 4\)\)](#)
- [Authentification des demandes : utilisation des paramètres de requête \(AWS Signature \(version 4\)\)](#)
- [Exemples : calculs de signature dans AWS Signature Version 4](#)
- [Demandes d'authentification : Browser-Based téléchargements à l'aide de POST \(AWS Signature \(version 4\)\)](#)
- [Clés de politique d'authentification spécifiques à Amazon S3 Signature version 4](#)

Chaque interaction avec Amazon S3 est authentifiée ou anonyme. Cette section explique l'authentification des demandes à l'aide de l'algorithme AWS Signature Version 4.

Note

Si vous utilisez les AWS SDK (voir [Exemples de code et bibliothèques](#)) pour envoyer vos demandes, vous n'avez pas besoin de lire cette section car les clients du SDK authentifient vos demandes à l'aide des clés d'accès que vous fournissez. À moins d'avoir une bonne raison de ne pas le faire, vous devez toujours utiliser les AWS SDK. Dans les régions qui prennent en charge les deux versions de signature, vous pouvez demander aux AWS kits de développement logiciel d'utiliser une version de signature spécifique. Pour de plus amples informations, veuillez consulter [Envoi de demandes authentifiées à l'aide du AWS Kits SDK](#). Vous devez lire cette section uniquement si vous implémentez l'algorithme AWS Signature Version 4 dans votre client personnalisé.

L'authentification avec AWS Signature Version 4 fournit tout ou partie des éléments suivants, selon la manière dont vous choisissez de signer votre demande :

- Vérification de l'identité du demandeur — Les demandes authentifiées nécessitent une signature que vous créez à l'aide de vos clés d'accès (ID de clé d'accès, clé d'accès secrète). Pour plus d'informations sur l'obtention de clés d'accès, voir [Comprendre et obtenir vos informations](#)

[d'identification de sécurité](#) dans le Références générales AWS. Si vous utilisez des informations d'identification de sécurité temporaires, les calculs de signature nécessitent également un jeton de sécurité. Pour plus d'informations, consultez [Demande d'informations d'identification temporaires de sécurité](#) dans le Guide de l'utilisateur IAM.

- In-transit protection des données — Afin d'empêcher toute manipulation d'une demande pendant son transit, vous utilisez certains éléments de la demande pour calculer la signature de la demande. À la réception de la demande, Amazon S3 calcule la signature en utilisant les mêmes éléments de demande. Si un composant de demande reçu par Amazon S3 ne correspond pas au composant qui a été utilisé pour calculer la signature, Amazon S3 rejettera la demande.
- Protection contre la réutilisation des parties signées de la demande — Les parties signées (à l'aide de AWS signatures) des demandes sont valides dans les 15 minutes suivant l'horodatage de la demande. Une partie non autorisée ayant accès à une demande signée peut modifier les parties non signées de la demande sans affecter la validité de la demande dans la fenêtre de 15 minutes. C'est pourquoi nous vous recommandons de maximiser la protection en signant les en-têtes et le corps des demandes, en envoyant des requêtes HTTPS à Amazon S3 et en utilisant la clé de `s3:x-amz-content-sha256` condition (voir [Clés de politique d'authentification spécifiques à Amazon S3 Signature version 4](#)) dans AWS les politiques pour obliger les utilisateurs à signer les corps de requêtes Amazon S3.

Note

Amazon S3 prend en charge Signature Version 4, un protocole permettant d'authentifier les demandes d'API entrantes destinées aux AWS services, dans l'ensemble. Régions AWSÀ l'heure actuelle, les versions Régions AWS créées avant le 30 janvier 2014 continueront de prendre en charge le protocole précédent, Signature Version 2. Toutes les nouvelles régions après le 30 janvier 2014 ne prendront en charge que la version 4 de la signature. Par conséquent, toutes les demandes adressées à ces régions doivent être effectuées avec la version 4 de la signature. Pour plus d'informations sur AWS Signature Version 2, consultez la section [Signature et authentification des requêtes REST](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Méthodes d'authentification

Vous pouvez exprimer les informations d'authentification à l'aide de l'une des méthodes suivantes :

- **En-tête d'autorisation HTTP** : l'utilisation de l'`Authorization`-en-tête HTTP est la méthode la plus courante pour authentifier une requête Amazon S3. Toutes les opérations REST d'Amazon S3 (à l'exception des chargements par navigateur utilisant des requêtes POST) nécessitent cet en-tête. Pour plus d'informations sur la valeur de l'`Authorization`-en-tête et sur la façon de calculer la signature et les options associées, consultez [Authentification des demandes : utilisation de l'en-tête d'autorisation \(AWS Signature \(version 4\)\)](#).
- **Paramètres de chaîne de requête** : vous pouvez utiliser une chaîne de requête pour exprimer une demande entièrement dans une URL. Dans ce cas, vous utilisez les paramètres de requête pour fournir les informations de demande, y compris les informations d'authentification. Comme la signature de la demande fait partie de l'URL, ce type d'URL est généralement appelé URL présignée. Vous pouvez utiliser des URL présignées pour intégrer des liens cliquables, dont la validité peut aller jusqu'à sept jours, au format HTML. Pour de plus amples informations, veuillez consulter [Authentification des demandes : utilisation des paramètres de requête \(AWS Signature \(version 4\)\)](#).

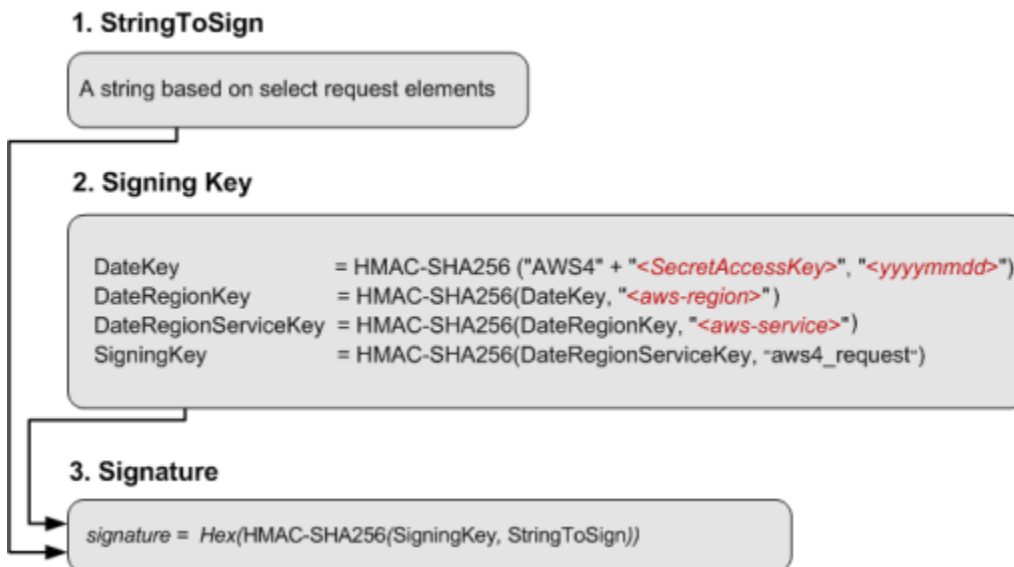
Amazon S3 prend également en charge les chargements via un navigateur qui utilisent des requêtes HTTP POST. Avec une requête HTTP POST, vous pouvez télécharger du contenu sur Amazon S3 directement depuis le navigateur. Pour plus d'informations sur l'authentification des requêtes POST, consultez [Browser-Based Téléversements à l'aide de POST \(AWS Signature \(version 4\)\)](#).

Présentation des demandes de signature

Les informations d'authentification que vous envoyez dans une demande doivent inclure une signature. Pour calculer une signature, vous devez d'abord concaténer les éléments de la demande de sélection pour former une chaîne, appelée chaîne à signer. Vous utilisez ensuite une clé de signature pour calculer le code d'authentification de message utilisant hash (HMAC) de la chaîne à signer.

Dans AWS la version 4 de Signature, vous n'utilisez pas votre clé d'accès secrète pour signer la demande. À la place, vous utilisez d'abord votre clé d'accès secrète pour dériver une clé de signature. La clé de signature dérivée est spécifique à la date, au service et à la région. Pour plus d'informations sur la dérivation d'une clé de signature dans différents langages de programmation, voir [Exemples de dérivation d'une clé de signature pour Signature Version 4](#).

Le schéma suivant illustre le processus général de calcul d'une signature.



La chaîne à signer dépend du type de demande. Par exemple, lorsque vous utilisez l'en-tête d'autorisation HTTP ou les paramètres de requête pour l'authentification, vous utilisez une combinaison variable d'éléments de demande pour créer la chaîne à signer. Pour une requête HTTP POST, la politique POST de la demande est la chaîne que vous signez. Pour plus d'informations sur le calcul d'une chaîne à signer, suivez les liens fournis à la fin de cette section.

Pour clé de signature, le schéma montre une série de calculs, dont le résultat de chaque étape est intégré à l'étape suivante. La dernière étape est la clé de signature.

À la réception d'une demande authentifiée, les serveurs Amazon S3 recréent la signature en utilisant les informations d'authentification contenues dans la demande. Si les signatures correspondent, Amazon S3 traite votre demande ; sinon, la demande est rejetée.

Pour plus d'informations sur l'authentification des demandes, consultez les rubriques suivantes :

- [Authentification des demandes : utilisation de l'en-tête d'autorisation \(AWS Signature \(version 4\)\)](#)
- [Authentification des demandes : utilisation des paramètres de requête \(AWS Signature \(version 4\)\)](#)
- [Browser-Based Téléversements à l'aide de POST \(AWS Signature \(version 4\)\)](#)

Authentification des demandes : utilisation de l'en-tête d'autorisation (AWS Signature (version 4))

Rubriques

- [Vue d'ensemble](#)

- [Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en un seul bloc \(AWS Signature \(version 4\)\)](#)
- [Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en plusieurs segments \(téléchargement fragmenté\) \(AWS Signature \(version 4\)\)](#)
- [Calculs de signature pour les en-têtes de fin \(téléchargements fragmentés\) \(AWS Signature \(version 4\)\)](#)

Vue d'ensemble

L'utilisation de l'Authorization en-tête HTTP est la méthode la plus courante pour fournir des informations d'authentification. À l'exception des requêtes [POST](#) et des requêtes signées à l'aide de paramètres de requête, toutes les opérations Amazon S3 utilisent l'en-tête de Authorization demande pour fournir des informations d'authentification.

Voici un exemple de valeur d'en-tête Authorization. Des sauts de ligne sont ajoutés à cet exemple pour faciliter la lecture :

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

Le tableau suivant décrit les différentes composantes de la valeur Authorization d'en-tête de l'exemple précédent :

Composant	Description
AWS4-HMAC-SHA256	Algorithme utilisé pour calculer la signature. Vous devez fournir cette valeur lorsque vous utilisez AWS Signature Version 4 pour l'authentification. La chaîne spécifie la version 4 de la AWS signature (AWS4) et l'algorithme de signature (HMAC-SHA256).
Credential	

Composant	Description
	Votre ID de clé d'accès et les informations relatives au champ d'application, notamment la date, la région et le service, qui ont été utilisés pour calculer la signature. Cette chaîne présente le format suivant : <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Où : • <i><date></i> la valeur est spécifiée à l'aide YYYYMMDD du format. • <i><aws-service></i> La valeur est définie s3 lors de l'envoi de la demande à Amazon S3.
SignedHeaders	Une liste d'en-têtes de demande séparés par des points-virgules que vous avez utilisés pour calculer Signature . La liste inclut uniquement les noms d'en-tête, qui doivent être en minuscules. Par exemple : <pre>host;range;x-amz-date</pre>
Signature	La signature de 256 bits exprimée en 64 caractères hexadécimaux minuscules. Par exemple : <pre>fe5f80f77d5fa3beca038a248ff027d0445342fe2855d dc963176630326f1024</pre> Veuillez noter que les calculs de signature varient en fonction de l'option que vous choisissez pour transférer la charge utile.

Les calculs de signature varient en fonction de la méthode que vous choisissez pour transférer la charge utile de la demande. S3 prend en charge les options suivantes :

- Transférer la charge utile en un seul bloc — Dans ce cas, vous disposez des options de calcul de signature suivantes :
- Option de charge utile signée : vous pouvez éventuellement calculer la somme de contrôle complète de la charge utile et l'inclure dans le calcul de la signature. Cela apporte une sécurité accrue, mais vous devez lire votre charge utile deux fois ou la mettre en mémoire tampon.

Par exemple, pour charger un fichier, vous devez d'abord le lire pour calculer un hachage de charge utile pour le calcul de la signature, puis de nouveau pour la transmission lorsque vous créez la demande. Pour les charges utiles plus petites, cette approche peut être préférable. Cependant, pour les fichiers volumineux, la lecture deux fois du fichier peut s'avérer inefficace. Vous pouvez donc préférer télécharger les données par morceaux.

Nous vous recommandons d'inclure la somme de contrôle de la charge utile pour plus de sécurité.

- Option de charge utile non signée : n'incluez pas la somme de contrôle de la charge utile dans le calcul des signatures.

Pour obtenir des instructions détaillées sur le calcul de la signature et la création de la valeur d'en-tête Authorization, consultez [Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en un seul bloc \(AWS Signature \(version 4\)\)](#).

- Transférer la charge utile en plusieurs blocs (téléchargement fragmenté) — Dans ce cas, vous transférez la charge utile en plusieurs morceaux. Vous pouvez transférer une charge utile par morceaux, quelle que soit la taille de la charge utile.

Vous pouvez diviser votre charge utile en plusieurs parties. Il peut s'agir de morceaux de taille fixe ou variable. En téléchargeant les données par morceaux, vous évitez de lire l'intégralité de la charge utile pour calculer la signature. Au lieu de cela, pour le premier segment, vous calculez une signature initiale qui utilise uniquement les en-têtes de demande. Le second bloc contient la signature du premier bloc, et chaque bloc suivant contient la signature du segment qui le précède. À la fin du téléchargement, vous envoyez un dernier bloc contenant 0 octet de données contenant la signature du dernier bloc de la charge utile. Pour de plus amples informations, veuillez consulter [Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en plusieurs segments \(téléchargement fragmenté\) \(AWS Signature \(version 4\)\)](#).

Lorsque vous signez vos demandes, vous pouvez utiliser AWS Signature Version 4 ou AWS Signature Version 4A. La principale différence entre les deux est déterminée par la façon dont la signature est calculée. Avec AWS la version 4A de Signature, la signature ne contient aucune Region-specific information et est calculée à l'aide de l'AWS4-ECDSA-P256-SHA256algorithme.

En plus de ces options, vous avez la possibilité d'inclure une remorque avec votre demande. Pour inclure une bande-annonce dans votre demande, vous devez le spécifier dans l'en-tête en réglant `x-amz-content-sha256` la valeur appropriée. Si vous utilisez un en-tête de fin, vous devez inclure `x-amz-trailer` dans l'en-tête et spécifier les noms d'en-tête de fin sous forme de chaîne dans une liste séparée par des virgules. Tous les en-têtes de fin sont écrits après le dernier morceau. Si vous chargez les données en plusieurs blocs, vous devez envoyer un dernier bloc contenant 0 octet de données avant d'envoyer l'en-tête de fin.

Lorsque vous envoyez une demande, vous devez indiquer à Amazon S3 laquelle des options précédentes vous avez choisie dans le calcul de votre signature, en ajoutant l'`x-amz-content-sha256`en-tête avec l'une des valeurs suivantes :

Valeur d'en-tête	Description
Valeur réelle de la somme de contrôle de la charge utile	Cette valeur est la somme de contrôle réelle de votre objet et n'est possible que lorsque vous chargez les données en un seul bloc.
UNSIGNED-PAYLOAD	Utilisez-le lorsque vous chargez l'objet sous la forme d'un seul bloc non signé.
STREAMING-UNSIGNED-PAYLOAD-TRAILER	Utilisez-le lorsque vous envoyez une charge utile non signée sur plusieurs segments. Dans ce cas, vous avez également un en-tête à la fin du téléchargement du bloc.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD	Utilisez-le lorsque vous envoyez une charge utile sur plusieurs segments et que les segments sont signés à l'aide de. AWS4-HMAC-SHA256 Cela produit une signature SIGv4.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER	Utilisez-le lorsque vous envoyez une charge utile sur plusieurs segments et que les segments sont signés à l'aide de. AWS4-HMAC-SHA256 Cela produit une signature SIGv4. De plus, le résumé des morceaux est inclus dans un en-tête de fin.

Valeur d'en-tête	Description
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD	Utilisez-le lorsque vous envoyez une charge utile sur plusieurs segments et que les segments sont signés à l'aide de. AWS4-ECDSA-P256-SHA256 Cela produit une signature SigV4a.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD-TRAILER	Utilisez-le lorsque vous envoyez une charge utile sur plusieurs segments et que les segments sont signés à l'aide de. AWS4-ECDSA-P256-SHA256 Cela produit une signature SigV4a. De plus, le résumé des morceaux est inclus dans un en-tête de fin.

À la réception de la demande, Amazon S3 recrée la chaîne à signer à l'aide des informations figurant dans l'`Authorization` en-tête et dans l'`date` en-tête. Il vérifie ensuite auprès du service d'authentification que les signatures correspondent. La date de la demande peut être spécifiée à l'aide du HTTP `Date` ou de l'`x-amz-date` en-tête. Si les deux en-têtes sont présents, c'`x-amz-date` est prioritaire.

Si les signatures correspondent, Amazon S3 traite votre demande ; sinon, votre demande échouera.

Pour plus d'informations, consultez les rubriques suivantes :

[Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en un seul bloc \(AWS Signature \(version 4\)\)](#)

[Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en plusieurs segments \(téléchargement fragmenté\) \(AWS Signature \(version 4\)\)](#)

[Calculs de signature pour les en-têtes de fin \(téléchargements fragmentés\) \(AWS Signature \(version 4\)\)](#)

Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en un seul bloc (AWS Signature (version 4))

Lorsque vous utilisez l'`Authorization` en-tête pour authentifier des demandes, la valeur de l'en-tête inclut, entre autres, une signature. Les calculs de signature varient en fonction du choix que vous faites pour transférer la charge utile ([Vue d'ensemble](#)). Cette section explique les calculs de

signature lorsque vous choisissez de transférer la charge utile en un seul bloc. La section d'exemple (voir [Exemples : calculs de signature](#)) présente les calculs de signature et Authorization les en-têtes qui en résultent, que vous pouvez utiliser comme suite de tests pour vérifier votre code.

Important

Lorsque vous transférez une charge utile en un seul bloc, vous pouvez éventuellement choisir d'inclure le hachage de la charge utile dans les calculs de signature, appelé charge utile signée (si vous ne l'incluez pas, la charge utile est considérée comme non signée). La procédure de signature décrite dans la section suivante s'applique aux deux, mais notez les différences suivantes :

- Option de charge utile signée : vous incluez le hachage de la charge utile lors de la création de la demande canonique (qui en fait ensuite partie intégrante StringToSign, comme expliqué dans la section sur le calcul de signature). Vous spécifiez également la même valeur que la valeur `x-amz-content-sha256` d'en-tête lors de l'envoi de la demande à S3.
- Option de charge utile non signée : vous incluez la chaîne littérale `UNSIGNED-PAYLOAD` lors de la création d'une demande canonique et vous définissez la même valeur que la valeur d'`x-amz-content-sha256` en-tête lors de l'envoi de la demande à Amazon S3.

Lorsque vous envoyez votre demande à Amazon S3, la valeur `x-amz-content-sha256` d'en-tête indique à Amazon S3 si la charge utile est signée ou non. Amazon S3 peut ensuite créer la signature en conséquence à des fins de vérification.

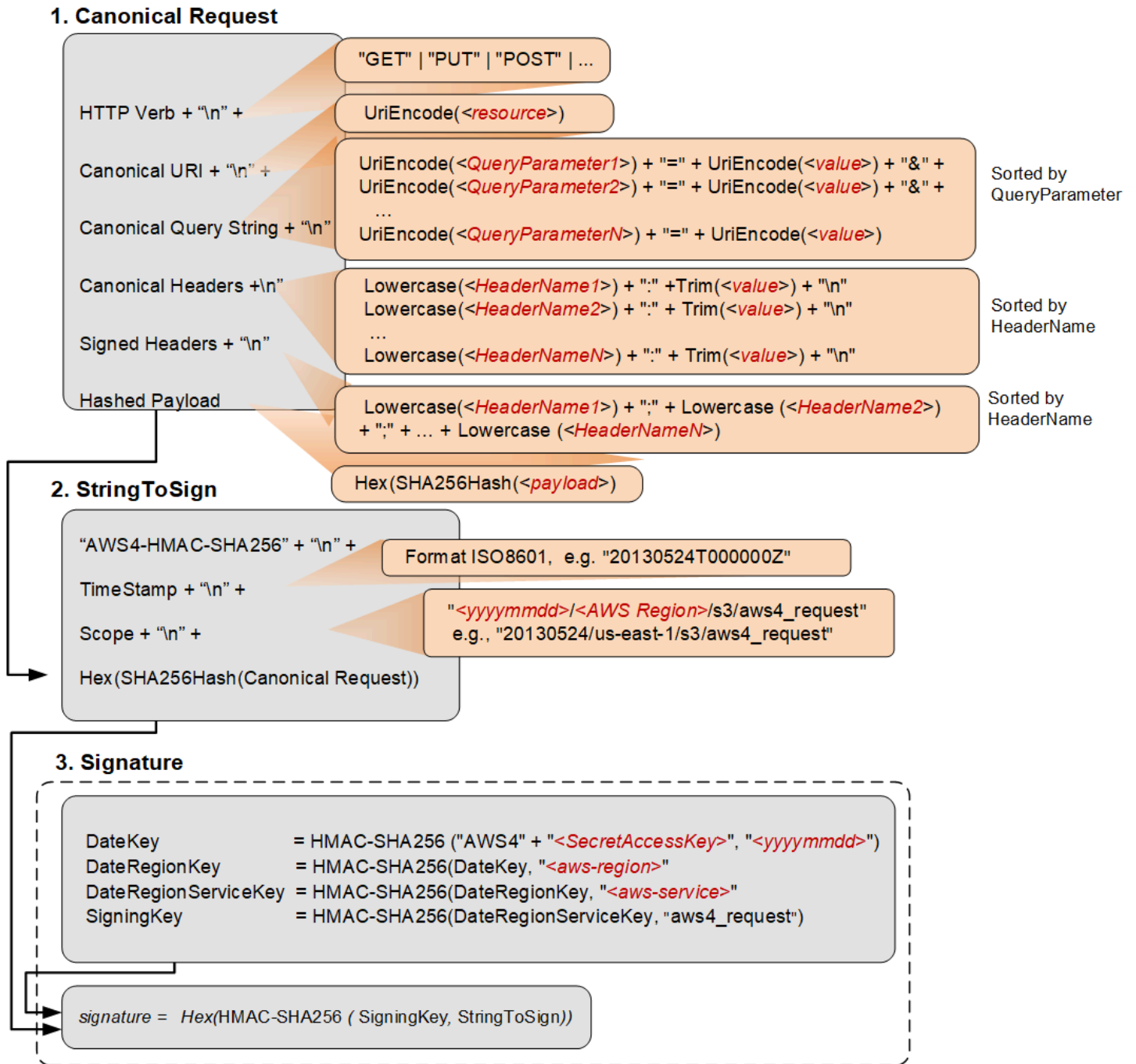
Dans les deux cas, étant donné que la valeur de l'`x-amz-content-sha256` en-tête fait déjà partie de votre contenu `HashedPayload`, vous n'êtes pas obligé de l'inclure en tant qu'`x-amz-content-sha256` en-tête canonique.

Calculer une signature

Pour calculer une signature, vous avez d'abord besoin d'une chaîne à signer. Vous calculez ensuite le HMAC-SHA256 hachage de la chaîne à signer à l'aide d'une clé de signature. Le schéma suivant illustre le processus, y compris les différents composants de la chaîne que vous créez pour la signature

Lorsqu'Amazon S3 reçoit une demande authentifiée, il calcule la signature, puis la compare à la signature que vous avez fournie dans la demande. C'est pourquoi vous devez calculer la signature en

utilisant la même méthode que celle utilisée par Amazon S3. Le processus qui consiste à présenter une demande sous une forme convenue pour la signature est appelé canonisation.



Le tableau suivant décrit les fonctions présentées dans le schéma. Vous devez implémenter du code pour ces fonctions.

Fonction	Description
<code>Lowercase()</code>	Convertit la chaîne en minuscules.
<code>Hex()</code>	Encodage en minuscules en base 16.
<code>SHA256Hash()</code>	Fonction de hachage cryptographique Secure Hash Algorithm (SHA, algorithme de hachage sécurisé).
<code>HMAC-SHA256()</code>	Calcule HMAC à l'aide de l'algorithme SHA256 avec la clé de signature fournie. Il s'agit de la signature finale.
<code>Trim()</code>	Supprimez tous les espaces blancs de début ou de fin.
<code>UriEncode()</code>	L'URI code chaque octet. UriEncode() doit appliquer les règles suivantes : • L'URI encode tous les octets sauf les caractères non réservés : A-Z, a-z, 0-9, -, ., _ et ~. • Le caractère d'espace est un caractère réservé qui doit être codé sous la forme « %20 » (et non sous la forme « + »). • Chaque octet encodé par URI est formé par un « % » et la valeur hexadécimale à deux chiffres de l'octet. • Les lettres de la valeur hexadécimale doivent être en majuscules, par exemple « %1A ». • Encodez la barre oblique « / » partout sauf dans le nom de la clé de l'objet. Par exemple, si le nom de clé de l'objet est <code>photos/Jan/sample.jpg</code> , la barre oblique du nom de la clé n'est pas encodée.  Important Les UriEncode fonctions standard fournies par votre plateforme de développement peuvent ne pas fonctionner en raison de différences d'implémentation et de l'ambiguïté associée dans les RFC sous-jacents. Nous vous recommandons d'écrire votre propre UriEncode

Fonction	Description
	fonction personnalisée pour garantir le bon fonctionnement de votre encodage. Voici un exemple de fonction UriEncode () en Java. <pre> public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Tâche 1 : créer une demande canonique

Cette section fournit une vue d'ensemble de la création d'une demande canonique.

Voici le format de demande canonique qu'Amazon S3 utilise pour calculer une signature. Pour que les signatures soient identiques, vous devez créer une requête canonique au format suivant :

```

<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n

```

```
<HashedPayload>
```

Où :

- **HTTPMethod** est l'une des méthodes HTTP, par exemple GET, PUT, HEAD et DELETE.
- **CanonicalURI** est la URI-encoded version du composant de chemin absolu de l'URI : tout ce qui commence par le « / » qui suit le nom de domaine et jusqu'à la fin de la chaîne ou jusqu'au point d'interrogation (« ? ») si vous avez des paramètres de chaîne de requête. L'URI de l'exemple suivant, /examplebucket/myphoto.jpg, est le chemin absolu et vous n'encodez pas la « / » dans le chemin absolu :

```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

Note

Vous ne normalisez pas les chemins d'URI pour les requêtes adressées à Amazon S3. Par exemple, vous pouvez avoir un compartiment contenant un objet nommé « my-object//example//photo.user ». La normalisation du chemin change le nom de l'objet dans la requête en « my-object/example/photo.user ». Il s'agit d'un chemin incorrect pour cet objet.

- **CanonicalQueryString** spécifie les paramètres URI-encoded de chaîne de requête. Vous URI-encode nommez et valorisez individuellement. Vous devez également trier les paramètres de la chaîne de requête canonique par ordre alphabétique par nom de clé. Le tri s'effectue après l'encodage. La chaîne de requête dans l'exemple d'URI suivant est prefix=somePrefix&marker=someMarker&max-keys=20 :

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

La chaîne de requête canonique est la suivante (des sauts de ligne sont ajoutés à cet exemple pour faciliter la lecture) :

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&"+  
UriEncode("max-keys")+"="+UriEncode("20") + "&" +
```

```
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

Lorsqu'une requête cible une sous-ressource, la valeur du paramètre de requête correspondant est une chaîne vide (« »). Par exemple, l'URI suivant identifie la sous-ressource ACL sur le compartiment `examplebucket` :

```
http://s3.amazonaws.com/examplebucket?acl
```

CanonicalQueryString Dans ce cas, c'est le suivant :

```
UriEncode("acl") + "=" + ""
```

Si l'URI ne contient pas de « ? », la demande ne contient aucune chaîne de requête et vous définissez la chaîne de requête canonique sur une chaîne vide (« »). Vous devrez tout de même inclure le « \n ».

- ***CanonicalHeaders*** est une liste d'en-têtes de requête avec leurs valeurs. Les paires de nom et de valeur d'en-tête individuelles sont séparées par le caractère de saut de ligne (« \n »). Les noms d'en-tête doivent être en minuscules. Vous devez trier les noms d'en-tête par ordre alphabétique pour construire la chaîne, comme indiqué dans l'exemple suivant :

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"  
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"  
...  
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

Les fonctions `Lowercase()` et `Trim()` utilisées dans cet exemple sont décrites dans la section précédente.

La ***CanonicalHeaders*** liste doit inclure les éléments suivants :

- En-tête `host` HTTP.
- Si l'`Content-MD5` en-tête est présent dans la demande, vous devez l'ajouter à la ***CanonicalHeaders*** liste.
- Les en-têtes `x-amz-*` que vous prévoyez d'inclure dans votre demande doivent également être ajoutés. Par exemple, si vous utilisez des informations d'identification de sécurité temporaires,

vous devez inclure `x-amz-security-token` dans votre demande. Vous devez ajouter cet en-tête dans la liste des *CanonicalHeaders*.

 Note

- L'`x-amz-content-sha256` en-tête est obligatoire pour toutes les demandes AWS Signature Version 4. Il fournit un hachage de la charge utile de la requête. Cependant, vous n'êtes pas obligé d'inclure le `x-amz-content-sha256` en tant qu'en-tête canonique car S3 utilisera automatiquement sa valeur lors du calcul du hachage de la charge utile envoyé dans la demande.
- S'il n'y a pas de charge utile, vous devez fournir le hachage d'une chaîne vide.
- Si vous ne souhaitez pas que S3 vérifie le hachage de la demande, vous pouvez utiliser la chaîne littérale à la "UNSIGNED-PAYLOAD" place.

Voici des exemples de CanonicalHeaders chaînes. Les noms d'en-tête sont en minuscules et sont triés.

Exemple Exemple 1

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

Exemple Exemple 2

```
host:s3.amazonaws.com
x-amz-content-sha256:UNSIGNED-PAYLOAD
x-amz-date:20130708T220855Z
```

Exemple Exemple 3

```
host:s3.amazonaws.com
x-amz-date:20130708T220855Z
```

 Note

Aux fins du calcul d'une signature d'autorisation, l'en-tête de l'hôte et tous les `x-amz-*` en-têtes, à l'exception des en-têtes `x-amz-content-sha256`, sont requis. Toutefois, afin d'éviter toute falsification des données, vous devriez envisager d'inclure tous les en-têtes dans le calcul de la signature.

La signature de l'`x-amz-content-sha256` en-tête est facultative car S3 utilisera sa valeur lors du calcul du hachage de la charge utile de la demande reçue.

- ***SignedHeaders*** est une liste alphabétique de noms d'en-têtes de demande séparés par des points-virgules et séparés par des points-virgules. Les en-têtes de demande de la liste sont les mêmes que ceux que vous avez inclus dans la chaîne `CanonicalHeaders`. Dans les exemples précédents, la valeur de ***SignedHeaders*** serait la suivante :

Exemple Exemples 1 et 2

```
host;x-amz-content-sha256;x-amz-date
```

Exemple Exemple 3

```
host;x-amz-date
```

- ***HashedPayload*** est la valeur hexadécimale du hachage SHA256 de la charge utile de la demande.

```
Hex(SHA256Hash(<payload>))
```

Si la demande ne contient aucune charge utile, vous calculez un hachage de la chaîne vide comme suit :

```
Hex(SHA256Hash(""))
```

Le hachage renvoie la valeur suivante :


```
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

Par exemple, lorsque vous chargez un objet à l'aide d'une demande PUT, vous fournissez les données de l'objet dans le corps du message. Lorsque vous récupérez un objet à l'aide d'une requête GET, vous calculez le hachage de chaîne vide.

Tâche 2 : créer une chaîne de connexion

Cette section fournit une vue d'ensemble de la création d'une chaîne à signer. Pour obtenir des instructions détaillées, voir [Tâche 2 : Création d'une chaîne de caractères pour se connecter](#) Références générales AWS.

La chaîne à signer est une concaténation des chaînes suivantes :

```
"AWS4-HMAC-SHA256" + "\n" +  
timeStampISO8601Format + "\n" +  
<Scope> + "\n" +  
Hex(SHA256Hash(<CanonicalRequest>))
```

La chaîne constante AWS4-HMAC-SHA256 spécifie l'algorithme de hachage que vous utilisez, HMAC-SHA256. timeStamp s'agit de l'heure UTC actuelle au format ISO 8601 (par exemple, 20130524T000000Z).

Scope est la signature obtenue à une date Région AWS, un et un service spécifiques. Ainsi, la signature qui en résultera ne fonctionnera que dans la région et pour un service spécifique. La signature est valide pendant sept jours après la date spécifiée.

```
date.Format(<YYYYMMDD>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

Pour Amazon S3, la chaîne de service est s3. Pour une liste de *region* chaînes, voir [Régions et points de terminaison](#) dans le Références générales AWS. La colonne Région de ce tableau fournit la liste des chaînes de région valides.

La portée suivante limite la signature résultante à la us-east-1 région et à Amazon S3.

```
20130606/us-east-1/s3/aws4_request
```

 Note

Scopo doit utiliser la même date que celle que vous utilisez pour calculer la clé de signature, comme indiqué dans la section suivante.

Tâche 3 : Calculer la signature

Dans la version 4 de AWS Signature, au lieu d'utiliser vos clés d' AWS accès pour signer une demande, vous créez d'abord une clé de signature limitée à une région et à un service spécifiques. Pour plus d'informations sur les clés de signature, consultez [Présentation des demandes de signature](#).

```
DateKey           = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<YYYYMMDD>")
DateRegionKey     = HMAC-SHA256(<DateKey>, "<aws-region>")
DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>, "<aws-service>")
SigningKey        = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")
```

 Note

Dans certains cas d'utilisation, les clés de signature peuvent être traitées pendant une durée maximale de 7 jours. Pour plus d'informations, voir [Partager un objet avec d'autres personnes](#).

Pour une liste des chaînes de régions, voir [Régions et points de terminaison](#) dans le Références générales AWS.

L'utilisation d'une clé de signature vous permet de conserver vos AWS informations d'identification en un seul endroit sûr. Par exemple, si plusieurs serveurs communiquent avec Amazon S3, vous partagez la clé de signature avec ces serveurs ; vous n'êtes pas obligé de conserver une copie de votre clé d'accès secrète sur chaque serveur. La clé de signature est valide pendant sept jours au maximum. Ainsi, chaque fois que vous calculez la clé de signature, vous devrez partager la clé de signature avec vos serveurs. Pour de plus amples informations, veuillez consulter [Demandes d'authentification \(AWS Signature \(version 4\)\)](#).

La signature finale est le HMAC-SHA256 hachage de la chaîne à signer, en utilisant la clé de signature comme clé.

```
HMAC-SHA256(SigningKey, StringToSign)
```

Pour obtenir des instructions détaillées sur la création d'une signature, voir [Tâche 3 : Créer une signature](#) dans le Références générales AWS.

Exemples : calculs de signature

Vous pouvez utiliser les exemples de cette section comme référence pour vérifier les calculs de signature dans votre code. Les calculs présentés dans les exemples utilisent les données suivantes :

- Exemples de clés d'accès.

Paramètre	Value
AWSSecretAccessKey	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- Horodatage de la demande 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Nom du bucket `examplebucket`.
- Le compartiment est supposé se trouver dans la région de l'Est des États-Unis (Virginie du Nord). Les informations d'identification Scope et les Signing Key calculs sont utilisés `us-east-1` comme spécificateur de région. Pour plus d'informations sur les autres régions, voir [Régions et points de terminaison](#) dans le Références générales AWS.
- Vous pouvez utiliser des requêtes de type chemin ou des requêtes de type hébergement virtuel. Les exemples suivants montrent comment signer une demande de type hébergement virtuel, par exemple :

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Pour plus d'informations, consultez la section Hébergement [virtuel de compartiments](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Exemple : objet GET

L'exemple suivant extrait les 10 premiers octets d'un objet (test.txt) depuis `examplebucket`. Pour plus d'informations sur l'action d'API, consultez [GetObject](#).

```
GET /test.txt HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
Range: bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date: 20130524T000000Z
```

Comme cette requête GET ne fournit aucun contenu de corps, la `x-amz-content-sha256` valeur peut être soit le hachage du corps de requête vide, soit la chaîne "UNSIGNED-PAYLOAD" littérale. Les étapes suivantes montrent les calculs de signature et la construction de l'`Authorization` en tête à l'aide du hachage d'une chaîne vide.

1. StringToSign

a. CanonicalRequest

```
GET
/test.txt

host:examplebucket.s3.amazonaws.com
range:bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;range;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Dans la chaîne de requête canonique, la dernière ligne est le hachage du corps de requête vide. La troisième ligne est vide car elle ne contient aucun paramètre de requête.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
```

```
20130524/us-east-1/s3/aws4_request
7344ae5b7ee6c3e7e6b0fe0640412a37625d1fbfff95c48bbb2dc43964946972
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

4. En-tête Authorization

L'Authorization en-tête qui en résulte est le suivant :

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;range;x-amz-content-sha256;x-amz-
date,Signature=f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

Exemple : objet PUT

Cet exemple de requête PUT crée un objet (`test$file.text`) dans `examplebucket`. Dans cet exemple il est supposé que :

- Vous faites une demande `REDUCED_REDUNDANCY` en tant que classe de stockage en ajoutant l'en-tête de `x-amz-storage-class` demande. Pour plus d'informations sur les classes de stockage, consultez la section [Classes de stockage](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.
- Le contenu du fichier téléchargé est une chaîne. `Welcome to Amazon S3.` La valeur de `x-amz-content-sha256` dans la demande est basée sur cette chaîne.

Pour plus d'informations sur l'action d'API, consultez [PutObject](#).

```
PUT test$file.text HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Fri, 24 May 2013 00:00:00 GMT
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
x-amz-content-sha256: 44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072

<Payload>
```

Les étapes suivantes présentent les calculs de signature.

1. StringToSign

a. CanonicalRequest

```
PUT
/test%24file.text

date:Fri, 24 May 2013 00:00:00 GMT
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
x-amz-date:20130524T000000Z
x-amz-storage-class:REDUCED_REDUNDANCY

date;host;x-amz-content-sha256;x-amz-date;x-amz-storage-class
44ce7dd67c959e0d3524ffac1771dfbba87d2b6b4b4e99e42034a8b803f8b072
```

Dans la requête canonique, la troisième ligne est vide car elle ne contient aucun paramètre de requête. L'en-tête `x-amz-content-sha256` canonique peut éventuellement être signé puisque son hachage de charge utile est déjà fourni au bas de la demande. La dernière ligne est le hachage du corps, qui doit être identique à la `x-amz-content-sha256` header valeur envoyée à S3 dans la requête HTTP.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
```

```
20130524/us-east-1/s3/aws4_request
9e0e90d9c76de8fa5b200d8c849cd5b8dc7a3be3951ddb7f6a76b4158342019d
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

4. En-tête Authorization

L'Authorization en-tête qui en résulte est le suivant :

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=date;host;x-amz-content-sha256;x-amz-date;x-amz-storage-
class,Signature=98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

Exemple : GET Bucket Lifecycle

La requête GET suivante récupère la configuration du cycle de vie de `examplebucket`. Pour plus d'informations sur l'action d'API, consultez [GetBucketLifecycleConfiguration](#).

```
GET ?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Étant donné que la demande ne fournit aucun contenu de corps, la valeur `x-amz-content-sha256` d'en-tête est le hachage du corps de demande vide. Les étapes suivantes présentent les calculs de signature.

1. StringToSign

a. CanonicalRequest

```
GET
/
lifecycle=
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Dans la requête canonique, la dernière ligne est le hash du corps vide de la requête.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
9766c798316ff2757b517bc739a67f6213b4ab36dd5da2f94eaebf79c77395ca
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

4. En-tête Authorization

L'Authorization en-tête qui en résulte est le suivant :


```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

Exemple : Get Bucket (List Objects)

L'exemple suivant récupère une liste d'objets à partir d'un examplebucket bucket. Pour plus d'informations sur l'action d'API, consultez [ListObjects](#).

```
GET ?max-keys=2&prefix=J HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Étant donné que la demande ne fournit pas de corps, la valeur de x-amz-content-sha256 est le hachage du corps de demande vide. Les étapes suivantes présentent les calculs de signature.

1. StringToSign

a. CanonicalRequest

```
GET
/
max-keys=2&prefix=J
host:examplebucket.s3.amazonaws.com
x-amz-content-
sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Dans la chaîne canonique, la dernière ligne est le hachage du corps de requête vide.

b. StringToSign

```
AWS4-HMAC-SHA256
```

```
20130524T000000Z
20130524/us-east-1/s3/aws4_request
df57d21db20da04d7fa30298dd4488ba3a2b47ca3a489c74750e0f1e7df1b9b7
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

4. En-tête Authorization

L'Authorization en-tête qui en résulte est le suivant :

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en plusieurs segments (téléchargement fragmenté) (AWS Signature (version 4))

Comme décrit dans le [Vue d'ensemble](#), lorsque vous authentifiez des demandes à l'aide de l'`Authorization`-en-tête, vous avez la possibilité de télécharger la charge utile par morceaux. Vous pouvez envoyer des données en blocs de taille fixe ou variable. Cette section décrit le processus de calcul de la signature lors d'un téléchargement fragmenté, comment créer le corps du bloc et comment fonctionne la signature différée lorsque vous chargez le bloc pour la première fois et que vous envoyez sa signature dans le bloc suivant. La section d'exemple (voir [Exemple : objet PUT](#)) présente les calculs de signature et `Authorization` les en-têtes qui en résultent, que vous pouvez utiliser comme suite de tests pour vérifier votre code.

Note

Lorsque vous transférez des données en plusieurs segments, vous devez effectuer l'une des opérations suivantes :

- Spécifiez explicitement la longueur totale du contenu (longueur de l'objet en octets plus les métadonnées de chaque bloc) à l'aide de l'en-tête `Content-Length` HTTP. Pour ce faire, vous devez précalculer la longueur totale de la charge utile, y compris les métadonnées que vous envoyez dans chaque bloc, avant de lancer votre demande.
- Spécifiez l'en-tête `Transfer-Encoding` HTTP. Si vous incluez l'`Transfer-Encoding`-en-tête et que vous spécifiez une valeur différente `identity`, vous devez omettre l'`Content-Length`-en-tête.

Pour toutes les demandes, vous devez inclure l'`x-amz-decoded-content-length`-en-tête, en spécifiant la taille de l'objet en octets.

Le calcul de la signature de chaque segment inclut la signature du segment précédent. Pour commencer, vous devez créer une signature initiale en utilisant uniquement les en-têtes. Vous utilisez la signature initiale dans le calcul de la signature du premier segment. Pour chaque segment suivant, vous créez une signature de bloc qui inclut la signature du bloc précédent. Ainsi, les signatures de bloc sont enchaînées ; c'est-à-dire que la signature du bloc `n` est une fonction `F` (bloc `n`, signature (bloc `n-1`)). Le chaînage garantit que vous envoyez les morceaux dans le bon ordre.

Pour effectuer un téléchargement fragmenté, procédez comme suit :

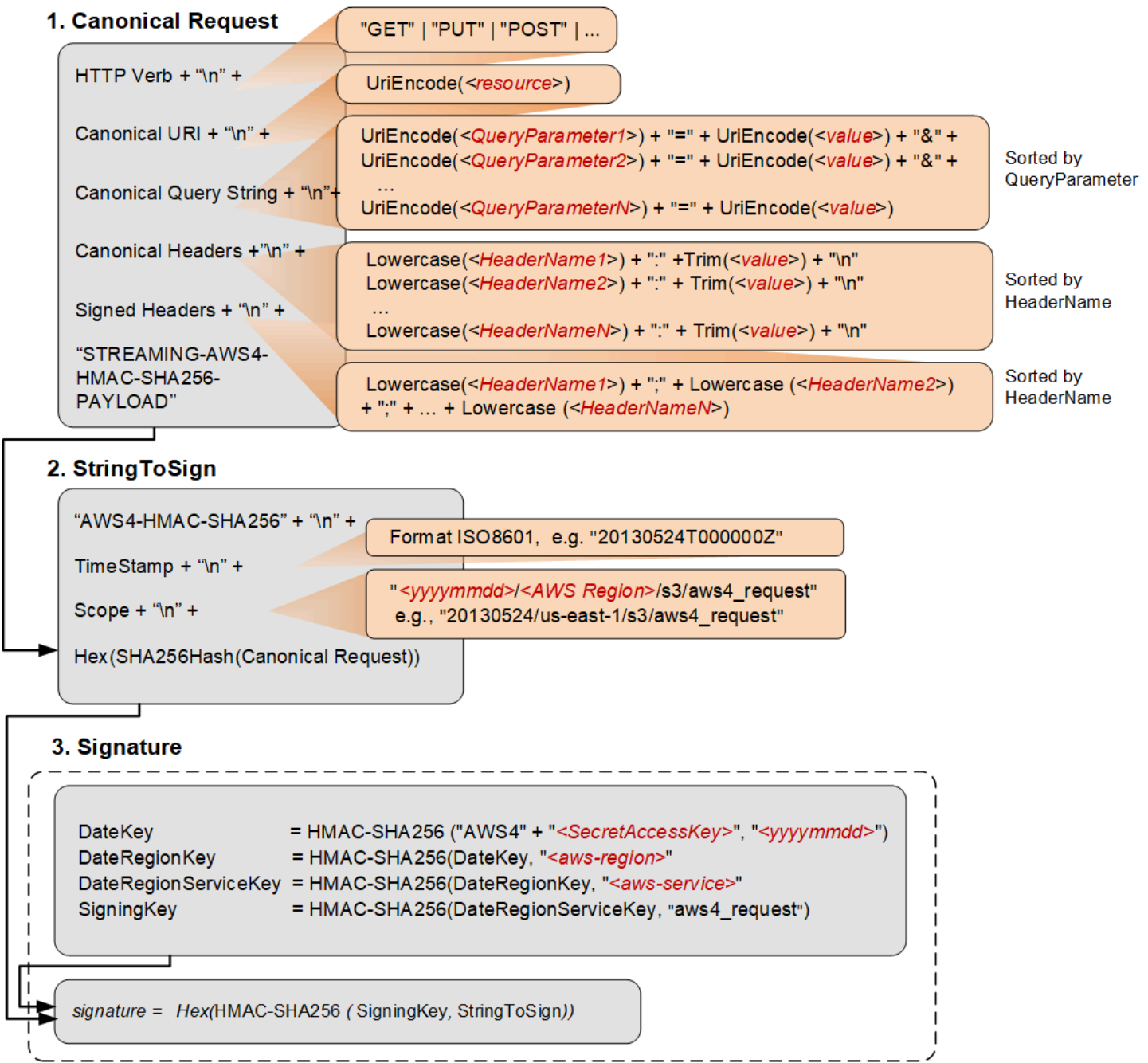
1. Décidez de la taille du bloc de charge utile. Vous en avez besoin lorsque vous écrivez le code.

La taille du bloc doit être d'au moins 8 Ko. Nous recommandons une taille de bloc d'au moins 64 Ko pour de meilleures performances. Cette taille de bloc s'applique à tous les morceaux sauf le dernier. Le dernier bloc que vous envoyez peut être inférieur à 8 Ko. Si votre charge utile est petite et peut tenir dans un seul bloc, elle peut être inférieure aux 8 Ko.

2. Créez la signature de départ à inclure dans le premier bloc. Pour de plus amples informations, veuillez consulter [Calcul de la signature de la graine](#).
3. Créez le premier morceau et diffusez-le. Pour de plus amples informations, veuillez consulter [Définition du corps du morceau](#).
4. Pour chaque segment suivant, calculez la signature du segment qui inclut la signature précédente dans la chaîne que vous signez, construisez le morceau et envoyez-le. Pour de plus amples informations, veuillez consulter [Définition du corps du morceau](#).
5. Envoyez le dernier bloc supplémentaire, qui est le même que les autres morceaux de la construction, mais il ne contient aucun octet de données. Pour de plus amples informations, veuillez consulter [Définition du corps du morceau](#).

Calcul de la signature de la graine

Le schéma suivant illustre le processus de calcul de la signature initiale.



Le tableau suivant décrit les fonctions présentées dans le schéma. Vous devez implémenter du code pour ces fonctions.

Fonction	Description
Lowercase()	Convertit la chaîne en minuscules.
Hex()	Encodage en minuscules en base 16.

Fonction	Description
SHA256Hash()	Fonction de hachage cryptographique Secure Hash Algorithm (SHA, algorithme de hachage sécurisé).
HMAC-SHA256()	Calcule HMAC à l'aide de l'algorithme SHA256 avec la clé de signature fournie. Il s'agit de la signature finale.
Trim()	Supprimez tous les espaces blancs de début ou de fin.

Fonction	Description
<code>UriEncode()</code>	L'URI code chaque octet. <code>UriEncode()</code> doit appliquer les règles suivantes : • L'URI encode tous les octets sauf les caractères non réservés : A-Z, a-z, 0-9, -, ., _ et ~. • Le caractère d'espace est un caractère réservé qui doit être codé sous la forme « %20 » (et non sous la forme « + »). • Chaque octet encodé par URI est formé par un « % » et la valeur hexadécimale à deux chiffres de l'octet. • Les lettres de la valeur hexadécimale doivent être en majuscules, par exemple « %1A ». • Encodez la barre oblique « / » partout sauf dans le nom de la clé de l'objet. Par exemple, si le nom de clé de l'objet est <code>photos/Jan/sample.jpg</code> , la barre oblique du nom de la clé n'est pas encodée.

 **Important**

Les `UriEncode` fonctions standard fournies par votre plateforme de développement peuvent ne pas fonctionner en raison de différences d'implémentation et de l'ambiguïté associée dans les RFC sous-jacents. Nous vous recommandons d'écrire votre propre `UriEncode` fonction personnalisée pour garantir le bon fonctionnement de votre encodage.

Voici un exemple de fonction `UriEncode ()` en Java.

```
public static String UriEncode(CharSequence input,
    boolean encodeSlash) {
    StringBuilder result = new StringBui
lder();
    for (int i = 0; i < input.length(); i++) {
        char ch = input.charAt(i);
```

Fonction	Description
	<pre> if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Pour plus d'informations sur le processus de signature, consultez [Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en un seul bloc \(AWS Signature \(version 4\)\)](#). Le processus est le même, sauf que la création de CanonicalRequest diffère comme suit :

- Outre les en-têtes de demande que vous prévoyez d'ajouter, vous devez inclure les en-têtes suivants :

En-tête	Description
x-amz-content-sha256	Cet en-tête est obligatoire pour toutes les demandes AWS Signature Version 4. Définissez la valeur sur STREAMING-AWS4-HMAC-SHA256-PAYLOAD pour indiquer que la signature ne couvre que les en-têtes et qu'il n'y a aucune charge utile.

En-tête	Description
Content-Encoding	Définissez la valeur sur <code>aws-chunked</code> . Amazon S3 prend en charge plusieurs valeurs de codage de contenu. Vous pouvez spécifier votre codage de contenu personnalisé lorsque vous utilisez l'API de streaming Signature Version 4. Par exemple : Content-Encoding : aws-chunked,gzip Amazon S3 stocke l'objet obtenu sans la <code>aws-chunked</code> valeur figurant dans l'<code>content-encoding</code> en-tête. Si <code>aws-chunked</code> c'est la seule valeur que vous transmettez dans l'<code>content-encoding</code> en-tête, S3 considère que l'<code>content-encoding</code> en-tête est vide et ne renvoie pas cet en-tête lorsque vous récupérez l'objet.
x-amz-decoded-content-length	Définissez la valeur sur la longueur, en octets, des données à fractionner, sans compter les métadonnées. Par exemple, si vous chargez un fichier de 4 Go, définissez la valeur sur 4294967296. Il s'agit de la taille brute de l'objet à charger (données que vous souhaitez stocker dans Amazon S3).
Content-Length	Définissez la valeur sur la taille réelle du corps HTTP transmis, qui inclut la longueur de vos données (valeur définie pour <code>x-amz-decoded-content-length</code>), ainsi que les métadonnées des segments. Chaque bloc possède des métadonnées, telles que la signature du bloc précédent. Les calculs de blocs sont abordés dans la section suivante. Si vous incluez l'<code>Transfer-Encoding</code> en-tête et que vous spécifiez une valeur autre que <code>identity</code> celle-ci, vous ne devez pas inclure l'<code>Content-Length</code> en-tête.

Vous envoyez le premier bloc avec la signature de départ. Vous devez construire le bloc comme décrit dans la section suivante.

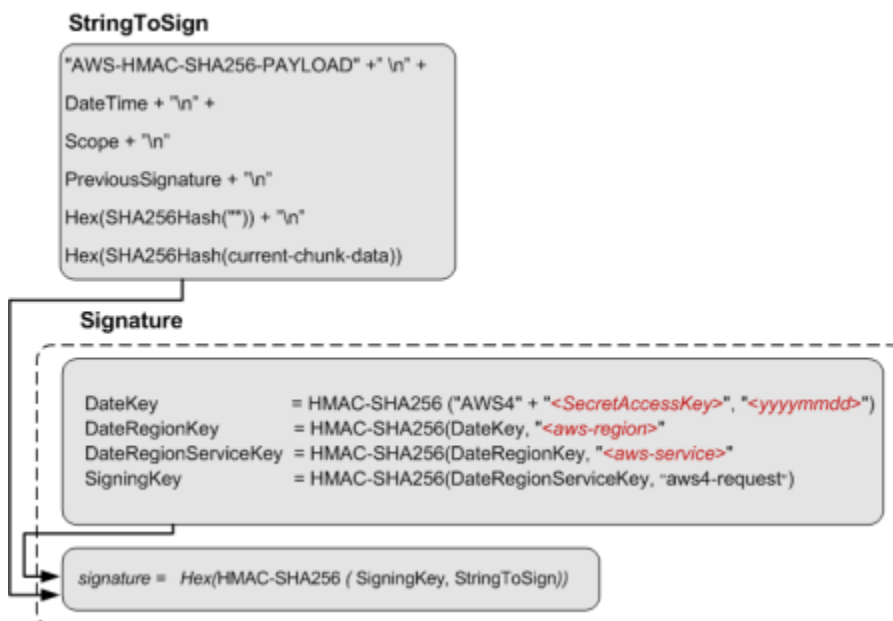
Définition du corps du morceau

Tous les segments incluent des métadonnées. Chaque morceau doit être conforme à la structure suivante :

```
string(IntHexBase(chunk-size)) + ";chunk-signature=" + signature + \r\n + chunk-data + \r\n
```

Où :

- `IntHexBase()` est une fonction que vous écrivez pour convertir un entier de la taille d'un bloc en hexadécimal. Par exemple, si la taille du morceau est 65536, la chaîne hexadécimale est « 10000 ».
- *chunk-size* est la taille, en octets, du bloc de données, sans les métadonnées. Par exemple, si vous chargez un objet de 65 Ko et que vous utilisez une taille de bloc de 64 Ko, vous chargez les données en trois blocs : le premier serait de 64 Ko, le second de 1 Ko et le dernier de 0 octet.
- *signature* Pour chaque segment, vous calculez la signature à l'aide de la chaîne de caractères suivante. Pour le premier segment, vous utilisez la signature de départ comme signature précédente.



La taille du bloc final de données que vous envoyez est de 0, bien que le corps du bloc contienne toujours des métadonnées, y compris la signature du bloc précédent.

Exemple : objet PUT

Vous pouvez utiliser les exemples de cette section comme référence pour vérifier les calculs de signature dans votre code. Avant de passer en revue les exemples, prenez note des points suivants :

- Les calculs de signature de ces exemples utilisent les informations d'identification de sécurité suivantes.

Paramètre	Value
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- Tous les exemples utilisent l'horodatage de la requête 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Tous les exemples sont utilisés `examplebucket` comme nom de compartiment.
- Le compartiment est supposé se trouver dans la région de l'Est des États-Unis (Virginie du Nord), et les informations d'identification Scope et les Signing Key calculs sont utilisés `us-east-1` comme spécificateur de région. Pour plus d'informations, consultez [Régions et points de terminaison](#) dans la Référence générale d'Amazon Web Services.
- Vous pouvez utiliser des requêtes de style de chemin ou des requêtes de style hébergées virtuellement. Les exemples suivants utilisent des requêtes de style hébergées virtuellement, par exemple :

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Pour plus d'informations, consultez la section Hébergement [virtuel de compartiments](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

L'exemple suivant envoie une PUT demande pour télécharger un objet. Les calculs de signature supposent ce qui suit :

- Vous chargez un fichier texte de 65 Ko dont le contenu est une chaîne d'un caractère composée de la lettre « a ».
- La taille du bloc est de 64 Ko. Par conséquent, la charge utile est téléchargée en trois blocs, 64 Ko, 1 Ko, et le dernier bloc contenant 0 octet de données de bloc.
- L'objet obtenu porte le nom de `chunkObject.txt`.
- Vous faites une demande `REDUCED_REDUNDANCY` en tant que classe de stockage en ajoutant l'en-tête de `x-amz-storage-class` demande.

Pour plus d'informations sur l'action d'API, consultez [PutObject](#). La syntaxe générale des requêtes est la suivante :

```
PUT /examplebucket/chunkObject.txt HTTP/1.1
Host: s3.amazonaws.com
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
Authorization: SignatureToBeCalculated
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 66560
Content-Length: 66824
<Payload>
```

Les étapes suivantes présentent les calculs de signature.

1. Signature d'origine — Créer une chaîne à signer
 - a. CanonicalRequest

```
PUT
/examplebucket/chunkObject.txt

content-encoding:aws-chunked
content-length:66824
host:s3.amazonaws.com
x-amz-content-sha256:STREAMING-AWS4-HMAC-SHA256-PAYLOAD
x-amz-date:20130524T000000Z
x-amz-decoded-content-length:66560
x-amz-storage-class:REDUCED_REDUNDANCY
```

```
content-encoding;content-length;host;x-amz-content-sha256;x-amz-date;x-amz-
decoded-content-length;x-amz-storage-class
STREAMING-AWS4-HMAC-SHA256-PAYLOAD
```

Dans la requête canonique, la troisième ligne est vide car elle ne contient aucun paramètre de requête. La dernière ligne est la chaîne constante fournie comme valeur de la charge utile hachée, qui doit être identique à la valeur de `x-amz-content-sha256` header

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
cee3fed04b70f867d036f722359b0b1f2f0e5dc0efadbc082b76c4c60e316455
```

Note

Pour plus d'informations sur chacune des lignes de la chaîne à signer, consultez le schéma qui explique le calcul de la signature initiale.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature de la graine

```
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

4. En-tête Authorization

L'en-tête Authorization qui en résulte est le suivant :

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=content-encoding;content-length;host;x-amz-
```

```
content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-
class,Signature=4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

5. Bloc 1 : (65536 octets, avec une valeur 97 pour la lettre « a »)

a. Chaîne en morceaux à signer :

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

Pour plus d'informations sur chaque ligne de la chaîne à signer, consultez le schéma précédent qui montre les différents composants de la chaîne à signer (par exemple, les trois dernières lignes sont `previous-signaturehash("")`, `ethash(current-chunk-data)`).

b. Signature du morceau :

```
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
```

c. Données partielles envoyées :

```
10000;chunk-
signature=ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
<65536-bytes>
```

6. Bloc 2 : (1024 octets, avec une valeur 97 pour la lettre « a »)

a. Chaîne en morceaux à signer :

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

```
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Signature du morceau :

```
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
```

c. Données partielles envoyées :

```
400; chunk-  
signature=0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
<1024 bytes>
```

7. Bloc 3 : (données de 0 octet)

a. Chaîne en morceaux à signer :

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Signature du morceau :

```
b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

c. Données partielles envoyées :

```
0; chunk-  
signature=b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

Calculs de signature pour les en-têtes de fin (téléchargements fragmentés) (AWS Signature (version 4))

Lorsque vous authentifiez des demandes à l'aide de l'Authorization en-tête, vous pouvez également télécharger la charge utile par morceaux. Lorsque vous envoyez les données de l'objet par morceaux, vous avez également la possibilité d'inclure des en-têtes de fin. (Pour plus d'informations, consultez [Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en plusieurs segments \(téléchargement fragmenté\) \(AWS Signature \(version 4\)\)](#).) Cette section décrit les étapes à suivre lorsque vous souhaitez inclure un en-tête de fin à la fin de votre téléchargement de plusieurs blocs.

Important

Lorsque vous incluez des en-têtes de fin, vous devez envoyer les informations suivantes dans votre en-tête initial :

- Vous devez `x-amz-content-sha256` définir une valeur appropriée qui indique qu'une remorque sera incluse. Pour connaître les valeurs acceptables pour `x-amz-content-sha256`, consultez [Authentification des demandes : utilisation de l'en-tête d'autorisation \(AWS Signature \(version 4\)\)](#).
- Vous devez configurer `x-amz-trailer` pour indiquer le contenu que vous incluez dans votre en-tête final.

Les en-têtes de fin ne sont envoyés qu'une fois que les morceaux ont été chargés. Les segments précédents sont envoyés normalement et signés comme décrit dans les sections précédentes, y compris l'envoi du dernier bloc avec une charge utile de 0 octet. Les en-têtes de fin sont inclus dans leur propre bloc et envoyés après le dernier bloc avec une charge utile de 0 octet. Par exemple, si vos données se terminaient par un bloc de 100 Ko, vous enverriez les informations suivantes :

- Morceaux de données précédents
- Bloc final de 100 Ko de l'objet
- Bloc de 0 octet indiquant la fin de l'objet
- Bloc d'en-têtes de fin

Exemples : vérification des calculs de signature

Vous pouvez utiliser les exemples de cette section comme référence pour vérifier les calculs de signature dans votre code. Avant de passer en revue les exemples, prenez note des points suivants :

- Les calculs de signature de ces exemples utilisent les informations d'identification de sécurité suivantes.

Paramètre	Value
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- Tous les exemples utilisent l'horodatage de la requête 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Tous les exemples sont utilisés `examplebucket` comme nom de compartiment.
- Le compartiment est supposé se trouver dans la région de l'Est des États-Unis (Virginie du Nord), et les informations d'identification Scope et les Signing Key calculs sont utilisés `us-east-1` comme spécificateur de région. Pour plus d'informations, consultez [Régions et points de terminaison](#) dans la Référence générale Amazon Web Services.
- Vous pouvez utiliser des requêtes de style de chemin ou des requêtes de style hébergées virtuellement. Les exemples suivants utilisent des requêtes de style hébergées virtuellement, par exemple :

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Pour plus d'informations, consultez la section Hébergement [virtuel de compartiments](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

L'exemple suivant envoie une PUT demande pour télécharger un objet. Les calculs de signature supposent ce qui suit :

- Vous chargez un fichier texte de 65 Ko dont le contenu est une chaîne d'un caractère composée de la lettre « a ».

- La taille du bloc est de 64 Ko. Par conséquent, la charge utile est téléchargée en trois blocs, 64 Ko, 1 Ko, et le dernier bloc contenant 0 octet de données de bloc.
- L'objet obtenu porte le nom de `chunkObject.txt`.
- Vous faites une demande `REDUCED_REDUNDANCY` en tant que classe de stockage en ajoutant l'en-tête de `x-amz-storage-class` demande.
- Le transfert inclut une valeur de somme de contrôle CRC32C en tant qu'en-tête de fin.

Pour plus d'informations sur l'action de l'API, consultez [PutObject](#). La syntaxe générale des requêtes est la suivante :

```
PUT /examplebucket/chunkObject.txt HTTP/1.1
Host: s3.amazonaws.com
x-amz-date: 20130524T000000Z
x-amz-storage-class: REDUCED_REDUNDANCY
Authorization: SignatureToBeCalculated
x-amz-content-sha256: STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
Content-Encoding: aws-chunked
x-amz-decoded-content-length: 66560
x-amz-trailer: x-amz-checksum-crc32c
Content-Length: 66946
<Payload>
```

Les étapes suivantes présentent les calculs de signature.

1. Signature d'origine — Créer une chaîne à signer

a. CanonicalRequest

```
PUT
/examplebucket/chunkObject.txt

content-encoding:aws-chunked
host:s3.amazonaws.com
x-amz-content-sha256:STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
x-amz-date:20130524T000000Z
x-amz-decoded-content-length:66560
x-amz-storage-class:REDUCED_REDUNDANCY
x-amz-trailer:x-amz-checksum-crc32c
```

```
content-encoding;host;x-amz-content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-class;x-amz-trailer
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER
```

Dans la requête canonique, la troisième ligne est vide car elle ne contient aucun paramètre de requête. La dernière ligne est la chaîne constante fournie comme valeur de la charge utile hachée, qui doit être identique à la valeur de `x-amz-content-sha256` header

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
44d48b8c2f70eae815a0198cc73d7a546a73a93359c070abbaa5e6c7de112559
```

Note

Pour plus d'informations sur chacune des lignes de la chaîne à signer, consultez le schéma qui explique le calcul de la signature initiale.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature de la graine

```
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

4. En-tête Authorization

L'en-tête Authorization qui en résulte est le suivant :

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/
aws4_request,SignedHeaders=content-encoding;content-length;host;x-amz-
```

```
content-sha256;x-amz-date;x-amz-decoded-content-length;x-amz-storage-
class,Signature=106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

5. Bloc 1 : (65536 octets, avec une valeur 97 pour la lettre « a »)

a. Chaîne en morceaux à signer :

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

Pour plus d'informations sur chaque ligne de la chaîne à signer, consultez le schéma de la rubrique précédente ([Calcul de la signature de la graine](#)) qui montre les différents composants de la chaîne à signer. Par exemple, les trois dernières lignes sont les suivantes :

- *previous-signature*
- `hash("")`
- `hash(current-chunk-data)`

b. Signature du morceau :

```
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
```

c. Données partielles envoyées :

```
10000;chunk-
signature=b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
<65536-bytes>
```

6. Bloc 2 : (1024 octets, avec une valeur 97 pour la lettre « a »)

a. Chaîne en morceaux à signer :

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Signature du morceau :

```
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
```

c. Données partielles envoyées :

```
400;chunk-
signature=1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
<1024-bytes>
```

7. Bloc 3 : (données de 0 octet)

a. Chaîne en morceaux à signer :

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Signature du morceau :

```
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

c. Données partielles envoyées :

```
0;chunk-
signature=2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

8. Partie 4 : En-têtes de fin

a. Chaîne en forme de bande annonce à signer :

```

AWS4-HMAC-SHA256-TRAILER
20130524T000000Z
20130524/us-east-1/s3/aws4_request
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
1e376db7e1a34a8ef1c4bcee131a2d60a1cb62503747488624e10995f448d774

```

 Note

Les deux dernières lignes sont `previous-signature` (la signature du bloc de données de 0 octet), hash (*`trailing-checksum-header-name: base64-encoded-trailing-checksum-value\n`*).

Le hachage est calculé comme suit, sans espace :

- Le nom de l'en-tête de la somme de contrôle de fin
- Un deux-points (:)
- La valeur de la somme de contrôle finale codée en base64
- Un caractère de nouvelle ligne (\n).

Dans cet exemple, où vous utilisez l'algorithme de `crc32c` hachage, avec la valeur de somme de contrôle codée en base64 `s008/Q==`, vous pouvez représenter le calcul comme suit : `hash( 'x-amz-checksum-crc32c:s008/Q==\n' )`

b. Signature du morceau :

```
d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

c. Données partielles envoyées :

```

x-amz-checksum-crc32c:s008/Q==
x-amz-trailer-
signature:d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435

```

Authentification des demandes : utilisation des paramètres de requête (AWS Signature (version 4))

Comme décrit dans la présentation de l'authentification (voir [Méthodes d'authentification](#)), vous pouvez fournir des informations d'authentification à l'aide de paramètres de chaîne de requête. L'utilisation de paramètres de requête pour authentifier les demandes est utile lorsque vous souhaitez exprimer une demande entièrement dans une URL. Cette méthode est également appelée présignature d'une URL.

Un scénario d'utilisation pour les URL présignées est que vous pouvez accorder un accès temporaire à vos ressources Amazon S3. Par exemple, vous pouvez intégrer une URL présignée sur votre site Web ou l'utiliser dans un client de ligne de commande (tel que Curl) pour télécharger des objets.

Note

Vous pouvez également utiliser le AWS CLI pour créer des URL présignées. Pour plus d'informations, consultez [presign](#) dans la Référence des commandes de l'AWS CLI .

Voici un exemple d'URL présignée.

```
https://examplebucket.s3.amazonaws.com/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=<your-access-key-id>/20130721/us-east-1/s3/aws4_request
&X-Amz-Date=20130721T201207Z
&X-Amz-Expires=86400
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

Dans l'exemple d'URL, notez ce qui suit :

- Les entrées de ligne sont ajoutées pour des raisons de lisibilité.
- La valeur X-Amz-Credential dans l'URL affiche le caractère « / » uniquement pour des raisons de lisibilité. En pratique, il doit être codé au format %2F. Par exemple :

```
&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request
```

Le tableau suivant décrit les paramètres de requête de l'URL qui fournissent des informations d'authentification.

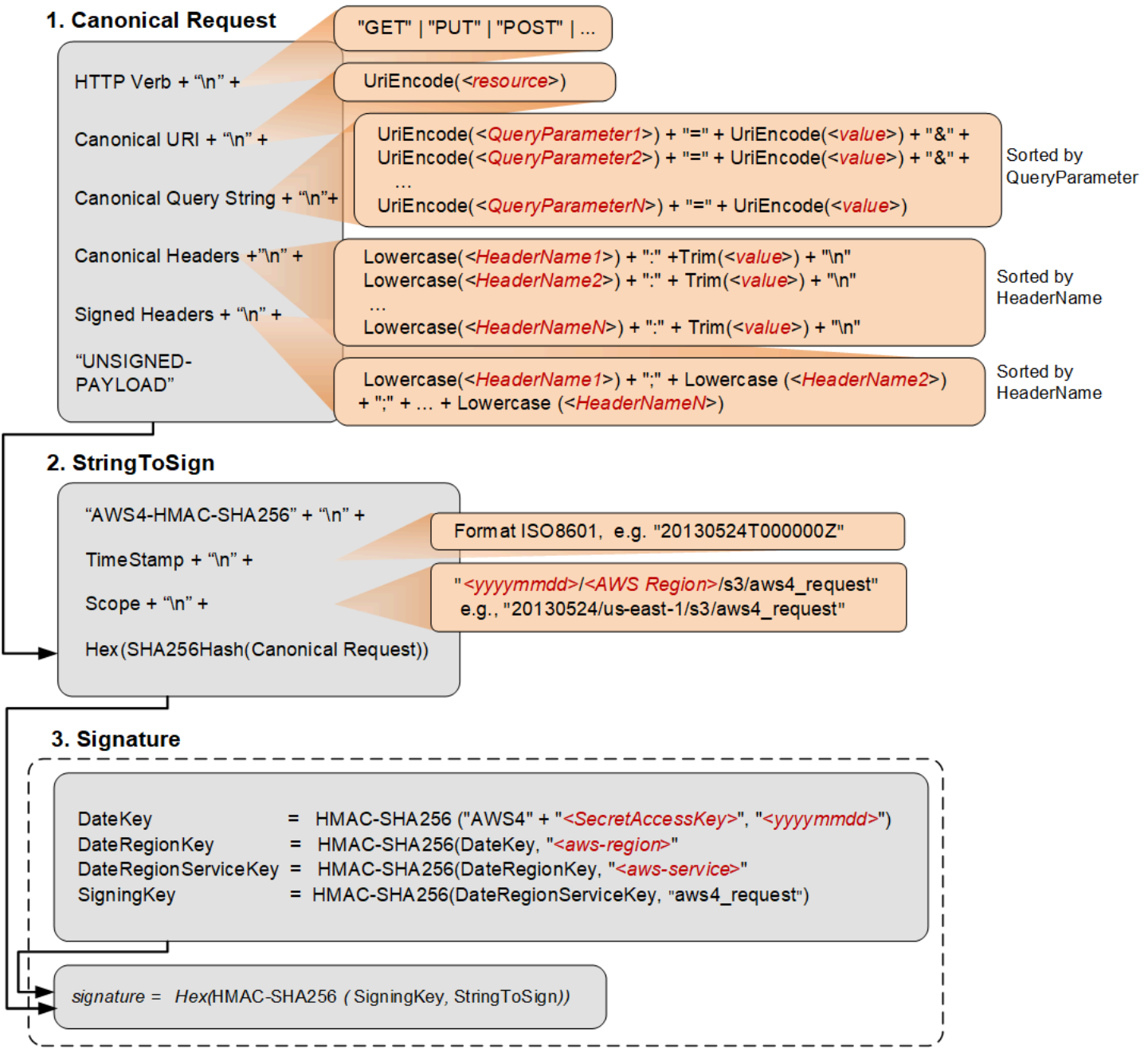
Nom du paramètre de la chaîne d'interrogation	Exemple de valeur
X-Amz-Algorithm	Identifie la version de AWS Signature et l'algorithme que vous avez utilisé pour calculer la signature. Pour AWS la version 4 de Signature, vous définissez la valeur de ce paramètre sur <code>AWS4-HMAC-SHA256</code>. Cette chaîne identifie AWS la version 4 de Signature (AWS 4) et l' HMAC-SHA256 algorithme (HMAC-SHA256).
X-Amz-Credential	Outre l'ID de votre clé d'accès, ce paramètre fournit également l'étendue (Région AWS et le service) pour lesquels la signature est valide. Cette valeur doit correspondre au champ d'application que vous utilisez dans les calculs de signature, comme indiqué dans la section suivante. Le format général de cette valeur de paramètre est le suivant : <your-access-key-id> /<date>/<Région AWS>/<AWS-service> /aws4_request Par exemple : AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request Pour Amazon S3, la <code>AWS-service</code> chaîne est <code>s3</code>. Pour une liste des AWS-region chaînes S3, voir Régions et points de terminaison dans le Références générales AWS.
X-Amz-Date	

Nom du paramètre de la chaîne d'interrogation	Exemple de valeur
	Le format de date et d'heure doit respecter la norme ISO 8601 et doit être formaté avec le format « <i>yyyyMMdd T HHmmss Z</i> ». Par exemple, si la date et l'heure étaient « 08/01 /2016 15:32:41.982-700 », elles doivent d'abord être converties en UTC (temps universel coordonné), puis soumises sous la forme « 20160801T223241Z ».
X-Amz-Expires	Indique la période, en secondes, pendant laquelle l'URL présignée générée est valide. Par exemple, 86400 (24 heures). Cette valeur est un nombre entier. La valeur minimale que vous pouvez définir est 1 et la valeur maximale est de 604800 (sept jours). Une URL pré-signée peut être valide pendant sept jours au maximum, car la clé de signature que vous utilisez pour le calcul des signatures est valide pendant sept jours au maximum.
X-Amz-SignedHeaders	Répertorie les en-têtes que vous avez utilisés pour calculer la signature. Les en-têtes suivants sont obligatoires pour les calculs de signature : • L'host-en-tête HTTP. • Tous x-amz-* les en-têtes que vous prévoyez d'ajouter à la demande.  Note Pour plus de sécurité, vous devez signer tous les en-têtes de demande que vous prévoyez d'inclure dans votre demande.

Nom du paramètre de la chaîne d'interrogation	Exemple de valeur
X-Amz-Signature	Fournit la signature pour authentifier votre demande. Cette signature doit correspondre à la signature calculée par Amazon S3 ; sinon, Amazon S3 refuse la demande. Par exemple, 733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7 Les calculs de signature sont décrits dans la section suivante.
X-Amz-Security-Token	Paramètre d'information d'identification facultatif si vous utilisez des informations d'identification provenant du service STS.

Calculer une signature

Le schéma suivant illustre le processus de calcul de la signature.



Le tableau suivant décrit les fonctions présentées dans le schéma. Vous devez implémenter du code pour ces fonctions.

Fonction	Description
Lowercase()	Convertit la chaîne en minuscules.
Hex()	Encodage en minuscules en base 16.

Fonction	Description
SHA256Hash()	Fonction de hachage cryptographique Secure Hash Algorithm (SHA, algorithme de hachage sécurisé).
HMAC-SHA256()	Calcule HMAC à l'aide de l'algorithme SHA256 avec la clé de signature fournie. Il s'agit de la signature finale.
Trim()	Supprimez tous les espaces blancs de début ou de fin.

Fonction	Description
<code>UriEncode()</code>	L'URI code chaque octet. <code>UriEncode()</code> doit appliquer les règles suivantes : • L'URI encode tous les octets sauf les caractères non réservés : A-Z, a-z, 0-9, -, ., _ et ~. • Le caractère d'espace est un caractère réservé qui doit être codé sous la forme « %20 » (et non sous la forme « + »). • Chaque octet encodé par URI est formé par un « % » et la valeur hexadécimale à deux chiffres de l'octet. • Les lettres de la valeur hexadécimale doivent être en majuscules, par exemple « %1A ». • Encodez la barre oblique « / » partout sauf dans le nom de la clé de l'objet. Par exemple, si le nom de clé de l'objet est <code>photos/Jan/sample.jpg</code> , la barre oblique du nom de la clé n'est pas encodée.

 **Important**

Les `UriEncode` fonctions standard fournies par votre plateforme de développement peuvent ne pas fonctionner en raison de différences d'implémentation et de l'ambiguïté associée dans les RFC sous-jacents. Nous vous recommandons d'écrire votre propre `UriEncode` fonction personnalisée pour garantir le bon fonctionnement de votre encodage.

Voici un exemple de fonction `UriEncode ()` en Java.

```
public static String UriEncode(CharSequence input,
    boolean encodeSlash) {
    StringBuilder result = new StringBui
lder();
    for (int i = 0; i < input.length(); i++) {
        char ch = input.charAt(i);
```

Fonction	Description
	<pre> if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Pour plus d'informations sur le processus de signature (détails sur la création d'une demande canonique, chaîne à signer et calculs de signature), consultez [Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en un seul bloc \(AWS Signature \(version 4\)\)](#). Le processus est généralement le même, sauf que la création de `CanonicalRequest` dans une URL présignée diffère comme suit :

- Vous n'incluez pas de hachage de charge utile dans la requête canonique, car lorsque vous créez une URL présignée, vous ne connaissez pas le contenu de la charge utile car l'URL est utilisée pour télécharger une charge utile arbitraire. Au lieu de cela, vous utilisez une chaîne constante `UNSIGNED-PAYLOAD`.
- La chaîne de requête canonique doit inclure tous les paramètres de requête du tableau précédent, à l'exception `X-Amz-Signature` de.
- Pour S3, vous devez inclure le paramètre de `X-Amz-Security-Token` requête dans l'URL si vous utilisez des informations d'identification provenant du service STS.
- Les en-têtes canoniques doivent inclure l'en-tête `HTTPHost`. Si vous prévoyez d'inclure l'un des `x-amz-*` en-têtes, ces en-têtes doivent également être ajoutés pour le calcul de la signature. Vous pouvez éventuellement ajouter tous les autres en-têtes que vous prévoyez d'inclure dans votre demande. Pour plus de sécurité, vous devez signer autant d'en-têtes que possible. Si vous ajoutez un en-tête signé qui est également un paramètre de requête signé et que leur valeur diffère, vous recevrez un `InvalidRequest` message d'erreur car l'entrée est en conflit.

Un exemple

Supposons que vous ayez un objet `test.txt` dans votre `examplebucket` seau. Vous souhaitez partager cet objet avec d'autres personnes pendant 24 heures (86 400 secondes) en créant une URL présignée.

```
https://examplebucket.s3.amazonaws.com/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request
&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

Les étapes suivantes illustrent d'abord les calculs de signature, puis la construction de l'URL présignée. L'exemple fait les hypothèses supplémentaires suivantes :

- L'horodatage de la demande est. `Fri, 24 May 2013 00:00:00 GMT`
- Le compartiment se trouve dans la région USA Est (Virginie du Nord), et les informations d'identification `Scope` et les `Signing Key` calculs sont utilisés `us-east-1` comme spécificateur de région. Pour de plus amples informations, veuillez consulter [Régions et points de terminaison](#) dans le manuel *Références générales AWS*.

Vous pouvez utiliser cet exemple comme scénario de test pour vérifier la signature calculée par votre code ; toutefois, vous devez utiliser le même nom de compartiment, la même clé d'objet, le même horodatage et les exemples d'informations d'identification suivants :

Paramètre	Value
<code>AWSSecretAccessKeyId</code>	<code>AKIAIOSFODNN7EXAMPLE</code>
<code>AWSSecretAccessKey</code>	<code>wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY</code>

1. StringToSign

a. CanonicalRequest

```
GET
/test.txt
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
host:examplebucket.s3.amazonaws.com

host
UNSIGNED-PAYLOAD
```

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
3bfa292879f6447bbcd47001decf97f4a54dc650c8942174ae0a9121cf58ad04
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signature

```
aeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```

Vous disposez maintenant de toutes les informations nécessaires pour créer une URL présignée. L'URL qui en résulte pour cet exemple s'affiche comme suit (vous pouvez l'utiliser pour comparer votre URL présignée) :

```
https://examplebucket.s3.amazonaws.com/test.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=aeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```


- **Objet PUT** : des exemples distincts illustrent à la fois le téléchargement de la charge utile complète en une seule fois et le téléchargement de la charge utile par morceaux. Pour plus d'informations sur l'utilisation de l'en-tête `Authorization` pour l'authentification, consultez [Authentification des demandes : utilisation de l'en-tête d'autorisation \(AWS Signature \(version 4\)\)](#).
- **OBJET GET** — Cet exemple génère une URL présignée pour obtenir un objet. Les paramètres de requête fournissent la signature et d'autres informations d'authentification. Les utilisateurs peuvent coller une URL présignée dans leur navigateur pour récupérer l'objet, ou vous pouvez utiliser l'URL pour créer un lien cliquable. Pour plus d'informations sur l'utilisation des paramètres de requête pour l'authentification, consultez [Authentification des demandes : utilisation des paramètres de requête \(AWS Signature \(version 4\)\)](#).

Le reste de cette section décrit les exemples en Java et en C#. Les rubriques incluent des instructions pour télécharger les exemples et les exécuter.

Exemples de calcul de signature à l'aide de Java (AWS Signature (version 4))

L'exemple Java qui montre le calcul des signatures peut être téléchargé à l'adresse <https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AWSS3SigV4JavaSamples.zip>. Notez que ces exemples utilisent le AWS SDK pour Java v1, et nous vous recommandons d'utiliser le [AWS SDK pour Java v2](#) pour les nouvelles applications. Dans `RunAllSamples.java`, la `main()` fonction exécute des exemples de requêtes pour créer un objet, récupérer un objet et créer une URL présignée pour l'objet. L'exemple crée un objet à partir de la chaîne de texte fournie dans le code :

```
PutS3ObjectSample.putS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
GetS3ObjectSample.getS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
PresignedUrlSample.getPresignedUrlToS3Object(bucketName, regionName, awsAccessKey,
    awsSecretKey);
PutS3ObjectChunkedSample.putS3ObjectChunked(bucketName, regionName, awsAccessKey,
    awsSecretKey);
```

Pour tester les exemples sur un Linux-based ordinateur

Les instructions suivantes concernent le système d'exploitation Linux.

1. Dans un terminal, accédez au répertoire qui contient `AWSS3SigV4JavaSamples.zip`.
2. Extrayez le fichier `.zip`.

3. Ouvrez le fichier `./com/amazonaws/services/s3/samples/RunAllSamples.java` dans un éditeur de texte. Mettez à jour le code avec les informations suivantes :

- Le nom du compartiment dans lequel le nouvel objet peut être créé.

 Note

Les exemples utilisent une requête de style hébergée virtuellement pour accéder au compartiment. Pour éviter d'éventuelles erreurs, assurez-vous que le nom de votre compartiment est conforme aux règles de dénomination des compartiments, comme expliqué dans [la section Restrictions et limitations des compartiments](#) du guide de l'utilisateur d'Amazon Simple Storage Service.

- Région AWS où se trouve le seau.

Si le bucket se trouve dans la région USA Est (Virginie du Nord), utilisez `us-east-1` pour spécifier la région. Pour en obtenir la liste des autres Régions AWS, accédez à [Amazon Simple Storage Service \(S3\)](#) dans le Références générales AWS.

4. Compilez le code source et stockez les classes compilées dans le `bin/` répertoire.

```
javac -d bin -source 6 -verbose com
```

5. Remplacez le répertoire par `bin/`, puis exécutez `RunAllSamples`.

```
java com.amazonaws.services.s3.sample.RunAllSamples
```

Le code exécute toutes les méthodes `main()`. Pour chaque demande, la sortie affichera la demande canonique, la chaîne à signer et la signature.

Exemples de calculs de signature en C# (AWS Signature (version 4))

L'exemple C# qui montre le calcul des signatures peut être téléchargé à l'adresse https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip. Notez que ces exemples utilisent le AWS SDK pour .NET v1, et nous vous recommandons d'utiliser le [AWS SDK pour .NET v3](#) pour les nouvelles applications. Dans `Program.cs`, la `main()` fonction exécute

des exemples de requêtes pour créer un objet, récupérer un objet et créer une URL présignée pour l'objet. Le code pour le calcul des signatures se trouve dans le \Signers dossier.

```
PutS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n*****");
PutS3ObjectChunkedSample.Run(awsRegion, bucketName, "MySampleFileChunked.txt");

Console.WriteLine("\n\n*****");
GetS3ObjectSample.Run(awsRegion, bucketName, "MySampleFile.txt");

Console.WriteLine("\n\n*****");
PresignedUrlSample.Run(awsRegion, bucketName, "MySampleFile.txt");
```

Pour tester les exemples avec Microsoft Visual Studio 2010 ou version ultérieure

1. Extrayez le fichier .zip.
2. Démarrez Visual Studio, puis ouvrez le fichier .sln.
3. Mettez à jour le App.config fichier avec des informations d'identification de sécurité valides.
4. Mettez à jour le code comme suit :
 - Dans Program.cs, indiquez le nom du Région AWS compartiment et son emplacement.
L'exemple crée un objet dans ce compartiment.
5. Exécutez le code.
6. Pour vérifier que l'objet a été créé, copiez l'URL présignée créée par le programme, puis collez-la dans une fenêtre de navigateur.

Demandes d'authentification : Browser-Based téléchargements à l'aide de POST (AWS Signature (version 4))

Amazon S3 prend en charge les requêtes HTTP POST afin que les utilisateurs puissent télécharger du contenu directement sur Amazon S3. L'utilisation de HTTP POST pour charger du contenu simplifie les chargements et réduit la latence de chargement lorsque les utilisateurs téléchargent des données à stocker dans Amazon S3. Cette section décrit comment authentifier les requêtes HTTP POST. Pour plus d'informations sur les requêtes HTTP POST, la création d'un formulaire, la création d'une politique POST et un exemple, consultez [Browser-Based Téléversements à l'aide de POST \(AWS Signature \(version 4\)\)](#).

Pour authentifier une requête HTTP POST, procédez comme suit :

1. Le formulaire doit inclure les champs suivants pour fournir la signature et les informations pertinentes qu'Amazon S3 peut utiliser pour recalculer la signature à la réception de la demande :

Nom d'élément	Description
policy	La politique Base64-encoded de sécurité qui décrit ce qui est autorisé dans la demande. Pour le calcul des signatures, cette règle correspond à la chaîne que vous signez. Amazon S3 doit appliquer cette politique pour pouvoir recalculer la signature.
x-amz-algorithm	Algorithme de signature utilisé. Pour AWS la version 4 de Signature, la valeur est <code>AWS4-HMAC-SHA256</code> .
x-amz-credential	Outre l'ID de votre clé d'accès, cela fournit les informations de portée que vous avez utilisées pour calculer la clé de signature pour le calcul de la signature. Il s'agit d'une chaîne de la forme suivante : <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Par exemple : <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .</pre> Pour Amazon S3, la chaîne <code>aws-service</code> est <code>s3</code> Pour obtenir la liste des <code>aws-region</code> chaînes Amazon S3, consultez la section Régions et points de terminaison dans le Références générales AWS.
x-amz-date	

Nom d'élément	Description
	Il s'agit de la valeur de date au format ISO8601. Par exemple, 20130728T000000Z . Il s'agit de la même date que celle que vous avez utilisée pour créer la clé de signature. Il doit également s'agir de la même valeur que celle que vous avez fournie dans la politique (x-amz-date) que vous avez signée.
x-amz-signature	(AWS Signature Version 4) Le HMAC-SHA256 hachage de la politique de sécurité. Pour plus d'informations sur les options de signature , voir Ajouter la signature à la requête HTTP dans le Références générales AWS.

2. La politique POST doit inclure les éléments suivants :

Nom d'élément	Description
x-amz-algorithm	Algorithme de signature que vous avez utilisé pour calculer la signature. Pour AWS la version 4 de Signature, la valeur estAWS4-HMAC-SHA256 .
x-amz-credential	Outre l'ID de votre clé d'accès, cela fournit les informations de portée que vous avez utilisées pour calculer la clé de signature pour le calcul de la signature. Il s'agit d'une chaîne de la forme suivante : <i><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</i> Par exemple,

Nom d'élément	Description
	AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .
x-amz-date	La valeur de date spécifiée dans la chaîne au format ISO8601. Par exemple, « 20130728T000000Z ». La date doit être la même que celle que vous avez utilisée lors de la création de la clé de signature pour le calcul de la signature.

3. Pour le calcul des signatures, la politique POST est la chaîne à signer.

Calculer une signature

Le schéma suivant illustre le processus de calcul de la signature.



Pour calculer une signature

1. Créez une politique à l'aide de l' UTF-8 encodage.
2. Convertissez la UTF-8-encoded politique en Base64. Le résultat est la chaîne à signer.
3. Créez la signature sous forme de HMAC-SHA256 hachage de la chaîne à signer. Vous fournirez la clé de signature comme clé de la fonction de hachage.
4. Encodrez la signature à l'aide d'un codage hexadécimal.

Pour plus d'informations sur la création de formulaires HTML, les politiques de sécurité et un exemple, consultez les sous-rubriques suivantes :

- [Création d'un formulaire HTML \(à l'aide de AWS Signature \(version 4\)\)](#)
- [Politique POST](#)
- [Exemple : Browser-Based téléchargement à l'aide de HTTP POST \(à l'aide de AWS Signature \(version 4\)\)](#)

Clés de politique d'authentification spécifiques à Amazon S3 Signature version 4

Le tableau suivant présente les clés de politique liées à l'authentification Amazon S3 Signature Version 4 qui peuvent figurer dans les politiques Amazon S3. Dans une stratégie de compartiment, vous pouvez ajouter ces conditions pour imposer un comportement spécifique lorsque les requêtes sont authentifiées en utilisant Signature Version 4. Pour obtenir des exemples de politiques, consultez [Exemples de politiques relatives aux compartiments utilisant des clés de condition associées à Signature Version 4](#).

Clés applicables	Description
s3:signatureversion	Identifie la version de AWS Signature que vous souhaitez prendre en charge pour les demandes authentifiées. Pour les demandes authentifiées, Amazon S3 prend en charge Signature Version 4 et Signature Version 2. Vous pouvez ajouter cette condition à votre

Clés applicables	Description
	politique de bucket pour exiger une version de signature spécifique. Valeurs valides : "AWS" identifie la version 2 de Signature "AWS4-HMAC-SHA256" identifie la version 4 de Signature
s3:authType	Amazon S3 prend en charge différentes méthodes d'authentification (voir Demandes d'authentification (AWS Signature (version 4))). Vous pouvez éventuellement utiliser cette clé de condition pour limiter les demandes entrantes à l'aide d'une méthode d'authentification spécifique. Par exemple, vous pouvez autoriser uniquement l'utilisation de l'Authorization en-tête HTTP pour l'authentification des demandes. Valeurs valides : REST-HEADER REST-QUERY-STRING POST

Clés applicables	Description
s3:signatureAge	La durée, en millisecondes, pendant laquelle une signature est valide dans une demande authentifiée. Cette condition fonctionne pour : • URL présignées : lorsque la condition la plus restrictive l'emporte. Pour plus d'informations, consultez la section Utilisation d'URL présignées. • POST présigné : chargez des fichiers directement sur S3 à l'aide du POST présigné. Pour plus d'informations, consultez la politique POST d'Amazon S3. Dans la version 2 de Signature, cette valeur est toujours définie sur 0. Dans Signature Version 4, la clé de signature est valide pendant sept jours au maximum. Par conséquent, les signatures ne restent valides que pendant sept jours. Vous pouvez utiliser cette condition pour limiter davantage la durée de la signature. Pour de plus amples informations, veuillez consulter Présentation des demandes de signature. Exemple de valeur : 100

Clés applicables	Description
s3:x-amz-content-sha256	Vous pouvez utiliser cette clé de condition pour interdire les contenus non signés dans votre compartiment. Lorsque vous utilisez Signature Version 4, pour les requêtes qui utilisent l'en-tête <code>Authorization</code>, vous ajoutez l'en-tête <code>x-amz-content-sha256</code> dans le calcul de signature, puis définissez sa valeur sur la charge utile du hachage. Notez que cette clé de condition ne prend pas en charge l'<code>x-amz-content-sha256</code> en-tête en tant que paramètre de chaîne de requête. Vous pouvez utiliser cette clé de condition dans votre politique de bucket pour refuser tout téléchargement dont les charges utiles ne sont pas signées. Par exemple, vous pouvez refuser les chargements qui utilisent l'<code>Authorization</code> en-tête pour authentifier les demandes mais ne signent pas la charge utile. Pour de plus amples informations, veuillez consulter Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en un seul bloc (AWS Signature (version 4)). Valeur valide : UNSIGNED-PAYLOAD

Exemples de politiques relatives aux compartiments utilisant des clés de condition associées à Signature Version 4

La politique de compartiment suivante refuse toute demande d'URL présignée Amazon S3 sur des objets `examplebucket` dont la signature date de plus de dix minutes.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Deny a presigned URL request if the signature is more than 10 min old",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket3/*",
      "Condition": {
        "NumericGreaterThan": {
          "s3:signatureAge": 600000
        }
      }
    }
  ]
}
```

La stratégie de compartiment suivante autorise uniquement les requêtes qui utilisent l'en-tête `Authorization` pour l'authentification des demandes. Toute demande POST ou URL présignée sera refusée.

JSON

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Allow only requests that use Authorization header for request authentication. Deny POST or presigned URL requests.",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::examplebucket3/*",
      "Condition": {
        "StringNotEquals": {
          "s3:authType": "REST-HEADER"
        }
      }
    }
  ]
}
```

```

    }
  }
]
}

```

La politique de bucket suivante refuse les demandes qui utilisent des URL présignées pour l'authentification des demandes :

JSON

```

{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"DenyUploadsUsingPresignedURL",
      "Effect":"Deny",
      "Principal":"*",
      "Action":"s3:*",
      "Resource":"arn:aws:s3:::amzn-s3-demo-bucket1/*",
      "Condition":{"
        "StringEquals":{"
          "s3:authType":"REST-query-string"
        }
      }
    }
  ]
}

```

La politique de bucket suivante interdit tout téléchargement contenant des charges utiles non signées :

JSON

```

{
  "Version":"2012-10-17",
  "Statement": [
    {

```

```
        "Sid": "Deny uploads with unsigned payloads that use the  
Authorization header.",  
        "Effect": "Deny",  
        "Principal": "*",  
        "Action": "s3:*",  
        "Resource": "arn:aws:s3:::examplebucket3/*",  
        "Condition": {  
            "StringEquals": {  
                "s3:x-amz-content-sha256": "UNSIGNED-PAYLOAD"  
            }  
        }  
    }  
]  
}
```

Browser-Based Téléversements à l'aide de POST (AWS Signature (version 4))

Cette section explique comment charger des fichiers directement sur Amazon S3 via un navigateur à l'aide de requêtes HTTP POST. Il contient également des informations sur l'utilisation de la JavaScript bibliothèque AWS Amplify pour le téléchargement de fichiers depuis un navigateur vers Amazon S3.

Rubriques

- [POST Object](#)
- [POST Object restore](#)
- [Browser-Based Téléversements à l'aide de HTTP POST](#)
- [Calculer une signature](#)
- [Création d'un formulaire HTML \(à l'aide de AWS Signature \(version 4\)\)](#)
- [Politique POST](#)
- [Exemple : Browser-Based téléchargement à l'aide de HTTP POST \(à l'aide de AWS Signature \(version 4\)\)](#)
- [Browser-based les téléchargements vers Amazon S3 à l'aide du AWS Bibliothèque Amplify](#)

POST Object

Description

L'POSTopération ajoute un objet à un compartiment spécifié à l'aide de formulaires HTML. POSTest une autre forme PUT qui permet les téléchargements via un navigateur afin de placer des objets dans des compartiments. Les paramètres transmis PUT via les en-têtes HTTP sont plutôt transmis sous forme de champs de formulaire POST dans le corps du message codé en multipart/form-data. Pour ajouter un objet à un compartiment, vous devez WRITE y avoir accès. Amazon S3 ne stocke jamais d'objets partiels. Si vous recevez une réponse positive, vous pouvez être sûr que l'objet entier a été stocké.

Amazon S3 est un système distribué. À moins que vous n'ayez activé la gestion des versions pour un compartiment, si Amazon S3 reçoit simultanément plusieurs demandes d'écriture pour le même objet, seule la dernière version de l'objet écrite est stockée.

Pour vous assurer que les données ne sont pas corrompues lors de la traversée du réseau, utilisez le champ du Content-MD5 formulaire. Lorsque vous utilisez ce champ de formulaire, Amazon S3 compare l'objet à la valeur MD5 fournie. S'ils ne correspondent pas, Amazon S3 renvoie un message d'erreur. En outre, vous pouvez calculer la valeur MD5 lors de la publication d'un objet sur Amazon S3 et comparer la valeur renvoyée ETag à la valeur MD5 calculée. L'ETag reflète uniquement les modifications apportées au contenu d'un objet, pas à ses métadonnées.

Note

Pour configurer votre application afin qu'elle envoie les en-têtes de demande avant d'envoyer le corps de la demande, utilisez le code d'état HTTP 100 (Continuer). Pour les POST opérations, l'utilisation de ce code d'état vous permet d'éviter d'envoyer le corps du message si celui-ci est rejeté sur la base des en-têtes (par exemple, en raison d'un échec d'authentification ou d'une redirection). Pour plus d'informations sur le code d'état HTTP 100 (Continuer), reportez-vous à la Section 8.2.3 de <http://www.ietf.org/rfc/rfc2616.txt>.

Amazon S3 chiffre automatiquement tous les nouveaux objets chargés dans un compartiment S3. Le paramètre de chiffrement d'un objet chargé dépend de la configuration de chiffrement par défaut du compartiment de destination. Par défaut, tous les compartiments ont une configuration de chiffrement par défaut qui utilise le chiffrement côté serveur avec des clés gérées par Amazon S3 (). SSE-S3

Si le compartiment de destination possède une configuration de chiffrement qui utilise le chiffrement côté serveur avec une clé AWS Key Management Service (AWS KMS) clé (SSE-KMS), un cryptage double couche côté serveur avec une AWS KMS clé (DSSE-KMS) ou une clé de cryptage fournie par le client (SSE-C), Amazon S3 utilise la clé KMS correspondante ou la clé fournie par le client pour chiffrer l'objet chargé. Lorsque vous chargez un objet, si vous souhaitez modifier le paramètre de cryptage de l'objet chargé, vous pouvez spécifier le type de cryptage côté serveur. Vous pouvez configurer SSE-S3 SSE-KMS, DSSE-KMS, ou SSE-C. Pour plus d'informations, consultez [Protection des données à l'aide du chiffrement côté serveur](#) dans le Guide de l'utilisateur Amazon Simple Storage Service.

Important

Lors de la création de votre demande, assurez-vous que le `file` champ est le dernier du formulaire.

Gestion des versions

Si vous activez la gestion des versions pour un bucket, un ID de version unique est POST automatiquement généré pour l'objet ajouté. Amazon S3 renvoie cet identifiant dans la réponse en utilisant l'en-tête de `x-amz-version-id` réponse.

Si vous suspendez la gestion des versions pour un compartiment, Amazon S3 utilise toujours l'ID de version de l'objet stocké dans un compartiment `null` comme ID de version.

Pour plus d'informations sur le retour de l'état de versionnement d'un bucket, consultez [GetBucketVersioning](#).

Amazon S3 est un système distribué. Si vous activez la gestion des versions pour un compartiment et qu'Amazon S3 reçoit simultanément plusieurs demandes d'écriture pour le même objet, toutes les versions de l'objet sont stockées.

Pour consulter des exemples de demandes utilisant le contrôle de version, voir [Exemple de demande](#).

Requêtes

Syntaxe

```
POST / HTTP/1.1
```

```
Host: destinationBucket.s3.amazonaws.com
User-Agent: browser_data
Accept: file_types
Accept-Language: Regions
Accept-Encoding: encoding
Accept-Charset: character_set
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: length

--9431149156168
Content-Disposition: form-data; name="key"

acl
--9431149156168
Content-Disposition: form-data; name="tagging"

<Tagging><TagSet><Tag><Key>Tag Name</Key><Value>Tag Value</Value></Tag></TagSet></
Tagging>
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

success_redirect
--9431149156168
Content-Disposition: form-data; name="Content-Type"

content_type
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

uuid
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

metadata
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

access-key-id
--9431149156168
Content-Disposition: form-data; name="Policy"

encoded_policy
```

```
--9431149156168
Content-Disposition: form-data; name="Signature"

signature=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

file_content
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

Paramètres de demande

Cette implémentation de l'opération n'utilise pas de paramètres de demande.

Champs de formulaire

Cette opération peut utiliser les champs de formulaire suivants.

Nom	Description	Obligatoire
AWSAccessKeyId	L'ID de clé d' AWS accès du propriétaire du bucket qui accorde à un utilisateur anonyme l'accès à une demande qui répond à l'ensemble de contraintes de la politique. Type : Chaîne Par défaut : aucun Contraintes : Obligatoire si un document de politique est inclus dans la demande.	Conditionnel
acl	La liste de contrôle d'accès (ACL) Amazon S3 spécifiée. Si l'ACL spécifiée n'est pas valide, une erreur est générée. Pour plus d'informations sur les	Non

Nom	Description	Obligatoire
	ACL, consultez la section Présentation de la liste de contrôle d'accès (ACL) dans le guide de l'utilisateur d'Amazon Simple Storage Service. Type : Chaîne Valeur par défaut : private Valeurs valides : private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	
Cache-Control , Content-Type , Content-Disposition , Content-Encoding , Expires	Les REST-specific en-têtes. Pour de plus amples informations, veuillez consulter PutObject. Type : Chaîne Par défaut : aucun	Non
file	Le contenu du fichier ou du texte. Le contenu du fichier ou du texte doit être le dernier champ du formulaire. Vous ne pouvez pas charger plus d'un fichier à la fois. Type : contenu de fichier ou de texte Par défaut : aucun	Oui

Nom	Description	Obligatoire
key	Nom de la clé chargée. Pour utiliser le nom de fichier fourni par l'utilisateur, utilisez la <code>\${filename}</code> variable. Par exemple, si un utilisateur nommé Mary charge le fichier <code>example.jpg</code> et que vous le spécifiez <code>/user/mary/\${filename}</code>, le nom de la clé est <code>user/mary/example.jpg</code>. Pour plus d'informations, consultez la section Clé d'objet et métadonnées dans le guide de l'utilisateur d'Amazon Simple Storage Service. Type : Chaîne Par défaut : aucun	Oui
policy	La politique de sécurité qui décrit ce qui est autorisé dans la demande. Les demandes sans politique de sécurité sont considérées comme anonymes et ne fonctionnent que sur des compartiments accessibles en écriture publique. Pour plus d'informations, consultez les formulaires HTML et les exemples de téléchargement dans le guide de l'utilisateur d'Amazon Simple Storage Service. Type : Chaîne Par défaut : aucun Contraintes : une politique de sécurité est requise si le bucket n'est pas accessible en écriture publique.	Conditionnel

Nom	Description	Obligatoire
<code>success_action_redirect</code> , <code>redirect</code>	URL vers laquelle le client est redirigé en cas de téléchargement réussi. Si <code>success_action_redirect</code> n'est pas spécifié, Amazon S3 retourne le type de document vide spécifié dans le champ <code>success_action_status</code>. Si Amazon S3 ne peut pas interpréter l'URL, il agit comme si le champ n'était pas présent. Si le téléchargement échoue, Amazon S3 affiche une erreur et ne redirige pas l'utilisateur vers une URL. Type : Chaîne Par défaut : aucun  Note Le nom du <code>redirect</code> champ est obsolète et la prise en charge du nom de <code>redirect</code> champ sera supprimée à l'avenir.	Non

Nom	Description	Obligatoire
success_action_status	Si vous ne le spécifiez pas <code>success_action_redirect</code>, le code d'état est renvoyé au client lorsque le téléchargement est réussi. Ce champ accepte les valeurs <code>200</code>, <code>201</code>, ou <code>204</code> (valeur par défaut). Si la valeur est définie sur <code>200</code> ou <code>204</code>, Amazon S3 renvoie un document vide avec un code de statut <code>200</code> ou <code>204</code>. Si la valeur est définie sur <code>201</code>, Amazon S3 renvoie un document XML avec un code d'état <code>201</code>. Si la valeur n'est pas définie ou si elle est définie sur une valeur non valide, Amazon S3 renvoie un document vide avec un code de statut <code>204</code>. Type : Chaîne Par défaut : aucun	Non

Nom	Description	Obligatoire
tagging	Ensemble de balises spécifié à ajouter à l'objet. Pour ajouter des balises, utilisez le schéma de codage suivant. <pre><Tagging> <TagSet> <Tag> <Key>TagName</Key> <Value>TagValue</Value> </Tag> ... </TagSet> </Tagging></pre>	Non
	Pour plus d'informations, consultez la section Marquage des objets dans le guide de l'utilisateur d'Amazon Simple Storage Service. Type : Chaîne Par défaut : aucun	

Nom	Description	Obligatoire
<code>x-amz-storage-class</code>	Classe de stockage à utiliser pour stocker l'objet. Si vous ne spécifiez aucune classe, Amazon S3 utilise la classe de stockage par défaut, STANDARD. Amazon S3 prend en charge d'autres classes de stockage. Pour plus d'informations, consultez la section Classes de stockage dans le guide de l'utilisateur d'Amazon Simple Storage Service. Type : Chaîne Valeur par défaut : STANDARD Valeurs valides : REDUCED_REDUNDANCY EXPRESS_ONEZONE DEEP_ARCHIVE GLACIER GLACIER_IR INTELLIGENT_TIERING ONEZONE_IA STANDARD STANDARD_IA	Non
<code>x-amz-meta-*</code>	Les en-têtes commençant par ce préfixe sont des métadonnées définies par l'utilisateur. Chacune est stockée et renvoyée sous la forme d'un ensemble de paires clé-valeur. Amazon S3 ne valide ni n'interprète les métadonnées définies par l'utilisateur. Pour de plus amples informations, veuillez consulter PutObject. Type : Chaîne Par défaut : aucun	Non

Nom	Description	Obligatoire
x-amz-security-token	Le jeton DevPay de sécurité Amazon. Chaque demande qui utilise Amazon DevPay nécessite deux champs de x-amz-security-token formulaire : un pour le jeton du produit et un pour le jeton utilisateur. Type : Chaîne Par défaut : aucun	Non
x-amz-signature	(AWS Signature Version 4) Le HMAC-SHA256 hachage de la politique de sécurité. Type : Chaîne Par défaut : aucun	Conditionnel

Nom	Description	Obligatoire
x-amz-website-redirect-location	Si le compartiment est configuré en tant que site Web, ce champ redirige les demandes relatives à cet objet vers un autre objet du même compartiment ou vers une URL externe. Amazon S3 stocke la valeur de cet en-tête dans les métadonnées de l'objet. Pour plus d'informations sur les métadonnées des objets, consultez la section Clé d'objet et métadonnées dans le guide de l'utilisateur d'Amazon Simple Storage Service. Dans l'exemple suivant, l'en-tête de demande définit la redirection vers un objet (anotherPage.html) dans le même compartiment : <pre>x-amz-website-redirect-location: /anotherPage.html</pre> Dans l'exemple suivant, l'en-tête de la demande définit la redirection de l'objet vers un autre site Web : <pre>x-amz-website-redirect-location: http://www.example.com/</pre> Pour plus d'informations sur l'hébergement de sites Web dans Amazon S3, consultez les rubriques Hébergement de sites Web sur Amazon S3 et Comment configurer les redirections de pages Web dans le guide de l'utilisateur d'Amazon Simple Storage Service. Type : Chaîne Par défaut : aucun	Non

Nom	Description	Obligatoire
	Contraintes : la valeur doit être préfixée par /http://, ouhttps://. La longueur de la valeur est limitée à 2 Ko.	

Champs supplémentaires du formulaire de demande de somme de contrôle

Lorsque vous chargez un objet, vous pouvez spécifier les différentes sommes de contrôle que vous souhaitez utiliser pour vérifier l'intégrité de vos données. Vous pouvez spécifier un algorithme de somme de contrôle supplémentaire à utiliser par Amazon S3. Pour plus d'informations sur les valeurs de somme de contrôle supplémentaires, consultez la section [Vérification de l'intégrité des objets](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Nom	Description	Obligatoire
x-amz-checksum-algorithm	Indique l'algorithme utilisé pour créer la somme de contrôle pour l'objet. Si une valeur est spécifiée, vous devez inclure l'en-tête de somme de contrôle correspondant. Dans le cas contraire, votre demande générera une erreur 400. Les valeurs possibles incluent CRC32, CRC32C, SHA1 et SHA256.	Non
x-amz-checksum-crc32	Spécifie la somme de contrôle CRC32 32 bits codée en base64 de l'objet. Ce paramètre est obligatoire si la valeur de x-amz-checksum-algorithm estCRC32.	Conditionnel
x-amz-checksum-crc32c	Spécifie la somme de contrôle CRC32C 32 bits codée en base64 de l'objet. Ce paramètre est obligatoire si la valeur de x-amz-checksum-algorithm estCRC32C.	Conditionnel

Nom	Description	Obligatoire
x-amz-checksum-sha1	Spécifie le SHA-1 condensé de 160 bits codé en base64 de l'objet. Ce paramètre est obligatoire si la valeur de x-amz-checksum-algorithm est SHA1.	Conditionnel
x-amz-checksum-sha256	Spécifie le SHA-256 condensé de 256 bits codé en base64 de l'objet. Ce paramètre est obligatoire si la valeur de x-amz-checksum-algorithm est SHA256.	Conditionnel

Server-Side Champs de formulaire de demande spécifiques au chiffrement

Server-side le chiffrement est le chiffrement des données au repos. Amazon S3 chiffre vos données lors de leur écriture sur des disques situés dans AWS des centres de données et les déchiffre lorsque vous y accédez. Lorsque vous chargez un objet, vous pouvez spécifier le type de chiffrement côté serveur qu'Amazon S3 doit utiliser pour chiffrer l'objet.

Il existe quatre types de chiffrement côté serveur :

- Server-side chiffrement à l'aide de clés gérées par Amazon S3 (SSE-S3) : à compter de mai 2022, le chiffrement est configuré par défaut pour tous les compartiments Amazon S3. L'option par défaut pour le chiffrement côté serveur est avec SSE-S3. Chaque objet est chiffré à l'aide d'une clé unique. Comme mesure de protection supplémentaire, SSE-S3 chiffre la clé elle-même à l'aide d'une clé racine qu'il change régulièrement. SSE-S3 utilise l'un des systèmes de chiffrement par blocs les plus puissants disponibles, le Advanced Encryption Standard (AES-256) 256 bits, pour chiffrer vos données.
- Server-side chiffrement à l'aide de AWS KMS clés (SSE-KMS) : SSE-KMS est fourni par le biais d'une intégration du AWS KMS service à Amazon S3. Avec AWS KMS, vous avez plus de contrôle sur vos clés. Par exemple, vous pouvez afficher des clés distinctes, modifier des politiques de contrôle et suivre les clés dans AWS CloudTrail. En outre, vous pouvez créer et gérer les clés gérées par le client ou utiliser les Clés gérées par AWS qui sont propres à vous-même, à votre service et à votre région.

- Dual-layer chiffrement côté serveur avec AWS KMS clés (DSSE-KMS) : le chiffrement côté Dual-layer serveur avec () est similaire au chiffrement côté serveur avec AWS KMS keys (DSSE-KMS) SSE-KMS, mais applique deux couches individuelles de chiffrement au niveau de l'objet au lieu d'une seule.
- Server-side chiffrement à l'aide de clés fournies par le client (SSE-C) : avec SSE-C, vous gérez les clés de chiffrement, et Amazon S3 gère le chiffrement lors de l'écriture sur les disques, ainsi que le déchiffrement lorsque vous accédez à vos objets.

Note

Si le chiffrement côté serveur avec les clés fournies par le client (SSE-C) est bloqué pour votre compartiment à usage général, une erreur HTTP 403 Access Denied s'affiche lorsque vous spécifiez les en-têtes de SSE-C demande lors de l'écriture de nouvelles données dans votre bucket. Pour plus d'informations, consultez la section [Blocage ou déblocage SSE-C d'un compartiment](#) à usage général.

Pour plus d'informations, consultez [la section Protection des données à l'aide du chiffrement côté serveur](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Selon le type de chiffrement côté serveur que vous souhaitez utiliser, spécifiez les champs de formulaire suivants.

- Utiliser SSE-S3 SSE-KMS, ou DSSE-KMS — Si vous souhaitez utiliser ces types de chiffrement côté serveur, spécifiez les champs de formulaire suivants dans la demande.

Nom	Description	Obligatoire
x-amz-server-side-encryption	Spécifie l'algorithme de chiffrement côté serveur à utiliser lorsqu'Amazon S3 crée un objet. Pour l'utiliser SSE-S3, spécifiez AES256. Pour l'utiliser SSE-KMS, spécifiez aws : kms. Pour l'utiliser DSSE-KMS, spécifiez aws : kms : dsse . Type : Chaîne Valeur valide : aws : kms, AES256, aws : kms : dsse	Oui

Nom	Description	Obligatoire
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	Si l'<code>x-amz-server-side-encryption</code> en-tête a une valeur valide de <code>aws:kms</code> ou <code>aws:kms:dsse</code>, cet en-tête spécifie l'ID de l'objet AWS KMS key qui a été utilisé pour chiffrer l'objet. Type : Chaîne	Oui, si la valeur de <code>x-amz-server-side-encryption</code> est <code>aws:kms</code> ou <code>aws:kms:dsse</code>
<code>x-amz-server-side-encryption-context</code>	Si la valeur <code>aws:kms</code> ou <code>x-amz-server-side-encryption</code> est valide <code>aws:kms:dsse</code>, cet en-tête spécifie le contexte de chiffrement de l'objet. La valeur de cet en-tête est une UTF-8 chaîne codée en base64 qui contient des paires JSON-formatted clé-valeur pour le contexte de chiffrement. Type : Chaîne	Non
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	Si la valeur <code>aws:kms</code> ou <code>x-amz-server-side-encryption</code> est valide <code>aws:kms:dsse</code>, cet en-tête indique si Amazon S3 doit utiliser une clé de compartiment S3 avec SSE-KMS ou DSSE-KMS. Si vous définissez cet en-tête <code>true</code> pour qu'Amazon S3 utilise une clé de compartiment S3 pour le chiffrement des objets avec SSE-KMS ou DSSE-KMS. Type : Boolean	Non

Note

Si vous spécifiez `x-amz-server-side-encryption:aws:kms` ou `x-amz-server-side-encryption:aws:kms:dsse`, mais que vous ne le fournissez pas `x-amz-server-side-encryption-aws-kms-key-id`, Amazon S3 utilise la Clé gérée par AWS (aws/S3) pour protéger les données.

- Utilisation SSE-C — Si vous souhaitez gérer vos propres clés de chiffrement, vous devez fournir tous les champs de formulaire suivants dans la demande.

Note

Si vous utilisez SSE-C, la ETag valeur renvoyée par Amazon S3 dans la réponse n'est pas le MD5 de l'objet.

Nom	Description	Obligatoire
<code>x-amz-server-side-encryption-customer-algorithm</code>	Spécifie l'algorithme à utiliser pour chiffrer l'objet. Type : Chaîne Par défaut : aucun Valeur valide : AES256 Contraintes : Doit être accompagné de <code>x-amz-server-side-encryption-customer-key-MD5</code> champs <code>x-amz-server-side-encryption-customer-key</code> et valides.	Oui
<code>x-amz-server-side-encryption-customer-key</code>	Spécifie la clé de chiffrement codée en base64 fournie par le client à utiliser par Amazon S3 pour chiffrer les données. Cette valeur est utilisée pour stocker l'objet, puis il est supprimé. Amazon ne stocke pas la clé de chiffrement. La clé doit être appropriée pour pouvoir être utilisée avec	Oui

Nom	Description	Obligatoire
	l'algorithme spécifié dans l'en-tête <code>x-amz-server-side-encryption-customer-algorithm</code> . Type : Chaîne Par défaut : aucun Contraintes : Doit être accompagné de <code>x-amz-server-side-encryption-customer-key-MD5</code> champs <code>x-amz-server-side-encryption-customer-algorithm</code> et valides.	
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Spécifie le condensé MD5 128 bits codé en base64 de la clé de chiffrement conformément à la RFC 1321. http://tools.ietf.org/html/rfc1321 Amazon S3 utilise cet en-tête pour vérifier l'intégrité des messages afin de s'assurer que la clé de chiffrement a été transmise sans erreur. Type : Chaîne Par défaut : aucun Contraintes : Doit être accompagné de <code>x-amz-server-side-encryption-customer-key</code> champs <code>x-amz-server-side-encryption-customer-algorithm</code> et valides.	Oui

Réponses

En-têtes de réponse

Cette implémentation de l'opération peut inclure les en-têtes de réponse suivants en plus des en-têtes de réponse communs à toutes les réponses. Pour plus d'informations, consultez la section En-têtes [de réponse](#) communs.

Nom	Description
<code>x-amz-checksum-crc32</code>	Somme de contrôle CRC32 32 bits codée en base64 de l'objet. Type : Chaîne
<code>x-amz-checksum-crc32c</code>	Somme de contrôle CRC32C 32 bits codée en base64 de l'objet. Type : Chaîne
<code>x-amz-checksum-sha1</code>	Résumé de 160 bits codé en base64 de l'objet SHA-1 . Type : Chaîne
<code>x-amz-checksum-sha256</code>	Résumé de 256 bits codé en base64 de l'objet SHA-256 . Type : Chaîne
<code>x-amz-expiration</code>	Si une <code>Expiration</code> action est configurée pour l'objet dans le cadre de la configuration du cycle de vie du bucket, Amazon S3 renvoie cet en-tête. La valeur d'en-tête comprend un <code>expiry-date</code> composant et un URL-encoded <code>rule-id</code> composant. Pour les buckets activés par version, cet en-tête s'applique uniquement aux versions actuelles. Amazon S3 ne fournit pas d'en-tête pour indiquer à quel moment une version non actuelle peut être supprimée définitivement. Pour de plus amples informations, veuillez consulter PutBucketLifecycleConfiguration. Type : Chaîne
<code>success_action_redirect, redirect</code>	URL vers laquelle le client est redirigé en cas de téléchargement réussi. Type : Chaîne

Nom	Description
	Ancêtre : PostResponse
<code>x-amz-server-side-encryption</code>	Algorithme de chiffrement côté serveur qui a été utilisé lors du stockage de cet objet dans Amazon S3 (par exemple,, AES256aws:kms,aws:kms:dsse). Type : Chaîne
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	Si la valeur de l'<code>x-amz-server-side-encryption</code> en-tête est valideaws:kms, cet en-tête spécifie l'ID de la clé KMS qui a été utilisée pour chiffrer l'objet. Type : Chaîne
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	S'il <code>x-amz-server-side-encryption</code> possède une valeur valide deaws:kms, cet en-tête indique si l'objet est chiffré à l'aide SSE-KMS d'une clé de compartiment S3. Si cet en-tête est défini surtrue, l'objet utilise une clé de compartiment S3 avec SSE-KMS. Type : Boolean
<code>x-amz-server-side-encryption-customer-algorithm</code>	Si cela SSE-C a été demandé, la réponse inclut cet en-tête, qui confirme l'algorithme de chiffrement utilisé. Type : Chaîne Valeurs valides : AES256
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Si cela SSE-C a été demandé, la réponse inclut cet en-tête pour vérifier l'intégrité du message aller-retour de la clé de cryptage fournie par le client. Type : Chaîne

Nom	Description
<code>x-amz-version-id</code>	Version de l'objet. Type : Chaîne

Éléments de réponse

Nom	Description
Bucket	Le nom du compartiment dans lequel l'objet a été stocké. Type : Chaîne Ancêtre : PostResponse
ETag	La balise d'entité (ETag) est un hachage MD5 de l'objet que vous pouvez utiliser pour effectuer des GET opérations conditionnelles en utilisant la balise de If-Modified requête avec l'opération de GET demande. ETag reflète uniquement les modifications apportées au contenu d'un objet, et non à ses métadonnées. Type : Chaîne Ancêtre : PostResponse
Key	Le nom de la clé de l'objet. Type : Chaîne Ancêtre : PostResponse
Location	L'URI de l'objet. Type : Chaîne Ancêtre : PostResponse

Erreurs spéciales

Cette implémentation de l'opération ne renvoie pas d'erreurs spéciales. Pour obtenir des informations générales sur les erreurs Amazon S3 et une liste des codes d'erreur, consultez la section Réponses aux [erreurs](#).

Exemples

Exemple de demande

```
POST /Neo HTTP/1.1
Content-Length: 4
Host: quotes.s3.amazonaws.com
Date: Wed, 01 Mar 2006 12:00:00 GMT
Authorization: authorization string
Content-Type: text/plain
Expect: the 100-continue HTTP status code
```

ObjectContent

Exemple de réponse avec suspension de la gestion des versions

Voici un exemple de réponse lorsque la gestion des versions des compartiments est suspendue :

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdm0dAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiI0Ix1+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: default
Date: Wed, 12 Oct 2009 17:50:00 GMT
ETag: "1b2cf535f27731c974343645a3985328"
Content-Length: 0
Connection: close
Server: AmazonS3
```

Dans cette réponse, l'ID de version est `null`.

Exemple de réponse avec la gestion des versions activée

Voici un exemple de réponse lorsque la gestion des versions des compartiments est activée.

```
HTTP/1.1 100 Continue
```

```
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdm0dAiIfgSm/F1YsViT1LW94/xUQxMsF7xiEb1a0wiI0Ix1+zbwZ163pt7
x-amz-request-id: 0A49CE4060975EAC
x-amz-version-id: 43jfkodU8493jnFJD9fjj3HHNVfdsQUIFDNsidf038jfdsjGFDSIRp
Date: Wed, 01 Mar 2006 12:00:00 GMT
ETag: "828ef3fdfa96f00ad9f27c383fc9ac7f"
Content-Length: 0
Connection: close
Server: AmazonS3
```

Ressources connexes

- [CopyObject](#)
- [POST Object](#)
- [GetObject](#)

POST Object restore

Description

Cette opération exécute les types de requêtes suivants :

- `select`— Effectue une requête de sélection sur un objet archivé
- `restore an archive`— Restaurer un objet archivé

Pour utiliser cette opération, vous devez disposer des autorisations nécessaires pour effectuer les `s3:GetObject` actions `s3:RestoreObject` et. Le propriétaire du bucket dispose de cette autorisation par défaut et peut accorder cette autorisation à d'autres personnes. Pour plus d'informations sur les autorisations, consultez [les sections Autorisations liées aux opérations des sous-ressources du bucket](#) et à [la gestion des autorisations d'accès à vos ressources Amazon S3](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Interroger les archives à l'aide de requêtes sélectionnées

Vous utilisez un type de requête sélectionné pour exécuter des requêtes SQL sur des objets archivés. Les objets archivés qui sont interrogés par la demande de sélection doivent être mis en forme sous forme de fichiers de valeurs séparées par des virgules (CSV) non compressés. Vous pouvez exécuter des requêtes et des analyses personnalisées sur vos données archivées sans avoir à restaurer vos données vers un niveau Amazon S3 plus avancé. Pour obtenir un aperçu de certaines demandes, consultez la section [Interroger des objets archivés](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Lorsque vous faites une demande de sélection, procédez comme suit :

- Définissez un emplacement de sortie pour la sortie de la requête sélectionnée. Il doit s'agir d'un compartiment Amazon S3 dans le même Région AWS compartiment que celui qui contient l'objet d'archive interrogé. La Compte AWS personne qui lance la tâche doit être autorisée à écrire dans le compartiment S3. Vous pouvez spécifier la classe de stockage et le chiffrement des objets de sortie stockés dans le bucket. Pour plus d'informations sur la sortie, consultez la section [Interroger des objets archivés](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Pour plus d'informations sur la S3 structure du corps de la requête, consultez les rubriques suivantes :

- [PutObject](#)

- [Gestion de l'accès à l'aide de listes de contrôle d'accès](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service
- [Protection des données à l'aide du Server-Side chiffrement](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service
- Définissez l'expression SQL pour le SELECT type de restauration de votre requête dans la SelectParameters structure du corps de la requête. Vous pouvez utiliser des expressions comme dans les exemples suivants.
- L'expression suivante renvoie tous les enregistrements de l'objet spécifié.

```
SELECT * FROM Object
```

- En supposant que vous n'utilisez aucun en-tête pour les données stockées dans l'objet, vous pouvez spécifier des colonnes avec des en-têtes positionnels.

```
SELECT s._1, s._2 FROM Object s WHERE s._3 > 100
```

- Si vous avez des en-têtes et que vous les définissez fileHeaderInfo dans la CSV structure du corps de la demande surUSE, vous pouvez spécifier des en-têtes dans la requête. (Si vous définissez le champ fileHeaderInfo sur IGNORE, la première ligne est ignorée pour la requête.) Vous ne pouvez pas combiner les positions ordinales avec des noms de colonnes d'en-têtes.

```
SELECT s.Id, s.FirstName, s.SSN FROM S3Object s
```

Pour plus d'informations sur l'utilisation de SQL avec la restauration d'Amazon Glacier Select, consultez la référence [SQL pour Amazon S3 Select et Amazon Glacier Select](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Lorsque vous faites une demande de sélection, vous pouvez également effectuer les opérations suivantes :

- Pour accélérer vos requêtes, spécifiez le Expedited niveau. Pour plus d'informations sur les niveaux, consultez la section « Restauration des archives », plus loin dans cette rubrique.
- Spécifiez des détails concernant le format de sérialisation des données de l'objet d'entrée interrogé et de la sérialisation des résultats de la requête. CSV-encoded

Voici d'autres informations importantes concernant la fonction de sélection :

- Les résultats en sortie sont de nouveaux objets Amazon S3. Contrairement aux extractions d'archives, elles sont stockées jusqu'à leur suppression explicite, manuellement ou dans le cadre d'une politique de cycle de vie.
- Vous pouvez émettre plusieurs demandes de sélection pour le même objet Amazon S3. Amazon S3 ne dédoublant pas les demandes, évitez donc d'émettre des demandes dupliquées.
- Amazon S3 accepte une demande de sélection même si l'objet a déjà été restauré. Une demande de sélection ne renvoie pas de réponse d'erreur 409.

Restauration des archives

Les objets des classes de stockage GLACIER et DEEP_ARCHIVE sont archivés. Pour accéder à un objet archivé, vous devez d'abord lancer une demande de restauration. Cela permet de restaurer une copie temporaire de l'objet archivé. Dans une demande de restauration, vous spécifiez le nombre de jours pendant lesquels vous souhaitez que la copie restaurée existe. Après la période spécifiée, Amazon S3 supprime la copie temporaire mais l'objet reste archivé dans la classe de stockage GLACIER ou DEEP_ARCHIVE à partir de laquelle l'objet a été restauré.

Pour restaurer une version spécifique d'un objet, vous pouvez fournir un ID de version. Si vous ne fournissez pas d'ID de version, Amazon S3 restaure la version actuelle.

Le temps nécessaire pour terminer les tâches de restauration dépend de la classe de stockage à partir de laquelle l'objet est restauré et du niveau d'accès aux données que vous spécifiez.

Lorsque vous restaurez un objet archivé (ou que vous utilisez une demande de sélection), vous pouvez spécifier l'une des options de niveau d'accès aux données suivantes dans l'`Tier` élément du corps de la demande :

- **Expedited**- Les extractions accélérées vous permettent d'accéder rapidement à vos données stockées dans la classe de stockage GLACIER lorsque des demandes urgentes occasionnelles concernant un sous-ensemble d'archives sont requises. Pour tous les objets archivés, à l'exception des objets les plus volumineux (250 Mo et plus), les données accessibles à l'aide des extractions accélérées sont généralement mises à disposition en 1 à 5 minutes. La capacité provisionnée garantit que la capacité des récupérations rapides est disponible lorsque vous en avez besoin. Les extractions accélérées et la capacité provisionnée ne sont pas disponibles pour la classe de stockage DEEP_ARCHIVE.

- **Standard**- Les récupérations Standard vous permettent d'accéder à vos objets archivés en quelques heures. Il s'agit de l'option par défaut pour les demandes de récupération GLACIER et DEEP_ARCHIVE qui ne spécifient pas l'option de récupération. Les extractions standard sont généralement effectuées dans les 3 à 5 heures suivant la classe de stockage GLACIER et généralement dans les 12 heures suivant la classe de stockage DEEP_ARCHIVE.
- **Bulk**- Les extractions en masse constituent l'option de récupération la moins coûteuse d'Amazon Glacier, vous permettant de récupérer de grandes quantités, voire des pétaoctets, de données à peu de frais en une journée. Les extractions en masse sont généralement effectuées dans les 5 à 12 heures suivant la classe de stockage GLACIER et généralement dans les 48 heures suivant la classe de stockage DEEP_ARCHIVE.

Pour plus d'informations sur les options de récupération des archives et la capacité provisionnée pour l'accès aux Expedited données, consultez la section [Restauration d'objets archivés](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Vous pouvez utiliser la mise à niveau de la vitesse de restauration d'Amazon S3 pour accélérer la vitesse de restauration pendant qu'elle est en cours. Vous améliorez la vitesse d'une restauration en cours en émettant une autre demande de restauration pour le même objet, en définissant un nouvel élément de Tier demande. Au moment d'émettre une demande de mise à niveau du niveau de restauration, vous devez choisir un niveau plus rapide que celui utilisé par la restauration en cours. Vous ne devez modifier aucun autre paramètre comme l'élément de demande Days. Pour plus d'informations, consultez la section [Améliorer la vitesse d'une In-Progress restauration](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Pour obtenir l'état de la restauration de l'objet, vous pouvez envoyer une HEAD demande. Les opérations renvoient l'`x-amz-restore` en-tête, qui fournit des informations sur l'état de restauration, dans la réponse. Vous pouvez utiliser les notifications d'événements Amazon S3 pour vous avertir lorsqu'une restauration est lancée ou terminée. Pour plus d'informations, consultez [la section Configuration des notifications d'événements Amazon S3](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Après avoir restauré un objet archivé, vous pouvez mettre à jour la période de restauration en réémettant la demande avec une nouvelle période. Amazon S3 met à jour la période de restauration par rapport à l'heure actuelle et ne facture que pour la demande. Aucun frais de transfert de données n'est facturé. Vous ne pouvez pas mettre à jour la période de restauration lorsqu'Amazon S3 traite activement votre demande de restauration actuelle pour l'objet.

Si votre bucket possède une configuration de cycle de vie avec une règle qui inclut une action d'expiration, l'expiration de l'objet remplace la durée de vie que vous spécifiez dans une demande de restauration. Par exemple, si vous restaurez une copie d'objet pendant 10 jours, mais que l'objet doit expirer dans 3 jours, Amazon S3 supprime l'objet dans 3 jours. Pour plus d'informations sur la configuration du cycle de vie, consultez [PutBucketLifecycleConfiguration](#) la section Gestion du cycle de vie des [objets](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Requêtes

Syntaxe

```
POST /ObjectName?restore&versionId=VersionID HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Demandes d'authentification \(AWS Signature \(version 4\)\))
Content-MD5: MD5

request body
```

Note

La syntaxe affiche certains en-têtes de requête. Pour une liste complète, voir « En-têtes de demande », plus loin dans cette rubrique.

Paramètres de demande

Cette implémentation de l'opération n'utilise pas de paramètres de demande.

En-têtes de demande

Nom	Description	Obligatoire
Content-MD5	Le condensé MD5 128 bits codé en base64 des données. Vous devez utiliser cet en-tête pour vérifier l'intégrité du message afin de vérifier que le corps de la demande n'a pas été endommagé	Oui

Nom	Description	Obligatoire
	pendant le transit. Pour plus d'informations, consultez la RFC 1864 . Type : Chaîne Par défaut : aucun	

Elements de la demande

Voici un exemple XML de corps de requête pour la restauration d'une archive.

```
<RestoreRequest>  
  <Days>2</Days>  
  <GlacierJobParameters>  
    <Tier>Bulk</Tier>  
  </GlacierJobParameters>  
</RestoreRequest>
```

Le tableau suivant explique le code XML pour la restauration des archives dans le corps de la demande.

Nom	Description	Obligatoire
RestoreRequest	Conteneur pour les informations de restauration. Type : Conteneur	Oui
Days	Durée de vie de la copie restaurée (active). Le nombre minimum de jours pendant lesquels vous pouvez restaurer un objet depuis Amazon Glacier est de 1. Une fois que la copie de l'objet a atteint la durée de vie spécifiée, Amazon S3 la supprime du compartiment. Si vous restaurez une archive, cet élément est obligatoire. N'utilisez pas cet élément avec un SELECT type de requête.	Oui, si vous restaurez une archive

Nom	Description	Obligatoire
GlacierJobParameters	Type : entier positif	Non
	Ancêtres : RestoreRequest	
Tier	Conteneur pour les paramètres de travail de la classe de stockage S3 Glacier.	Non
	N'utilisez pas cet élément avec un SELECT type de requête.	
	Type : Conteneur	
	Ancêtres : RestoreRequest	
	Le niveau d'accès aux données à utiliser lors de la restauration de l'archive. Standard est la valeur par défaut.	Non
	Type : énumération	
	Valeurs valides : Expedited Standard Bulk	
	Ancêtres : GlacierJobParameters	

Le code XML suivant constitue le corps de requête d'une requête de sélection sur un objet archivé :

```
<RestoreRequest>
  <Type>SELECT</Type>
  <Tier>Expedited</Tier>
  <Description>Job description</Description>
  <SelectParameters>
    <Expression>Select * from Object</Expression>
    <ExpressionType>SQL</ExpressionType>
    <InputSerialization>
      <CSV>
        <FileHeaderInfo>IGNORE</FileHeaderInfo>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
```

```

        <Comments>#</Comments>
    </CSV>
</InputSerialization>
<OutputSerialization>
    <CSV>
        <QuoteFields>ASNEEDED</QuoteFields>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
    </CSV>
</OutputSerialization>
</SelectParameters>
<OutputLocation>
    <S3>
        <BucketName>Name of bucket</BucketName>
        <Prefix>Key prefix</Prefix>
        <CannedACL>Canned ACL string</CannedACL>
        <AccessControlList>
            <Grantee>
                <Type>Grantee Type</Type>
            <ID>Grantee identifier</ID>
            <URI>Grantee URI</URI>
            <Permission>Granted permission</Permission>
            <DisplayName>Display Name</DisplayName>
            <EmailAddress>email</EmailAddress>
        </Grantee>
    </AccessControlList>
    <Encryption>
        <EncryptionType>Encryption type</EncryptionType>
    <KMSKeyId>KMS Key ID</KMSKeyId>
    <KMSContext>Base64-encoded JSON<KMSContext>
    </Encryption>
    <UserMetadata>
        <MetadataEntry>
            <Name>Key</Name>
            <Value>Value</Value>
        </MetadataEntry>
    </UserMetadata>
    <Tagging>
        <TagSet>
            <Tag>
                <Key>Tag name</Key>
                <Value>Tag value</Value>
            </Tag>
        </TagSet>
    </Tagging>

```

```
        </Tag>
      </TagSet>
    </Tagging>
    <StorageClass>Storage class</StorageClass>
  </S3>
</OutputLocation>
</RestoreRequest>
```

Les tableaux suivants décrivent le code XML pour un SELECT type de restauration dans le corps de la demande.

Nom	Description	Obligatoire
RestoreRequest	Conteneur pour les informations de restauration. Type : Conteneur	Oui
Tier	Le niveau d'accès aux données à utiliser lors de la restauration de l'archive. Standard est la valeur par défaut. Type : énumération Valeurs valides : Expedited Standard Bulk Ancêtres : RestoreRequest	Non
Description	Description facultative de la demande. Type : Chaîne Ancêtres : RestoreRequest	Non
SelectParameters	Décrit les paramètres de la demande de tâche sélectionnée. Type : Conteneur Ancêtres : RestoreRequest	Oui, si le type de demande est SELECT

Nom	Description	Obligatoire
OutputLocation	Décrit l'emplacement qui reçoit les résultats de la demande de restauration sélectionnée. Type : Conteneur pour Amazon S3 Ancêtres : RestoreRequest	Oui, si le type de demande est SELECT

L'élément **SelectParameters** conteneur contient les éléments suivants.

Nom	Description	Obligatoire
Expression	Expression SQL. Par exemple : - L'expression SQL suivante extrait la première colonne des données de l'objet stocké au format CSV : <pre>SELECT s._1 FROM Object s</pre> - L'expression SQL suivante renvoie tout ce qui provient de l'objet : <pre>SELECT * FROM Object</pre> Type : Chaîne Ancêtres : SelectParameters	Oui
ExpressionType	Identifie le type d'expression. Type : Chaîne Valeurs valides : SQL Ancêtres : SelectParameters	Oui

Nom	Description	Obligatoire
InputSerialization	Décrit le format de sérialisation de l'objet. Type : Conteneur pour CSV Ancêtres : <code>SelectParameters</code>	Oui
OutputSerialization	Décrit la manière dont les résultats de la tâche de sélection sont sérialisés. Type : Conteneur pour CSV Ancêtres : <code>SelectParameters</code>	Oui

L'élément conteneur CSV de l'**InputSerialization** élément contient les éléments suivants.

Nom	Description	Obligatoire
RecordDelimiter	Caractère unique utilisé pour séparer les enregistrements individuels dans l'entrée. Au lieu de la valeur par défaut, vous pouvez spécifier un délimiteur arbitraire. Type : Chaîne Valeur par défaut : <code>\n</code> Ancêtres : <code>CSV</code>	Non
FieldDelimiter	Caractère unique utilisé pour séparer les différents champs d'un enregistrement. Vous pouvez spécifier un délimiteur arbitraire. Type : Chaîne Valeur par défaut : <code>,</code>	Non

Nom	Description	Obligatoire
	Ancêtres : CSV	
QuoteCharacter	Caractère unique utilisé pour échapper lorsque le délimiteur de champ fait partie de la valeur. Prenons cet exemple dans un fichier CSV : "a, b" Placer la valeur entre guillemets fait de cette valeur un champ unique. Si vous n'utilisez pas de guillemets, la virgule est un délimiteur de champ (ce qui en fait deux valeurs de champ distinctes, a et b). Type : Chaîne Valeur par défaut : " Ancêtres : CSV	Non
QuoteEscapeCharacter	Caractère unique utilisé pour échapper au guillemet à l'intérieur d'une valeur déjà échappée. Par exemple, la valeur ""a , b "" est analysée comme " a , b ". Type : Chaîne Valeur par défaut : " Ancêtres : CSV	Non

Nom	Description	Obligatoire
FileHeaderInfo	Décrit la première ligne des données d'entrée. Il s'agit de l'une des valeurs ENUM. • NONE: La première ligne n'est pas un en-tête. • IGNORE: La première ligne est un en-tête, mais vous ne pouvez pas utiliser les valeurs d'en-tête pour indiquer la colonne d'une expression. Vous pouvez utiliser la position de la colonne (telle que _1, _2,...) pour indiquer la colonne (SELECT s._1 FROM OBJECT s). • Use: La première ligne est un en-tête, et vous pouvez utiliser la valeur de l'en-tête pour identifier une colonne dans une expression (SELECT "name" FROM OBJECT). Type : énumération Valeurs valides : NONE USE IGNORE Ancêtres : CSV	Non
Comments	Caractère nique utilisé pour indiquer qu'une ligne doit être ignorée lorsque le caractère est présent au début de cette ligne. Vous pouvez spécifier n'importe quel caractère pour indiquer une ligne de commentaire. Type : Chaîne Ancêtres : CSV	Non

L'élément conteneur CSV (dans les **OutputSerialization** éléments) contient les éléments suivants.

Nom	Description	Obligatoire
QuoteFields	Indique s'il faut utiliser des guillemets autour des champs de sortie. • ALWAYS: utilisez toujours des guillemets pour les champs de sortie. • ASNEEDED: utilisez des guillemets pour les champs de sortie si nécessaire. Type : énumération Valeurs valides : ALWAYS ASNEEDED Valeur par défaut : AsNeeded Ancêtres : CSV	Non
RecordDelimiter	Caractère unique utilisé pour séparer les enregistrements individuels dans la sortie. Au lieu de la valeur par défaut, vous pouvez spécifier un délimiteur arbitraire. Type : Chaîne Valeur par défaut : \n Ancêtres : CSV	Non
FieldDelimiter	Caractère unique utilisé pour séparer les différents champs d'un enregistrement. Vous pouvez spécifier un délimiteur arbitraire. Type : Chaîne	Non

Nom	Description	Obligatoire
	Valeur par défaut : , Ancêtres : CSV	
QuoteCharacter	Caractère unique utilisé pour échapper lorsque le délimiteur de champ fait partie de la valeur. Par exemple, si la valeur est a , b, Amazon S3 met la valeur de ce champ entre guillemets, comme suit : " a , b " Type : Chaîne Valeur par défaut : " Ancêtres : CSV	Non
QuoteEscapeCharacter	Caractère unique utilisé pour échapper au guillemet à l'intérieur d'une valeur déjà échappée. Par exemple, si la valeur est " a , b " , Amazon S3 met la valeur entre guillemets, comme suit : "" " a , b "" Type : Chaîne Ancêtres : CSV	Non

L'élément conteneur S3 (dans l'**OutputLocation**élément) contient les éléments suivants.

Nom	Description	Obligatoire
AccessControlList	Liste des subventions qui contrôlent l'accès aux résultats par étapes. Type : Conteneur pour subvention Ancêtres : S3	Non
BucketName		Oui

Nom	Description	Obligatoire
	Nom du compartiment S3 dans lequel les résultats de la restauration sélectionnée sont stockés. Le compartiment doit se trouver dans le même compartiment Région AWS que celui qui contient l'objet d'archive en entrée. Type : Chaîne Ancêtres : S3	
CannedACL	La liste de contrôle d'accès (ACL) prédéfinie à appliquer aux résultats de restauration sélectionnés. Type : Chaîne Valeurs valides : <code>private</code> <code>public-read</code> <code>public-read-write</code> <code>aws-exec-read</code> <code>authenticated-read</code> <code>bucket-owner-read</code> <code>bucket-owner-full-control</code> Ancêtres : S3	Non
Encryption	Contient des informations de chiffrement pour les résultats enregistrés. Type : Conteneur pour chiffrement Ancêtres : S3	Non
Prefix	Le préfixe ajouté aux résultats de la restauration sélectionnée. La longueur maximale du préfixe est de 512 caractères. Type : Chaîne Ancêtres : S3	Oui

Nom	Description	Obligatoire
StorageClass	Classe de stockage utilisée pour stocker les résultats de la demande de sélection. Type : Chaîne Valeurs valides : STANDARD REDUCED_REDUNDANCY STANDARD_IA ONEZONE_IA Ancêtres : S3	Non
Tagging	Conteneur pour les informations sur les étiquettes. Type : Structure des balises Ancêtres : S3	Non
UserMetadata	Contient une liste de métadonnées à stocker avec les résultats de restauration sélectionnés. Type : MetadataEntry structure Ancêtres : S3	Non

L'élément conteneur du bénéficiaire (dans l'**AccessControlList** élément) contient les éléments suivants.

Nom	Description	Obligatoire
DisplayName	Nom de l'écran du bénéficiaire. Type : Chaîne Ancêtres : Grantee	Non
EmailAddress	Adresse e-mail du bénéficiaire.	Non

Nom	Description	Obligatoire
ID	Type : Chaîne	Non
	Ancêtres : Grantee	
	ID d'utilisateur canonique du bénéficiaire.	
Type	Type : Chaîne	Non
	Ancêtres : Grantee	
	Type du bénéficiaire.	
URI	Type : Chaîne	Non
	Ancêtres : Grantee	
	URI du groupe du bénéficiaire.	
Permission	Type : Chaîne	Non
	Ancêtres : Grantee	
	Autorisation accordée.	

L'élément **Conteneur de** chiffrement (dans S3) contient les éléments suivants.

Nom	Description	Obligatoire
EncryptionType	Algorithme de chiffrement côté serveur utilisé lors du stockage des résultats des tâches. La valeur par défaut est sans chiffrement.	Non

Nom	Description	Obligatoire
	Type : Chaîne Valeurs valides aws : kms AES256 Ancêtres : Encryption	
KMSContext	Facultatif. Si le type de cryptage est aws : kms, vous pouvez utiliser cette valeur pour spécifier le contexte de chiffrement pour les résultats de restauration sélectionnés. Type : Chaîne Ancêtres : Encryption	Non
KMSKeyId	L'ID de clé AWS Key Management Service (AWS KMS) à utiliser pour le chiffrement des objets. Type : Chaîne Ancêtres : Encryption	Non

L'élément **TagSet** conteneur (dans l'**élément de** balisage) contient l'élément suivant.

Nom	Description	Obligatoire
Tag	Contient des étiquettes. Type : Conteneur Ancêtres : TagSet	Non

L'élément conteneur Tag (dans l'**TagSet** élément) contient les éléments suivants.

Nom	Description	Obligatoire
Key		Non

Nom	Description	Obligatoire
	Nom de la balise. Type : Chaîne Ancêtres : Tag	
Value	Valeur de la balise. Type : Chaîne Ancêtres : Tag	Non

L'élément **MetadataEntry** conteneur (dans l'**UserMetadata**élément) contient les éléments de paire clé-valeur suivants à stocker avec un objet.

Nom	Description	Obligatoire
MetadataKey	Clé de métadonnées. Type : Chaîne Ancêtres :	Non
MetadataEntry	La valeur des métadonnées. Type : Chaîne Ancêtres :	Non

Réponses

Une opération réussie renvoie le code 200 OK d'202 Accepted état ou.

- Si la copie de l'objet n'a pas été restaurée précédemment, Amazon S3 le renvoie 202 Accepted dans la réponse.
- Si la copie de l'objet a déjà été restaurée, Amazon S3 renvoie 200 OK la réponse.

En-têtes de réponse

Cette implémentation de l'opération utilise uniquement les en-têtes de réponse communs à la plupart des réponses. Pour plus d'informations, consultez la section En-têtes [de réponse](#) communs.

Éléments de réponse

Cette opération ne renvoie pas d'éléments de réponse.

Erreurs spéciales

Code d'erreur	Description	HTTP Status Code	Préfixe du code d'erreur SOAP
RestoreAlreadyInProgress	La restauration des objets est déjà en cours. (Cette erreur ne s'applique pas aux demandes de type SELECT.)	409 – Conflit	Client
GlacierExpeditedRetrievalNotAvailable	Les récupérations accélérées de la classe de stockage S3 Glacier ne sont actuellement pas disponibles. Réessayez ultérieurement. (Renvoyé si la capacité est insuffisante pour traiter la Expedited demande. Cette erreur s'applique uniquement aux Expedited extractions et non aux Standard Bulk extractions.)	503	N/A

Exemples

Restaurez un objet pendant deux jours à l'aide de l'option de récupération accélérée

La demande de restauration suivante restaure une copie de l'photo1.jpg objet depuis Amazon Glacier pendant une période de deux jours à l'aide de l'option de récupération accélérée.

```
POST /photo1.jpg?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Mon, 22 Oct 2012 01:49:52 GMT
Authorization: authorization string
Content-Length: content length
```

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Expedited</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

Si examplebucket aucune copie restaurée de l'objet n'est disponible, Amazon S3 renvoie la 202 Accepted réponse suivante.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZ1zYQvPiBlZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

Si une copie de l'objet est déjà restaurée, Amazon S3 renvoie une 200 OK réponse et met à jour uniquement l'heure d'expiration de la copie restaurée.

Interroger une archive à l'aide d'une requête SELECT

Voici un exemple de demande de restauration sélective.

```
POST /object-one.csv?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Date: Sat, 20 Oct 2012 23:54:05 GMT
Authorization: authorization string
Content-Length: content length

<RestoreRequest xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Type>SELECT</Type>
  <Tier>Expedited</Tier>
  <Description>this is a description</Description>
  <SelectParameters>
```

```

<InputSerialization>
  <CSV>
    <FileHeaderInfo>IGNORE</FileHeaderInfo>
    <Comments>#</Comments>
    <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
    <RecordDelimiter>\n</RecordDelimiter>
    <FieldDelimiter>,</FieldDelimiter>
    <QuoteCharacter>"</QuoteCharacter>
  </CSV>
</InputSerialization>
<ExpressionType>SQL</ExpressionType>
<Expression>select * from object</Expression>
<OutputSerialization>
  <CSV>
    <QuoteFields>ALWAYS</QuoteFields>
    <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
    <RecordDelimiter>\n</RecordDelimiter>
    <FieldDelimiter>\t</FieldDelimiter>
    <QuoteCharacter>\'</QuoteCharacter>
  </CSV>
</OutputSerialization>
</SelectParameters>
<OutputLocation>
  <S3>
    <BucketName>example-output-bucket</BucketName>
    <Prefix>test-s3</Prefix>
    <AccessControllist>
      <Grant>
        <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail">
          <EmailAddress>jane-doe@example.com</EmailAddress>
        </Grantee>
        <Permission>FULL_CONTROL</Permission>
      </Grant>
    </AccessControllist>
    <UserMetadata>
      <MetadataEntry>
        <Name>test</Name>
        <Value>test-value</Value>
      </MetadataEntry>
      <MetadataEntry>
        <Name>other</Name>
        <Value>something else</Value>
      </MetadataEntry>
    </UserMetadata>
  </S3>
</OutputLocation>

```

```
</UserMetadata>
<StorageClass>STANDARD</StorageClass>
</S3>
</OutputLocation>
</RestoreRequest>
```

Amazon S3 renvoie la 202 Accepted réponse suivante.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZlZYQvPiBlZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
x-amz-restore-output-path: js-test-s3/qE8nk5M0XIj-LuZE2HXNw6empQm3znLkHlMWInRYPS-
0r12W0uj6LyYm-neTvm1-btz3wbBxfMhPykd3jk1-lvZE7w42/
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

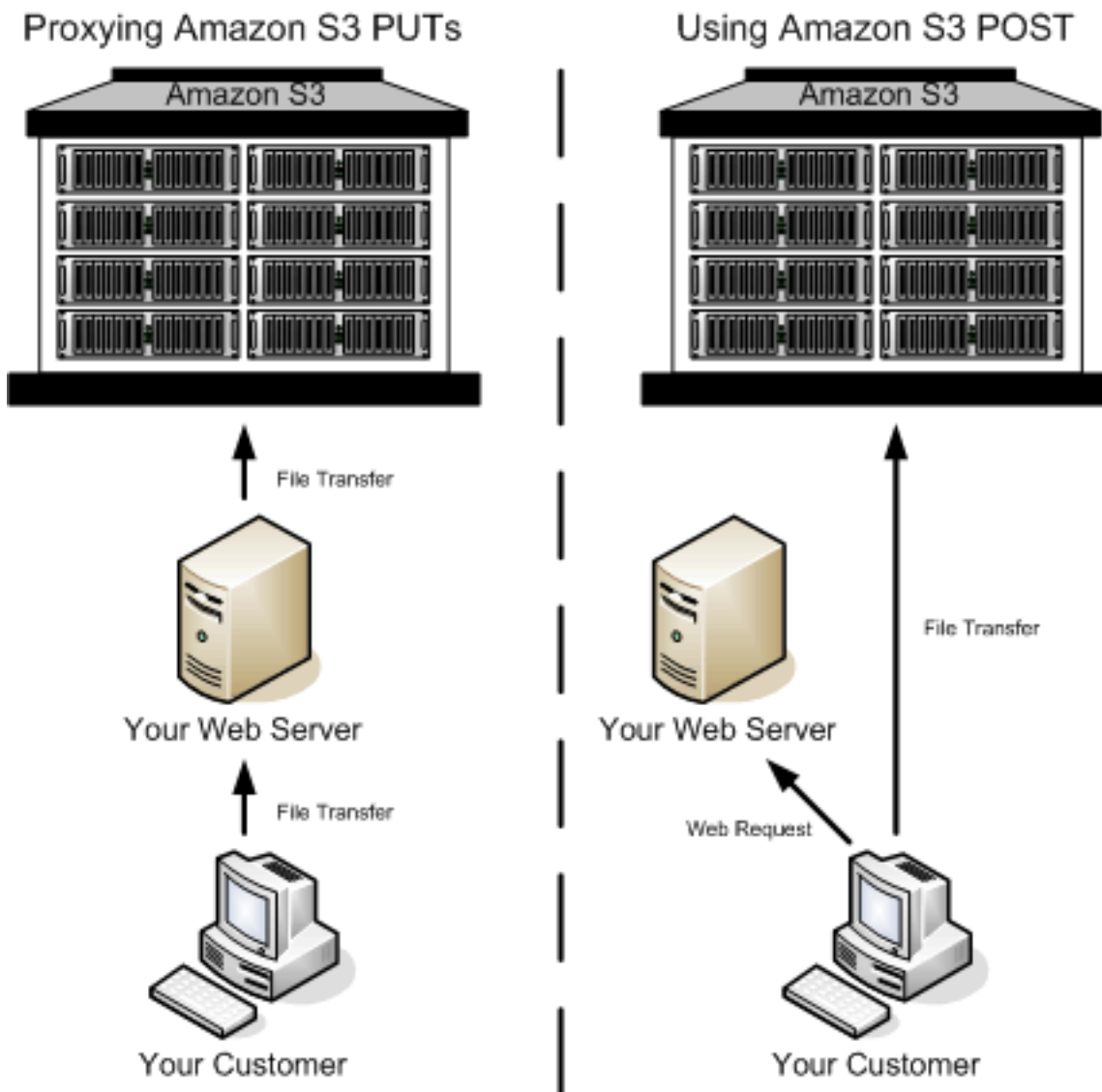
Plus d'informations

- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)
- [Référence SQL pour Amazon S3 Select et Amazon Glacier Select](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service

Browser-Based Téléversements à l'aide de HTTP POST

Amazon S3 prend en charge les requêtes HTTP POST afin que les utilisateurs puissent télécharger du contenu directement sur Amazon S3. En utilisant POST, les utilisateurs finaux peuvent authentifier les demandes sans avoir à transmettre les données via un nœud intermédiaire sécurisé qui protège vos informations d'identification. Ainsi, HTTP POST a le potentiel de réduire la latence.

La figure suivante montre un téléchargement Amazon S3 à l'aide d'une requête POST.



1. L'utilisateur accède à votre page à partir d'un navigateur Web.
2. Votre page Web contient un formulaire HTML qui contient toutes les informations nécessaires pour que l'utilisateur télécharge du contenu sur Amazon S3.
3. L'utilisateur télécharge du contenu sur Amazon S3 via le navigateur Web.

Le processus d'envoi de requêtes POST depuis un navigateur est le suivant :

1. Créez une politique de sécurité spécifiant les conditions qui limitent ce que vous souhaitez autoriser dans la demande, telles que le nom du compartiment dans lequel les objets peuvent être chargés et les préfixes de noms de clé que vous souhaitez autoriser pour l'objet en cours de création.

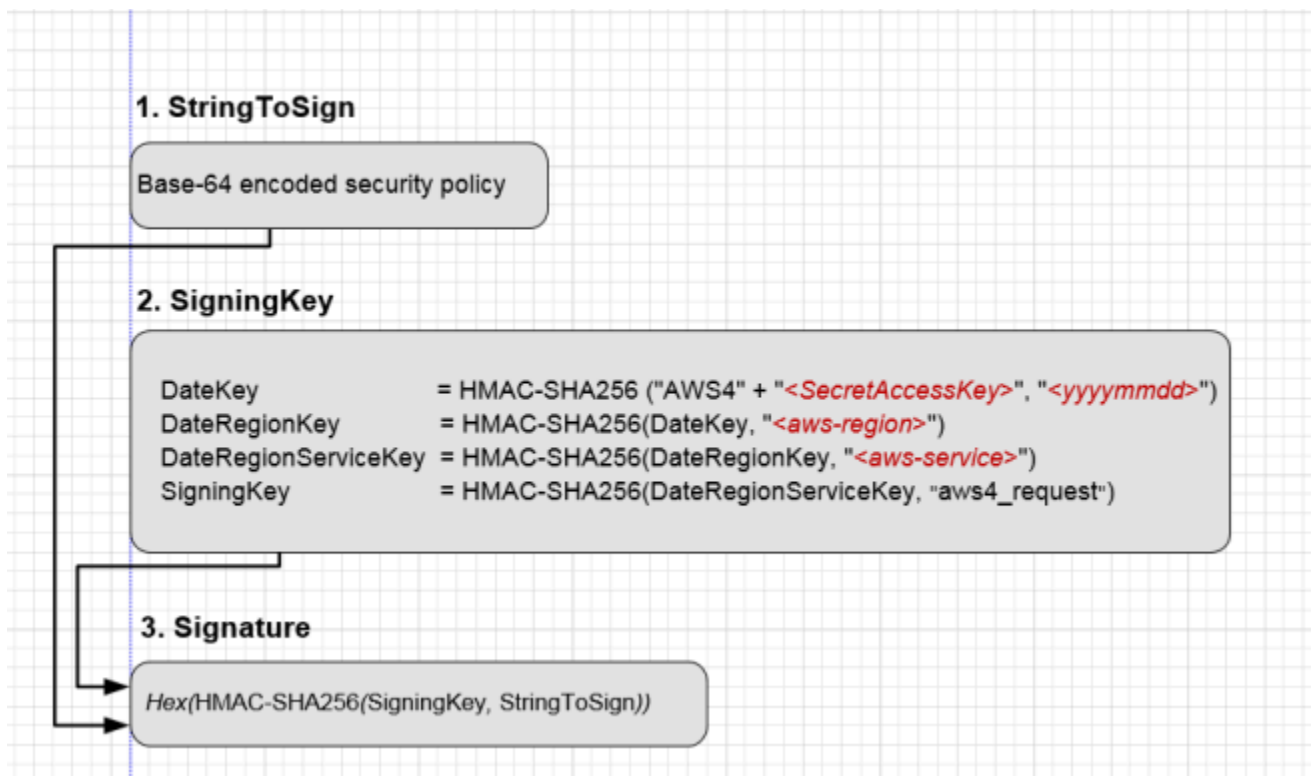
2. Créez une signature basée sur la politique. Pour les demandes authentifiées, le formulaire doit inclure une signature valide et la politique.
3. Créez un formulaire HTML auquel vos utilisateurs peuvent accéder afin de charger des objets dans votre compartiment Amazon S3.

La section suivante décrit comment créer une signature pour authentifier une demande. Pour plus d'informations sur la création de formulaires et de politiques de sécurité, consultez [Création d'un formulaire HTML \(à l'aide de AWS Signature \(version 4\)\)](#).

Calculer une signature

Pour les demandes authentifiées, le formulaire HTML doit inclure des champs pour une politique de sécurité et une signature.

- Une politique de sécurité (voir [Politique POST](#)) contrôle ce qui est autorisé dans la demande.
- La politique de sécurité est StringToSign (voir [Présentation des demandes de signature](#)) dans le calcul de votre signature.



Pour calculer une signature

1. Créez une politique à l'aide de l' UTF-8 encodage.
2. Convertissez les octets UTF-8-encodés de politique en base64. Le résultat est `leStringToSign`.
3. Créez une clé de signature.
4. Utilisez la clé de signature pour signer l'algorithme de signature à l'`StringToSign` de l'algorithme de HMAC-SHA256 signature.

Pour plus d'informations sur la création de formulaires HTML, les politiques de sécurité et un exemple, consultez les rubriques suivantes :

- [Création d'un formulaire HTML \(à l'aide de AWS Signature \(version 4\)\)](#)
- [Politique POST](#)
- [Exemple : Browser-Based téléchargement à l'aide de HTTP POST \(à l'aide de AWS Signature \(version 4\)\)](#)

Création d'un formulaire HTML (à l'aide de AWS Signature (version 4))

Rubriques

- [Déclaration de formulaire HTML](#)
- [Champs de formulaire HTML](#)

Pour permettre aux utilisateurs de télécharger du contenu sur Amazon S3 à l'aide de leur navigateur (requêtes HTTP POST), vous utilisez des formulaires HTML. Les formulaires HTML se composent d'une déclaration de formulaire et de champs de formulaire. La déclaration de formulaire contient des informations de haut niveau sur la demande. Les champs du formulaire contiennent des informations détaillées sur la demande.

Cette section explique comment créer des formulaires HTML. Pour un exemple concret de téléchargement basé sur un navigateur à l'aide de HTTP POST et des calculs de signature associés

pour l'authentification des demandes, consultez. [Exemple : Browser-Based téléchargement à l'aide de HTTP POST \(à l'aide de AWS Signature \(version 4\)\)](#)

Le formulaire et la politique doivent être UTF-8 codés. Vous pouvez appliquer le UTF-8 codage au formulaire en le spécifiant `charset=UTF-8` dans l'`contentattribut`. Voici un exemple de UTF-8 codage dans le titre HTML.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

Voici un exemple d' UTF-8 encodage dans un en-tête de demande.

```
Content-Type: text/html; charset=UTF-8
```

Note

Les données et les limites du formulaire (à l'exclusion du contenu du fichier) ne peuvent pas dépasser 20 Ko.

Déclaration de formulaire HTML

La déclaration de formulaire HTML possède les trois attributs suivants :

- **action**— L'URL qui traite la demande, qui doit être définie sur l'URL du bucket. Par exemple, si le nom de votre compartiment est `examplebucket`, l'URL est `http://examplebucket.s3.amazonaws.com/`.

Note

Le nom de clé est spécifié dans un champ du formulaire.

- **method**— La méthode doit être POST.

- **enctype**— Le type de pièce jointe (**enctype**) doit être défini sur **multipart/form-data** pour les téléchargements de fichiers et les téléchargements de zones de texte. Pour plus d'informations à ce sujet **enctype**, consultez la [RFC 1867](#).

Il s'agit d'un formulaire de déclaration pour le bucket `examplebucket`.

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

Champs de formulaire HTML

Le tableau suivant décrit la liste des champs que vous pouvez utiliser dans un formulaire. Parmi les autres champs, il existe un champ de signature que vous pouvez utiliser pour authentifier les demandes. Certains champs vous permettent de spécifier l'algorithme de calcul de signature (**x-amz-algorithm**), l'étendue des informations d'identification (**x-amz-credential**) que vous avez utilisée pour générer la clé de signature et la date (**x-amz-date**) utilisée pour calculer la signature. Amazon S3 utilise ces informations pour recréer la signature. Si les signatures correspondent, Amazon S3 traite la demande.

Note

La variable `${filename}` est automatiquement remplacée par le nom du fichier fourni par l'utilisateur et est reconnue par l'ensemble des champs de formulaire. Si le navigateur ou le client fournit un chemin complet ou partiel vers le fichier, seul le texte suivant la dernière barre oblique (/) ou la dernière barre oblique inverse (\) est utilisé (par exemple, `C:\Program Files\directory1\file.txt` est interprété comme). `file.txt` Si aucun fichier ni nom de fichier n'est fourni, la variable est remplacée par une chaîne vide.

Si vous ne fournissez pas les éléments requis pour les demandes authentifiées, tels que l'**policy**élément, la demande est considérée comme anonyme et n'aboutira que si vous avez configuré le compartiment pour la lecture et l'écriture publiques.

Nom d'élément	Description	Obligatoire
<code>acl</code>		Non

Nom d'élément	Description	Obligatoire
	Une liste de contrôle d'accès (ACL) Amazon S3 Si une ACL non valide est spécifiée, Amazon S3 refuse la demande. Pour plus d'informations sur les listes de contrôle d'accès, consultez la section Utilisation des listes de contrôle d'accès Amazon S3. Type : Chaîne Valeur par défaut : private Valeurs valides : private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	
Cache-Control	REST-specific en-têtes. Pour de plus amples informations, veuillez consulter PutObject .	Non
Content-Type		
Content-Disposition		
Content-Encoding		
Expires		

Nom d'élément	Description	Obligatoire
key	Le nom clé de l'objet chargé. Pour utiliser le nom de fichier fourni par l'utilisateur, utilisez la variable <code>\${filename}</code>. Par exemple, si vous chargez un fichier <code>photo1.jpg</code> et que vous le spécifiez <code>/user/user1/\${filename}</code> comme nom de clé, le fichier est enregistré sous <code>/user/user1/photo1.jpg</code>. Pour plus d'informations, consultez la section Clé d'objet et métadonnées dans le guide de l'utilisateur d'Amazon Simple Storage Service.	Oui
policy	Politique de sécurité codée en base64 qui décrit ce qui est autorisé dans la demande. Pour les demandes authentifiées, une politique est requise. Les demandes sans politique de sécurité sont considérées comme anonymes et ne réussiront que sur un bucket accessible en écriture publique.	Obligatoire pour les demandes authentifiées

Nom d'élément	Description	Obligatoire
<code>success_action_redirect</code>	URL vers laquelle le client est redirigé en cas d'échec du chargement. Si <code>success_action_redirect</code> ce n'est pas spécifié, ou si Amazon S3 ne peut pas interpréter l'URL, Amazon S3 renvoie le type de document vide spécifié dans le <code>success_action_status</code> champ. Si le chargement échoue, Amazon S3 renvoie une erreur et ne redirige pas l'utilisateur vers une autre URL.	Non

Nom d'élément	Description	Obligatoire
success_action_status	Le code d'état renvoyé au client une fois le téléchargement réussi s'il n'<code>success_action_redirect</code> est pas spécifié. Les valeurs valides sont 200201, ou 204 (par défaut). Si la valeur est définie sur 200 ou 204, Amazon S3 renvoie un document vide avec le code de statut spécifié. Si la valeur est définie sur 201, Amazon S3 retourne un document XML avec un code de statut égal à 201. Pour plus d'informations sur le contenu du document XML, consultez POST Object. Si la valeur n'est pas définie ou n'est pas valide, Amazon S3 renvoie un document vide avec un code de statut 204.  Note Certaines versions d'Adobe Flash Player ne traitent pas correctement les réponses HTTP dont le corps est vide. Pour prendre en charge les chargements via Adobe Flash, nous vous recommandons de définir <code>success_action_status</code> sur 201.	Non

Nom d'élément	Description	Obligatoire
x-amz-algorithm	Algorithme de signature utilisé pour authentifier la demande. Pour AWS la version 4 de Signature, la valeur est AWS4-HMAC-SHA256. Ce champ est requis si un document de stratégie est inclus dans la demande.	Obligatoire pour les demandes authentifiées
x-amz-credential	Outre l'ID de votre clé d'accès, ce champ fournit également des informations d'étendue identifiant la région et le service pour lesquels la signature est valide. Il doit s'agir de la même étendue que celle que vous avez utilisée pour calculer la clé de signature pour le calcul de la signature. Il s'agit d'une chaîne de la forme suivante : <i><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</i> Par exemple : AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request Pour Amazon S3, la chaîne aws-service est. s3 Pour obtenir la liste des aws-region chaînes Amazon S3, consultez la section Régions et points de terminaison dans le Références générales AWS. Cela est nécessaire si un document de politique est inclus dans la demande.	Obligatoire pour les demandes authentifiées

Nom d'élément	Description	Obligatoire
x-amz-date	Il s'agit de la valeur de date au format ISO8601. Par exemple, 20130728T000000Z . Il s'agit de la même date que celle que vous avez utilisée lors de la création de la clé de signature (par exemple, 20130728). Il doit également s'agir de la même valeur que celle que vous avez fournie dans la politique (x-amz-date) que vous avez signée. Cela est nécessaire si un document de politique est inclus dans la demande.	Obligatoire pour les demandes authentifiées
x-amz-security-token	Un jeton de sécurité utilisé par Amazon DevPay et des informations d'identification de session Si la demande utilise Amazon DevPay, elle nécessite deux champs de x-amz-security-token formulaire : un pour le jeton du produit et un pour le jeton utilisateur. Pour plus d'informations, consultez la section Utilisation DevPay dans le guide de l'utilisateur d'Amazon Simple Storage Service. Si la demande utilise des informations d'identification de session, elle nécessite un x-amz-security-token formulaire. Pour plus d'informations, consultez Demande d'informations d'identification temporaires de sécurité dans le Guide de l'utilisateur IAM.	Non

Nom d'élément	Description	Obligatoire
x-amz-signature	(AWS Signature Version 4) Le HMAC-SHA256 hachage de la politique de sécurité. Ce champ est requis si un document de stratégie est inclus dans la demande.	Obligatoire pour les demandes authentifiées
x-amz-meta-*	Les noms de champs commençant par ce préfixe sont des métadonnées définies par l'utilisateur. Chacune est stockée et renvoyée sous la forme d'un ensemble de paires clé-valeur. Amazon S3 ne valide ni n'interprète les métadonnées définies par l'utilisateur. Pour de plus amples informations, veuillez consulter PutObject .	Non
x-amz-*	Voir POST Object (POST Object pour les autres x-amz-* en-têtes).	Non
file	Fichier ou contenu de texte. Le fichier ou le contenu doit être le dernier champ du formulaire. Vous ne pouvez pas charger plus d'un fichier à la fois.	Oui

Les éléments conditionnels sont obligatoires pour les demandes authentifiées et sont facultatifs pour les demandes anonymes.

Maintenant que vous savez comment créer des formulaires, vous pouvez ensuite créer une politique de sécurité que vous pouvez signer. Pour de plus amples informations, veuillez consulter [Politique POST](#).

Politique POST

Rubriques

- [Expiration](#)
- [Correspondance des conditions](#)
- [Conditions](#)
- [Echappement de caractère](#)

La politique requise pour effectuer des demandes authentifiées à l'aide de HTTP POST est un UTF-8 document codé en base64 écrit en notation JavaScript objet (JSON) qui spécifie les conditions que la demande doit remplir. Selon la manière dont vous concevez votre document de politique, vous pouvez contrôler la granularité de l'accès par téléchargement, par utilisateur, pour tous les téléchargements ou selon d'autres modèles répondant à vos besoins.

Cette section décrit la politique POST. Pour des exemples de calculs de signature à l'aide de la politique POST, consultez [Exemple : Browser-Based téléchargement à l'aide de HTTP POST \(à l'aide de AWS Signature \(version 4\)\)](#).

Note

Bien que le document de politique soit facultatif, nous vous recommandons vivement d'en utiliser un afin de contrôler ce qui est autorisé dans la demande. Si vous rendez le compartiment accessible en écriture publique, vous n'avez aucun contrôle sur les utilisateurs qui peuvent y écrire.

Voici un exemple de document de politique POST.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

La politique POST contient toujours les conditions éléments expiration et. L'exemple de politique utilise deux types de correspondance de conditions (correspondance exacte et correspondance commençant par une correspondance). Les sections suivantes décrivent ces éléments.

Expiration

L'expirationélément spécifie la date et l'heure d'expiration de la politique POST au format de date ISO8601 GMT. Par exemple, 2013-08-01T12:00:00.000Z spécifie que la politique POST n'est pas valide après minuit GMT le 1er août 2013.

Correspondance des conditions

Le tableau suivant décrit les types de correspondance de conditions que vous pouvez utiliser pour spécifier les conditions de politique POST (décrites dans la section suivante). Bien que vous deviez spécifier au moins une condition pour chaque champ de formulaire que vous spécifiez dans le formulaire, vous pouvez créer des critères de correspondance plus complexes en spécifiant plusieurs conditions pour un champ de formulaire.

Type de correspon dance de condition	Description
Correspon dances exactes	La valeur du champ de formulaire doit correspondre à la valeur spécifiée. Cet exemple indique que la liste ACL doit être définie sur public-read : <pre>{"acl": "public-read" }</pre> Cet exemple représente une méthode alternative pour indiquer que la liste ACL doit être définie sur public-read : <pre>["eq", "\$acl", "public-read"]</pre>
Commence par	La valeur doit commencer par la valeur spécifiée. Cet exemple indique que la clé de l'objet doit commencer par user/user1 :

Type de correspondance de condition	Description
	<pre>["starts-with", "\$key", "user/user1/"]</pre>
Correspondance Content-Types dans une Comma-Separated liste	Content-Types les valeurs d'une <code>starts-with</code> condition contenant des virgules sont interprétées comme des listes. Chaque valeur de la liste doit remplir la condition pour que l'ensemble de la condition soit réussi. Par exemple, étant donné la condition suivante : <pre>["starts-with", "\$Content-Type", "image/"]</pre> La valeur suivante satisferait à la condition : <pre>"image/jpg,image/png,image/gif"</pre> La valeur suivante ne satisferait pas à la condition : <pre>["image/jpg,text/plain"]</pre>  Note Les éléments de données autres que ceux <code>Content-Type</code> sont traités comme des chaînes, indépendamment de la présence de virgules.
Correspondance avec un contenu quelconque	Pour configurer la politique POST afin d'autoriser tout contenu dans un champ de formulaire, utilisez <code>starts-with</code> avec une valeur vide (« »). Cet exemple autorise n'importe quelle valeur pour <code>success_action_redirect</code> : <pre>["starts-with", "\$success_action_redirect", ""]</pre>

Type de correspondance de condition	Description
Spécification de plages	Pour les champs de formulaire qui acceptent une plage, séparez les limites supérieure et inférieure par une virgule. Cet exemple autorise une taille de fichier comprise entre 1 et 10 Mo : <pre>["content-length-range", 1048576, 10485760]</pre>

Les conditions spécifiques prises en charge dans une politique POST sont décrites dans [Conditions](#).

Conditions

Dans une politique POST, il y a un tableau d'objets, dont chacun est utilisé pour valider la demande. Vous pouvez utiliser ces conditions pour restreindre ce qui est autorisé dans la demande. Par exemple, les conditions de politique précédentes exigent ce qui suit :

- La demande doit spécifier le nom du johnsmith compartiment.
- Le nom de la clé de l'objet doit comporter le user/eric préfixe.
- L'ACL de l'objet doit être définie sur public-read.

Chaque champ de formulaire que vous spécifiez dans un formulaire (à l'exception des noms de champs x-amz-signature filepolicy,, et des noms de champs x-ignore- comportant un préfixe) doit figurer dans la liste des conditions.

Note

Toutes les variables du formulaire sont développées avant la validation de la politique POST. Par conséquent, toutes les correspondances de conditions doivent se faire par rapport aux champs de formulaire étendus. Supposons que vous souhaitiez limiter le nom de la clé de votre objet à un préfixe spécifique (user/user1). Dans ce cas, vous définissez le champ du formulaire clé sur user/user1/\${filename}. Votre politique POST doit être ["starts-with", "\$key", "user/user1/"] (ne pas entrer ["starts-with", "\$key",

"user/user1/\${filename}"]). Pour de plus amples informations, veuillez consulter [Correspondance des conditions](#).

Les conditions du document de politique sont décrites dans le tableau suivant.

Nom d'élément	Description
acl	Spécifie la valeur ACL qui doit être utilisée lors de la soumission du formulaire. Cette condition prend en charge la correspondance exacte et le type de correspondance des <code>starts-with</code> conditions discuté dans la section suivante.
bucket	Spécifie le nom de compartiment acceptable. Cette condition prend en charge le type de correspondance de condition correspondant exactement.
content-length-range	Taille minimale et maximale autorisée pour le contenu téléchargé. Cette <code>content-length-range</code> condition prend en charge le type de correspondance des conditions.
Cache-Control Content-Type Content-Disposition Content-Encoding Expires	REST-specific en-têtes. Pour de plus amples informations, veuillez consulter POST Object. Cette condition prend en charge la correspondance exacte et le type de correspondance des <code>starts-with</code> conditions.
key	Le nom de clé acceptable ou un préfixe de l'objet chargé.

Nom d'élément	Description
	Cette condition prend en charge la correspondance exacte et le type de correspondance des <code>starts-with</code> conditions.
<code>success_action_redirect</code> <code>redirect</code>	URL vers laquelle le client est redirigé en cas d'échec du chargement. Cette condition prend en charge la correspondance exacte et le type de correspondance des <code>starts-with</code> conditions.
<code>success_action_status</code>	Le code d'état renvoyé au client une fois le téléchargement réussi s'il n'y a pas de <code>success_action_redirect</code> spécifié. Cette condition permet une correspondance exacte.
<code>x-amz-algorithm</code>	Algorithme de signature qui doit être utilisé lors du calcul de signature. Pour AWS la version 4 de Signature, la valeur est <code>AWS4-HMAC-SHA256</code>. Cette condition permet une correspondance exacte.

Nom d'élément	Description
<code>x-amz-credential</code>	Les informations d'identification que vous avez utilisées pour calculer la signature. Il fournit l'identifiant de la clé d'accès et les informations de portée identifiant la région et le service pour lesquels la signature est valide. Il doit s'agir de la même étendue que celle que vous avez utilisée pour calculer la clé de signature pour le calcul de la signature. Il s'agit d'une chaîne de la forme suivante : <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Par exemple : <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Pour Amazon S3, la chaîne <code>aws-service</code> est <code>s3</code>. Pour obtenir la liste des <code>aws-region</code> chaînes Amazon S3, consultez la section Régions et points de terminaison dans le Référence s générales AWS. Cela est obligatoire si un document de politique POST est inclus dans la demande. Cette condition permet une correspondance exacte.
<code>x-amz-date</code>	La valeur de date spécifiée dans la chaîne au format ISO8601. Par exemple, <code>20130728T000000Z</code> . La date doit être la même que celle que vous avez utilisée lors de la création de la clé de signature pour le calcul de la signature. Cela est obligatoire si un document de politique POST est inclus dans la demande. Cette condition permet une correspondance exacte.

Nom d'élément	Description
<code>x-amz-security-token</code>	Jeton DevPay de sécurité Amazon. Chaque demande qui utilise Amazon DevPay nécessite deux champs de <code>x-amz-security-token</code> : un pour le jeton du produit et un pour le jeton utilisateur. Par conséquent, les valeurs doivent être séparées par des virgules. Par exemple, si le jeton utilisateur est <code>eW91dHVIZQ==</code> et que le jeton de produit l'est <code>b0hnNVNKWVJIQTA=</code>, vous définissez l'entrée de politique POST sur : <pre>{ "x-amz-security-token": "eW91dHVIZQ==,b0hnNVNKWVJIQTA=" }</pre> Pour plus d'informations sur Amazon DevPay, consultez la section Utilisation DevPay dans le guide de l'utilisateur d'Amazon Simple Storage Service.
<code>x-amz-meta-*</code>	User-specified métadonnées. Cette condition prend en charge la correspondance exacte et le type de correspondance des <code>starts-with</code> conditions.
<code>x-amz-*</code>	Voir POST Object (POST Object pour les autres <code>x-amz-*</code> en-têtes). Cette condition permet une correspondance exacte.

 Note

Si votre boîte à outils ajoute d'autres champs de formulaire (par exemple, `Flash ajoutefilename`), vous devez les ajouter au document de politique POST. Si vous pouvez contrôler cette fonctionnalité, ajoutez le préfixe `x-ignore-` au champ afin qu'Amazon S3 ignore la fonction et que les versions futures de cette fonction ne soient pas affectées.

Echappement de caractère

Les caractères qui doivent être supprimés dans un document de politique POST sont décrits dans le tableau suivant.

Séquence d'échappement	Description
\\	Barre oblique inverse
\\\$	Symbole du dollar
\\b	Retour arrière
\\f	Saut de page
\\n	Nouvelle ligne
\\r	Retour chariot
\\t	Tabulation horizontale
\\v	Tabulation verticale
\\u xxxx	Tous les caractères Unicode

Maintenant que vous connaissez les formulaires et les politiques, et que vous comprenez comment fonctionne la signature, vous pouvez essayer un exemple de téléchargement POST. Vous devez écrire le code pour calculer la signature. L'exemple fournit un exemple de formulaire et une politique POST que vous pouvez utiliser pour tester vos calculs de signature. Pour de plus amples informations, veuillez consulter [Exemple : Browser-Based téléchargement à l'aide de HTTP POST \(à l'aide de AWS Signature \(version 4\)\)](#).

Exemple : Browser-Based téléchargement à l'aide de HTTP POST (à l'aide de AWS Signature (version 4))

Cette section présente un exemple d'utilisation d'une requête HTTP POST pour charger du contenu directement sur Amazon S3.

Pour plus d'informations sur Signature Version 4, voir Processus de [signature Signature Version 4](#).

Chargement d'un fichier sur Amazon S3 à l'aide du protocole HTTP POST

Cet exemple fournit un exemple de politique POST et un formulaire que vous pouvez utiliser pour charger un fichier. La rubrique utilise l'exemple de politique et des informations d'identification fictives pour vous montrer le flux de travail et le hachage de signature et de stratégie qui en résultent. Vous pouvez utiliser ces données comme suite de tests pour vérifier votre code de calcul de signature.

L'exemple utilise les informations d'identification suivantes pour les calculs de signature. Vous pouvez utiliser ces informations d'identification pour vérifier votre code de calcul de signature. Cependant, vous devez ensuite les remplacer par vos propres informations d'identification lorsque vous envoyez des demandes à AWS.

Paramètre	Value
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

Exemple de politique et de formulaire

La politique POST suivante prend en charge les chargements vers Amazon S3 sous certaines conditions.

```
{ "expiration": "2015-12-30T12:00:00.000Z",  
  "conditions": [  
    {"bucket": "sigv4examplebucket"},  
    ["starts-with", "$key", "user/user1/"],  
    {"acl": "public-read"},
```

```

    {"success_action_redirect": "http://sigv4examplebucket.s3.amazonaws.com/
successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    {"x-amz-server-side-encryption": "AES256"},
    ["starts-with", "$x-amz-meta-tag", ""],

    {"x-amz-credential": "AKIAIOSFODNN7EXAMPLE/20151229/us-east-1/s3/aws4_request"},
    {"x-amz-algorithm": "AWS4-HMAC-SHA256"},
    {"x-amz-date": "20151229T000000Z" }
  ]
}

```

Cette politique POST définit les conditions suivantes pour la demande :

- Le téléchargement doit avoir lieu avant midi UTC le 30 décembre 2015.
- Le contenu ne peut être téléchargé que sur `sigv4examplebucket`. Le compartiment doit se trouver dans la région que vous avez spécifiée dans l'étendue des informations d'identification (paramètre du `x-amz-credential` formulaire), car la signature que vous avez fournie n'est valide que dans cette étendue.
- Vous pouvez fournir n'importe quel nom de clé commençant par `user/user1`. Par exemple, `user/user1/MyPhoto.jpg`.
- L'ACL doit être définie sur `public-read`.
- Si le téléchargement aboutit, le navigateur de l'utilisateur est redirigé vers `http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html`.
- L'objet doit être un fichier image.
- La `x-amz-meta-uuid` balise doit être définie sur `14365123651274`.
- Le `x-amz-meta-tag` peut contenir n'importe quelle valeur.

Voici une Base64-encoded version de cette politique POST. Vous utilisez cette valeur pour le calcul `StringToSign` de votre signature.

```
eyAiZXhwaXJhdGlvbiI6IClyMDE1LTEyLTMwVDEyOjAwOjAwLjAwMFoiLA0KICAiY29uZGl0aW9ucyI6IFsNCiAgICB7ImJ
```

Lorsque vous utilisez copy/paste la politique précédente, elle devrait comporter des retours de chariot et de nouvelles lignes pour que votre hachage calculé corresponde à cette valeur (c'est-à-dire. Texte ASCII, avec terminaisons de ligne CRLF).

À l'aide d'exemples d'informations d'identification pour créer une signature, la valeur de signature est la suivante (dans le calcul de la signature, la date est la même que x-amz-date dans la politique (20151229)) :

```
8afdbf4008c03f22c2cd3cdb72e4afbb1f6a588f3255ac628749a66d7f09699e
```

L'exemple de formulaire suivant spécifie la politique POST précédente et prend en charge une requête POST adressée à `sigv4examplebucket`. Copy/paste le contenu dans un éditeur de texte et enregistrez-le sous le nom `exampleform.html`. Vous pouvez ensuite télécharger des fichiers image dans le compartiment concerné à l'aide du fichier `exampleform.html`. Votre demande aboutira si la signature que vous fournissez correspond à la signature calculée par Amazon S3.

Note

Vous devez mettre à jour le nom du compartiment, les dates, les informations d'identification, la politique et la signature avec des valeurs valides pour que le téléchargement vers S3 soit réussi.

```
<html>
<head>

  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

</head>
<body>

  <form action="http://sigv4examplebucket.s3.amazonaws.com/" method="post"
  enctype="multipart/form-data">
    Key to upload:
    <input type="input" name="key" value="user/user1/${filename}" /><br />
    <input type="hidden" name="acl" value="public-read" />
    <input type="hidden" name="success_action_redirect" value="http://
sigv4examplebucket.s3.amazonaws.com/successful_upload.html" />
    Content-Type:
    <input type="input" name="Content-Type" value="image/jpeg" /><br />
    <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
    <input type="hidden" name="x-amz-server-side-encryption" value="AES256" />
    <input type="text" name="X-Amz-Credential" value="AKIAIOSFODNN7EXAMPLE/20151229/
us-east-1/s3/aws4_request" />
```

```
<input type="text"    name="X-Amz-Algorithm" value="AWS4-HMAC-SHA256" />
<input type="text"    name="X-Amz-Date" value="20151229T000000Z" />

Tags for File:
<input type="input"   name="x-amz-meta-tag" value="" /><br />
<input type="hidden" name="Policy" value='<Base64-encoded policy string>' />
<input type="hidden" name="X-Amz-Signature" value="<signature-value>" />
File:
<input type="file"    name="file" /> <br />
<!-- The elements after this will be ignored -->
<input type="submit" name="submit" value="Upload to Amazon S3" />
</form>

</html>
```

Les paramètres de publication ne font pas la distinction entre majuscules et minuscules. Par exemple, vous définissez `x-amz-signature` ou `X-Amz-Signature`.

Browser-based les téléchargements vers Amazon S3 à l'aide du AWS Bibliothèque Amplify

Cette section explique comment charger des fichiers sur Amazon S3 à l'aide de la JavaScript bibliothèque AWS Amplify.

Pour plus d'informations sur la configuration de la bibliothèque AWS Amplify, voir [Installation et configuration d'AWS Amplify](#).

Utilisation de AWS JavaScript Bibliothèque Amplify pour charger des fichiers vers Amazon S3

Le `Storage` module de bibliothèque AWS Amplify fournit un mécanisme de téléchargement simple basé sur un navigateur pour gérer le contenu utilisateur dans un stockage Amazon S3 public ou privé.

Exemple: AWS Configuration manuelle d'Amplifier

L'exemple suivant montre la configuration manuelle pour utiliser le module `AWS Amplify Storage`. L'implémentation par défaut du module de stockage utilise Amazon S3.

```
import Amplify from 'aws-amplify';
```

```
Amplify.configure(  
  Auth: {  
    identityPoolId: 'XX-XXXX-X:XXXXXXXXX-XXXX-1234-abcd-1234567890ab', //REQUIRED -  
    Amazon Cognito Identity Pool ID  
    region: 'XX-XXXX-X', // REQUIRED - Amazon Cognito Region  
    userPoolId: 'XX-XXXX-X_abcd1234', //OPTIONAL - Amazon Cognito User Pool ID  
    userPoolWebClientId: 'XX-XXXX-X_abcd1234', //OPTIONAL - Amazon Cognito Web  
    Client ID  
  },  
  Storage: {  
    bucket: '', //REQUIRED - Amazon S3 bucket  
    region: 'XX-XXXX-X', //OPTIONAL - Amazon service region  
  }  
);
```

Exemple: Insérez des données dans Amazon S3

L'exemple suivant montre comment put publier des données dans Amazon S3.

```
Storage.put('test.txt', 'Hello')  
  .then (result => console.log(result))  
  .catch(err => console.log(err));
```

L'exemple suivant montre comment put privatiser des données dans Amazon S3.

```
Storage.put('test.txt', 'Private Content', {  
  level: 'private',  
  contentType: 'text/plain'  
})  
  .then (result => console.log(result))  
  .catch(err => console.log(err));
```

Pour plus d'informations sur l'utilisation du module AWS Amplify Storage, voir [AWS Amplify Storage](#).

Plus d'informations

[AWS Amplifier le démarrage rapide](#)

En-têtes de requête courants

Le tableau suivant décrit les en-têtes qui peuvent être utilisés par différents types de requêtes REST Amazon S3.

Nom de l'en-tête	Description
Authorization	Les informations requises pour l'authentification de la demande. Pour plus d'informations, consultez l'en-tête d'authentification du guide du développeur Amazon Simple Storage Service. Pour les demandes anonymes, cet en-tête n'est pas obligatoire.
Access-Control-Request-Method	Liste des méthodes HTTP envoyées sous forme de requête CORS avant le vol. Si l'évaluation CORS avant vol est réussie, les méthodes spécifiées peuvent être utilisées dans la demande CORS suivante.
Content-Length	Longueur du message (sans les en-têtes) selon la RFC 2616. Cet en-tête est requis pour les PUT et les opérations qui chargent du XML, telles que la journalisation et les ACL.
Content-Type	Type de contenu de la ressource si le corps de la demande contient du contenu. Exemple : <code>text/plain</code>
Content-MD5	Le condensé MD5 128 bits codé en base64 du message (sans les en-têtes) conformément à la RFC 1864. Cet en-tête peut être utilisé pour vérifier l'intégrité des messages afin de vérifier que les données sont les mêmes que celles qui ont été envoyées à l'origine. Bien qu'il soit facultatif, nous vous recommandons d'utiliser le Content-MD5 mécanisme pour vérifier l'intégrité de bout en bout. Pour plus d'informations sur l'authentification des requêtes REST, consultez la rubrique Authentification REST dans le manuel Amazon Simple Storage Service Developer Guide.

Nom de l'en-tête	Description
Date	Date qui peut être utilisée pour créer la signature contenue dans l'en-tête <code>Authorization</code> . Si l'en-tête <code>Date</code> doit être utilisé pour la signature, il doit être spécifié au format de base ISO 8601. Dans ce cas, l'en-tête <code>x-amz-date</code> n'est pas nécessaire. Notez que lorsque <code>x-amz-date</code> est présent, il remplace toujours la valeur de l'en-tête <code>Date</code>. Si l'en-tête <code>Date</code> n'est pas utilisé pour la signature, il peut se présenter sous l'un des formats de date complets spécifiés par RFC 2616, section 3.3. Par exemple, date/time <code>Wed, 01 Mar 2006 12:00:00 GMT</code> il s'agit d'un date/time en-tête valide à utiliser avec Amazon S3. Si vous utilisez l'en-tête <code>Date</code> pour la signature, il doit être au format de base ISO 8601 <code>YYYYMMDD'T'HHMMSS'Z'</code> . S'il <code>Date</code> est spécifié mais n'est pas au format de base ISO 8601, vous devez également inclure l'<code>x-amz-date</code> en-tête. Si <code>Date</code> est spécifié au format de base ISO 8601, cela suffit pour signer les demandes et l'en-tête <code>x-amz-date</code> n'est pas nécessaire. Pour plus d'informations, consultez la page Gestion des dates dans Signature Version 4 dans le Glossaire Amazon Web Services.
Expect	Lorsque votre application utilise <code>100-continue</code>, elle n'envoie pas le corps de la demande tant qu'elle n'a pas reçu d'accusé de réception. Si le message est rejeté sur la base des en-têtes, le corps du message n'est pas envoyé. Cet en-tête ne peut être utilisé que si vous envoyez un corps. Valeurs valides : <code>100-continuer</code>

Nom de l'en-tête	Description
Host	Pour les requêtes de type chemin, la valeur est. <code>s3.amazonaws.com</code> Pour les requêtes de style virtuel, la valeur est. <code>BucketName.s3.amazonaws.com</code> Pour plus d'informations, consultez la rubrique Hébergement virtuel dans le guide de l'utilisateur d'Amazon Simple Storage Service. Cet en-tête est obligatoire pour HTTP 1.1 (la plupart des boîtes à outils ajoutent cet en-tête automatiquement) ; facultatif pour les HTTP/1.0 requêtes.
Origin	Point de terminaison qui spécifie le nom du serveur du demandeur initial.
x-amz-content-sha256	Lorsque vous utilisez la version 4 de la signature pour authentifier une demande, cet en-tête fournit un hachage de la charge utile de la demande. Pour de plus amples informations, veuillez consulter Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en un seul bloc (AWS Signature (version 4)). Lorsque vous chargez un objet en morceaux, vous définissez la valeur <code>STREAMING-AWS4-HMAC-SHA256-PAYLOAD</code> pour indiquer que la signature ne couvre que les en-têtes et qu'il n'y a aucune charge utile. Pour de plus amples informations, veuillez consulter Calculs de signature pour l'en-tête d'autorisation : transfert de la charge utile en plusieurs segments (téléchargement fragmenté) (AWS Signature (version 4)).

Nom de l'en-tête	Description
<code>x-amz-date</code>	Date utilisée pour créer la signature dans l'en-tête <code>Authorization</code>. Le format doit correspondre à la norme ISO 8601 de base au format <code>YYYYMMDD'T'HHMMSS'Z'</code>. Par exemple, <code>date/time 20170210T120000Z</code> il est valide <code>x-amz-date</code> pour une utilisation avec Amazon S3. <code>x-amz-date</code> est facultatif pour toutes les demandes ; il peut être utilisé pour modifier la date utilisée pour la signature des demandes. Si l'en-tête <code>Date</code> est spécifié au format de base ISO 8601, <code>x-amz-date</code> n'est pas nécessaire. Lorsque <code>x-amz-date</code> est présent, il remplace toujours la valeur de l'en-tête <code>Date</code>. Pour plus d'informations, consultez la page Gestion des dates dans Signature Version 4 dans le Glossaire Amazon Web Services.

Nom de l'en-tête	Description
x-amz-security-token	Cet en-tête peut être utilisé dans les scénarios suivants : • Pour fournir des jetons de sécurité pour les DevPay opérations Amazon : chaque demande utilisant Amazon DevPay nécessite deux x-amz-security-token en-têtes : un pour le jeton du produit et un pour le jeton utilisateur. Lorsqu'Amazon S3 reçoit une demande authentifiée, il compare la signature calculée à la signature fournie. Les en-têtes à valeurs multiples mal formatés utilisés pour calculer une signature peuvent entraîner des problèmes d'authentification. • Pour fournir un jeton de sécurité lors de l'utilisation d'informations d'identification de sécurité temporaires - Lorsque vous envoyez des demandes à l'aide des informations d'identification de sécurité temporaires que vous avez obtenues auprès d'IAM, vous devez fournir un jeton de sécurité à l'aide de cet en-tête. Pour en savoir plus sur les informations d'identification de sécurité temporaires, voir Faire des demandes. Cet en-tête est obligatoire pour les demandes qui utilisent Amazon DevPay et les demandes qui sont signées à l'aide d'informations d'identification de sécurité temporaires.

En-têtes de réponse courants

Le tableau suivant décrit les en-têtes de réponse communs à la plupart des réponses Amazon S3.

Nom	Description
Access-Control-Allow-Credentials	Booléen qui détermine si le serveur autorise les requêtes CORS à contenir des informations d'identification. Si l'en-tête de Access-Control-Allow-Origin demande est défini sur '*' alors l'en-tête de Access-Control-Allow-Credentials réponse sera omis, sinon il est défini sur true lorsque l'évaluation CORS est réussie. Type : Boolean Par défaut : aucun
Access-Control-Allow-Headers	Liste des en-têtes HTTP autorisés pour vos requêtes CORS. L'en-tête de Access-Control-Allow-Headers réponse est renvoyé pour les évaluations CORS réussies et spécifie explicitement tout ce qui est autoriséAccess-Control-Request-Headers . Type : Chaîne Par défaut : aucun
Access-Control-Allow-Methods	Liste qui indique quelles méthodes HTTP sont autorisées. Amazon S3 n'autorisera les requêtes CORS provenant de méthodes CORS autorisées que si l'évaluation CORS est réussie. Type : Chaîne Par défaut : aucun
Access-Control-Allow-Origin	L'emplacement de l'origine autorisée. Amazon S3 n'enverra l'en-tête de Access-Control-Allow-Origin réponse que lorsque l'évaluation CORS est réussie. Si l'origine de la demande correspond '*' aux origines autorisées par la configuration CORS, elle '*' est renvoyée dans cet en-tête de réponse au lieu de l'origine d'origine.

Nom	Description
	Type : Chaîne Par défaut : aucun
Access-Control-Expose-Headers	Liste qui permet à un serveur d'identifier un en-tête de réponse qui expose l'accès aux applications lorsque l'évaluation CORS est réussie. Type : Chaîne Par défaut : aucun
Access-Control-Max-Age	Durée en secondes pendant laquelle votre navigateur peut mettre en cache la réponse à une demande de pré-vol CORS identifiée par la ressource, la méthode HTTP et l'origine. L'en-tête de Access-Control-Max-Age réponse n'est renvoyé que lorsque l'évaluation CORS est réussie. Type : Integer Par défaut : aucun
Vary	Une liste qui indique sur quels en-têtes de demande varie le résultat de l'évaluation CORS. L'en-tête de Vary réponse n'est renvoyé que lorsque l'évaluation CORS est réussie. Type : Chaîne Par défaut : aucun
Content-Length	Longueur en octets du corps de la réponse. Type : Chaîne Par défaut : aucun
Content-Type	Type MIME du contenu. Par exemple, Content-Type: text/html; charset=utf-8 . Type : Chaîne Par défaut : aucun

Nom	Description
Connection	Valeur qui indique si la connexion au serveur est ouverte ou fermée. Type : énumération Valeurs valides : open close Par défaut : aucun
Date	Date et heure auxquelles Amazon S3 a répondu ; par exemple, le mercredi 1er mars 2006 à 12:00:00 GMT. Type : Chaîne Par défaut : aucun
ETag	La balise d'entité (ETag) représente une version spécifique de l'objet. ETag reflète les modifications uniquement appliquées au contenu d'un objet, pas ses métadonnées. ETag peut ou ne peut pas être une valeur de hachage MD5 des données de l'objet. Cela dépend de la manière dont l'objet a été créé et de la manière dont il est chiffré, comme suit : • Objets créés par le Console de gestion AWS biais de l'PUTobjet ou POST Object de Copy l'opération : • Les objets en texte clair ou chiffrés par chiffrement côté serveur avec des clés gérées par Amazon S3 (SSE-S3) possèdent des ETags qui sont un condensé MD5 de leurs données. • Les objets chiffrés par chiffrement côté serveur à l'aide de clés (SSE-C) ou AWS Key Management Service () clés () fournies par le client possèdent des ETags qui ne sont pas un condensé MD5 de leurs données d'objet.AWS KMS SSE-KMS • Les objets créés par l'opération de téléchargement partitionné ou de téléchargement partiel comportent des ETags qui ne sont pas des résumés MD5, quelle que soit la méthode de cryptage. Type : Chaîne

Nom	Description
Server	Le nom du serveur qui a créé la réponse. Type : Chaîne Valeur par défaut : AmazonS3
x-amz-delete-marker	Valeur qui indique si l'objet renvoyé était (true) ou non (false) un marqueur de suppression. Type : Boolean Valeurs valides : true false Valeur par défaut : false
x-amz-id-2	Jeton spécial utilisé conjointement avec l'<code>x-amz-request-id</code> en-tête pour AWS résoudre les problèmes. Pour plus d'informations sur Support l'utilisation de ces ID de demande, consultez Résolution des problèmes liés à Amazon S3. Type : Chaîne Par défaut : aucun
x-amz-request-id	Valeur créée par Amazon S3 qui identifie la demande de manière unique. Cette valeur est utilisée conjointement avec l'<code>x-amz-id-2</code> en-tête pour aider à AWS résoudre les problèmes. Pour plus d'informations sur Support l'utilisation de ces ID de demande, consultez Résolution des problèmes liés à Amazon S3. Type : Chaîne Par défaut : aucun
x-amz-server-side-encryption	Algorithme de chiffrement côté serveur utilisé lors du stockage de cet objet dans Amazon S3 (par exemple, AES256, aws:kms). Valeurs valides : AES256 aws:kms

Nom	Description
<code>x-amz-version-id</code>	Version de l'objet. Lorsque vous activez le contrôle de version, Amazon S3 génère un nombre aléatoire pour les objets ajoutés à un compartiment. La valeur est UTF-8 codée et l'URL est prête. Lorsque vous placez PUT un objet dans un compartiment dont la gestion des versions a été suspendue, l'ID de version est toujours <code>null</code>. Type : Chaîne Valeurs valides : <code>null</code> n'importe quelle URL-ready chaîne UTF-8 codée Par défaut:<code>null</code>

Réponses d'erreur

Cette section fournit des informations de référence sur les erreurs Amazon S3.

Note

- En général, les propriétaires de compartiments S3 se voient facturer les demandes comportant des réponses positives HTTP 200 OK et des réponses d'erreur client HTTP 4XX. Les propriétaires de compartiments ne se voient pas facturer les réponses d'erreur de serveur HTTP 5XX, telles que les erreurs HTTP 503 Slow Down. Pour plus d'informations sur les codes d'erreur S3 sous HTTP 3XX et les codes de 4XX statut qui ne sont pas facturés, consultez la section [Facturation des réponses d'erreur Amazon S3](#) dans le guide de l'utilisateur Amazon S3. Pour plus d'informations sur les frais de facturation si votre compartiment est configuré comme un compartiment Requester Pays, consultez [Comment fonctionnent les frais Requester Pays](#) dans le guide de l'utilisateur Amazon S3.
- La prise en charge de SOAP sur HTTP est obsolète, mais SOAP continue d'être disponible sur HTTP. Les nouvelles fonctions Amazon S3 ne sont pas prises en charge pour SOAP. Au lieu d'utiliser SOAP, nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

Rubriques

- [Réponses aux erreurs REST](#)
- [Liste des codes d'erreur](#)
- [Liste des codes d'erreur relatifs au contenu de l'objet SELECT](#)
- [Liste des codes Replication-related d'erreur](#)
- [Liste des codes Tagging-related d'erreur](#)
- [Liste des codes d'erreur liés aux annotations Amazon S3](#)
- [Liste des codes d'erreur Amazon S3 on Outposts](#)
- [Liste des codes d'erreur d'Amazon S3 Storage Lens](#)
- [Liste des codes d'erreur Amazon S3 Object Lambda](#)
- [Liste des codes d'erreur asynchrones Amazon S3](#)
- [Liste des codes d'erreur Amazon S3 Access Grants](#)

- [Liste des codes FSx-related d'erreur Amazon](#)
- [Liste des codes d'erreur d'Amazon S3 Tables](#)
- [Bonnes pratiques concernant les erreurs Amazon S3](#)

Réponses aux erreurs REST

En cas d'erreur, les informations d'en-tête contiennent les informations suivantes :

- Content-Type: application/xml
- un code de statut HTTP 3xx, 4xx ou 5xx.

Le corps de la réponse contient également des informations sur l'erreur. L'exemple de réponse d'erreur suivant montre la structure des éléments de réponse communs à toutes les réponses d'erreur REST.

```
<?xml version="1.0" encoding="UTF-8"?>
<Error>
  <Code>NoSuchKey</Code>
  <Message>The resource you requested does not exist</Message>
  <Resource>/mybucket/myfoto.jpg</Resource>
  <RequestId>4442587FB7D0A2F9</RequestId>
</Error>
```

Le tableau suivant explique les éléments de réponse aux erreurs REST.

Nom	Description
Code	Le code d'erreur est une chaîne qui identifie de façon univoque une condition d'erreur. Il doit être lu et compris par des programmes qui détectent et gèrent des erreurs par type. Pour plus d'informations, consultez la section Liste des codes d'erreur. Type : Chaîne Ancêtre : Error
Error	Conteneur pour tous les éléments d'erreur.

Nom	Description
	Type : Conteneur Ancêtre : Aucun
Message	Le message d'erreur contient une description générique de la condition d'erreur en anglais. Il s'adresse à un public d'individus. Des programmes basiques affichent le message directement pour l'utilisateur final s'ils trouvent une condition d'erreur qu'ils ne savent, ni souhaitent gérer. Des programmes sophistiqués avec une gestion d'erreur plus complète et une stratégie d'internationalisation personnalisée risquent plus d'ignorer le message d'erreur. Type : Chaîne Ancêtre : Error
RequestId	ID de la demande associée à l'erreur. Type : Chaîne Ancêtre : Error
Resource	Le compartiment ou l'objet impliqué dans l'erreur. Type : Chaîne Ancêtre : Error

Beaucoup de réponses d'erreur contiennent des données structurées complémentaires destinées à être lues et comprises par un développeur diagnostiquant des erreurs de programmation. Par exemple, si vous envoyez un Content-MD5 en-tête avec une requête REST PUT qui ne correspond pas au résumé calculé sur le serveur, vous recevez un BadDigest message d'erreur. La réponse d'erreur inclut également comme éléments de détail le résumé calculé par le serveur et le résumé que vous avez indiqué au serveur d'attendre. Pendant le développement, vous pouvez utiliser ces informations pour diagnostiquer l'erreur. En production, un programme performant doit comprendre ces informations dans son journal des erreurs.

Pour plus d'informations sur les éléments de réponse généraux, consultez la rubrique [Réponses aux erreurs](#).

Liste des codes d'erreur

Le tableau suivant répertorie les codes d'erreur Amazon S3.

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
AccessControlListNotSupported	Le bucket n'autorise pas les ACL.	400 Bad Request	Client
AccessDenied	Accès refusé	403 Forbidden	Client
AccessPointAlreadyOwnedByYou	Un point d'accès portant le même nom existe déjà dans votre compte.	409 Conflict	Client
AccountProblem	Un problème vous empêche Compte AWS de mener à bien l'opération. Pour obtenir de l'aide, consultez Contactez-nous .	403 Forbidden	Client
AllAccessDisabled	Tous les accès à cette ressource Amazon S3 ont été désactivés. Pour obtenir de l'aide, consultez Contactez-nous .	403 Forbidden	Client
AmbiguousGrantByEmailAddress	L'adresse e-mail que vous avez fournie est associée à plusieurs comptes.	400 Bad Request	Client
AuthorizationHeaderMalformed	L'en-tête d'autorisation que vous avez fourni n'est pas valide.	400 Bad Request	N/A

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
AuthorizationQueryParametersError	Les paramètres de requête d'autorisation que vous avez fournis ne sont pas valides.	400 Bad Request	N/A
BadDigest	La valeur Content-MD5 ou la somme de contrôle que vous avez spécifiée ne correspond pas à ce que le serveur a reçu.	400 Bad Request	Client
BucketAlreadyExists	Le nom de compartiment demandé n'est pas disponible. L'espace de noms de compartiments est partagé par tous les utilisateurs du système. Spécifiez un nom différent et réessayez.	409 Conflict	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
BucketAlreadyOwnedByYou	Le compartiment que vous avez essayé de créer existe déjà et vous en êtes le propriétaire. Amazon S3 renvoie cette erreur partout Régions AWS sauf dans la région USA Est (Virginie du Nord) (us-east-1). Pour des raisons de compatibilité avec les versions précédentes, si vous recréez un compartiment existant que vous possédez déjà dans us-east-1, Amazon S3 renvoie 200 OK et réinitialise les listes de contrôle d'accès (ACL) du compartiment. Pour Amazon S3 on Outposts, le compartiment que vous avez essayé de créer existe déjà dans votre Outpost et vous en êtes propriétaire.	409 Conflict toutes les régions sauf us-east-1)	Client
BucketHasAccessPointsAttached	Des points d'accès sont attachés au compartiment que vous avez essayé de supprimer. Supprimez vos points d'accès avant de supprimer votre bucket.	400 Bad Request	Client
BucketNotEmpty	Le compartiment que vous avez essayé de supprimer n'est pas vide.	409 Conflict	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>ClientTokenConflict</code>	Le jeton d'idempotence de votre point d' Multi-Region accès a déjà été utilisé pour une autre demande.	409 Conflict	Client
<code>ConnectionClosedByRequester</code>	Renvoyé à l'appelant d'origine lorsqu'une erreur est rencontrée lors de la lecture du <code>WriteGetObjectResponse</code> corps.	400 Bad Request	Client
<code>ConditionalRequestConflict</code>	Une opération contradictoire s'est produite. Si <code>PutObject</code> vous l'utilisez, vous pouvez réessayer la demande. Si vous utilisez le téléchargement en plusieurs parties, vous devez lancer une autre <code>CreateMultipartUpload</code> demande et télécharger à nouveau chaque partie.	409 Conflict	Client
<code>CredentialsNotSupported</code>	Cette demande ne prend pas en charge les informations d'identification.	400 Bad Request	Client
<code>CrossLocationLoggingProhibited</code>	Cross-Region la journalisation n'est pas autorisée. Les compartiments d'une région Région AWS ne peuvent pas enregistrer d'informations dans un compartiment d'une autre région.	403 Forbidden	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
DeviceNotActiveError	L'appareil n'est actuellement pas actif.	400 Bad Request	Client
EndpointNotFound	Dirigez les demandes vers le point de terminaison approprié.	400 Bad Request	Client
EntityTooSmall	Le téléchargement que vous proposez est inférieur à la taille d'objet minimale autorisée.	400 Bad Request	Client
EntityTooLarge	Le téléchargement que vous proposez dépasse la taille d'objet maximale autorisée. Pour plus d'informations, consultez la section Points de terminaison et quotas Amazon Simple Storage Service dans le Références générales AWS.	400 Bad Request	Client
	Le téléchargement que vous proposez dépasse la taille maximale autorisée.	405 Method Not Allowed	Client
ExpiredToken	Le jeton fourni a expiré.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>IllegalLocationConstraintException</code>	Cette erreur peut se produire dans les conditions suivantes : • Vous essayez d'accéder à un compartiment depuis une région différente de celle où le compartiment existe. • Vous essayez de créer un compartiment avec une contrainte de localisation qui correspond à une région différente de celle du point de terminaison régional auquel la demande a été envoyée.	400 Bad Request	Client
<code>IllegalVersioningConfigurationException</code>	La configuration de contrôle de version spécifiée dans la demande n'est pas valide.	400 Bad Request	Client
<code>IncompleteBody</code>	Vous n'avez pas indiqué le nombre d'octets spécifié par l'en-tête Content-Length HTTP.	400 Bad Request	Client
<code>IncorrectEndpoint</code>	Le compartiment spécifié existe dans une autre région. Dirigez les demandes vers le point de terminaison approprié.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>IncorrectNumberOfFilesInPostRequest</code>	POST nécessite exactement un téléchargement de fichier par demande.	400 Bad Request	Client
<code>InlineDataTooLarge</code>	Les données en ligne dépassent la taille maximale autorisée.	400 Bad Request	Client
<code>InternalError</code>	Une erreur interne s'est produite. Réessayez.	500 Internal Server Error	Serveur
<code>InvalidAccessKeyId</code>	L'identifiant de clé d' AWS accès que vous avez fourni n'existe pas dans nos dossiers.	403 Forbidden	Client
<code>InvalidAccessPoint</code>	Le nom du point d'accès ou le compte spécifié n'est pas valide.	400 Bad Request	Client
<code>InvalidAccessPointAliasError</code>	Le nom d'alias du point d'accès spécifié n'est pas valide.	400 Bad Request	Client
<code>InvalidAddressingHeader</code>	Vous devez spécifier le rôle anonyme.	N/A	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidArgument	Cette erreur peut se produire dans les conditions suivantes : • Une ListBuckets demande est envoyée à un point de terminais on régional différent de la région spécifiée dans le bucket-region paramètre. • L'argument spécifié n'était pas valide. • Il manquait un en-tête obligatoire à la demande. • L'argument spécifié était incomplet ou n'était pas au bon format. • L'argument spécifié doit avoir une longueur supérieure ou égale à 3.	400 Bad Request	Client
InvalidBucketAclWithObjectOwnership	Les ACL du bucket ne peuvent pas être définies avec ObjectOwnership le BucketOwnerEnforced paramètre S.	400 Bad Request	Client
InvalidBucketName	Le compartiment spécifié n'est pas valide.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidBucketOwnerAWSAccountID	La valeur du paramètre du propriétaire du compartiment attendu doit être un Compte AWS ID.	400 Bad Request	Client
InvalidBucketState	La demande n'est pas valide pour l'état actuel du compartiment.	409 Conflict	Client
InvalidDigest	La valeur Content-MD5 ou la somme de contrôle que vous avez spécifiée n'est pas valide.	400 Bad Request	Client
InvalidEncryptionAlgorithmError	La demande de chiffrement que vous avez spécifiée n'est pas valide. La valeur valide est AES256.	400 Bad Request	Client
InvalidHostHeader	Les en-têtes d'hôte fournis dans la demande utilisaient un style d'adressage incorrect.	400 Bad Request	Client
InvalidHttpMethod	La demande est effectuée à l'aide d'une méthode HTTP inattendue.	400 Bad Request	Client
InvalidLocationConstraint	La contrainte de localisation (région) spécifiée n'est pas valide. Pour plus d'informations sur la sélection d'une région pour vos compartiments, consultez la section Présentation des compartiments .	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidObjectState	L'opération n'est pas valide pour l'état actuel de l'objet.	403 Forbidden	Client
InvalidPart	Une ou plusieurs des pièces spécifiées n'ont pas pu être trouvées. L'article n'a peut-être pas été chargé ou la balise d'entité spécifiée ne correspondait peut-être pas à la balise d'entité de l'article.	400 Bad Request	Client
InvalidPartOrder	La liste des pièces n'était pas classée par ordre croissant. La liste des pièces doit être spécifiée dans l'ordre par numéro de pièce.	400 Bad Request	Client
InvalidPayer	Tous les accès à cet objet ont été désactivés. Pour obtenir de l'aide, consultez Contactez-nous .	403 Forbidden	Client
InvalidPolicyDocument	Le contenu du formulaire ne répond pas aux conditions spécifiées dans le document de politique.	400 Bad Request	Client
InvalidRange	La plage demandée ne peut pas être satisfaite.	416 – Plage demandée impossible à respecter	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidRegion	Vous avez essayé de créer un point d' Multi-Region accès dans une région pour laquelle vous n'avez pas souscrit.	403 Forbidden	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidRequest	Cette erreur peut se produire dans les conditions suivantes : • Une ListBuckets demande non paginée est faite à partir d'un compte dont le quota de compartiments à usage général approuvé est supérieur à 10 000. Vous devez effectuer des requêtes paginées pour répertorier les compartiments d'un compte contenant plus de 10 000 compartiments. • La version de signature utilisée dans la demande est incorrecte. Utilisez AWS4-HMAC-SHA256 (Signature Version 4). • Un point d'accès ne peut être créé que pour un compartiment existant. • Le point d'accès n'est pas dans un état où il peut être supprimé. • Un point d'accès ne peut être répertorié que pour un compartiment existant. • Le jeton suivant n'est pas valide.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
	• Au moins une action doit être spécifiée dans une règle du cycle de vie. • Au moins une règle du cycle de vie doit être spécifiée. • Le nombre de règles du cycle de vie ne doit pas dépasser la limite autorisée de 1 000 règles. • La plage pour le paramètre <code>MaxResults</code> n'est pas valide. • Les demandes SOAP doivent être effectuées via une connexion HTTPS. • Amazon S3 Transfer Acceleration n'est pas pris en charge pour les compartiments dont les noms ne sont pas conformes à DNS. • Amazon S3 Transfer Acceleration n'est pas pris en charge pour les compartiments dont le nom contient des points (.). • Le point de terminaison Amazon S3 Transfer Acceleration		

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
	ne prend en charge que les demandes de style virtuelles. • Amazon S3 Transfer Acceleration n'est pas configuré sur ce compartiment. • Amazon S3 Transfer Acceleration est désactivé sur ce compartiment. • Amazon S3 Transfer Acceleration n'est pas pris en charge sur ce compartiment. Pour obtenir de l'aide, contactez Support. • Amazon S3 Transfer Acceleration ne peut pas être activé sur ce compartiment. Pour obtenir de l'aide, contactez Support. • Valeurs contradictoires fournies dans les en-têtes HTTP et les paramètres de requête. • Valeurs contradictoires fournies dans les en-têtes HTTP et les champs de formulaire POST. •		

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
	CopyObject demande effectuée sur des objets d'une taille supérieure à 5 Go.		
InvalidSessionException	Renvoyé si la session n'existe plus après avoir expiré.	400 Bad Request	Client
InvalidSignature	La signature de demande que le serveur a calculée ne correspond pas à la signature que vous avez fournie. Vérifiez votre clé d'accès AWS secrète et votre méthode de signature. Pour plus d'informations, consultez Signature et authentification des demandes REST .	400 Bad Request	Client
InvalidSecurity	Les informations d'identification de sécurité fournies ne sont pas valides.	403 Forbidden	Client
InvalidSOAPRequest	Le corps de la demande SOAP n'est pas valide.	400 Bad Request	Client
InvalidStorageClass	La classe de stockage que vous avez spécifiée n'est pas valide.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidTargetBucketForLogging	Le compartiment cible pour la journalisation n'existe pas, ne vous appartient pas ou ne dispose pas des autorisations appropriées pour le groupe de distribution de journaux.	400 Bad Request	Client
InvalidToken	Le jeton fourni est mal formé ou n'est pas valide.	400 Bad Request	Client
InvalidURI	L'URI spécifié n'a pas pu être analysé.	400 Bad Request	Client
KeyTooLongError	Votre clé est trop longue.	400 Bad Request	Client
KMS.DisabledException	La demande a été rejetée, car la clé KMS spécifiée n'est pas activée.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
KMS.InvalidKeyUsageException	Cette demande a été rejetée pour l'une des raisons suivantes : • La KeyUsage valeur de la clé KMS n'est pas compatible avec le fonctionnement de l'API. • L'algorithme de chiffrement ou l'algorithme de signature spécifié pour l'opération est incompatible avec le type de matériel clé contenu dans la clé KMS (KeySpec). Pour chiffrer, déchiffrer, rechiffrer et générer des clés de données, la valeur doit être ENCRYPT_DECRYPT. KeyUsage Pour signer et vérifier les messages, KeyUsage il doit être SIGN_VERIFY. Pour générer et vérifier les codes d'authentification des messages (MAC), celui-ci KeyUsage doit être GENERATE_VERIFY_MAC. Pour obtenir des secrets d'accord clés, KeyUsage il doit être KEY_AGREEMENT. Pour trouver la clé KeyUsage d'une clé KMS, utilisez l' DescribeKey opération. Pour trouver les algorithmes de chiffrement ou de signature pris	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
	en charge pour une clé KMS particulière, utilisez l' DescribeKey opération.		

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
KMS.KMSInvalidStateException	La demande a été rejetée, car l'état de la ressource spécifiée n'est pas valide pour cette demande. Cette exception a l'une des significations suivantes : • L'état de clé de la clé KMS n'est pas compatible avec l'opération. Pour trouver l'état de la clé, utilisez l' <code>DescribeKey</code> opération . Pour plus d'informations sur les états de clé compatibles avec chaque opération KMS, consultez États de clé de AWS KMS keys dans le Guide du développeur AWS Key Management Service . • Pour les opérations cryptographiques sur les clés KMS dans les magasins de clés personnalisés, cette exception représente un échec général avec de nombreuses causes possibles . Pour identifier la cause, consultez le message d'erreur qui accompagne l'exception.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>KMS.NotFoundException</code>	La demande a été rejetée, car l'entité ou la ressource spécifiée est introuvable.	400 Bad Request	Client
<code>MalformedACLError</code>	La liste ACL que vous avez fournie n'est pas bien formée ou n'a pas été validée par rapport à notre schéma publié.	400 Bad Request	Client
<code>MalformedPOSTRequest</code>	Le corps de votre requête POST n'est pas bien formé multipart/form-data.	400 Bad Request	Client
<code>MalformedXML</code>	Le code XML que vous avez fourni n'est pas bien formé ou n'a pas été validé par rapport à notre schéma publié.	400 Bad Request	Client
<code>MaxMessageLengthExceeded</code>	Votre demande était trop importante.	400 Bad Request	Client
<code>MaxPostPreDataLengthExceededError</code>	Les champs de votre demande POST précédant le fichier de chargement étaient trop volumineux.	400 Bad Request	Client
<code>MetadataTooLarge</code>	Vos en-têtes de métadonnées dépassent la taille de métadonnées maximale autorisée.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
MethodNotAllowed	La méthode spécifiée n'est pas autorisée sur cette ressource.	405 Method Not Allowed	Client
MissingAttachment	Une pièce jointe SOAP était attendue, mais aucune n'a été trouvée.	400 Requête erronée	Client
MissingAuthenticationToken	La demande n'a pas été signée.	403 Forbidden	Client
MissingContentLength	Vous devez fournir l'en-tête Content-Length HTTP.	411 Length Required	Client
MissingRequestBodyError	Vous avez envoyé un document XML vide sous forme de demande.	400 Bad Request	Client
MissingSecurityElement	Il manque un élément de sécurité à la requête SOAP 1.1.	400 Bad Request	Client
MissingSecurityHeader	Il manque un en-tête obligatoire dans votre demande.	400 Bad Request	Client
NoLoggingStatusForKey	Il n'existe pas de sous-ressource de statut de journalisation pour une clé.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
NoSuchAsyncRequest	La demande spécifiée est introuvable.	404 Not Found	Client
NoSuchBucket	Le compartiment spécifié n'existe pas.	404 Not Found	Client
NoSuchBucketPolicy	Le compartiment spécifié n'a pas de politique de compartiment.	404 Not Found	Client
NoSuchCORSConfiguration	Le compartiment spécifié n'a pas de configuration CORS.	404 Not Found	Client
NoSuchKey	La clé spécifiée n'existe pas.	404 Not Found	Client
NoSuchLifecycleConfiguration	La configuration de cycle de vie spécifiée n'existe pas.	404 Not Found	Client
NoSuchMultiRegionAccessPoint	Le point Multi-Region d'accès spécifié n'existe pas.	404 Not Found	Client
NoSuchObjectLockConfiguration	L'objet spécifié n'a pas de ObjectLock configuration.	404 Not Found	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
NoSuchWebsiteConfiguration	Le compartiment spécifié n'a aucune configuration de site Web.	404 Not Found	Client
NoSuchTagSet	La balise spécifiée n'existe pas.	404 Not Found	Client
NoSuchUpload	Le chargement partitionné spécifié n'existe pas. L'ID de chargement n'est peut-être pas valide ou le chargement partitionné a peut-être été abandonné ou terminé.	404 Not Found	Client
NoSuchVersion	L'ID de version spécifié dans la demande ne correspond pas à une version existante.	404 Not Found	Client
NotDeviceOwnerError	L'appareil qui a généré le jeton n'appartient pas à l'utilisateur authentifié.	400 Bad Request	Client
NotImplemented	Un en-tête que vous avez fourni implique une fonctionnalité qui n'est pas implémentée.	501 Not Implemented	Serveur
NotModified	La ressource n'a pas été modifiée.	304 Not Modified	Serveur

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
NoTransformationDefined	Aucune transformation n'a été trouvée pour ce point d'accès Object Lambda.	404 Not Found	Client
NotSignedUp	Votre compte n'est pas inscrit pour le service Amazon S3. Vous devez vous inscrire avant de pouvoir utiliser Amazon S3. Vous pouvez vous inscrire à l'adresse suivante : https://aws.amazon.com/s3	403 Forbidden	Client
ObjectLockConfigurationNotFoundError	La configuration de verrouillage d'objet n'existe pas pour ce compartiment.	404 Not Found	Client
OwnershipControlsNotFoundError	Les contrôles de propriété du compartiment n'ont pas été trouvés.	404 Not Found	Client
OperationAborted	Une opération conditionnelle conflictuelle est actuellement en cours sur cette ressource. Réessayez.	409 Conflict	Client
PermanentRedirect	Le compartiment auquel vous tentez d'accéder doit être adressé à l'aide du point de terminaison spécifié. Envoyez toutes les demandes futures à ce point de terminaison.	301 Moved Permanently	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
PermanentRedirectControlError	L'opération d'API à laquelle vous tentez d'accéder doit être adressée à l'aide du point de terminaison spécifié. Envoyez toutes les demandes futures à ce point de terminaison.	301 Moved Permanently	Client
PreconditionFailed	Au moins l'une des conditions préalables que vous avez spécifiées n'était pas remplie.	412 Precondition Failed	Client
Redirect	Redirection temporaire. Vous êtes redirigé vers le compartiment pendant la mise à jour du serveur du système de nom de domaine (DNS).	307 Temporary Redirect	Client
RequestHeaderSectionTooLarge	L'en-tête de la demande et les paramètres de requête utilisés pour faire en sorte que la demande dépasse la taille maximale autorisée.	400 Bad Request	Client
RequestIsNotMultipartContent	Une requête POST de bucket doit être du type de pièce jointe -data. multipart/form	412 Precondition Failed	Client
RequestTimeout	Votre connexion socket au serveur n'a pas été lue ou écrite pendant le délai imparti.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>RequestTimeTooSkewed</code>	La différence entre l'heure de la demande et celle du serveur est trop importante.	403 Forbidden	Client
<code>RequestTorrentOfBucketError</code>	La demande du fichier torrent d'un bucket n'est pas autorisée.	400 Bad Request	Client
<code>ResponseInterrupted</code>	Renvoyé à l'appelant d'origine lorsqu'une erreur est rencontrée lors de la lecture du <code>WriteGetObjectResponse</code> corps.	400 Bad Request	Client
<code>RestoreAlreadyInProgress</code>	La restauration de l'objet est déjà en cours.	409 Conflict	Client
<code>ServerSideEncryptionConfigurationNotFound</code>	La configuration de chiffrement côté serveur n'a pas été trouvée.	400 Bad Request	Client
<code>ServiceUnavailable</code>	Le service n'est pas en mesure de traiter la demande.	503 Service Unavailable	Serveur

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
SignatureDoesNotMatch	La signature de demande que le serveur a calculée ne correspond pas à la signature que vous avez fournie. Vérifiez votre clé d'accès AWS secrète et votre méthode de signature. Pour plus d'informations, consultez Authentification REST et Authentification SOAP .	403 Forbidden	Client
SlowDown	Veuillez réduire votre taux de demandes.	503 Slow Down	Serveur
503 SlowDown	Ralentissez	503 Slow Down	Serveur
TagPolicyException	La politique de balise n'autorise pas la valeur spécifiée pour la clé de balise suivante.	400 Bad Request	Client
TemporaryRedirect	Vous êtes redirigé vers le compartiment pendant la mise à jour du serveur du système de nom de domaine (DNS).	307 Temporary Redirect	Client
TokenCodeInvalidError	Le code du and/or jeton du numéro de série que vous avez fourni n'est pas valide.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
TokenRefreshRequired	Le jeton fourni doit être actualisé.	400 Bad Request	Client
TooManyAccessPoints	Vous avez essayé de créer plus de points d'accès que ce qui est autorisé pour un compte. Pour plus d'informations, consultez la section Points de terminaison et quotas Amazon Simple Storage Service dans le Références générales AWS.	400 Bad Request	Client
TooManyBuckets	Vous avez essayé de créer plus de compartiments que ce qui est autorisé pour un compte. Pour plus d'informations, consultez la section Points de terminaison et quotas Amazon Simple Storage Service dans le Références générales AWS.	400 Bad Request	Client
TooManyMultiRegion AccessPointregionsError	Vous avez essayé de créer un point d' Multi-Region accès avec plus de régions que ce qui est autorisé pour un compte. Pour plus d'informations, consultez la section Points de terminaison et quotas Amazon Simple Storage Service dans le Références générales AWS.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
TooManyMultiRegionAccessPoints	Vous avez essayé de créer plus de points Multi-Region d'accès que ce qui est autorisé pour un compte. Pour plus d'informations, consultez la section Points de terminaison et quotas Amazon Simple Storage Service dans le Références générales AWS.	400 Bad Request	Client
UnauthorizedAccessError	Applicable dans les régions de Chine uniquement. Renvoyé lorsqu'une demande est adressée à un compartiment qui ne possède pas de licence ICP. Pour plus d'informations, consultez Enregistrement ICP .	403 Forbidden	Client
UnexpectedContent	Cette demande contient du contenu non pris en charge.	400 Bad Request	Client
UnexpectedIPError	Applicable dans les régions de Chine uniquement. Cette demande a été rejetée, car l'adresse IP était inattendue.	403 Forbidden	Client
UnsupportedArgument	La demande contenait un argument non pris en charge.	400 Bad Request	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
UnsupportedSignature	La demande fournie est signée avec une version de jeton STS non prise en charge ou la version de signature n'est pas prise en charge.	400 Bad Request	Client
UnresolvableGrantByEmailAddress	L'adresse e-mail que vous avez fournie ne correspond à aucun compte enregistré.	400 Bad Request	Client
UserKeyMustBeSpecified	La demande POST du compartiment doit contenir le nom de champ spécifié. S'il est spécifié, vérifiez l'ordre des champs.	400 Bad Request	Client
NoSuchAccessPoint	Le point d'accès spécifié n'existe pas.	404 Not Found	Client
InvalidTag	Votre demande contient une entrée de balise qui n'est pas valide. Par exemple, votre demande peut contenir des clés dupliquées, des clés ou des valeurs trop longues ou des balises système.	400 Bad Request	Client
MalformedPolicy	Votre police contient un principal qui n'est pas valide.	400 Bad Request	Client

Liste des codes d'erreur relatifs au contenu de l'objet SELECT

Important

Amazon S3 Select n'est plus disponible pour les nouveaux clients. Les clients existants d'Amazon S3 Select peuvent continuer à utiliser cette fonctionnalité comme d'habitude. [En savoir plus](#)

Le tableau suivant contient des erreurs spéciales SELECT Object Content susceptibles de se reproduire. Pour obtenir des informations générales sur les erreurs Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
AmbiguousFieldName	Le nom du champ correspond à plusieurs champs du fichier. Vérifiez l'expression SQL et le fichier, puis réessayez.	400	Client
Busy	Le service n'est pas disponible. Réessayez ultérieurement.	503	Client
CastFailed	Une tentative de conversion d'un type de données à un autre à l'aide de l'expression SQL CAST a échoué.	400	Client
ColumnTooLong	La longueur d'une colonne dans le résultat est supérieure à <code>maxCharsPerColumn</code> 1 Mo.	400	Client
CSVEscapingRecordDelimiter	Un délimiteur d'enregistrement entre guillemets a été trouvé dans le fichier. Pour autoriser les	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
	séparateurs d'enregistrements entre guillemets, définissez <code>surAllowQuotedRecordDelimiter</code> à <code>'TRUE'</code>		
<code>CSVParsingError</code>	Une erreur s'est produite lors de l'analyse du fichier CSV. Vérifiez le fichier et réessayez.	400	Client
<code>CSVUnescapedQuote</code>	Une citation non échappée a été trouvée lors de l'analyse du fichier CSV. Pour autoriser les séparateurs d'enregistrements entre guillemets, définissez <code>surAllowQuotedRecordDelimiter</code> à <code>'TRUE'</code>	400	Client
<code>EmptyRequestBody</code>	Le corps de la demande ne peut pas être vide.	400	Client
<code>EvaluatorBindingDoesNotExist</code>	Aucun nom de colonne ou chemin fourni n'existe dans l'expression SQL.	400	Client
<code>EvaluatorInvalidArguments</code>	Le nombre d'arguments dans l'appel de fonction de l'expression SQL est incorrect.	400	Client
<code>EvaluatorInvalidTimestampFormatPattern</code>	La chaîne de format d'horodatage dans l'expression SQL n'est pas valide.	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>EvaluatorInvalidTimestampFormatPatternSymbol</code>	Le modèle de format d'horodatage contient un symbole dans l'expression SQL qui n'est pas valide.	400	Client
<code>EvaluatorInvalidTimestampFormatPatternSymbolForParsing</code>	Le modèle de format d'horodatage contient un symbole de format valide qui ne peut pas être appliqué à l'analyse de l'horodatage dans l'expression SQL.	400	Client
<code>EvaluatorInvalidTimestampFormatPatternToken</code>	Le modèle de format d'horodatage contient un jeton dans l'expression SQL qui n'est pas valide.	400	Client
<code>EvaluatorLikePatternInvalidEscapeSequence</code>	Un argument donné à l' <code>LIKE</code> expression n'était pas valide.	400	Client
<code>EvaluatorNegativeLimit</code>	<code>LIMIT</code> ne doit pas être négatif.	400	Client
<code>EvaluatorTimestampFormatPatternDuplicateFields</code>	Le modèle de format d'horodatage contient plusieurs indicateurs de format représentant le champ d'horodatage de l'expression SQL.	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>EvaluatorTimestampFormatPatternHourClockAmPmMismatch</code>	Le modèle de format d'horodatage contient un symbole de format d'heure de jour sur 12 heures mais ne contient pas également de AM/PM champ, ou il contient un spécificateur de format sur 24 heures et contient un AM/PM champ dans l'expression SQL.	400	Client
<code>EvaluatorUnterminatedTimestampFormatPatternToken</code>	Le modèle de format d'horodatage contient un jeton non terminé dans l'expression SQL.	400	Client
<code>ExpressionTooLong</code>	L'expression SQL est trop longue. La longueur d'octet maximale pour une expression SQL est de 256 Ko.	400	Client
<code>ExternalEvalException</code>	La requête ne peut pas être évaluée. Vérifiez le fichier et réessayez.	400	Client
<code>IllegalSqlFunctionArgument</code>	Un argument illégal a été utilisé dans la fonction SQL.	400	Client
<code>IncorrectSqlFunctionArgumentType</code>	Un type d'argument incorrect a été spécifié dans un appel de fonction dans l'expression SQL.	400	Client
<code>IntegerOverflow</code>	Un dépassement ou un sous-débit d'entiers s'est produit dans l'expression SQL.	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>InternalError</code>	Une erreur interne s'est produite.	500	Client
<code>InvalidCast</code>	Une tentative de conversion d'un type de données à un autre à l'aide de l'expression SQL CAST a échoué.	400	Client
<code>InvalidColumnIndex</code>	L'index de colonne dans l'expression SQL n'est pas valide.	400	Client
<code>InvalidCompressionFormat</code>	Le format de compression du fichier n'est pas pris en charge. Seuls GZIP et BZIP2 sont pris en charge.	400	Client
<code>InvalidDataSource</code>	Le type de source de données n'est pas valide. Seuls les formats CSV, JSON et Parquet sont pris en charge.	400	Client
<code>InvalidDataType</code>	L'expression SQL contient un type de données qui n'est pas valide.	400	Client
<code>InvalidExpressionType</code>	La <code>ExpressionType</code> valeur n'est pas valide. Seules les expressions SQL sont prises en charge.	400	Client
<code>InvalidFileHeaderInfo</code>	La <code>FileHeaderInfo</code> valeur n'est pas valide. Seuls NONE, USE et IGNORE sont pris en charge.	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>InvalidJsonType</code>	La <code>JsonType</code> valeur n'est pas valide. Seuls <code>DOCUMENT</code> et <code>LINES</code> sont pris en charge.	400	Client
<code>InvalidKeyPath</code>	Le chemin clé de l'expression SQL n'est pas valide.	400	Client
<code>InvalidQuoteFields</code>	La <code>QuoteFields</code> valeur n'est pas valide. Seuls <code>ALWAYS</code> et <code>ASNEEDED</code> sont pris en charge.	400	Client
<code>InvalidRequestParameter</code>	La valeur d'un paramètre de l' <code>SelectRequest</code> élément n'est pas valide. Consultez la documentation de l'API du service et réessayez.	400	Client
<code>InvalidScanRange</code>	La plage de numérisation fournie n'est pas valide.	400	Client
<code>InvalidTableAlias</code>	L'expression SQL contient un alias de table qui n'est pas valide.	400	Client
<code>InvalidTextEncoding</code>	Le type de codage n'est pas valide. Seul le UTF-8 codage est pris en charge.	400	Client
<code>JSONParsingError</code>	Une erreur s'est produite lors de l'analyse du fichier JSON. Vérifiez le fichier et réessayez.	400	Client
<code>LexerInvalidChar</code>	L'expression SQL contient un caractère qui n'est pas valide.	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>LexerInvalidIONLiteral</code>	L'expression SQL contient un opérateur qui n'est pas valide.	400	Client
<code>LexerInvalidLiteral</code>	L'expression SQL contient un opérateur qui n'est pas valide.	400	Client
<code>LexerInvalidOperator</code>	L'expression SQL contient un littéral qui n'est pas valide.	400	Client
<code>LikeInvalidInputs</code>	L'argument donné à la LIKE clause dans l'expression SQL n'est pas valide.	400	Client
<code>MalformedXML</code>	Le code XML fourni n'était pas bien formé ou n'a pas été validé par rapport à notre schéma publié. Consultez la documentation du service et réessayez.	400	Client
<code>MaxOperatorsExceeded</code>	Impossible d'analyser l'expression SQL, essayez de réduire la complexité. Par exemple, réduisez le nombre d'opérateurs utilisés.	400	Client
<code>MethodNotAllowed</code>	La méthode spécifiée n'est pas autorisée sur cette ressource.	405 Method Not Allowed	Client
<code>MissingRequiredParameter</code>	Il manque un paramètre obligatoire à l' <code>SelectRequest</code> entité. Consultez la documentation du service et réessayez.	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>MultipleDataSourceUnsupported</code>	Les sources de données multiples ne sont pas prises en charge.	400	Client
<code>NumberFormatError</code>	Une erreur s'est produite lors de l'analyse d'un nombre. Cette erreur peut être causée par un sous-débit ou un débordement d'entiers.	400	Client
<code>ObjectSerializationConflict</code>	<code>InputSerialization</code> spécifie plusieurs formats (CSV, JSON ou Parquet), ou <code>OutputSerialization</code> spécifie plusieurs formats (CSV ou JSON). Pour <code>InputSerialization</code> et <code>OutputSerialization</code> , vous ne pouvez spécifier qu'un seul format pour chacun.	400	Client
<code>OverMaxColumn</code>	Le nombre de colonnes du résultat est supérieur au nombre de colonnes maximum autorisé.	400	Client
<code>OverMaxParquetBlockSize</code>	La taille maximale du fichier Parquet est supérieure à la taille maximale du groupe de lignes.	400	Client
<code>OverMaxRecordSize</code>	La longueur d'un enregistrement dans l'entrée ou le résultat est supérieure à la <code>maxCharsPerRecord</code> limite de 1 Mo.	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
ParquetParsingError	Une erreur s'est produite lors de l'analyse du fichier Parquet. Vérifiez le fichier et réessayez.	400	Client
ParquetUnsupportedCompressionCodec	Le codec de compression Parquet spécifié n'est pas pris en charge.	400	Client
ParseAsteriskIsNotAloneInSelectList	Les autres expressions ne sont pas autorisées dans la SELECT liste lorsqu'elle * est utilisée sans notation par points dans l'expression SQL.	400	Client
ParseCannotMixSqbandWildcardInSelectList	Impossible de mélanger [] et * d'insérer la même expression dans une SELECT liste de l'expression SQL.	400	Client
ParseCastArity	L'arité de l'expression SQL CAST est incorrecte.	400	Client
ParseEmptySelect	L'expression SQL contient une SELECT clause vide.	400	Client
ParseExpected2TokenTypes	Le jeton attendu dans l'expression SQL est introuvable.	400	Client
ParseExpectedArgumentDelimiter	Le délimiteur d'arguments attendu dans l'expression SQL est introuvable.	400	Client
ParseExpectedDatePart	La partie date attendue dans l'expression SQL est introuvable.	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
ParseExpectedExpression	L'expression SQL attendue n'a pas été trouvée.	400	Client
ParseExpectedIdentifierForAlias	L'identifiant attendu pour l'alias dans l'expression SQL est introuvable.	400	Client
ParseExpectedIdentifierForAt	L'identifiant attendu pour le AT nom dans l'expression SQL est introuvable.	400	Client
ParseExpectedIdentifierForGroupName	GROUP n'est pas pris en charge dans l'expression SQL.	400	Client
ParseExpectedKeyword	Le mot clé attendu dans l'expression SQL est introuvable.	400	Client
ParseExpectedLeftParenAfterCast	La parenthèse gauche attendue après CAST l'expression SQL est introuvable.	400	Client
ParseExpectedLeftParenBuiltinFunctionCall	La parenthèse gauche attendue dans l'expression SQL est introuvable.	400	Client
ParseExpectedLeftParenValueConstructor	La parenthèse gauche attendue dans l'expression SQL est introuvable.	400	Client
ParseExpectedMember	L'expression SQL contient une utilisation non prise en charge de. MEMBER	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
ParseExpectedNumber	Le nombre attendu dans l'expression SQL est introuvable.	400	Client
ParseExpectedRightParenBuiltinFunctionCall	Le caractère de parenthèse droite attendu dans l'expression SQL est introuvable.	400	Client
ParseExpectedTokenType	Le jeton attendu dans l'expression SQL est introuvable.	400	Client
ParseExpectedTypeName	Le nom de type attendu dans l'expression SQL est introuvable.	400	Client
ParseExpectedWhenClause	La WHEN clause attendue dans l'expression SQL est introuvable. CASE n'est pas pris en charge.	400	Client
ParseInvalidContextForWildcardInSelectList	L'utilisation de * dans la SELECT liste dans l'expression SQL n'est pas valide.	400	Client
ParseInvalidPathComponent	L'expression SQL contient un composant de chemin qui n'est pas valide.	400	Client
ParseInvalidTypeParameter	L'expression SQL contient une valeur de paramètre qui n'est pas valide.	400	Client
ParseMalformedJoin	JOIN n'est pas pris en charge dans l'expression SQL.	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
ParseMissingIdentifierAt	L'identifiant attendu après le symbole @ dans l'expression SQL est introuvable.	400	Client
ParseNonUnaryAggregateFunctionCall	Un seul argument est pris en charge pour les fonctions d'agrégation dans l'expression SQL.	400	Client
ParseSelectMissingFrom	L'expression SQL contient un élément manquant FROM après la SELECT liste.	400	Client
ParseUnexpectedKeyword	L'expression SQL contient un mot clé inattendu.	400	Client
ParseUnexpectedOperator	L'expression SQL contient un opérateur inattendu.	400	Client
ParseUnexpectedTerm	L'expression SQL contient un terme inattendu.	400	Client
ParseUnexpectedToken	L'expression SQL contient un jeton inattendu.	400	Client
ParseUnknownOperator	L'expression SQL contient un opérateur qui n'est pas valide.	400	Client
ParseUnsupportedAlias	L'expression SQL contient une utilisation non prise en charge de. ALIAS	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
ParseUnsupportedCallWithStar	Seul COUNT (*) le paramètre with est pris en charge dans l'expression SQL.	400	Client
ParseUnsupportedCase	L'expression SQL contient une utilisation non prise en charge de. CASE	400	Client
ParseUnsupportedCaseClause	L'expression SQL contient une utilisation non prise en charge de. CASE	400	Client
ParseUnsupportedLiteralsGroupBy	L'expression SQL contient une utilisation non prise en charge de. GROUP BY	400	Client
ParseUnsupportedSelect	L'expression SQL contient une utilisation non prise en charge de. SELECT	400	Client
ParseUnsupportedSyntax	L'expression SQL contient une syntaxe non prise en charge.	400	Client
ParseUnsupportedToken	L'expression SQL contient un jeton non pris en charge.	400	Client
TruncatedInput	La décompression de l'objet a échoué. Vérifiez que l'objet est correctement compressé en utilisant le format spécifié dans la demande.	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
UnauthorizedAccess	Vous n'êtes pas autorisé à effectuer cette opération.	401	Client
UnrecognizedFormatException	Le type d'enregistrement n'est pas valide.	400	Client
UnsupportedFunction	S3 a détecté une fonction SQL non prise en charge.	400	Client
UnsupportedParquetType	Le type de parquet spécifié n'est pas pris en charge.	400	Client
UnsupportedRangeHeader	Un en-tête de plage n'est pas pris en charge pour cette opération.	400	Client
UnsupportedScanRangeInput	Les requêtes par plage de numérisation ne sont pas prises en charge sur ce type d'objet.	400	Client
UnsupportedSqlOperation	S3 a rencontré une opération SQL non prise en charge.	400	Client
UnsupportedSqlStructure	S3 a rencontré une structure SQL non prise en charge. Consultez la référence SQL.	400	Client
UnsupportedStorageClass	La classe de stockage n'est pas prise en charge. Seules STANDARD, STANDARD_IA , et les classes ONEZONE_IA de stockage sont prises en charge.	400	Client
UnsupportedSyntax	La syntaxe n'est pas valide.	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
UnsupportedTypeFor Querying	Votre requête contient un type non pris en charge pour la comparaison (par exemple, vérifier qu'un type de colonne Parquet INT96 est supérieur à 0).	400	Client
ValueParseFailure	Un échec d'analyse de l'horodatage s'est produit dans l'expression SQL.	400	Client

Liste des codes Replication-related d'erreur

Le tableau suivant contient les erreurs spéciales que l'opération `Replication` peut renvoyer. Pour obtenir des informations générales sur les erreurs Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidArgument	Cette erreur peut se produire dans les conditions suivantes : - L'<Account> élément est vide. Il doit contenir un identifiant de compte valide. - La Compte AWS valeur spécifiée dans l'<Account> élément doit correspondre au	400	Client

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
	propriétaire du compartiment de destination. • <code>ReplicationTime-Status</code> doit contenir une valeur. • <code>ReplicationTime-ReplicationTimeValue</code> doit contenir une valeur. • <code>Replication-ReplicationTimeValue-Minutes</code> la valeur doit être 15. • <code>ReplicationMetrics</code> doit contenir un <code>Status</code>. • <code>ReplicationMetrics</code> doit contenir un <code>EventThreshold</code>. • <code>EventThreshold-ReplicationTimeValue-Minutes</code> la valeur doit être 15. • <code>Rule ID</code> ne doit pas contenir de caractères non ASCII.		

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidRequest	Cette erreur peut se produire dans les conditions suivantes : • L'<Owner>entrée <AccessControlTranslation> a une valeur, donc l'<Account> élément doit être spécifié. • L'<Account> élément est vide. Il doit contenir un identifiant de compte valide. • La réplication destination doit contenir à la fois ReplicationTime et Metrics, ou aucun des deux. • ReplicationTime et ReplicationMetrics doivent avoir le même statut. • Le contrôle du temps de réplication S3 (S3 RTC) n'est pas pris en charge à cet Région AWSégard.	400	Client
ReplicationConfigurationNotFoundErro r	Il n'existe aucune configuration de réplication pour ce compartiment.	404 – Non trouvé	Client

Liste des codes Tagging-related d'erreur

Le tableau suivant contient des erreurs spéciales que les `ListTagsForResource` opérations `TagResource` `UntagResource`, et peuvent renvoyer pour les groupes de lentilles de stockage. Pour obtenir des informations générales sur les erreurs générales d'Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Code d'erreur	Description	HTTP Status Code	Préfixe du code d'erreur SOAP
<code>InvalidRequest</code>	L'ARN Région AWS de la ressource ne correspond pas à la région spécifiée dans cette demande. Le AWS compte dans l'ARN de la ressource ne correspond pas à l'ID de compte spécifié dans cette demande. La AWS partition du <code>ResourceArn</code> n'est pas valide.	400 Bad Request	Non pris en charge
<code>InvalidTag</code>	Cette demande contient une clé ou une valeur de balise qui n'est pas valide. Les caractères valides sont les suivants : <code>[a-zA-Z+-=._:/]</code> . Les clés de tag peuvent contenir jusqu'à 128 caractères. Les valeurs des balises peuvent contenir jusqu'à 256 caractères. Il y a des clés de tag dupliquées dans votre demande. User-defined les clés de tag ne peuvent pas commencer par <code>aws:</code> .	400 Bad Request	Non pris en charge
<code>NoSuchResource</code>	La ressource spécifiée n'existe pas.	404 Not Found	Non pris en charge

Code d'erreur	Description	HTTP Status Code	Préfixe du code d'erreur SOAP
TagPolicyException	La politique de balise n'autorise pas la valeur spécifiée pour la clé de balise suivante.	400 Bad Request	Non pris en charge
TooManyTags	Le nombre de balises dépasse la limite de 50 balises.	400 Bad Request	Non pris en charge

Liste des codes d'erreur liés aux annotations Amazon S3

Le tableau suivant contient des erreurs spéciales que les opérations d'annotation (`PutObjectAnnotation`, `GetObjectAnnotation`, `ListObjectAnnotations`, `DeleteObjectAnnotation`, `UpdateBucketMetadataAnnotationTable` et le `CopyObject` comportement lié aux annotations peuvent renvoyer. Pour obtenir des informations générales sur les erreurs Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Code d'erreur	Description	HTTP Status Code	Préfixe du code d'erreur SOAP
AccessDenied	Accès refusé. L'appelant n'est pas autorisé à effectuer l'opération d'annotation demandée, ou l'objet est protégé par Object Lock en mode COMPLIANCE.	403 Forbidden	Non pris en charge
AnnotationLimitExceeded	Le nombre d'annotations d'objets dépasse la limite (1000).	400 Bad Request	Non pris en charge
AnnotationNameTooLong	Le nom de votre annotation est trop long. Les noms d'annotat	400 Bad Request	Non pris en charge

Code d'erreur	Description	HTTP Status Code	Préfixe du code d'erreur SOAP
	ions peuvent avoir une longueur maximale de 512 octets.		
AnnotationRequestConflict	Une opération contradictoire est actuellement en cours concernant cette annotation. Cela peut se produire lorsqu'une annotation source est supprimée lors d'une copie entre régions.	409 Conflict	Non pris en charge
EntityTooSmall	La taille de téléchargement que vous proposez est inférieure à la taille minimale autorisée. La charge utile des annotations doit être d'au moins 1 octet.	400 Bad Request	Non pris en charge
InvalidAnnotationName	Le nom de votre annotation contient des caractères non valides, ou il commence par un préfixe réservé (aws ou sans distinction s3 majuscules/minuscules). Les noms d'annotations ne peuvent contenir que des lettres Unicode, des chiffres, un trait de soulignement (_), un point (.) et un tiret (-).	400 Bad Request	Non pris en charge

Code d'erreur	Description	HTTP Status Code	Préfixe du code d'erreur SOAP
InvalidArgument	La directive d'annotation n'est pas valide. Les valeurs valides sont COPY et EXCLUDE. Pour ListObjectAnnotations, max-annotation-results doit être compris entre 1 et 1000.	400 Bad Request	Non pris en charge
InvalidPrefix	Le préfixe d'annotation que vous avez fourni n'est pas valide.	400 Bad Request	Non pris en charge
InvalidRequest	Cette erreur peut se produire pour les raisons suivantes : L'objet est chiffré avec SSE-C, ce qui ne prend pas en charge les annotations. L'objet source comporte des annotations qui ne peuvent pas être copiées vers la destination car celle-ci se trouve dans une région non prise en charge. La demande de configuration de la table d'annotations n'est pas valide (par exemple, désactivation de la table tout en spécifiant le chiffrement, activation d'une table déjà activée, mise à jour d'une table avec le statut CREATING ou soumission d'une demande vide).	400 Bad Request	Non pris en charge

Code d'erreur	Description	HTTP Status Code	Préfixe du code d'erreur SOAP
MethodNotAllowed	La méthode spécifiée n'est pas autorisée sur cette ressource . Par exemple, cette erreur se produit lorsque la demande cible un marqueur de suppression.	405 Method Not Allowed	Non pris en charge
NoSuchAnnotation	L'annotation spécifiée n'existe pas.	404 Not Found	Non pris en charge
NoSuchKey	La clé spécifiée n'existe pas.	404 Not Found	Non pris en charge
NoSuchVersion	L'ID de version spécifié dans la demande ne correspond pas à une version existante.	404 Not Found	Non pris en charge
NotImplemented	Les annotations ne sont pas prises en charge pour les compartiments de répertoires Amazon S3.	501 Not Implemented	Non pris en charge
OperationAborted	Une opération conditionnelle conflictuelle est actuellement en cours sur cette ressource. Cela peut se produire lors de modifications simultanées ou lorsqu'un nombre d'annotations source ou un ETag change au cours d'une copie entre régions. Réessayez.	409 Conflict	Non pris en charge

Code d'erreur	Description	HTTP Status Code	Préfixe du code d'erreur SOAP
PreconditionFailed	Au moins l'une des conditions préalables que vous avez spécifiées n'était pas remplie. L' <code>x-amz-object-if-match</code> ETag ne correspond pas à l'ETag actuel de l'objet.	412 Precondition Failed	Non pris en charge
UnsupportedMediaType	La charge utile de la demande contient un type de média non pris en charge. La charge utile des annotations doit être un texte UTF-8 codé valide.	415 Unsupported Media Type	Non pris en charge
UserAnnotationNameMustBeSpecified	Vous devez spécifier un nom d'annotation utilisateur.	400 Bad Request	Non pris en charge

Liste des codes d'erreur Amazon S3 on Outposts

Le tableau suivant contient des erreurs spéciales qu'une opération Amazon S3 on Outposts peut renvoyer. Pour obtenir des informations générales sur les erreurs Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
BadRequest	Le compartiment est dans un état de transition en raison d'une précédente tentative de suppression. Réessayez ultérieurement.	400 Requête erronée	Non pris en charge

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidRequest	Cette erreur peut se produire dans les conditions suivantes : - La configuration d'Amazon VPC est requise. - L'accès public n'est pas autorisé sur les points d'accès S3 on Outposts.	400 Requête erronée	Client
InvalidOutpostState	La demande n'est pas valide dans l'état actuel de l'Avant-poste.	409 – Conflit	Non pris en charge
InvalidRequest	Le point d'accès n'est pas dans un état où il peut être supprimé.	400 Requête erronée	Non pris en charge
NoSuchOutpost	L'avant-poste spécifié n'existe pas.	404 – Non trouvé	Non pris en charge
UnsupportedOperation	L'action spécifiée n'a pas été prise en charge.	404 – Non trouvé	Non pris en charge
InsufficientCapacity	Capacité insuffisante.	507 Stockage insuffisant	Non pris en charge

Liste des codes d'erreur d'Amazon S3 Storage Lens

Le tableau suivant contient des erreurs spéciales que les opérations d'Amazon S3 Storage Lens peuvent renvoyer. Pour obtenir des informations générales sur les erreurs générales d'Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
AccessDenied	Cette région n'est pas prise en charge en tant que région d'origine pour S3 Storage Lens.	403 Forbidden	Non pris en charge
AccountNotAuthorized	L'utilisation de ce compte n'est pas autorisée AWS Organizations. Utilisez votre compte de gestion ou votre compte d'administrateur délégué.	403 Forbidden	Non pris en charge
ActivityMetricsMustEnabled	Les mesures d'activité doivent être activées.	400 Requête erronée	Non pris en charge
AWSOrganizationsNotInUseException	Ce compte ne fait pas partie de votre organisation.	403 Forbidden	Non pris en charge
DefaultConfigurationDeleteForbidden	La configuration par défaut ne peut pas être supprimée.	403 Forbidden	Non pris en charge
DuplicateStorageLensGroupARN	Il existe deux entrées ou plus du même ARN du groupe Storage Lens dans cette configuration.	400 Bad Request	Non pris en charge
EmptyExcludeContainer	Cette erreur se produit pour les raisons suivantes : - Le conteneur d'exclusion ne peut pas être vide. - Le conteneur d'exclusion ne peut pas contenir aucun seau.	400 Requête erronée	Non pris en charge

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
	Le conteneur d'exclusion ne peut pas contenir aucune région.		
EmptyExcludeElement	Vous devez spécifier un groupe de lentilles de stockage avec votre élément d'exclusion.	400 Bad Request	Non pris en charge
EmptyIncludeContainer	Cette erreur se produit pour les raisons suivantes : - Le conteneur inclus ne peut pas être vide. - Le conteneur inclus ne peut pas contenir aucun seau. - Le conteneur d'inclusion ne peut pas avoir aucune région.	400 Requête erronée	Non pris en charge
InvalidAWSOrgArn	Il y a un AWS Organizations ARN mal formé dans la configuration.	400 Requête erronée	Non pris en charge
EmptyIncludeElement	Vous devez spécifier un groupe de lentilles de stockage avec votre élément Include.	400 Bad Request	Non pris en charge
InvalidBucketFilter	Organization-level les configurations ne prennent pas en charge les filtres à godets.	400 Requête erronée	Non pris en charge
InvalidConfigId	L'ID de configuration n'est pas valide.	400 Requête erronée	Non pris en charge

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidDestination	L'ARN du compartiment S3 est mal formé.	400 Requête erronée	Non pris en charge
InvalidEncryptionMethod	Une seule méthode de chiffrement peut être spécifiée.	400 Requête erronée	Non pris en charge
InvalidFilterForDefaultConfiguration	La configuration par défaut ne doit inclure aucun filtre.	400 Requête erronée	Non pris en charge
InvalidIncludeExcludeContainers	Vous pouvez spécifier un conteneur d'inclusion ou un conteneur d'exclusion dans une configuration. Vous ne pouvez pas spécifier les deux dans une configuration.	400 Requête erronée	Non pris en charge
InvalidIncludeExcludeElements	Un seul élément d'inclusion ou d'exclusion est autorisé. Au moins un élément d'inclusion ou d'exclusion doit être présent.	400 Bad Request	Non pris en charge
InvalidKMSEncryptionKeyId	L'ARN de l'ID de clé KMS n'est pas valide.	400 Requête erronée	Non pris en charge
InvalidMaximumPrefixDepth	MaxDepth doit se situer dans la plage [1,10].	400 Requête erronée	Non pris en charge
InvalidMinimumStorageBytesPercentage	MinStorageBytesPercentage doit se situer dans la plage [1,00,100,00].	400 Requête erronée	Non pris en charge
InvalidOrganizationARN	L' AWS Organizations ARN de la configuration n'est pas valide.	400 Requête erronée	Non pris en charge

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
<code>InvalidOrganizationForDefaultConfiguration</code>	La configuration par défaut ne prend pas en charge les métriques au niveau de l'organisation.	400 Requête erronée	Non pris en charge
<code>InvalidRegionForDefaultConfiguration</code>	La région spécifiée n'est pas prise en charge pour la configuration par défaut.	400 Requête erronée	Non pris en charge
<code>InvalidRegionName</code>	Le nom de la région n'est pas valide.	400 Requête erronée	Non pris en charge
<code>InvalidStorageLensArn</code>	L'ARN S3 Storage Lens n'est pas requis en entrée.	400 Requête erronée	Non pris en charge
<code>InvalidStorageLensGroupARN</code>	Cet ARN de groupe Storage Lens n'est pas valide ou seuls les groupes Storage Lens de votre compte sont autorisés. En outre, vous devez suivre la structure ARN du groupe Storage Lens : <code>arn::s3::storage-lens-group/</code> et respecter la limite de 64 caractères. Les noms des groupes Storage Lens peuvent également contenir uniquement les caractères suivants : a-z, 0-9 A-Z, des traits d'union (-) et des traits de soulignement (_).	400 Bad Request	Non pris en charge
<code>MissingAccountLevelActivityMetrics</code>	Les mesures d'activité doivent être activées au niveau du compte lorsque les mesures d'activité sont activées au niveau du bucket.	400 Requête erronée	Non pris en charge

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
MissingBucketLevelActivityMetrics	Les mesures d'activité doivent être activées au niveau du compartiment lorsque les mesures d'activité sont activées au niveau du compte.	400 Requête erronée	Non pris en charge
MissingEncryptionMethod	La méthode de cryptage ne peut pas être vide. Spécifiez SSE-KMS ou SSE-S3.	400 Requête erronée	Non pris en charge
MissingPrefixLevelStorageMetrics	Les mesures de stockage au niveau du préfixe sont obligatoires lorsque le niveau du préfixe est activé.	400 Requête erronée	Non pris en charge
OrganizationAccessDenied	L'ajout de ce compte n'est pas autorisé AWS Organizations.	403 Forbidden	Non pris en charge
OrgConfigurationNotSupported	La région spécifiée n'est pas prise en charge AWS Organizations en charge dans la configuration.	403 Forbidden	Non pris en charge
ServiceNotEnabledForOrg	Le rôle lié au service S3 Storage Lens n'est pas activé pour l'organisation.	403 Forbidden	Non pris en charge
StorageMetricsMustBeEnabled	Prefix-level les métriques de stockage doivent être activées.	400 Requête erronée	Non pris en charge
TooManyBuckets	Le conteneur à godets ne peut pas contenir plus de 50 seaux.	400 Requête erronée	Non pris en charge

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
TooManyRegions	Le conteneur Régions ne peut pas contenir plus de 50 régions.	400 Requête erronée	Non pris en charge
TooManyStorageLens Groups	Vous ne pouvez pas associer plus de 50 groupes Storage Lens à votre tableau de bord Storage Lens.	400 Bad Request	Non pris en charge

Le tableau suivant contient des erreurs spéciales que les opérations des groupes S3 Storage Lens peuvent renvoyer. Pour obtenir des informations générales sur les erreurs générales d'Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
AccessDenied	Vous n'êtes pas autorisé à effectuer des actions de groupe Storage Lens. Cette région n'est pas prise en charge en tant que région d'origine pour les groupes S3 Storage Lens.	403 Forbidden	Non pris en charge
ConfigurationAlreadyExists	La configuration spécifiée existe déjà.	409 Conflict	Non pris en charge
DuplicateElement	Les tags doivent être uniques. L'opérateur And logique inclut des clés de balise dupliquées. L'opérateur Or logique inclut des balises dupliquées. L'opérateur	400 Bad Request	Non pris en charge

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
	logique inclut des préfixes ou des suffixes dupliqués.		
InvalidAge	DaysLessThan et DaysGreaterThan doivent être des nombres positifs.	400 Bad Request	Non pris en charge
InvalidFilter	Un filtre doit inclure l'un des éléments suivants : AndOr, MatchAnyTag, MatchAnyPrefix, MatchAnySuffix, MatchObjectAge, MatchObjectSize .	400 Bad Request	Non pris en charge
InvalidLogicalOperator	Au moins deux sous-éléments doivent être présents dans les opérateurs logiques And ou Or.	400 Bad Request	Non pris en charge
InvalidMatchAnyPrefix	Le MatchAnyPrefix paramètre ne peut pas être vide.	400 Bad Request	Non pris en charge
InvalidMatchAnySuffix	Le MatchAnySuffix paramètre ne peut pas être vide.	400 Bad Request	Non pris en charge
InvalidMatchAnyTag	Le MatchAnyTag paramètre ne peut pas être vide.	400 Bad Request	Non pris en charge
InvalidMatchObjectAge	Le MatchObjectAge paramètre ne peut pas être vide.	400 Bad Request	Non pris en charge
InvalidMatchObjectSize	Le MatchObjectSize paramètre ne peut pas être vide.	400 Bad Request	Non pris en charge

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidName	Le paramètre Storage Lens group Name doit comporter entre 1 et 64 caractères. Le Name paramètre du groupe Storage Lens doit utiliser le <code>^[a-zA-Z0-9\-_]+\$</code> modèle.	400 Bad Request	Non pris en charge
InvalidNumericCombination	Cette combinaison d'âge ou de taille d'objet n'est pas valide.	400 Bad Request	Non pris en charge
InvalidPrefix	La longueur maximale d'un préfixe est de 1 024 caractères. La chaîne de préfixe ne peut pas être vide.	400 Bad Request	Non pris en charge
InvalidSize	BytesLessThan et BytesGreaterThan doivent être des nombres positifs. La taille maximale de l'objet ne peut pas dépasser 50 To. La taille minimale de l'objet ne peut pas être supérieure ou égale à 50 To.	400 Bad Request	Non pris en charge
InvalidSuffix	La longueur maximale d'un suffixe est de 1 024 caractères. La chaîne de suffixe ne peut pas être vide.	400 Bad Request	Non pris en charge

Code d'erreur	Description	Code de statut HTTP	Préfixe du code d'erreur SOAP
InvalidTag	La clé de balise d'objet ne peut pas dépasser 128 caractères. La chaîne clé de la balise d'objet ne peut pas être nulle ou vide. La longueur maximale d'une valeur de balise est de 256 caractères. La clé de balise de l'objet contient des caractères qui ne sont pas valides. La clé de balise d'objet ne doit contenir que a-z A-Z, 0-9, des espaces et les caractères suivants : <code>^(_ . : / = + \ - @] *) \$</code>	400 Bad Request	Non pris en charge
MismatchedName	Le nom spécifié dans la demande ne correspond pas au nom du groupe Storage Lens.	400 Bad Request	Non pris en charge
TooManyConfigurations	Vous avez essayé de créer un plus grand nombre de configurations de groupes de lentilles de stockage que les 50 autorisées.	400 Bad Request	Non pris en charge
TooManyElements	Cela Element dépasse le nombre maximum d'éléments autorisés dans un opérateur logique. Seuls 10 préfixes, suffixes ou balises sont autorisés.	400 Bad Request	Non pris en charge

Liste des codes d'erreur Amazon S3 Object Lambda

Le tableau suivant contient des erreurs spéciales que S3 Object Lambda peut renvoyer. Pour plus d'informations sur les erreurs générales d'Amazon S3 et pour obtenir la liste des codes d'erreur, consultez [Réponses d'erreur](#).

Les réponses d'erreur reçues des points d'accès compatibles lorsqu'il n'y a pas de `GetObject` demande sont envoyées à l'appelant sans modification.

Code d'erreur	Description	Code de statut HTTP
<code>LambdaInvalidResponse</code>	Retourné à l'appelant d'origine lorsqu'il <code>WriteGetObjectResponse</code> répond par <code>ValidationError</code> à AWS Lambda. Consultez le <code>ValidationError</code> message pour plus de détails. Tous les cas ne se <code>ValidationError</code> traduisent pas par une <code>LambdaInvalidResponse</code> erreur.	400 Bad Request
<code>LambdaInvocationFailed</code>	L'invocation de la fonction Lambda a échoué. Les appelants peuvent recevoir l'erreur suivante lorsque S3 Object Lambda ne parvient pas à invoquer correctement la fonction Lambda configurée. Le message d'erreur peut contenir des informations sur une éventuelle erreur renvoyée par le AWS Lambda service lors de l'appel de la fonction (par exemple, code d'état, code	400 Bad Request

Code d'erreur	Description	Code de statut HTTP	
	d'erreur, message d'erreur et ID de demande).		
LambdaNotFound	La AWS Lambda fonction n'a pas été trouvée. La fonction, la version ou l'alias Lambda configurés n'ont pas été trouvés lors de la tentative d'invocation. Veillez à ce que la configuration du point d'accès S3 Object Lambda pointe vers l'ARN correct de la fonction Lambda. Le message d'erreur peut contenir des informations sur une éventuelle erreur renvoyée par le AWS Lambda service lors de l'appel de la fonction (par exemple, code d'état, code d'erreur, message d'erreur et ID de demande).	404 Not Found	

Code d'erreur	Description	Code de statut HTTP	
LambdaPermissionError	L'appelant n'est pas autorisé à invoquer la fonction Lambda. L'appelant doit être autorisé à invoquer la fonction Lambda. Vérifiez les politiques associées à l'appelant et assurez-vous qu'elles ont été autorisées à utiliser <code>lambda:Invoke</code> pour la fonction configurée. Le message d'erreur peut contenir des informations sur une éventuelle erreur renvoyée par le AWS Lambda service lors de l'appel de la fonction (par exemple, code d'état, code d'erreur, message d'erreur et ID de demande).	403 Forbidden	
LambdaResponseNotReceived	La fonction Lambda s'est terminée sans avoir réussi à appeler <code>WriteGetObjectResponse</code>. <code>GetObject</code> les données de réponse sont fournies par la fonction Lambda en appelant l'opération <code>WriteGetObjectResponse</code> API. Les CloudWatch journaux Amazon de la fonction peuvent fournir plus d'informations sur les raisons pour lesquelles la fonction n'a pas réussi à appeler cette opération d'API alors qu'elle se terminait normalement.	500 InternalServiceError	

Code d'erreur	Description	Code de statut HTTP	
LambdaRuntimeError	La fonction Lambda a échoué lors de l'exécution. Une erreur explicite a été reçue de la fonction Lambda. Pour plus de détails sur la panne, consultez les AWS CloudFormation journaux.	500 Internal Service Error	
LambdaTimeout	La fonction Lambda n'a pas répondu dans le délai imparti. La fonction Lambda n'a pas réussi à terminer son appel WriteGetObjectResponse dans les 60 secondes.	500 Internal Service Error	

Code d'erreur	Description	Code de statut HTTP
SlowDown	Réduisez votre taux de demandes pour les opérations impliquant AWS Lambda. L'invocation de la fonction a été limitée par AWS Lambda, peut-être parce qu'elle a atteint sa limite de simultanéité configurée. Pour plus d'informations, consultez la section Gestion de la simultanéité pour une fonction Lambda dans le Guide du AWS Lambda développeur. Le message d'erreur peut contenir des informations sur une éventuelle erreur renvoyée par le AWS Lambda service lors de l'appel de la fonction (par exemple, code d'état, code d'erreur, message d'erreur et ID de demande).	503 Slow Down
ValidationError	Des erreurs de validation peuvent être renvoyées par l'opération de l'WriteGetObjectResponse API et peuvent se produire pour de nombreuses raisons. Consultez le message d'erreur pour plus de détails.	400 Bad Request

Liste des codes d'erreur asynchrones Amazon S3

Le tableau suivant contient les erreurs spéciales que les requêtes asynchrones peuvent renvoyer. Pour obtenir des informations générales sur les erreurs Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Ces erreurs sont renvoyées lorsque vous interrogez sur l'état d'une demande asynchrone, par exemple en utilisant `DescribeMultiRegionAccessPointOperation`. Ces demandes étant asynchrones, le code d'état de toutes ces erreurs est de `200 OK`.

Code d'erreur	Description	Code de statut HTTP	
<code>AccessDenied</code>	Accès refusé.	<code>200 OK</code>	
<code>InternalErrors</code>	Une erreur interne du serveur s'est produite.	<code>200 OK</code>	
<code>MalformedPolicy</code>	La syntaxe de politique spécifiée n'est pas valide.	<code>200 OK</code>	
<code>MultiRegionAccessPointAlreadyOwnedByYou</code>	Vous disposez déjà d'un point Multi-Region d'accès portant le même nom.	<code>200 OK</code>	
<code>MultiRegionAccessPointModifiedByAnotherRequest</code>	L'action a échoué car une autre demande modifie la ressource spécifiée. Essayez de soumettre à nouveau votre demande une fois que la demande précédente aura été traitée.	<code>200 OK</code>	
<code>MultiRegionAccessPointNotReady</code>	Le point Multi-Region d'accès spécifié n'est pas prêt à être mis à jour.	<code>200 OK</code>	
<code>MultiRegionAccessPointSameBucketRegion</code>	Les compartiments utilisés pour créer un point d' Multi-Region accès ne peuvent pas se trouver dans la même région.	<code>200 OK</code>	
<code>MultiRegionAccessPointUnsupportedRegion</code>	L'un des compartiments fournis pour créer le point d' Multi-Region accès	<code>200 OK</code>	

Code d'erreur	Description	Code de statut HTTP	
	se trouve dans une région qui n'est pas prise en charge.		
NoSuchBucket	Le compartiment spécifié n'existe pas.	200 OK	
NoSuchMultiRegionAccessPoint	Le point Multi-Region d'accès spécifié n'existe pas.	200 OK	

Liste des codes d'erreur Amazon S3 Access Grants

Le tableau suivant contient des erreurs spéciales que les demandes S3 Access Grants peuvent renvoyer. Pour obtenir des informations générales sur les erreurs Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Code d'erreur	Description	HTTP Status Code	
AccessGrantAlreadyExists	L'autorisation d'accès spécifiée existe déjà	409	
AccessGrantsInstanceAlreadyExists	L'instance Access Grants existe déjà	409	
AccessGrantsInstanceNotEmptyError	Veuillez nettoyer les emplacements avant de supprimer l'instance des autorisations d'accès	400	
AccessGrantsInstanceDoesNotExistError	L'instance Access Grants n'existe pas	404	
AccessGrantsInstanceResourcePolicyDoesNotExist	La politique relative aux ressources d'instance Access Grants n'existe pas	404	

Code d'erreur	Description	HTTP Status Code	
AccessGrantsLocationAlreadyExistsError	L'emplacement des autorisations d'accès spécifié existe déjà	409	
AccessGrantsLocationNotEmptyError	Veuillez nettoyer les autorisations d'accès avant de supprimer l'emplacement des autorisations d'accès	400	
AccessGrantsLocationsQuotaExceededError	Le quota de localisation des autorisations d'accès a été dépassé. Quota des emplacements des subventions d'accès : <i><value></i> . Veuillez contacter S3 si une augmentation est requise.	409	
AccessGrantsQuotaExceededError	Le quota d'autorisations d'accès a été dépassé. Quota de subventions d'accès : <i><value></i> . Veuillez contacter S3 si une augmentation est requise.	409	
InvalidTag	Votre demande contient des clés de tag dupliquées. Supprimez les clés de tag dupliquées et réessayez.	400	
InvalidAccessGrant	L'autorisation d'accès spécifiée n'est pas valide	400	
InvalidAccessGrantLocation	L'emplacement des autorisations d'accès spécifié n'est pas valide	400	
InvalidIamRole	Le rôle IAM spécifié n'est pas valide	400	
InvalidIdentityCenterInstance	L'instance de centre d'identité spécifiée n'est pas valide	400	

Code d'erreur	Description	HTTP Status Code	
InvalidResourcePolicy	La politique de ressources spécifiée n'est pas valide	400	
InvalidResourcePolicy	La politique de ressources spécifiée n'est pas valide	400	
InvalidTag	Cette demande contient une clé ou une valeur de balise qui n'est pas valide. Les caractères valides sont les suivants : [a-z A-Z +-=._ :/]. Les clés de tag peuvent contenir jusqu'à 128 caractères. Les valeurs des balises peuvent contenir jusqu'à 256 caractères.	400	
NoSuchAccessGrantError	L'autorisation d'accès spécifiée n'existe pas	404	
NoSuchAccessGrantsLocationError	L'emplacement des autorisations d'accès spécifié n'existe pas	404	
AccessDenied	Vous n'êtes pas autorisé <i><requested permission></i> à accéder au préfixe S3 demandé : <i><requested target></i>	403 Forbidden	
StsNotAuthorizedError	Une erreur s'est produite (StsNotAuthorizedError) lors de l'appel de l' GetDataAccess opération : Utilisateur : n'access-grants.s3.amazonaws.com est pas autorisé à effectuer : sts:AssumeRole sur la ressource : <i><IAM Role ARN></i>	403	

Code d'erreur	Description	HTTP Status Code	
StsPackedPolicyTooLargeError	Une erreur s'est produite (StsPackedPolicyTooLargeError) lors de l'appel de l' GetDataAccess opération : jeton sérialisé trop volumineux pour la session	400	
StsValidationError	Le message d'erreur varie en fonction de l'erreur de validation.	400	
InvalidTags	Les clés de balise ne peuvent pas commencer par le préfixe AWS réservé aux balises système. »	400	
TooManyTags	Le nombre de balises dépasse la limite de 50 balises. Supprimez certaines balises et réessayez.	400	

Liste des codes FSx-related d'erreur Amazon

Le tableau suivant contient des erreurs spéciales que les demandes adressées à Amazon FSx via un point d'accès Amazon S3 peuvent renvoyer. Pour obtenir des informations générales sur les erreurs Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Code d'erreur	Description	HTTP Status Code	
AccessDenied	Cette erreur peut se produire dans les conditions suivantes : - L'appelant n'est pas autorisé à effectuer l'opération S3 spécifiée.	403 Forbidden	

Code d'erreur	Description	HTTP Status Code	
	L'appelant n'est pas autorisé à accéder à la ressource Amazon FSx pour la clé spécifiée.		
InsufficientCapacity	La capacité de stockage maximale du système de fichiers a été atteinte.	507 Insufficient Capacity	
InvalidKey	La clé spécifiée n'est pas valide.	400 Bad Request	
SlowDown	Veuillez réduire votre taux de demandes.	503 Slow Down	

Liste des codes d'erreur d'Amazon S3 Tables

Le tableau suivant contient des erreurs spéciales que les opérations d'Amazon S3 Tables peuvent renvoyer. Pour obtenir des informations générales sur les erreurs Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Code d'erreur	Description	HTTP Status Code
AccessDeniedException	L'action ne peut pas être exécutée car vous ne disposez pas de l'autorisation requise.	403 Forbidden
BadRequestException	La demande n'est pas valide ou n'est pas valide.	400 Bad Request
ConflictException	La demande a échoué en raison d'un conflit avec une écriture précédente. Vous pouvez réessayer la demande.	409 Conflict

Code d'erreur	Description	HTTP Status Code
ForbiddenException	L'appelant n'est pas autorisé à faire la demande.	403 Forbidden
InternalServerErrorException	La demande a échoué en raison d'une erreur interne du serveur.	500 Internal Server Error
NotFoundException	La demande a été rejetée car la ressource spécifiée est introuvable.	404 Not Found
TooManyRequestsException	La limite du nombre de requêtes par seconde a été dépassée.	429 Too Many Requests

Bonnes pratiques concernant les erreurs Amazon S3

Beaucoup de réponses d'erreur contiennent des données structurées complémentaires destinées à être lues et comprises par un développeur diagnostiquant des erreurs de programmation. Par exemple, si vous envoyez un Content-MD5 en-tête avec une requête REST PUT qui ne correspond pas au résumé calculé sur le serveur, vous recevez un BadDigest message d'erreur. La réponse d'erreur inclut également comme éléments de détail le résumé qui a été calculé et le résumé que vous avez indiqué comme prévu. Pendant le développement, vous pouvez utiliser ces informations pour diagnostiquer l'erreur. En production, un programme performant doit comprendre ces informations dans son journal des erreurs.

Lors de la conception d'une application devant être utilisée avec Amazon S3, il est important de gérer les erreurs Amazon S3 de façon appropriée. Cette section décrit les erreurs à prendre en compte lors de la conception de votre application.

Réessayez InternalErrors

Les erreurs internes sont des erreurs qui se produisent au sein de l'environnement Amazon S3.

Les demandes qui reçoivent une `InternalServerError` réponse n'ont peut-être pas été traitées. Par exemple, si une requête `PUT` est `InternalServerError` renvoyée, un `GET` suivant peut récupérer l'ancienne valeur ou la valeur mise à jour.

Si Amazon S3 renvoie une `InternalServerError` réponse, réessayez la demande.

Réglez l'application pour éviter les SlowDown erreurs répétées

Comme tout système distribué, S3 possède des mécanismes de protection qui détectent la surconsommation intentionnelle ou non intentionnelle des ressources et réagissent en conséquence. SlowDown des erreurs peuvent survenir lorsqu'un taux de demandes élevé déclenche l'un de ces mécanismes. Réduire le taux de demande diminuera ou éliminera des erreurs de ce type. D'une manière générale, la plupart des utilisateurs ne rencontrent pas ces erreurs régulièrement. Toutefois, si vous souhaitez obtenir plus d'informations ou si vous rencontrez des SlowDown erreurs importantes ou inattendues, veuillez publier un message [sur notre forum des développeurs Amazon S3](#) ou vous inscrire à Support <https://aws.amazon.com/premiumsupport/>.

Isoler les erreurs

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

Amazon S3 fournit un ensemble de codes d'erreur qui sont utilisés par les deux API SOAP et REST. L'API SOAP renvoie des codes d'erreur Amazon S3 standard. L'API REST est conçue pour fonctionner comme un serveur HTTP standard et interagir avec les clients HTTP existants (par exemple, les navigateurs, les bibliothèques client HTTP, les serveurs proxy, les mémoires cache, etc.). Pour garantir que les clients HTTP gèrent correctement les erreurs, Amazon S3 associe chaque erreur Amazon S3 à un code d'état HTTP.

Les codes de statut HTTP sont moins parlants que les codes d'erreur Amazon S3 et contiennent moins d'informations sur l'erreur. Par exemple, les erreurs Amazon S3 `NoSuchKey` et `NoSuchBucket` sont toutes les deux liées au code de statut HTTP `404 Not Found`.

Même si les codes de statut HTTP contiennent moins d'informations sur l'erreur, les clients qui comprennent les codes HTTP, mais pas l'API Amazon S3, sont généralement à même de gérer l'erreur correctement.

Par conséquent, lors de la gestion d'erreurs ou du signalement des erreurs Amazon S3 aux utilisateurs finaux, utilisez le code d'erreur Amazon S3 à la place du code de statut HTTP, car il contient plus d'informations sur l'erreur. De même, lors du débogage de votre application, vous devrez également consulter l'élément lisible <Details> de la réponse d'erreur XML.

AWS Glossaire

Pour la AWS terminologie la plus récente, consultez le [AWS glossaire](#) dans la Glossaire AWS référence.

Ressources Amazon S3

Le tableau suivant liste les ressources connexes qui sont utiles lors de l'utilisation de ce service.

Ressource	Description
Guide de l'utilisateur d'Amazon Simple Storage Service	Le guide de démarrage fournit un didacticiel rapide du service basé sur un cas d'utilisation simple.
Guide de l'utilisateur d'Amazon Simple Storage Service	Le guide du développeur explique comment accomplir des tâches à l'aide des opérations Amazon S3.
FAQ technique sur Amazon S3	La FAQ couvre les 20 questions les plus fréquemment posées par les développeurs à propos de ce produit.
Notes de mise à jour d'Amazon S3	Les notes de publication donnent un aperçu général de la version actuelle. Elles indiquent, en particulier, les nouvelles fonctions et dernières corrections apportées, ainsi que les problèmes connus.
Outils pour Amazon Web Services	Point de départ central pour trouver de la documentation, des exemples de code, des notes de version et d'autres informations qui vous aideront à créer des applications innovantes à l'aide de AWS kits de développement logiciel et d'outils.
Console de gestion AWS	La console vous permet d'exécuter la plupart des fonctions d'Amazon S3 sans programmation.
Forums de discussion	Forum communautaire permettant aux développeurs de discuter de questions techniques liées à Amazon Web Services.
Support Centre	La page d'accueil du support AWS technique, y compris l'accès à nos forums de développeurs, à nos FAQ techniques, à la page d'état du service et à l'assistance Premium.

Ressource	Description
Support	La page Web principale contenant des informations sur Support un canal d'assistance personnalisé et rapide pour vous aider à créer et à exécuter des applications sur AWS les services d'infrastructure.
Informations sur le produit Amazon S3	Page web principale pour de plus amples informations sur Amazon S3.
Contactez-nous	Un point de contact central pour les demandes concernant AWS la facturation, le compte, les événements, les abus, etc.
Conditions d'utilisation	Informations détaillées sur le droit d'auteur et l'utilisation des marques Amazon.com et sur d'autres sujets.

Annexe

Rubriques

- [Annexe : SelectObjectContent Réponse](#)
- [Annexe : objet OPTIONS](#)
- [Annexe : API SOAP](#)
- [Annexe : Demandes d'authentification \(AWS signature \(version 2\)\)](#)
- [Annexe : API de configuration du cycle de vie \(obsolète\)](#)

Annexe : SelectObjectContent Réponse

Description

L'opération Amazon S3 Select filtre le contenu d'un objet Amazon S3 sur la base d'une instruction SQL (Structured Query Language) simple. Étant donné que la taille de réponse de cette opération est inconnue, Amazon S3 Select diffuse la réponse sous la forme d'une série de messages et inclut un `Transfer-Encoding` en-tête dont **chunked** la valeur est la valeur dans la réponse.

Pour plus d'informations sur Amazon S3 Select, consultez la section [Sélection de contenu à partir d'objets](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Pour plus d'informations sur l'utilisation de SQL avec Amazon S3 Select, consultez la section Référence [SQL pour Amazon S3 Select et Amazon Glacier Select](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Réponses

Une opération Amazon S3 Select réussie renvoie un code 200 OK d'état.

En-têtes de réponse

Cette implémentation de l'opération utilise uniquement les en-têtes de réponse communs à la plupart des réponses. Pour plus d'informations, consultez la section En-têtes [de réponse](#) communs.

Corps de la réponse

La taille de réponse d'Amazon S3 Select étant inconnue, Amazon S3 diffuse la réponse sous la forme d'une série de messages et inclut un `Transfer-Encoding` en-tête dont **chunked** la valeur est la valeur dans la réponse. L'exemple suivant montre le format de réponse au niveau supérieur :

```
<Message 1>
<Message 2>
<Message 3>
.....
<Message n>
```

Chaque message est constitué de deux sections : le préambule et les données. La section de prélude comprend 1) la longueur totale en octets du message et 2) la longueur en octets combinée de tous les en-têtes. La section des données comprend 1) les en-têtes et 2) une charge utile.

Chaque section se termine par une somme de contrôle en entier big-endian (CRC) de 4 octets. Amazon S3 Select utilise CRC32 (souvent appelé GZIP CRC32) pour calculer les deux CRC. Pour plus d'informations sur CRC32, consultez la rubrique [Spécification du format de fichiers GZIP version 4.3](#).

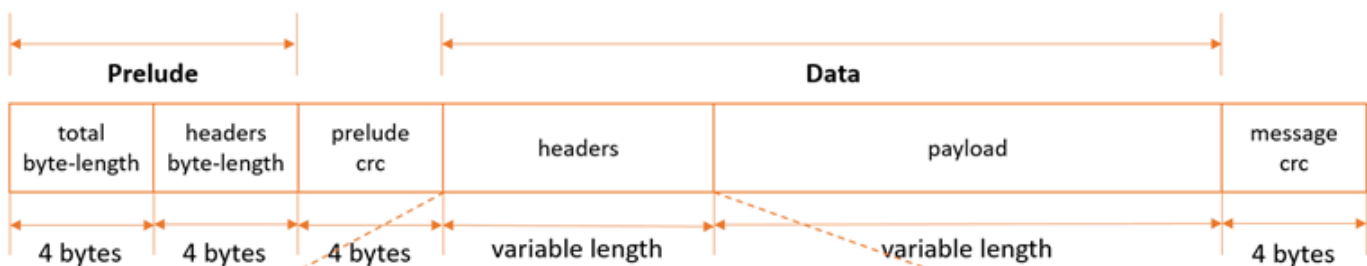
La charge totale des messages, y compris le prélude et les deux sommes de contrôle, est de 16 octets.

Note

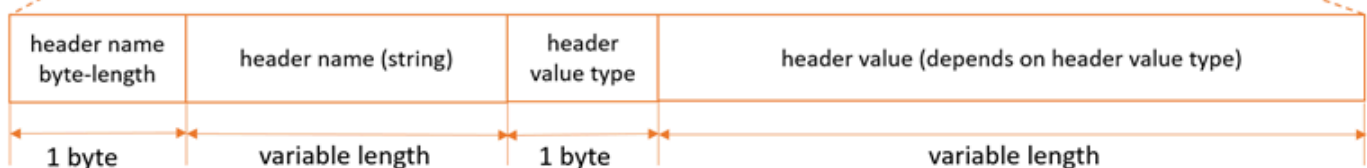
Toutes les integer valeurs contenues dans les messages sont dans l'ordre des octets du réseau, ou dans l'ordre big-endian.

Le schéma suivant montre les composants qui constituent un message et un en-tête. Notez qu'il existe plusieurs en-têtes par message.

Message:



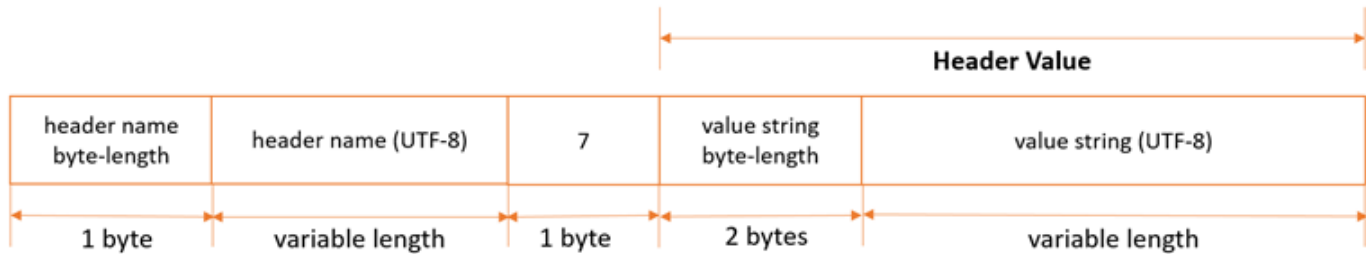
Headers (multiple headers per message):



Note

Pour Amazon S3 Select, le type de valeur d'en-tête est toujours 7 (Type=String). Pour ce type, la valeur d'en-tête se compose de deux composants, un entier big-endian de 2 octets et une UTF-8 chaîne de cette longueur d'octet. Le schéma suivant montre les composants qui constituent les en-têtes Amazon S3 Select.

Amazon S3 Select Headers (type=String):



Calculs de la longueur en octets de la charge utile (ces deux calculs sont équivalents) :

- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - \text{SizeOf}(\text{total\_length}) - \text{SizeOf}(\text{header\_length}) - \text{SizeOf}(\text{prelude\_crc}) - \text{SizeOf}(\text{message\_crc})$
- longueur de la charge utile = longueur_totale - longueur_de l'en-tête - 16

Chaque message comporte les composants suivants :

- **Prélude** : Taille toujours fixe de 8 octets (deux champs de 4 octets chacun) :
 - Quatre premiers octets : longueur totale en octets : longueur en octets Big-endian entière de l'ensemble du message (y compris le champ de longueur totale de 4 octets lui-même).
 - Quatre seconds octets : longueur en octets des en-têtes : longueur en octets Big-endian entière de la partie des en-têtes du message (à l'exclusion du champ de longueur des en-têtes lui-même).
- **CRC du prélu**de : somme de contrôle en entier big-endian (CRC) de 4 octets pour la partie du prélu
de du message (à l'exclusion du CRC lui-même). Le prélude possède un CRC distinct du message CRC (voir ci-dessous), afin de garantir que les informations de longueur d'octet corrompues puissent être détectées immédiatement, sans provoquer de comportement de mise en mémoire tampon pathologique.
- **En-têtes** : ensemble de métadonnées annotant le message, telles que le type de message, le format de la charge utile, etc. Les messages peuvent avoir plusieurs en-têtes, de sorte que cette partie du message peut avoir des longueurs d'octets différentes en fonction du type de message. Les en-têtes sont des paires clé-valeur, où la clé et la valeur sont des chaînes. UTF-8 Les en-têtes peuvent apparaître dans n'importe quel ordre dans la partie des en-têtes du message, et un type d'en-tête donné ne peut apparaître qu'une seule fois.

Pour Amazon S3 Select, voici la liste des noms d'en-têtes et l'ensemble des valeurs valides en fonction du type de message.

- **MessageType En-tête** :
 - HeaderName => « :type de message »
 - Valide HeaderValues => « erreur », « événement »
- **EventType En-tête** :
 - HeaderName => « :type d'événement »
 - Valide HeaderValues => « Enregistrements », « Suite », « Progression », « Statistiques », « Fin »
- **ErrorCode En-tête** :
 - HeaderName => « :code d'erreur »
 - Valide HeaderValues => Code d'erreur indiqué dans le tableau de la [Liste des codes d'erreur relatifs au contenu de l'objet SELECT](#) section.
- **ErrorMessage En-tête** :
 - HeaderName => « :message d'erreur »
 - Valide HeaderValues => Message d'erreur renvoyé par le service, pour aider à diagnostiquer les erreurs au niveau de la demande.
- **Charge utile** : Cela peut être n'importe quoi.
- **Message CRC** : somme de contrôle en entier big-endian (CRC) de 4 octets entre le début du message et le début de la somme de contrôle (c'est-à-dire tout le contenu du message à l'exception du message CRC lui-même).

Chaque en-tête comporte les composants suivants : Il peut y avoir plusieurs en-têtes par message.

- **Nom de l'en-tête Byte-Length** : Byte-length du nom de l'en-tête.
- **Nom de l'en-tête** : nom de l'en-tête, indiquant le type d'en-tête. Valeurs valides : « :message-type » « :event-type » « :error-code » « :error-message »
- **Type de valeur d'en-tête** : énumération indiquant le type de valeur d'en-tête. Pour Amazon S3 Select, c'est toujours 7.
- **Chaîne de valeur Byte-Length** : (pour Amazon S3 Select) Byte-length de la chaîne de valeur d'en-tête.

- Chaîne de valeur d'en-tête : (pour Amazon S3 Select) Valeur de la chaîne d'en-tête. Les valeurs valides pour ce champ varient en fonction du type d'en-tête. Consultez les sections ci-dessous pour connaître les valeurs valides pour chaque type d'en-tête et de message.

Pour Amazon S3 Select, les réponses peuvent être des messages des types suivants :

- Message relatif aux enregistrements : peut contenir un seul enregistrement, des enregistrements partiels ou plusieurs enregistrements. Selon la taille du résultat, une réponse peut contenir un ou plusieurs de ces messages.
- Message de continuation : Amazon S3 envoie régulièrement ce message pour maintenir la connexion TCP ouverte. Ces messages apparaissent au hasard dans les réponses. Le client doit détecter le type de message et le traiter en conséquence.
- Message de progression : Amazon S3 envoie régulièrement ce message, sur demande. Il contient des informations sur la progression d'une requête qui a commencé mais qui n'est pas encore terminée.
- Message de statistiques : Amazon S3 envoie ce message à la fin de la demande. Il contient des statistiques relatives à la requête.
- Message de fin : indique que la demande est terminée et qu'aucun autre message ne sera envoyé. Vous ne devez pas présumer que la demande est complète tant que le client n'a pas reçu de End message.
- RequestLevelError message : Amazon S3 envoie ce message si la demande a échoué pour une raison quelconque. Il contient le code d'erreur et le message d'erreur correspondant à la panne. Si Amazon S3 envoie un RequestLevelError message, il n'en envoie End pas.

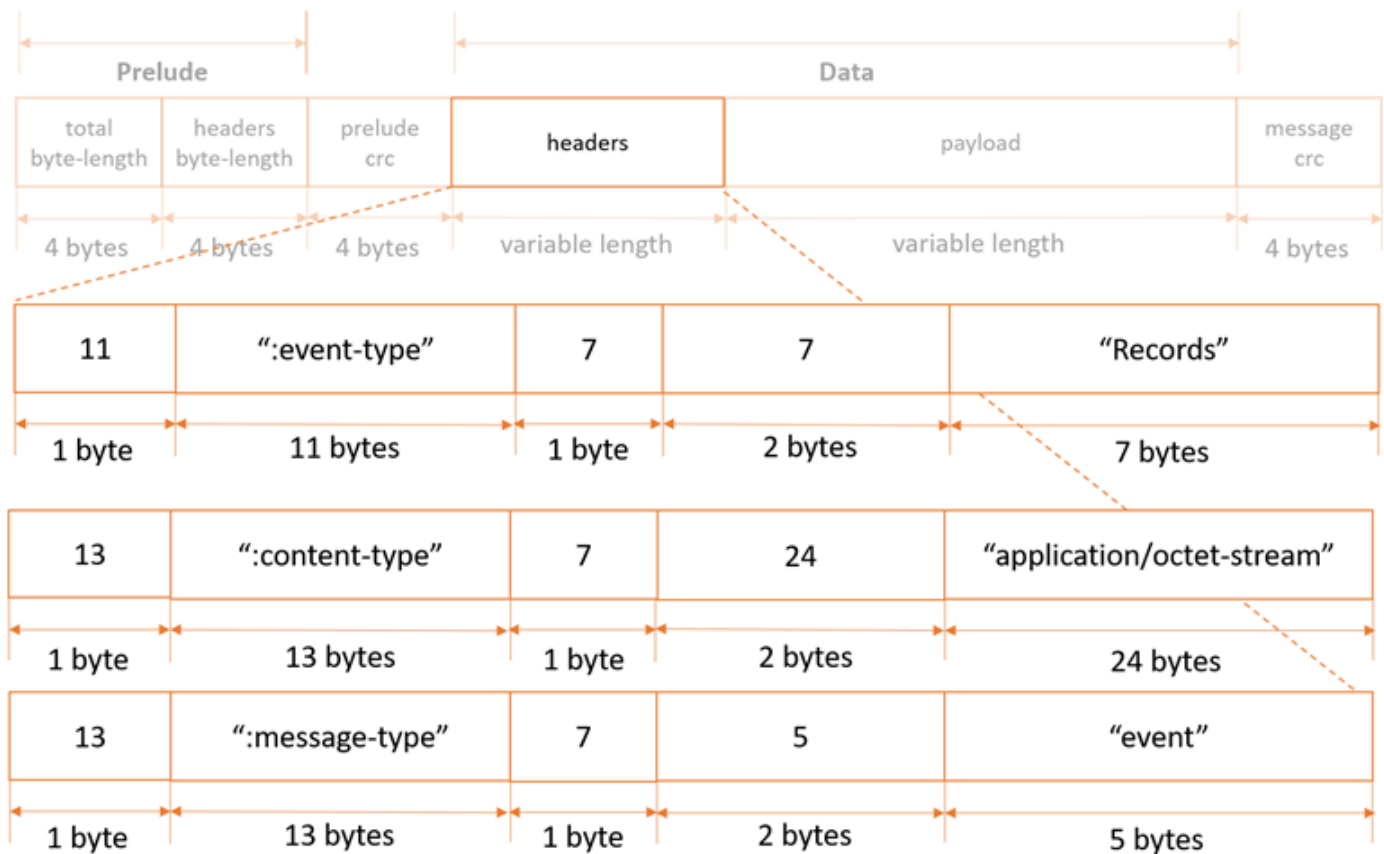
Les sections suivantes expliquent plus en détail la structure de chaque type de message.

Pour des exemples de code et des tests unitaires utilisant ce protocole, consultez [AWS C Event Stream](#) sur le GitHub site Web.

Message d'enregistrement

Spécification de l'en-tête

Les messages d'enregistrement contiennent trois en-têtes, comme suit :



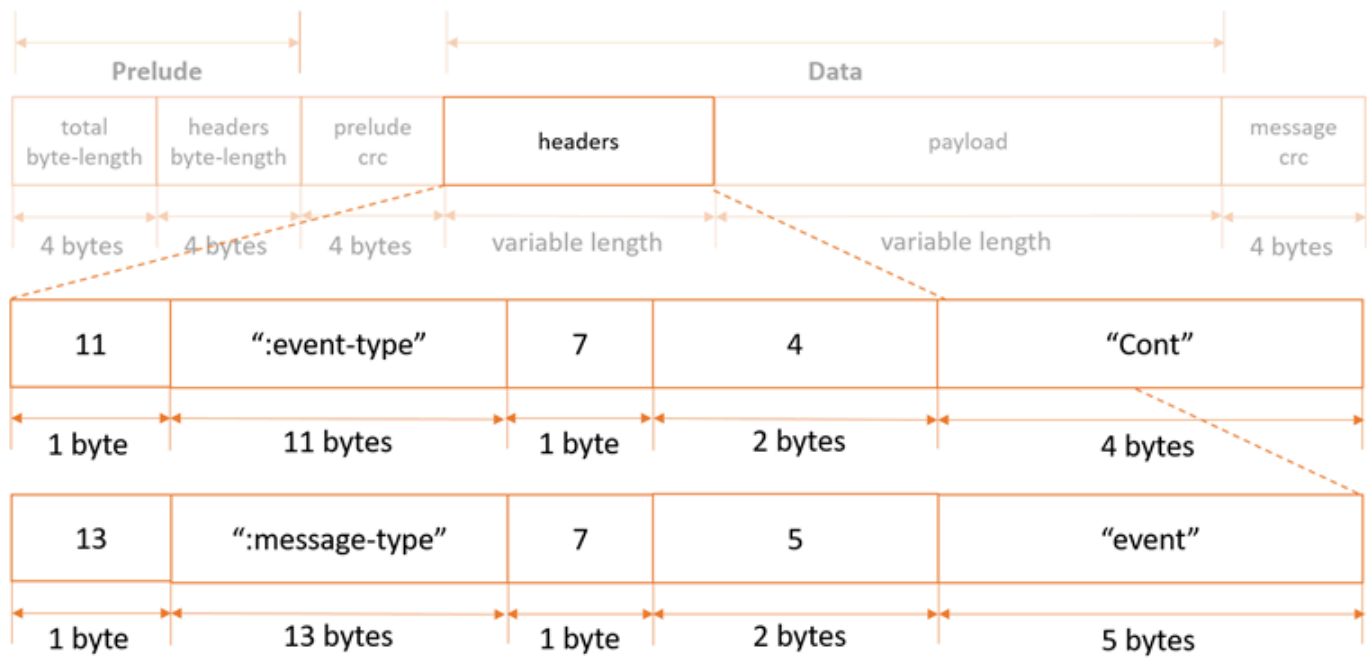
Spécification de la charge utile

Les charges utiles des messages d'enregistrements peuvent contenir un seul enregistrement, des enregistrements partiels ou plusieurs enregistrements.

Message de continuation

Spécification de l'en-tête

Les messages de suite contiennent deux en-têtes, comme suit :



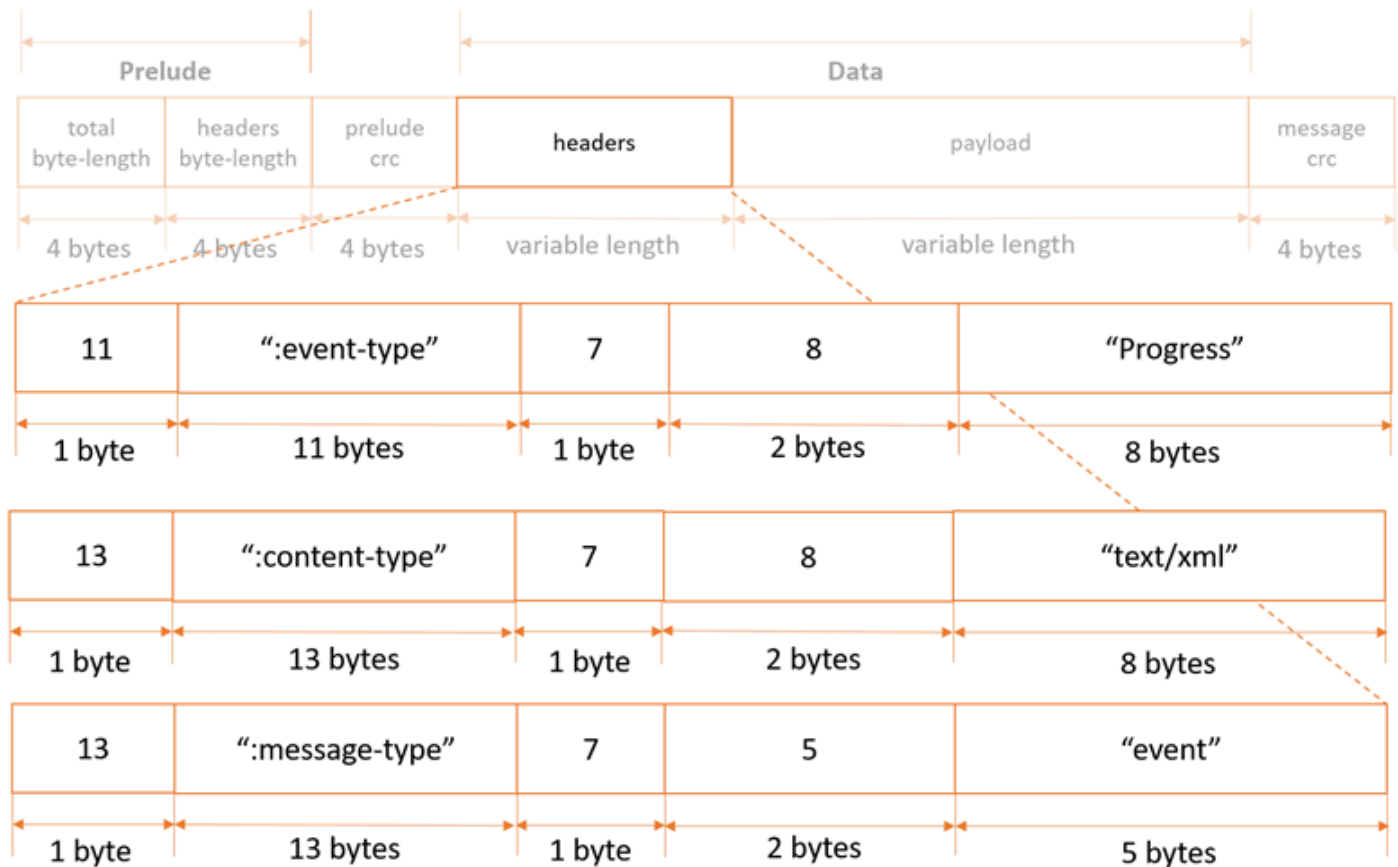
Spécification de la charge utile

Les messages de continuation n'ont aucune charge utile.

Message de progression

Spécification de l'en-tête

Les messages de progression contiennent trois en-têtes, comme suit :



Spécification de la charge utile

La charge utile du message de progression est un document XML contenant des informations sur la progression d'une demande.

- BytesScanned=> Nombre d'octets qui ont été traités avant d'être décompressés (si le fichier est compressé).
- BytesProcessed=> Nombre d'octets qui ont été traités après avoir été décompressés (si le fichier est compressé).
- BytesReturned=> Nombre actuel d'octets de données utiles d'enregistrements renvoyés par Amazon S3.

Pour les fichiers non compressés, BytesScanned et BytesProcessed sont identiques.

Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

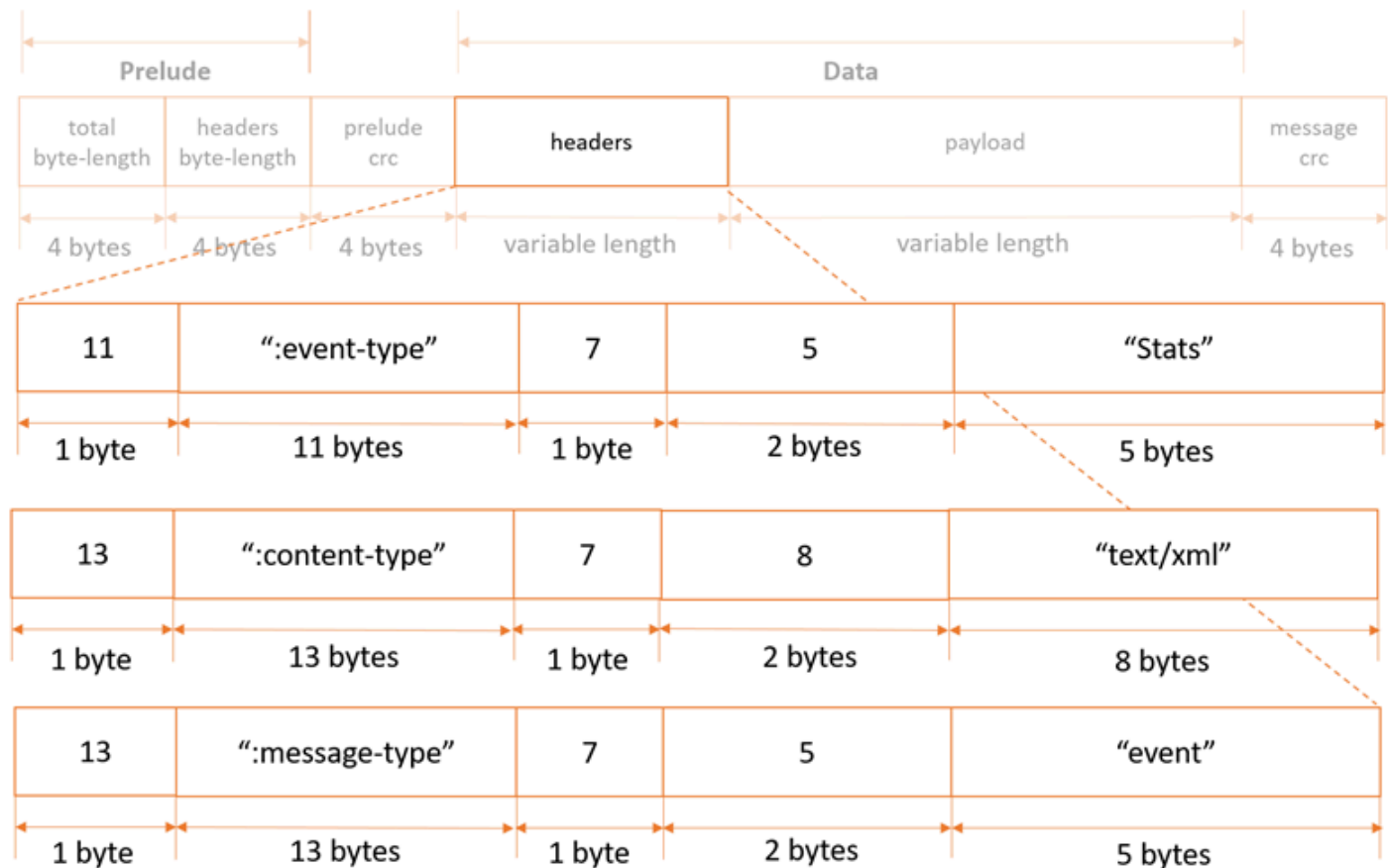
<Progress>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Progress>

```

Message de statistiques

Spécification de l'en-tête

Les messages de statistiques contiennent trois en-têtes, comme suit :



Spécification de la charge utile

La charge utile du message de statistiques est un document XML contenant des informations sur les statistiques d'une demande une fois le traitement terminé.

- BytesScanned=> Nombre d'octets qui ont été traités avant d'être décompressés (si le fichier est compressé).

- BytesProcessed=> Nombre d'octets qui ont été traités après avoir été décompressés (si le fichier est compressé).
- BytesReturned=> Nombre total d'octets de données utiles d'enregistrements renvoyés par Amazon S3.

Pour les fichiers non compressés, BytesScanned et BytesProcessed sont identiques.

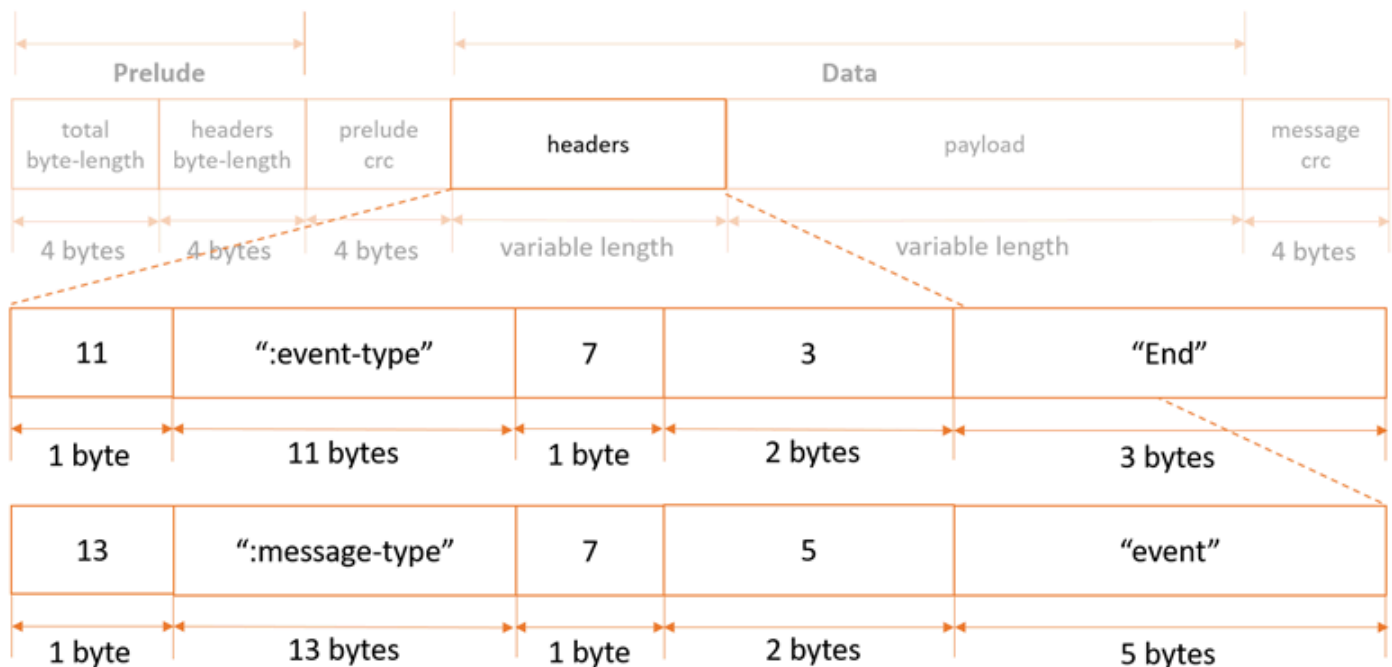
Exemple :

```
<?xml version="1.0" encoding="UTF-8"?>
<Stats>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Stats>
```

Fin du message

Spécification de l'en-tête

Les messages de fin contiennent deux en-têtes, comme suit :



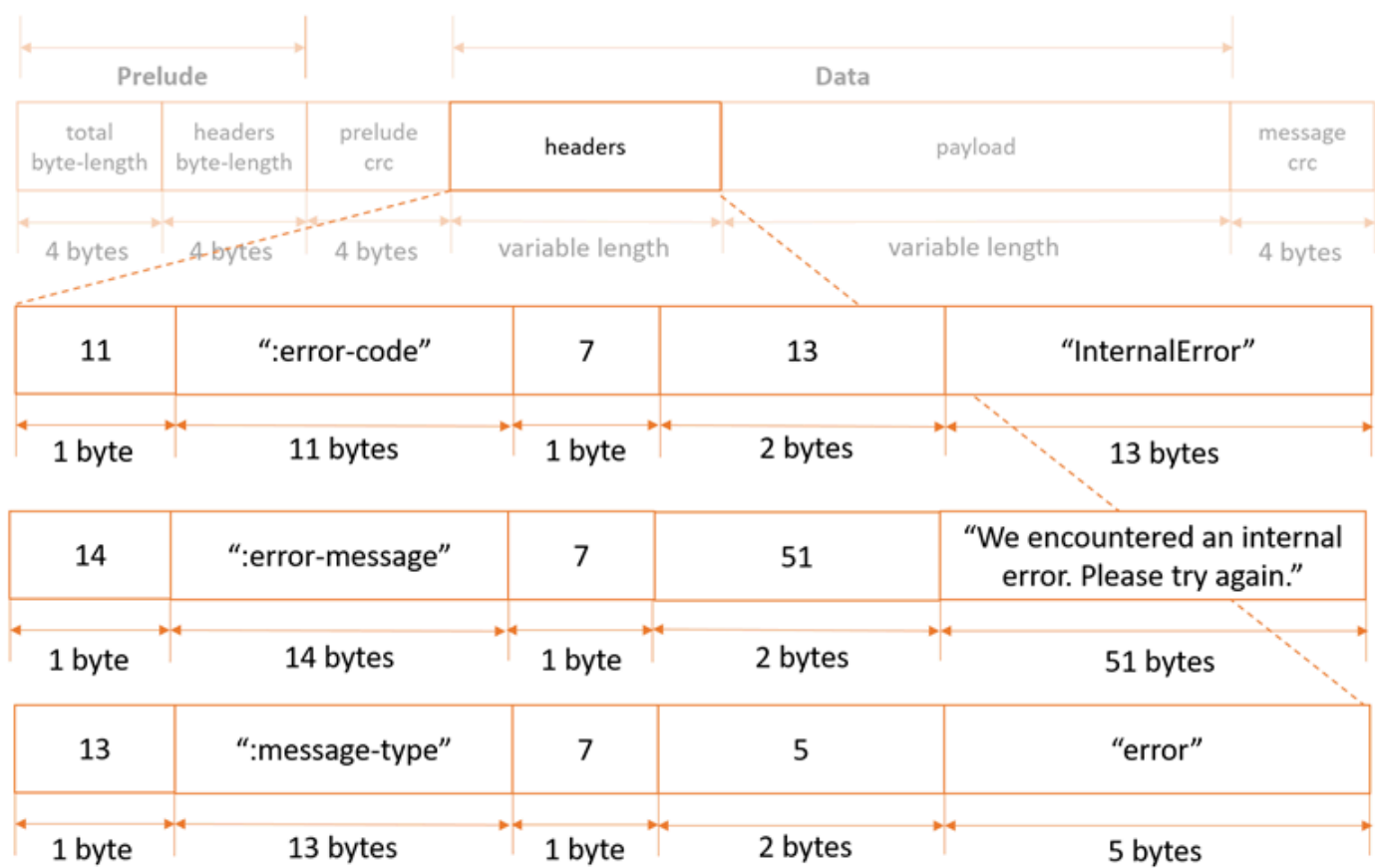
Spécification de la charge utile

Les messages de fin n'ont aucune charge utile.

Message d'erreur au niveau de la demande

Spécification de l'en-tête

Request-level les messages d'erreur contiennent trois en-têtes, comme suit :



Pour obtenir la liste des codes d'erreur et des messages d'erreur possibles, consultez le [Liste des codes d'erreur relatifs au contenu de l'objet SELECT](#).

Spécification de la charge utile

Request-level les messages d'erreur n'ont aucune charge utile.

Ressources connexes

- [SelectObjectContent](#)

- [GetObject](#)
- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)

Annexe : objet OPTIONS

Description

Un navigateur peut envoyer cette demande de contrôle préalable à Amazon S3 afin de déterminer s'il peut envoyer une demande réelle avec l'origine, la méthode HTTP et les en-têtes spécifiques.

Amazon S3 prend en charge le partage de ressources entre origines (CORS) en vous permettant d'ajouter une `cors` sous-ressource dans un compartiment. Lorsqu'un navigateur envoie cette demande de contrôle préalable, Amazon S3 répond en évaluant les règles définies dans la `cors` configuration.

Si cette option `cors` n'est pas activée sur le compartiment, Amazon S3 renvoie une 403 Forbidden réponse.

Pour plus d'informations sur CORS, consultez la rubrique [Activation du partage Cross-Origin des ressources](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Requêtes

Syntaxe

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

Paramètres de demande

Cette opération n'introduit aucun paramètre de demande spécifique, mais elle peut contenir tous les paramètres de demande requis par la demande elle-même.

En-têtes de demande

Nom	Description	Obligatoire
Origin		Oui

Nom	Description	Obligatoire
	Identifie l'origine de la demande d'origine croisée adressée à Amazon S3. Par exemple, <code>http://www.example.com</code>. Type : Chaîne Par défaut : aucun	
Access-Control-Request-Method	Identifie la méthode HTTP qui sera utilisée dans la demande réelle. Type : Chaîne Par défaut : aucun	Oui
Access-Control-Request-Headers	Une liste d'en-têtes HTTP séparés par des virgules qui seront envoyés dans la demande proprement dite. Par exemple, pour placer un objet avec un chiffrement côté serveur, cette demande de contrôle préalable déterminera si l'<code>x-amz-server-side-encryption</code> en-tête peut être inclus dans la demande. Type : Chaîne Par défaut : aucun	Non

Elements de la demande

Cette implémentation de l'opération n'utilise pas d'éléments de requête.

Réponses

En-têtes de réponse

En-tête	Description
Access-Control-Allow-Origin	L'origine que vous avez envoyée dans votre demande. Si l'origine de votre demande n'est pas autorisée, Amazon S3 n'inclura pas cet en-tête dans la réponse. Type : Chaîne
Access-Control-Max-Age	Durée, en secondes, pendant laquelle les résultats de la demande de contrôle préalable peuvent être mis en cache. Type : Chaîne
Access-Control-Allow-Methods	Méthode HTTP qui a été envoyée dans la demande d'origine. Si la méthode de la demande n'est pas autorisée, Amazon S3 n'inclura pas cet en-tête dans la réponse. Type : Chaîne
Access-Control-Allow-Headers	Liste d'en-têtes HTTP séparés par des virgules que le navigateur peut envoyer dans la demande proprement dite. Si l'un des en-têtes demandés n'est pas autorisé, Amazon S3 n'inclura pas cet en-tête dans la réponse, et celle-ci ne contiendra aucun des en-têtes avec le <code>Access-Control</code> préfixe. Type : Chaîne
Access-Control-Expose-Headers	Liste d'en-têtes HTTP séparés par des virgules. Cet en-tête permet au JavaScript client d'accéder à ces en-têtes dans la réponse à la demande proprement dite. Type : Chaîne

Éléments de réponse

Cette implémentation de l'opération ne renvoie pas d'éléments de réponse.

Exemples

Exemple : envoyer une requête Preflight OPTIONS à un bucket compatible **CORS**

Un navigateur peut envoyer cette demande de contrôle préalable à Amazon S3 afin de déterminer s'il peut envoyer la véritable demande PUT depuis l'`http://www.example.com`origine vers le compartiment Amazon S3 nommé `examplebucket`.

Exemple de demande

```
OPTIONS /exampleobject HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

Exemple de réponse

```
HTTP/1.1 200 OK
x-amz-id-2: 6SvaESv3VULYPLik5LL171SPPtSnBvDdGmnk1X1HfU17uS2m1DF6td6KWKNjYMXZ
x-amz-request-id: BDC4B83DF5096BBE
Date: Wed, 21 Aug 2012 23:09:55 GMT
Etag: "1f1a1af1f111111111111111c11aed1da1"
Access-Control-Allow-Origin: http://www.example.com
Access-Control-Allow-Methods: PUT
Access-Control-Expose-Headers: x-amz-request-id
Content-Length: 0
Server: AmazonS3
```

Ressources connexes

- [GetBucketCors](#)
- [DeleteBucketCors](#)
- [PutBucketCors](#)

Annexe : API SOAP

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

Cette section décrit l'API SOAP en ce qui concerne les opérations relatives aux services, aux compartiments et aux objets. Notez que les requêtes SOAP, à la fois authentifiées et anonymes, doivent être envoyées à Amazon S3 via SSL. Amazon S3 renvoie un message d'erreur lorsque vous envoyez une demande SOAP via HTTP.

La dernière version d'Amazon S3 WSDL est disponible sur docs.aws.amazon. [com/2006-03- 01/AmazonS3](https://docs.aws.amazon.com/2006-03-01/AmazonS3) .wsdl.

Rubriques

- [Opérations sur le service \(API SOAP\)](#)
- [Opérations sur les buckets \(API SOAP\)](#)
- [Opérations sur les objets \(API SOAP\)](#)
- [Authentification des demandes SOAP](#)
- [Configuration d'une stratégie d'accès avec SOAP](#)
- [Éléments communs](#)
- [Réponses aux erreurs SOAP](#)

Opérations sur le service (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

Cette section décrit les opérations que vous pouvez effectuer sur le service Amazon S3.

Rubriques

- [ListAllMyBuckets \(API SOAP\)](#)

ListAllMyBuckets (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'opération `ListAllMyBuckets` renvoie une liste de tous les compartiments appartenant à l'expéditeur de la demande.

Exemple

Sample Request

```
<ListAllMyBuckets xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListAllMyBuckets>
```

Sample Response

```
<ListAllMyBucketsResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Owner>
    <ID>bcaf1ffd86f41161ca5fb16fd081034f</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes</Name>
      <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>samples</Name>
      <CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```



```
</Buckets>
</ListAllMyBucketsResult>
```

Corps de la réponse

- **Owner :**

Cela fournit des informations qu'Amazon S3 utilise pour représenter votre identité à des fins d'authentification et de contrôle d'accès. L'ID est un identifiant unique et permanent pour le développeur qui a fait la demande. DisplayName est un nom lisible par l'homme représentant le développeur qui a fait la demande. Il n'est pas unique et peut changer au fil du temps. Nous vous recommandons de faire correspondre votre nom DisplayName à celui de votre forum.

- **Name :**

Le nom d'un seau. Notez que si l'un de vos compartiments a été supprimé récemment, le nom du compartiment supprimé peut encore figurer dans cette liste pendant un certain temps.

- **CreationDate :**

Heure à laquelle le bucket a été créé.

Contrôle d'accès

Vous devez vous authentifier à l'aide d'un ID de clé AWS d'accès valide. Les demandes anonymes ne sont jamais autorisées à répertorier des compartiments, et vous ne pouvez répertorier que des compartiments dont vous êtes le propriétaire.

Opérations sur les buckets (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

Cette section décrit les opérations que vous pouvez effectuer sur les compartiments Amazon S3.

Rubriques

- [CreateBucket \(API SOAP\)](#)

- [DeleteBucket \(API SOAP\)](#)
- [ListBucket \(API SOAP\)](#)
- [GetBucketAccessControlPolicy \(API SOAP\)](#)
- [SetBucketAccessControlPolicy \(API SOAP\)](#)
- [GetBucketLoggingStatus \(API SOAP\)](#)
- [SetBucketLoggingStatus \(API SOAP\)](#)

CreateBucket (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'CreateBucket opération crée un compartiment. Toutes les chaînes ne constituent pas un nom de compartiment acceptable. Pour plus d'informations sur les restrictions relatives à la dénomination des compartiments, consultez la section [Utilisation des compartiments](#) Amazon S3.

Note

Pour déterminer s'il existe un nom de compartiment, utilisez-le ListBucket et MaxKeys définissez-le sur 0. Une NoSuchBucket réponse indique que le compartiment est disponible, une AccessDenied réponse indique que quelqu'un d'autre est propriétaire du compartiment et une réponse de réussite indique que vous êtes propriétaire du compartiment ou que vous êtes autorisé à y accéder.

Exemple Créez un bucket nommé « quotes »

Sample Request

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
```

```
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

Sample Response

```
<CreateBucketResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <CreateBucketResponse>
    <Bucket>quotes</Bucket>
  </CreateBucketResponse>
</CreateBucketResponse>
```

Eléments

- **Bucket** : Le nom du compartiment que vous essayez de créer.
- **AccessControlList** : La liste de contrôle d'accès pour le nouveau compartiment. Cet élément est facultatif. S'il n'est pas fourni, le bucket est créé avec une politique d'accès qui donne au demandeur un accès FULL_CONTROL.

Contrôle d'accès

Vous devez vous authentifier à l'aide d'un ID de clé AWS d'accès valide. Les demandes anonymes ne sont jamais autorisées pour créer des buckets.

Ressources connexes

- [ListBucket \(API SOAP\)](#)

DeleteBucket (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'**DeleteBucket** opération supprime un compartiment. Tous les objets du compartiment doivent être supprimés pour que le compartiment lui-même puisse être supprimé.

Exemple

Cet exemple supprime le compartiment « guillemets ».

Sample Request

```
<DeleteBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteBucket>
```

Sample Response

```
<DeleteBucketResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <DeleteBucketResponse>
    <Code>204</Code>
    <Description>No Content</Description>
  </DeleteBucketResponse>
</DeleteBucketResponse>
```

Éléments

- **Bucket** : Le nom du compartiment que vous souhaitez supprimer.

Contrôle d'accès

Seul le propriétaire d'un bucket est autorisé à le supprimer, quelle que soit la politique de contrôle d'accès du bucket.

ListBucket (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'opération `ListBucket` renvoie des informations sur certains éléments du compartiment.

Pour une présentation générale du fonctionnement de la liste, consultez la section [Clés des objets de liste](#).

Requêtes

Cet exemple répertorie jusqu'à 1 000 clés dans le compartiment « guillemets » qui ont le préfixe « notes ».

Syntaxe

```
<ListBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Prefix>notes</Prefix>
  <Delimiter></Delimiter>
  <MaxKeys>1000</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Paramètres

Nom	Description	Obligatoire
prefix	Limite la réponse aux touches commençant par le préfixe indiqué. Vous pouvez utiliser des préfixes pour séparer un compartiment en différents ensembles de clés de la même manière qu'un système de fichiers utilise les dossiers. Type : Chaîne Par défaut : aucun	Non
marker	Indique où commencer la mise en vente dans le compartiment. La liste inclura uniquement les clés qui apparaissent lexicographiquement après le marqueur. C'est pratique pour la pagination : pour obtenir la page de résultats suivante, utilisez la dernière touche de la page en cours comme marqueur. Type : Chaîne	Non

Nom	Description	Obligatoire
	Par défaut : aucun	
max-keys	Le nombre maximum de touches que vous souhaitez voir apparaître dans le corps de la réponse. Le serveur peut renvoyer un nombre inférieur à ce nombre de clés, mais il n'en renverra pas plus. Type : Chaîne Par défaut : aucun	Non
delimiter	Fait en sorte que les clés contenant la même chaîne entre le préfixe et la première occurrence du délimiteur soient regroupées en un seul élément de résultat de la CommonPrefixes collection. Ces clés enroulées ne sont pas renvoyées ailleurs dans la réponse. Type : Chaîne Par défaut : aucun	Non

Réponse réussie

Cette réponse suppose que le compartiment contient les clés suivantes :

```
notes/todos.txt
notes/2005-05-23/customer_mtg_notes.txt
notes/2005-05-23/phone_notes.txt
notes/2005-05-28/sales_notes.txt
```

Syntaxe

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>backups</Name>
  <Prefix>notes</Prefix>
  <MaxKeys>1000</MaxKeys>
  <Delimiter></Delimiter>
```

```
<IsTruncated>false</IsTruncated>
<Contents>
  <Key>notes/todos.txt</Key>
  <LastModified>2006-01-01T12:00:00.000Z</LastModified>
  <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
  <Size>5126</Size>
  <StorageClass>STANDARD</StorageClass>
  <Owner>
    <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <StorageClass>STANDARD</StorageClass>
</Contents>
<CommonPrefixes>
  <Prefix>notes/2005-05-23/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
  <Prefix>notes/2005-05-28/</Prefix>
</CommonPrefixes>
</ListBucketResult>
```

Comme vous pouvez le constater, de nombreux champs de la réponse font écho aux paramètres de la demande. `IsTruncatedContents`, et `CommonPrefixes` sont les seuls éléments de réponse qui peuvent contenir de nouvelles informations.

Éléments de réponse

Nom	Description
Contents	Métadonnées relatives à chaque objet renvoyé. Type : métadonnées XML Ancêtre : ListBucketResult
CommonPrefixes	Une réponse ne peut contenir <code>CommonPrefixes</code> que si vous spécifiez <code>undelimiter</code> . Lorsque vous le faites, <code>CommonPrefixes</code> contient toutes les clés (le cas échéant) comprises entre <code>Prefix</code> la prochaine occurrence de la chaîne spécifiée par <code>delimiter</code> . En fait, <code>CommonPrefixes</code> répertorie les clés qui agissent comme des sous-répertoires dans le répertoire spécifié par <code>Prefix</code> . Par exemple, si <code>prefix</code> est <code>notes/</code> et <code>delimiter</code> est une

Nom	Description
	barre oblique (/), dans <code>notes/summer/july</code> , le préfixe commun est. <code>notes/summer/</code> Type : Chaîne Ancêtre : ListBucketResult
Delimiter	Fait en sorte que les clés contenant la même chaîne entre le préfixe et la première occurrence du délimiteur soient regroupées en un seul élément de résultat de la CommonPrefixes collection. Ces clés enroulées ne sont pas renvoyées ailleurs dans la réponse. Type : Chaîne Ancêtre : ListBucketResult
IsTruncated	Spécifie si (vrai) ou non (faux) tous les résultats ont été renvoyés. Tous les résultats peuvent ne pas être renvoyés si le nombre de résultats dépasse celui spécifié par MaxKeys. Type : Chaîne Ancêtre : booléen
Marker	Indique où commencer la mise en vente dans le compartiment. Type : Chaîne Ancêtre : ListBucketResult
MaxKeys	Le nombre maximum de clés renvoyées dans le corps de la réponse. Type : Chaîne Ancêtre : ListBucketResult

Nom	Description
Name	Nom du seau. Type : Chaîne Ancêtre : ListBucketResult
Prefix	Clés commençant par le préfixe indiqué. Type : Chaîne Ancêtre : ListBucketResult

Corps de la réponse

Pour plus d'informations sur la réponse de la liste, voir [Listing Keys Response](#).

Contrôle d'accès

Pour répertorier les clés d'un compartiment, vous devez avoir obtenu l'autorisation d'READY accéder.

GetBucketAccessControlPolicy (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'GetBucketAccessControlPolicyopération récupère la politique de contrôle d'accès pour un bucket.

Exemple

Cet exemple récupère la politique de contrôle d'accès pour le compartiment « quotes ».

Sample Request

```
<GetBucketAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
```

```
<Bucket>quotes</Bucket>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetBucketAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd2441bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd2441bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Corps de la réponse

La réponse contient la politique de contrôle d'accès pour le bucket. Pour une explication de cette réponse, voir Politique d'accès [SOAP](#).

Contrôle d'accès

Vous devez disposer de READ_ACP droits sur le compartiment pour pouvoir récupérer la politique de contrôle d'accès d'un compartiment.

SetBucketAccessControlPolicy (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'opération `SetBucketAccessControlPolicy` définit la politique de contrôle d'accès pour un compartiment existant. En cas de succès, la politique de contrôle d'accès précédente pour le bucket est entièrement remplacée par la politique de contrôle d'accès spécifiée.

Exemple

Donnez à l'utilisateur spécifié (généralement le propriétaire) l'`FULL_CONTROL` accès au compartiment « citations ».

Sample Request

```
<SetBucketAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b8863000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE= </Signature>
</SetBucketAccessControlPolicy >
```

Sample Response

```
<GetBucketAccessControlPolicyResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetBucketAccessControlPolicyResponse>
    <Code>200</Code>
```

```
<Description>OK</Description>
</GetBucketAccessControlPolicyResponse>
</GetBucketAccessControlPolicyResponse>
```

Contrôle d'accès

Vous devez disposer de WRITE_ACP droits sur le compartiment afin de définir la politique de contrôle d'accès pour un compartiment.

GetBucketLoggingStatus (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

GetBucketLoggingStatus Récupère l'état de journalisation d'un compartiment existant.

Pour une présentation générale de cette fonctionnalité, consultez la section Journaux [du serveur](#).

Exemple

Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
    <soap:Body>
      <GetBucketLoggingStatus xmlns="https://doc.s3.amazonaws.com/2006-03-01">
        <Bucket>mybucket</Bucket>
        <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
        <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
        <Signature>YOUR_SIGNATURE_HERE</Signature>
      </GetBucketLoggingStatus>
    </soap:Body>
  </soap:Envelope>
```

Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <GetBucketLoggingStatusResponse xmlns="https://s3.amazonaws.com/
doc/2006-03-01">
        <GetBucketLoggingStatusResponse>
          <LoggingEnabled>
            <TargetBucket>mylogs</TargetBucket>
            <TargetPrefix>mybucket-access_log-</TargetPrefix>
          </LoggingEnabled>
        </GetBucketLoggingStatusResponse>
      </GetBucketLoggingStatusResponse>
    </soapenv:Body>
  </soapenv:Envelope>
```

Contrôle d'accès

Seul le propriétaire d'un bucket est autorisé à invoquer cette opération.

SetBucketLoggingStatus (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'opération `SetBucketLoggingStatus` met à jour l'état de journalisation d'un compartiment existant.

Pour une présentation générale de cette fonctionnalité, consultez la section Journaux [du serveur](#).

Exemple

Cet exemple de demande active la journalisation des accès au serveur pour le compartiment « mybucket » et configure les journaux à transmettre à « mylogs » sous le préfixe « access_log- »

Sample Request

```
<?xml version="1.0" encoding="utf-8"?>
  <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://
www.w3.org/2001/XMLSchema">
    <soap:Body>
      <SetBucketLoggingStatus xmlns="https://doc.s3.amazonaws.com/2006-03-01">
        <Bucket>myBucket</Bucket>
        <AWSAccessKeyId>YOUR_AWS_ACCESS_KEY_ID</AWSAccessKeyId>
        <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
        <Signature>YOUR_SIGNATURE_HERE</Signature>
        <BucketLoggingStatus>
          <LoggingEnabled>
            <TargetBucket>mylogs</TargetBucket>
            <TargetPrefix>mybucket-access_log-</TargetPrefix>
          </LoggingEnabled>
        </BucketLoggingStatus>
      </SetBucketLoggingStatus>
    </soap:Body>
  </soap:Envelope>
```

Sample Response

```
<?xml version="1.0" encoding="utf-8"?>
  <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/
XMLSchema-instance" >
    <soapenv:Header>
    </soapenv:Header>
    <soapenv:Body>
      <SetBucketLoggingStatusResponse xmlns="https://s3.amazonaws.com/
doc/2006-03-01"/>
    </soapenv:Body>
  </soapenv:Envelope>
```

Contrôle d'accès

Seul le propriétaire d'un bucket est autorisé à invoquer cette opération.

Opérations sur les objets (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

Cette section décrit les opérations que vous pouvez effectuer sur les objets Amazon S3.

Rubriques

- [PutObjectInline \(API SOAP\)](#)
- [PutObject \(API SOAP\)](#)
- [CopyObject \(API SOAP\)](#)
- [GetObject \(API SOAP\)](#)
- [GetObjectExtended \(API SOAP\)](#)
- [DeleteObject \(API SOAP\)](#)
- [GetObjectAccessControlPolicy \(API SOAP\)](#)
- [SetObjectAccessControlPolicy \(API SOAP\)](#)

PutObjectInline (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'PutObjectInlineopération ajoute un objet à un compartiment. Les données de l'objet sont fournies dans le corps du message SOAP.

Si un objet existe déjà dans un compartiment, le nouvel objet le remplacera car Amazon S3 stocke la dernière demande d'écriture. Amazon S3 est toutefois un système distribué. Si Amazon S3 reçoit

plusieurs demandes d'écriture pour le même objet presque simultanément, tous les objets peuvent être stockés, même si un seul gagne au final. Amazon S3 ne fournit pas de verrouillage des objets ; si vous en avez besoin, assurez-vous de l'intégrer à votre couche d'application.

Pour vous assurer qu'un objet n'est pas endommagé sur le réseau, vous pouvez calculer le MD5 d'un objet, le placer dans Amazon S3 et comparer l'Etag renvoyé à la valeur MD5 calculée.

PutObjectInline ne convient pas à une utilisation avec de gros objets. Le système limite cette opération à l'utilisation d'objets de 1 Mo ou moins. PutObjectInline échouera avec le code `InlineDataTooLargeError` d'état si le paramètre `Data` code un objet de plus de 1 Mo. Pour charger des objets volumineux, pensez à utiliser l'API non intégrée ou PutObject l'API REST à la place.

Exemple

Cet exemple écrit du texte et des métadonnées dans l'objet « Nelson » du compartiment « guillemets », permet à un utilisateur (généralement le propriétaire) `FULL_CONTROL` d'accéder à l'objet et le rend lisible par des personnes anonymes.

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fde241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
```



```
<Grant>
  <Grantee xsi:type="Group">
    <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
  </Grantee>
  <Permission>READ</Permission>
</Grant>
</AccessControlList>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <LastModified>2006-01-01T12:00:00.000Z</lastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

Éléments

- **Bucket** : Le compartiment dans lequel ajouter l'objet.
- **Key** : La touche à attribuer à l'objet.
- **Metadata** : Vous pouvez fournir des paires de métadonnées nom-valeur dans l'élément de métadonnées. Ils seront stockés avec l'objet.
- **Data** : La forme codée en base 64 des données.
- **ContentLength** : Longueur des données en octets.
- **AccessControlList** : Une liste de contrôle d'accès pour la ressource. Cet élément est facultatif. En cas d'omission, le demandeur a FULL_CONTROL accès à l'objet. Si l'objet existe déjà, la politique de contrôle d'accès préexistante est remplacée.

Réponses

- **ETag** : La balise d'entité est un hachage MD5 de l'objet que vous pouvez utiliser pour effectuer des extractions conditionnelles de l'objet. `GetObjectExtended` L'ETag reflète uniquement les modifications apportées au contenu d'un objet, pas à ses métadonnées.
- **LastModified** : L'horodatage Amazon S3 de l'objet enregistré.

Contrôle d'accès

Vous devez avoir **WRITE** accès au seau pour y placer des objets.

Ressources connexes

- [PutObject \(API SOAP\)](#)
- [CopyObject \(API SOAP\)](#)

PutObject (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'opération `PutObject` ajoute un objet à un compartiment. Les données de l'objet sont jointes en tant que pièce jointe MIME.

Pour vous assurer qu'un objet n'est pas endommagé sur le réseau, vous pouvez calculer le MD5 d'un objet, le placer dans Amazon S3 et comparer l'ETag renvoyé à la valeur MD5 calculée.

Si un objet existe déjà dans un compartiment, le nouvel objet le remplacera car Amazon S3 stocke la dernière demande d'écriture. Amazon S3 est toutefois un système distribué. Si Amazon S3 reçoit plusieurs demandes d'écriture pour le même objet presque simultanément, il est possible que tous les objets soient stockés, même si un seul gagne au final. Amazon S3 ne fournit pas de verrouillage des objets ; si vous en avez besoin, assurez-vous de l'intégrer à votre couche d'application.

Example

Cet exemple place certaines données et métadonnées dans l'objet « Nelson » du compartiment « quotes », permet à un utilisateur (généralement le propriétaire) d'`FULL_CONTROL` accéder à l'objet et le rend lisible par des personnes anonymes. Dans cet exemple, la pièce jointe réelle n'est pas affichée.

Sample Request

```
<PutObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers<URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2007-05-11T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObject>
```

Sample Response

```
<PutObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2006-03-01T12:00:00.183Z</LastModified>
  </PutObjectResponse>
</PutObjectResponse>
```

Éléments

- **Bucket** : Le compartiment dans lequel ajouter l'objet.
- **Key** : La touche à attribuer à l'objet.
- **Metadata** : Vous pouvez fournir des paires de métadonnées nom-valeur dans l'élément de métadonnées. Ils seront stockés avec l'objet.
- **ContentLength** : Longueur des données en octets.
- **AccessControlList** : Une liste de contrôle d'accès pour la ressource. Cet élément est facultatif. En cas d'omission, le demandeur a FULL_CONTROL accès à l'objet. Si l'objet existe déjà, la politique de contrôle d'accès préexistante est remplacée.

Réponses

- **ETag** : La balise d'entité est un hachage MD5 de l'objet que vous pouvez utiliser pour effectuer des extractions conditionnelles de l'objet. GetObjectExtended L'ETag reflète uniquement les modifications apportées au contenu d'un objet, pas à ses métadonnées.
- **LastModified** : L'horodatage Amazon S3 de l'objet enregistré.

Contrôle d'accès

Pour placer des objets dans un seau, vous devez avoir WRITE accès au seau.

Ressources connexes

- [CopyObject \(API SOAP\)](#)

CopyObject (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

Description

L'opération `CopyObject` crée une copie d'un objet lorsque vous spécifiez la clé et le compartiment d'un objet source et la clé et le compartiment d'une destination cible.

Lorsque vous copiez un objet, vous pouvez conserver toutes les métadonnées (par défaut) ou spécifier de nouvelles métadonnées. Cependant, l'ACL n'est pas conservée et est configurée `private` pour l'utilisateur qui fait la demande. Pour modifier le paramètre ACL par défaut, spécifiez une nouvelle ACL lors de la génération d'une demande de copie. Pour de plus amples informations, veuillez consulter [Utilisation des listes ACL](#).

Toutes les demandes de copie doivent être authentifiées. En outre, vous devez disposer d'un accès en lecture à l'objet source et d'un accès en écriture au compartiment de destination. Pour plus d'informations, consultez la section [Utilisation de Auth Access](#).

Pour copier un objet uniquement sous certaines conditions, par exemple si l'Etag correspond ou si l'objet a été modifié avant ou après une date spécifiée, utilisez les paramètres de requête `CopySourceIfUnmodifiedSince`, `CopyIfUnmodifiedSince`, `CopySourceIfMatch`, ou `CopySourceIfNoneMatch`.

Note

Vous devrez peut-être configurer le délai d'expiration du socket de la pile SOAP pour copier des objets volumineux.

Syntaxe de la requête

```
<CopyObject xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <SourceBucket>source_bucket</SourceBucket>
```

```
<SourceObject>source_object</SourceObject>
<DestinationBucket>destination_bucket</DestinationBucket>
<DestinationObject>destination_object</DestinationObject>
<MetadataDirective>{REPLACE | COPY}</MetadataDirective>
<Metadata>
  <Name>metadata_name</Name>
  <Value>metadata_value</Value>
</Metadata>
...
<AccessControlList>
  <Grant>
    <Grantee xsi:type="user_type">
      <ID>user_id</ID>
      <DisplayName>display_name</DisplayName>
    </Grantee>
    <Permission>permission</Permission>
  </Grant>
  ...
</AccessControlList>
<CopySourceIfMatch>etag</CopySourceIfMatch>
<CopySourceIfNoneMatch>etag</CopySourceIfNoneMatch>
<CopySourceIfModifiedSince>date_time</CopySourceIfModifiedSince>
<CopySourceIfUnmodifiedSince>date_time</CopySourceIfUnmodifiedSince>
<AWSAccessKeyId>AWSAccessKeyId</AWSAccessKeyId>
<Timestamp>TimeStamp</Timestamp>
<Signature>Signature</Signature>
</CopyObject>
```

Paramètres de demande

Nom	Description	Obligatoire
SourceBucket	Le nom du compartiment source.	Oui
	Type : Chaîne	
	Par défaut : aucun	
	Contraintes : compartiment source valide.	
SourceKey	Le nom clé de l'objet source.	Oui
	Type : Chaîne	

Nom	Description	Obligatoire
	Par défaut : aucun Contraintes : clé d'un objet source valide auquel vous avez un accès READ.	
DestinationBucket	Nom du compartiment de destination. Type : Chaîne Par défaut : aucun Contraintes : vous devez disposer d'un accès WRITE au compartiment de destination.	Oui
DestinationKey	La clé de l'objet de destination. Type : Chaîne Par défaut : aucun Contraintes : vous devez disposer d'un accès WRITE au compartiment de destination.	Oui

Nom	Description	Obligatoire
MetadataDirective	Spécifie si les métadonnées sont copiées depuis l'objet source ou remplacées par les métadonnées fournies dans la demande. Type : Chaîne Par défaut : COPIER Valeurs valides : COPIER REMPLACER Contraintes : les valeurs autres que COPY ou REPLACE qui entraîneront une erreur immédiate. Vous ne pouvez pas copier un objet sur lui-même à moins que l' MetadataDirective en-tête ne soit spécifié et que sa valeur ne soit définie surREPLACE.	Non
Metadata	Spécifie les paires nom-valeur des métadonnées à définir pour l'objet. Si cette valeur MetadataDirective est définie surCOPY, toutes les métadonnées sont ignorées. Type : Chaîne Par défaut : aucun Contraintes : Aucune.	Non

Nom	Description	Obligatoire
AccessControlList	Accorde l'accès aux utilisateurs par adresse e-mail ou identifiant utilisateur canonique. Type : Chaîne Par défaut : aucun Contraintes: Aucune	Non
CopySourceIfMatch	Copie l'objet si sa balise d'entité (ETag) correspond à la balise spécifiée ; sinon, renvoie un PreconditionFailed. Type : Chaîne Par défaut : aucun Contraintes : Aucune. Si l'Etag ne correspond pas, l'objet n'est pas copié.	Non
CopySourceIfNoneMatch	Copie l'objet si sa balise d'entité (ETag) est différente de l'Etag spécifié ; sinon, renvoie une erreur. Type : Chaîne Par défaut : aucun Contraintes : Aucune.	Non
CopySourceIfUnmodifiedSince	Copie l'objet s'il n'a pas été modifié depuis l'heure spécifiée ; sinon, renvoie un PreconditionFailed. Type : DateTime Par défaut : aucun	Non

Nom	Description	Obligatoire
CopySourceIfModifiedSince	Copie l'objet s'il a été modifié depuis l'heure spécifiée ; sinon, renvoie une erreur. Type : DateTime Par défaut : aucun	Non

Syntaxe de la réponse

```
<CopyObjectResponse xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"etag"</ETag>
    <LastModified>timestamp</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Éléments de réponse

Voici une liste des éléments de réponse.

Note

L'API SOAP ne renvoie pas d'espace blanc supplémentaire. Les espaces supplémentaires ne sont renvoyés que par l'API REST.

Nom	Description
Etag	Renvoie l'etag du nouvel objet. L'ETag reflète uniquement les modifications apportées au contenu d'un objet, pas à ses métadonnées. Type : Chaîne Ancêtre : CopyObjectResult

Nom	Description
LastModified	Renvoie la date à laquelle l'objet a été modifié pour la dernière fois. Type : Chaîne Ancêtre : CopyObjectResult

Pour plus d'informations sur les éléments de réponse généraux, consultez la section [Utilisation des en-têtes](#) de réponse aux erreurs REST.

Erreurs spéciales

Il n'y a pas d'erreur particulière pour cette opération. Pour plus d'informations sur les erreurs générales d'Amazon S3, consultez [Liste des codes d'erreur](#).

Exemples

L'exemple copie l'objet flotsam du compartiment pacific vers l'objet jetsam du compartiment atlantic, en préservant les métadonnées.

Exemple de demande

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>pacific</SourceBucket>
  <SourceObject>flotsam</SourceObject>
  <DestinationBucket>atlantic</DestinationBucket>
  <DestinationObject>jetsam</DestinationObject>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```

Exemple de réponse

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
```

```
</CopyObjectResponse>
</CopyObjectResponse>
```

Cet exemple copie l'objet « tweedledee » du compartiment Wonderland vers l'objet « tweedledum » du bucket Wonderland, en remplaçant ses métadonnées.

Exemple de demande

```
<CopyObject xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <SourceBucket>wonderland</SourceBucket>
  <SourceObject>tweedledee</SourceObject>
  <DestinationBucket>wonderland</DestinationBucket>
  <DestinationObject>tweedledum</DestinationObject>
  <MetadataDirective>REPLACE</MetadataDirective>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>relationship</Name>
    <Value>twins</Value>
  </Metadata>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2008-02-18T13:54:10.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbq7RrtSFmw=</Signature>
</CopyObject>
```

Exemple de réponse

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Ressources connexes

- [PutObject \(API SOAP\)](#)
- [PutObjectInline \(API SOAP\)](#)

GetObject (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'GetObject opération renvoie la version actuelle d'un objet. Si vous essayez d'GetObject utiliser un objet dont la version actuelle comporte un marqueur de suppression, S3 renvoie une erreur 404. Vous ne pouvez pas utiliser l'API SOAP pour récupérer une version spécifiée d'un objet. Pour ce faire, utilisez l'API REST. Pour plus d'informations, consultez la section [Gestion des versions](#). Pour plus d'options, utilisez l'[GetObjectExtended \(API SOAP\)](#) opération.

Note

Les noms de clé d'objet contenant la valeur « soap » ne sont pas pris en charge pour les [demandes de type hébergement virtuel](#). Pour les noms de clé d'objet contenant la valeur « soap », une [URL de type chemin d'accès](#) doit être utilisée à la place.

Exemple

Cet exemple extrait l'objet « Nelson » du compartiment « quotes ».

Sample Request

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Sample Response

```
<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>
      <Name>family</Name>
      <Value>Muntz</Value>
    </Metadata>
    <Data>aGEtaGE=</Data>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

Éléments

- **Bucket** : Le seau dans lequel récupérer l'objet.
- **Key** : La clé qui identifie l'objet.
- **GetMetadata** : Les métadonnées sont renvoyées avec l'objet si cela est vrai.
- **GetData** : Les données de l'objet sont renvoyées si cela est vrai.
- **InlineData** : Si cela est vrai, les données sont renvoyées, codées en base 64, dans le corps SOAP de la réponse. Si la valeur est fausse, les données sont renvoyées sous forme de pièce jointe SOAP. L' **InlineData** option n'est pas adaptée à une utilisation avec de gros objets. Le système limite cette opération à l'utilisation de 1 Mo de données ou moins. Une **GetObject** demande dont l' **InlineData** indicateur est défini échouera avec le code d'**InlineDataTooLargeError** état si le paramètre **Data** résultant aurait encodé plus de 1 Mo. Pour télécharger des objets volumineux, pensez à appeler **GetObject** sans définir d' **InlineData** indicateur, ou utilisez plutôt l'API REST.

Éléments retournés

- **Metadata** : Métadonnées associées nom-valeur stockées avec l'objet.

- **Data** : Si la requête `InlineData` était vraie, cela contient les données d'objet codées en base 64.
- **LastModified** : Heure à laquelle l'objet a été stocké dans Amazon S3.
- **ETag** : L'étiquette d'entité de l'objet. Il s'agit d'un hachage de l'objet qui peut être utilisé pour effectuer des requêtes conditionnelles. L'`ETag` reflète uniquement les modifications apportées au contenu d'un objet, pas à ses métadonnées.

Contrôle d'accès

Vous ne pouvez lire un objet que si vous avez obtenu `READ` l'autorisation d'y accéder.

Téléchargements tronqués et reprenant le format SOAP

Pour `GET` plus de flexibilité, Amazon S3 prend en charge les téléchargements fragmentés et pouvant être repris.

Sélectionnez l'une des options suivantes :

- Pour les téléchargements d'objets volumineux, vous souhaitez peut-être les diviser en petits morceaux. Pour de plus amples informations, consultez [Gamme GETS](#).
- Pour les `GET` opérations qui échouent, vous pouvez concevoir votre application de manière à télécharger le reste du fichier au lieu de télécharger l'intégralité du fichier. Pour de plus amples informations, consultez [Récupération d'erreurs REST GET](#).

Gamme GETS

Pour certains clients, vous souhaitez peut-être diviser les gros téléchargements en téléchargements plus petits. Pour diviser un `GET` en unités plus petites, utilisez `Range`.

Avant de pouvoir diviser un `GET` en unités plus petites, vous devez déterminer sa taille. Par exemple, la requête suivante permet d'obtenir la taille de l'objet `bigfile`.

```
<ListBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Prefix>bigfile</Prefix>
  <MaxKeys>1</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Amazon S3 renvoie la réponse suivante.

```
<ListBucketResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Name>quotes</Name>
  <Prefix>N</Prefix>
  <MaxKeys>1</MaxKeys>
  <IsTruncated>false</IsTruncated>
  <Contents>
    <Key>bigfile</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <Size>2023276</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>bcaf1ffd86f41161ca5fb16fd081034f</ID>
      <DisplayName>bigfile</DisplayName>
    </Owner>
  </Contents>
</ListBucketResult>
```

Voici une requête qui télécharge le premier mégaoctet depuis l'objet bigfile.

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>0</ByteRangeStart>
  <ByteRangeEnd>1048576</ByteRangeEnd>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Amazon S3 renvoie le premier mégaoctet du fichier et l'Etag du fichier.

```
<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
```



```

</Status>
<Metadata>
  <Name>Content-Type</Name>
  <Value>text/plain</Value>
</Metadata>
<Metadata>
  <Name>family</Name>
  <Value>Muntz</Value>
</Metadata>
<Data>--first megabyte of bigfile--</Data>
<LastModified>2006-01-01T12:00:00.000Z</LastModified>
<ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
</GetObjectResponse>
</GetObjectResponse>

```

Pour vous assurer que le fichier n'a pas été modifié depuis le téléchargement de la partie précédente, spécifiez l' `IfMatch` élément. Bien que l' `IfMatch` élément ne soit pas obligatoire, il est recommandé pour le contenu susceptible de changer.

La requête suivante permet d'obtenir le reste du fichier à l'aide de l'en-tête de la `IfMatch` demande.

```

<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>10485761</ByteRangeStart>
  <ByteRangeEnd>2023276</ByteRangeEnd>
  <IfMatch>"828ef3fdfa96f00ad9f27c383fc9ac7f"</IfMatch>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>

```

Amazon S3 renvoie la réponse suivante et le reste du fichier.

```

<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>

```

```

</Status>
<Metadata>
  <Name>Content-Type</Name>
  <Value>text/plain</Value>
</Metadata>
<Metadata>
  <Name>family</Name>
  <Value>>Muntz</Value>
</Metadata>
<Data>--remainder of bigfile--</Data>
<LastModified>2006-01-01T12:00:00.000Z</LastModified>
<ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
</GetObjectResponse>
</GetObjectResponse>

```

Versionné GetObject

La requête suivante renvoie la version spécifiée de l'objet dans le compartiment.

```

<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>

```

Exemple de réponse

```

<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
      <Description>OK</Description>
    </Status>
    <Metadata>
      <Name>Content-Type</Name>
      <Value>text/plain</Value>
    </Metadata>
    <Metadata>

```

```
<Name>family</Name>
<Value>Muntz</Value>
</Metadata>
<Data>aGEtaGE=</Data>
<LastModified>2006-01-01T12:00:00.000Z</LastModified>
<ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
</GetObjectResponse>
</GetObjectResponse>
```

Récupération d'erreurs REST GET

Si un objet GET échoue, vous pouvez obtenir le reste du fichier en spécifiant la plage à télécharger. Pour ce faire, vous devez obtenir la taille de l'objet en utilisant `ListBucket` et effectuer une plage GET sur le reste du fichier. Pour de plus amples informations, veuillez consulter [GetObjectExtended \(API SOAP\)](#).

Ressources connexes

[Opérations sur les objets \(API SOAP\)](#)

GetObjectExtended (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

`GetObjectExtended` est exactement pareil [GetObject \(API SOAP\)](#), sauf qu'il prend en charge les éléments supplémentaires suivants qui peuvent être utilisés pour exécuter en grande partie les mêmes fonctionnalités que celles fournies par les en-têtes HTTP GET (allez à <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>).

`GetObjectExtended` prend en charge les éléments suivants en plus de ceux pris en charge par `GetObject` :

- `ByteRangeStart`, `ByteRangeEnd` : Ces éléments indiquent que seule une partie des données de l'objet doit être récupérée. Ils suivent le comportement des plages d'octets HTTP (rendez-vous sur [http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html #sec14.35](http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35)).

- **IfModifiedSince** : Renvoie l'objet uniquement si l'horodatage de l'objet est postérieur à l'horodatage spécifié. (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.25>)
- **IfUnmodifiedSince** : Renvoie l'objet uniquement si l'horodatage de l'objet est antérieur ou égal à l'horodatage spécifié. (allez à <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.28>)
- **IfMatch** : Renvoie l'objet uniquement si son ETag correspond à la ou aux balises fournies. (allez à <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24>)
- **IfNoneMatch** : Renvoie l'objet uniquement si son ETag ne correspond pas à la ou aux balises fournies. (allez à <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26>)
- **ReturnCompleteObjectOnConditionFailure** : **ReturnCompleteObjectOnConditionFailure** : Si c'est vrai, alors si la demande inclut un élément de plage et l'un des **IfUnmodifiedSince**/**IfMatch** éléments ou les deux, et que la condition échoue, renvoie l'objet entier plutôt qu'une erreur. Cela active la **If-Range** fonctionnalité (rendez-vous sur <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27>).

DeleteObject (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'**DeleteObject** opération supprime l'objet spécifié d'Amazon S3. Une fois supprimé, il n'existe aucune méthode pour restaurer ou annuler la suppression d'un objet.

Note

Si vous supprimez un objet qui n'existe pas, Amazon S3 renverra un message de réussite (pas un message d'erreur).

Exemple

Cet exemple supprime l'objet « Nelson » du compartiment « quotes ».

Sample Request

```
<DeleteObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId> AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</DeleteObject>
```

Sample Response

```
<DeleteObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <DeleteObjectResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </DeleteObjectResponse>
</DeleteObjectResponse>
```

Éléments

- **Bucket** : Le seau qui contient l'objet.
- **Key** : La clé qui identifie l'objet.

Contrôle d'accès

Vous ne pouvez supprimer un objet que si vous avez **WRITE** accès au compartiment, quels que soient le propriétaire de l'objet ou les droits qui lui sont accordés.

GetObjectAccessControlPolicy (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'GetObjectAccessControlPolicy opération récupère la politique de contrôle d'accès pour un objet.

Exemple

Cet exemple extrait la politique de contrôle d'accès pour l'objet « Nelson » à partir du compartiment « quotes ».

Sample Request

```
<GetObjectAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObjectAccessControlPolicy>
```

Sample Response

```
<AccessControlPolicy>
  <Owner>
    <ID>a9a7b886d6fd24a541bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Owner>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b841bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
</AccessControlPolicy>
```

Corps de la réponse

La réponse contient la politique de contrôle d'accès pour le bucket. Pour une explication de cette réponse, Politique d'accès [SOAP](#).

Contrôle d'accès

Vous devez disposer de READ_ACP droits sur l'objet afin de récupérer la politique de contrôle d'accès pour un objet.

SetObjectAccessControlPolicy (API SOAP)

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

L'SetObjectAccessControlPolicy opération définit la politique de contrôle d'accès pour un objet existant. En cas de succès, la politique de contrôle d'accès précédente pour l'objet est entièrement remplacée par la politique de contrôle d'accès spécifiée.

Exemple

Cet exemple permet à l'utilisateur spécifié (généralement le propriétaire) FULL_CONTROL d'accéder à l'objet « Nelson » depuis le compartiment « quotes ».

Sample Request

```
<SetObjectAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</SetObjectAccessControlPolicy>
```

Sample Response

```
<SetObjectAccessControlPolicyResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <SetObjectAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </SetObjectAccessControlPolicyResponse>
</SetObjectAccessControlPolicyResponse>
```

Clé

Contrôle d'accès

Vous devez disposer de WRITE_ACP droits sur l'objet afin de définir la politique de contrôle d'accès pour un bucket.

Authentification des demandes SOAP

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les AWS SDK.

Chaque demande non anonyme doit contenir des informations d'authentification pour établir l'identité de la personne habilitée créant la demande. Dans SOAP, les informations d'authentification sont placées dans les éléments suivants de la demande SOAP :

- ID de votre clé d' AWS accès

Note

Lorsque des demandes SOAP authentifiées sont créées, les informations d'identification temporaires ne sont pas prises en charge. Pour obtenir plus d'informations sur les types d'informations d'identification, consultez [Demandes](#).

- **Timestamp** : Il doit s'agir d'une DateTime (passez à <http://www.w3.org/TR/xmlschema-2/#dateTime>) dans le fuseau horaire du temps universel coordonné (heure moyenne de Greenwich),

par exemple. 2009-01-01T12:00:00.000Z L'autorisation échouera si cet horodatage a plus de 15 minutes d'écart avec l'horloge sur les serveurs Amazon S3.

- **Signature :** Le HMAC-SHA1 résumé de la RFC 2104 (allez à <http://www.ietf.org/rfc/rfc2104.txt>) de la concaténation de « AMAZONS3 » + OPERATION + Timestamp, en utilisant votre clé d'accès secrète comme clé. AWS Par exemple, dans l' CreateBucket exemple de demande suivant, l'élément de signature contiendrait le HMAC-SHA1 résumé de la valeur AmazonS3CreateBucket2009-01-01T12:00:00.000Z « » :

Par exemple, dans l' CreateBucket exemple de demande suivant, l'élément de signature contiendrait le HMAC-SHA1 résumé de la valeur AmazonS3CreateBucket2009-01-01T12:00:00.000Z « » :

Exemple

```
<CreateBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Acl>private</Acl>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2009-01-01T12:00:00.000Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</CreateBucket>
```

Note

Les demandes SOAP, aussi bien authentifiées qu'anonymes, doivent être envoyées à Amazon S3 à l'aide de SSL. Amazon S3 renvoie un message d'erreur lorsque vous envoyez une demande SOAP via HTTP.

Important

En raison des diverses propositions existant sur la façon de renoncer à une précision horaire supplémentaire, les utilisateurs de .NET doivent faire attention à ne pas envoyer des horodatages Amazon S3 trop spécifiques. Pour y parvenir, il est possible de créer manuellement des objets `DateTime` avec une précision limitée aux millisecondes.

Configuration d'une stratégie d'accès avec SOAP

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les kits de AWS développement logiciel (SDK).

Le contrôle d'accès peut être défini au moment où un bucket ou un objet est écrit en incluant l'élément `AccessControlList` « » dans la demande adressée à `CreateBucketPutObjectInline`, ou `PutObject`. L' `AccessControlList` élément est décrit dans la section Gestion des [identités et des accès pour Amazon S3](#). Si aucune liste de contrôle d'accès n'est spécifiée avec ces opérations, la ressource est créée avec une politique d'accès par défaut qui donne au demandeur un accès `FULL_CONTROL` (c'est le cas même s'il s'agit d'une demande `PutObjectInline` ou d'une `PutObject` demande pour un objet qui existe déjà).

Vous trouverez ci-dessous une demande qui écrit des données dans un objet, rend l'objet lisible par des mandataires anonymes et donne à l'utilisateur spécifié les droits `FULL_CONTROL` sur le compartiment (la plupart des développeurs voudront s'accorder un accès `FULL_CONTROL` à leur propre compartiment).

Exemple

Vous trouverez ci-dessous une demande qui écrit des données dans un objet et rend l'objet lisible par des mandataires anonymes.

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
```

```

    <Grantee xsi:type="CanonicalUser">
      <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
      <DisplayName>chriscustomer</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
    <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
    </Grantee>
    <Permission>READ</Permission>
  </Grant>
</AccessControlList>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2009-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>

```

Sample Response

```

<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>

```

La stratégie de contrôle d'accès peut être lue ou spécifiée pour un compartiment ou un objet existant à l'aide des méthodes `GetBucketAccessControlPolicy`, `GetObjectAccessControlPolicy`, `SetBucketAccessControlPolicy` et `SetObjectAccessControlPolicy`. Pour de plus amples informations, veuillez consulter la description détaillée de ces méthodes.

Éléments communs

Vous pouvez inclure les éléments suivants relatifs à l'autorisation dans une demande SOAP :

- **AWSAccessKeyId** : L'ID de clé d'AWS accès du demandeur
- **Timestamp** : heure actuelle sur votre système
- **Signature** : signature de la demande

Réponses aux erreurs SOAP

 Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les kits de AWS développement logiciel (SDK).

Dans SOAP, un résultat d'erreur est renvoyé au client comme une erreur SOAP avec le code réponse HTTP 500. Si vous ne recevez pas une erreur SOAP, alors votre demande a abouti. Le code d'erreur SOAP Amazon S3 est composé d'un code d'erreur SOAP 1.1 standard (« Serveur » ou « Client ») concaténé avec le code d'erreur Amazon S3-specific . Par exemple : «Server.InternalError" ou "Client.NoSuchBucket». L'élément de chaîne d'erreur SOAP contient un message d'erreur générique directement lisible en anglais. Finalement, l'élément détaillé de l'erreur SOAP contient différentes informations concernant l'erreur.

Par exemple, si vous tentez de supprimer l'objet « Fred », qui n'existe pas, le corps de la réponse SOAP contient un « NoSuchKey » défaut SOAP.

L'exemple suivant montre un exemple de réponse d'erreur SOAP.

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
    <Faultstring>The specified key does not exist.</Faultstring>
    <Detail>
      <Key>Fred</Key>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

Le tableau suivant explique les éléments de réponse aux erreurs SOAP

Nom	Description
Detail	Conteneur pour la clé impliquée dans l'erreur
	Type : Conteneur

Nom	Description
	Ancêtre : Body.Fault
Fault	Conteneur contenant les informations relatives aux erreurs. Type : Conteneur Ancêtre : Body
Faultcode	Le code d'erreur est une chaîne qui identifie de manière unique une condition d'erreur. Il doit être lu et compris par des programmes qui détectent et gèrent des erreurs par type. Pour plus d'informations, consultez la section Liste des codes d'erreur. Type : Chaîne Ancêtre : Body.Fault
Faultstring	La chaîne d'erreur contient une description générique de la condition d'erreur en anglais. Il s'adresse à un public d'individus. Des programmes basiques affichent le message directement pour l'utilisateur final s'ils trouvent une condition d'erreur qu'ils ne savent, ni souhaitent gérer. Les programmes sophistiqués dotés d'une gestion des erreurs plus exhaustive et d'une internationalisation appropriée sont plus susceptibles d'ignorer la chaîne d'erreur. Type : Chaîne Ancêtre : Body.Fault
Key	Identifie la clé impliquée dans l'erreur Type : Chaîne Ancêtre : Body.Fault

Annexe : Demandes d'authentification (AWS signature (version 2))

Important

Cette section décrit comment authentifier les demandes à l'aide de AWS Signature Version 2. Signature Version 2 est en cours de désactivation (obsolète), Amazon S3 acceptera uniquement les requêtes d'API signées avec Signature Version 4. Pour de plus amples informations, consultez [AWS Signature Version 2 désactivée \(obsolète\) pour Amazon S3](#). La version 4 de Signature est prise en charge dans tous les cas Régions AWS, et c'est la seule version prise en charge pour les nouvelles régions. Pour plus d'informations, consultez la section Demandes [d'authentification \(AWS Signature version 4\)](#) dans le manuel Amazon Simple Storage Service API Reference.

Amazon S3 vous permet d'identifier la version de signature d'API qui a été utilisée pour signer une demande. Il est important de déterminer si l'un de vos workflows utilise Signature Version 2 et de procéder à une mise à niveau vers Signature Version 4 afin d'éviter tout impact sur votre entreprise.

- Si vous utilisez des journaux d' CloudTrail événements (option recommandée), consultez la section [Identifier les demandes Amazon S3 Signature Version 2 CloudTrail](#) en utilisant comment interroger et identifier ces demandes.
- Si vous utilisez les journaux d'accès au serveur Amazon S3, voir [Utilisation des journaux d'accès au serveur Amazon S3 pour identifier les demandes](#)

Rubriques

- [Authentification des demandes à l'aide de l'API REST \(AWS signature \(version 2\)\)](#)
- [Signature et authentification des requêtes REST \(AWS signature \(version 2\)\)](#)
- [Browser-based téléchargements à l'aide de POST \(AWS signature \(version 2\)\)](#)

Authentification des demandes à l'aide de l'API REST (AWS signature (version 2))

Lorsque vous accédez à Amazon S3 grâce à REST, vous devez fournir les éléments suivants dans la demande afin qu'elle puisse être authentifiée :

Éléments d'une demande

- **AWS ID de clé d'accès** — Chaque demande doit contenir l'ID de clé d'accès de l'identité que vous utilisez pour envoyer votre demande.
- **Signature** – Chaque demande doit contenir une signature de demande valide, sinon, la demande est rejetée.

Une signature de demande est définie grâce à la clé d'accès secrète, clé qui est connue uniquement par vous et AWS.

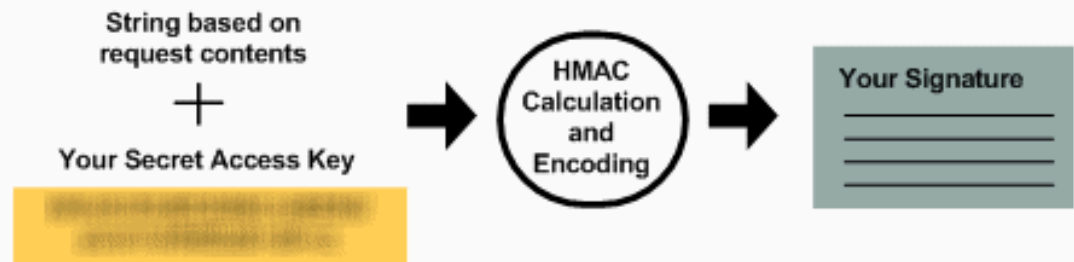
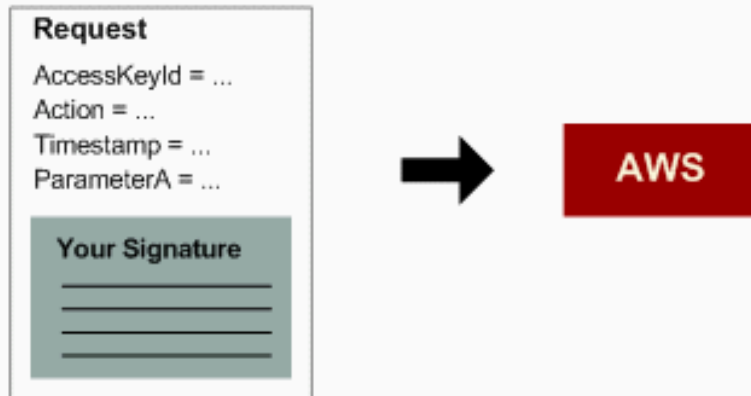
- **Horodatage** – Chaque demande doit contenir la date et l'heure de création de la demande sous la forme d'une chaîne de caractères en UTC.
- **Date** – Toute demande doit contenir son horodatage.

Selon l'action de l'API que vous utilisez, vous pouvez fournir une date et une heure d'expiration de la demande au lieu ou en plus de l'horodatage. Consultez la rubrique sur l'authentification concernant l'action particulière pour déterminer ce qu'elle nécessite.

Voici les étapes générales pour authentifier des demandes sur Amazon S3. Nous supposons que vous disposez des informations d'identification de sécurité nécessaires, de l'ID de clé d'accès et de la clé d'accès secrète.

You**1 Create a request:****Request**

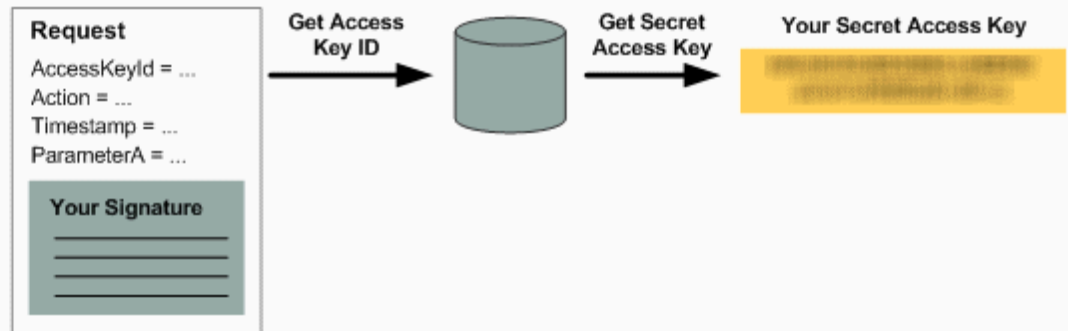
AccessKeyId = ...
Action = ...
Timestamp = ...
ParameterA = ...

2 Create an HMAC-SHA1 signature:**3 Send the request and signature to AWS:**

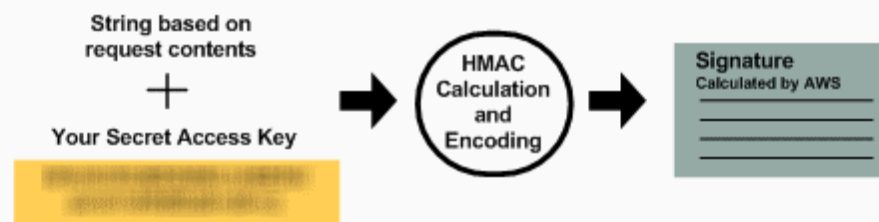
- 1 Créez une demande pour AWS.
- 2 Calculez la signature grâce à la clé d'accès secrète.
- 3 Envoyez la demande à Amazon S3. Incluez l'ID de clé d'accès et la signature dans la demande. Amazon S3 réalise les trois prochaines étapes.

AWS

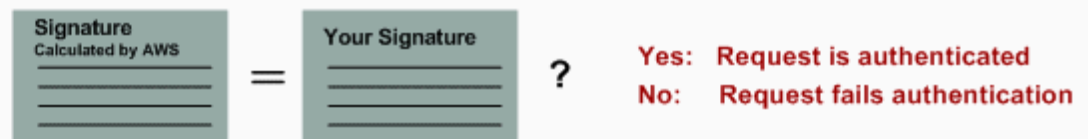
4 Retrieve your Secret Access Key:



5 Create an HMAC-SHA1 signature:



6 Compare the two signatures:



4 Amazon S3 utilise l'ID de clé d'accès pour rechercher la clé d'accès secrète.

5 Amazon S3 calcule une signature à partir des données de la demande et de la clé d'accès secrète grâce au même algorithme que vous avez utilisé pour calculer la signature envoyée dans la demande.

6 Si la signature générée par Amazon S3 correspond à celle figurant dans la demande que vous avez envoyée, la demande est considérée comme authentique. Si la comparaison échoue, la demande est rejetée, et Amazon S3 renvoie une réponse d'erreur.

Informations d'authentification détaillées

Pour obtenir des informations détaillées sur l'authentification REST, consultez [Signature et authentification des requêtes REST \(AWS signature \(version 2\)\)](#).

Signature et authentification des requêtes REST (AWS signature (version 2))

Rubriques

- [Utilisation d'informations d'identification de sécurité temporaires](#)
- [L'en-tête Authentification](#)
- [Conversion sous forme canonique d'une demande pour signature](#)
- [Construction de l' CanonicalizedResource élément](#)
- [Construction de l' CanonicalizedAmzHeaders élément](#)
- [Éléments d'en-tête HTTP positionnels et éléments d'en-tête StringToSign HTTP nommés](#)
- [Exigence d'horodatage](#)
- [Exemples d'authentification](#)
- [Problèmes de signature de la demande REST](#)
- [Alternative à l'authentification d'une demande par chaîne d'interrogation](#)

Note

Cette rubrique explique les demandes d'authentification à l'aide de la Signature Version 2. Amazon S3 prend désormais en charge le dernier processus Signature Version 4. Ce protocole récent est pris en charge dans toutes les Régions et chaque nouvelle Région après le 30 janvier 2014 prendra uniquement en charge Signature Version 4. Pour plus d'informations, consultez la rubrique Demandes [d'authentification \(AWS Signature version 4\)](#) dans le manuel Amazon Simple Storage Service API Reference.

L'authentification est le processus qui consiste à prouver l'identité au système. L'identité est un facteur important dans les décisions liées au contrôle d'accès Amazon S3. Les demandes sont autorisées ou refusées en partie en fonction l'identité du demandeur. Par exemple, le droit de créer des compartiments est réservé aux développeurs inscrits et (par défaut) le droit de créer des objets dans un compartiment est réservé au propriétaire du compartiment en question. En tant que développeur, vous faites des demandes qui appellent ces privilèges, vous devez donc prouver

l'identité au système grâce à l'authentification des demandes. Cette section vous montre comment le faire.

Note

Le contenu de cette section ne s'applique pas à HTTP POST. Pour de plus amples informations, veuillez consulter [Browser-based téléchargements à l'aide de POST \(AWS signature \(version 2\)\)](#).

L'API REST Amazon S3 utilise un schéma HTTP personnalisé basé sur un HMAC (Hash Message Authentication Code) à clés pour l'authentification. Pour authentifier une demande, vous devez d'abord concaténer les éléments sélectionnés de la demande pour former une chaîne. Vous utilisez ensuite votre clé d'accès AWS secrète pour calculer le HMAC de cette chaîne. De manière informelle, ce processus est appelé « signature de la demande » et le résultat de l'algorithme HMAC est la signature, car il simule les propriétés de sécurité d'une signature réelle. Enfin, vous ajoutez cette signature comme un paramètre de la demande grâce à la syntaxe décrite dans cette section.

Lorsque le système reçoit une demande authentifiée, il récupère la clé d'accès AWS secrète que vous prétendez détenir et l'utilise de la même manière pour calculer la signature du message qu'il a reçu. Il compare ensuite la signature calculée et la signature présentée par le demandeur. Si les deux signatures correspondent, le système conclut que le demandeur doit avoir accès à la clé d'accès AWS secrète et agit donc avec l'autorité du mandant à qui la clé a été délivrée. Si les deux signatures ne correspondent pas, la demande est abandonnée et le système répond par un message d'erreur.

Exemple Demande REST Amazon S3 authentifiée

```
GET /photos/puppy.jpg HTTP/1.1
Host: awsexamplebucket1.us-west-1.s3.amazonaws.com
Date: Tue, 27 Mar 2007 19:36:42 +0000
```

```
Authorization: AWS AKIAIOSFODNN7EXAMPLE:
qgk2+6Sv9/oM7G3qLEjTH1a1l1g=
```

Utilisation d'informations d'identification de sécurité temporaires

Si vous signez la demande grâce à des informations d'identification de sécurité temporaires (consultez [Demandes](#)), vous devez inclure le token de sécurité correspondant dans la demande et ajouter l'en-tête `x-amz-security-token`.

Lorsque vous obtenez des informations d'identification de sécurité temporaires grâce à l'API AWS Security Token Service , la réponse inclut des informations d'identification de sécurité temporaires et un token de session. Vous indiquez la valeur du jeton de session dans l'en-tête `x-amz-security-token` lorsque vous envoyez des demandes à Amazon S3. Pour plus d'informations sur l' AWS Security Token Service API fournie par IAM, consultez la rubrique [Action](#) dans le Guide de référence des AWS Security Token Service API.

L'en-tête Authentification

L'API REST Amazon S3 utilise l'en-tête `Authorization` HTTP standard pour transmettre les informations d'authentification. (Le nom de l'en-tête standard est maladroit car il comporte des informations d'authentification, pas d'autorisation.) Dans le schéma d'authentification Amazon S3, l'en-tête `Autorisation` possède la forme suivante :

```
Authorization: AWS AWSAccessKeyId:Signature
```

Les développeurs reçoivent un identifiant de clé d' AWS accès et une clé d'accès AWS secrète lors de leur inscription. Pour une authentification de demande, l'élément `AWSAccessKeyId` identifie l'ID de clé d'accès utilisé pour calculer la signature et, indirectement, le développeur à l'origine de la demande.

L'`Signature`élément est la RFC 2104 HMAC-SHA1 des éléments sélectionnés de la demande. La `Signature` partie de l'en-tête `Authorization` varie donc d'une demande à l'autre. Si la signature de la demande calculée par le système correspond à `Signature` celle incluse dans la demande, le demandeur aura démontré qu'il possédait la clé d'accès AWS secrète. La demande sera ensuite traitée sous l'identité, et avec l'autorité, du développeur pour qui la clé a été émise.

Voici une pseudo-grammaire qui illustre la construction de l'en-tête de la demande `Authorization`. (Dans l'exemple, `\n` signifie le point de code Unicode U+000A, communément appelé nouvelle ligne).

```
Authorization = "AWS" + " " + AWSAccessKeyId + ":" + Signature;

Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of(YourSecretAccessKey), UTF-8-Encoding-Of( StringToSign ) ) );

StringToSign = HTTP-Verb + "\n" +
  Content-MD5 + "\n" +
  Content-Type + "\n" +
  Date + "\n" +
```

```
CanonicalizedAmzHeaders +  
CanonicalizedResource;  
  
CanonicalizedResource = [ "/" + Bucket ] +  
<HTTP-Request-URI, from the protocol name up to the query string> +  
[ subresource, if present. For example "?acl", "?location", or "?logging"];  
  
CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1 est un algorithme défini par la [RFC 2104 Keyed-Hashing pour l'authentification des messages](#). L'algorithme utilise comme paramètres des chaînes encodées sur deux octets, une clé et un message. Pour l'authentification des demandes Amazon S3, utilisez votre clé d'accès AWS secrète (YourSecretAccessKey) comme clé et son UTF-8 encodage StringToSign comme message. La sortie de HMAC-SHA1 est également une chaîne d'octets, appelée condensé. Le paramètre de la demande Signature est construit en encodant en Base64 cette valeur de hachage.

Conversion sous forme canonique d'une demande pour signature

Rappelez-vous que lorsque le système reçoit une demande authentifiée, il compare la signature calculée de la demande et celle fournie dans la demande dans StringToSign. C'est pourquoi, vous devez calculer la signature grâce à la même méthode utilisée par Amazon S3. Processus qui consiste à présenter une demande sous une forme convenue pour la signature de la canonisation.

Construction de l' CanonicalizedResource élément

CanonicalizedResource représente la ressource Amazon S3 ciblée par la demande. Construisez-la pour une demande REST de la manière suivante :

1. Commencez par une chaîne de caractères vide ("").
2. Si la demande spécifie un compartiment qui utilise l'en-tête hôte HTTP (demande d'hébergement virtuel), ajoutez le nom du compartiment précédé d'un "/" (par exemple, "/nom du compartiment"). Pour les demandes de type chemin et les demandes qui ne font pas référence à un compartiment, ne faites rien. Pour plus d'informations sur les requêtes de type hébergement virtuel, consultez la section Hébergement [virtuel de buckets](#).

Pour une requête de type hébergé virtuel « », il s'CanonicalizedResource agit de https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg « /awsexamplebucket1".

Pour la requête de style path, https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg « », CanonicalizedResource c'est « ».

3. Ajoutez la partie chemin du HTTP non décodé, jusqu'à la chaîne de requête Request-URI, mais sans l'inclure.

Pour une requête de style hébergé virtuel `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg` « », il CanonicalizedResource s'agit de «`/awsexamplebucket1/photos/puppy.jpg`».

Pour une requête de type chemin, `https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg` « », CanonicalizedResource c'est «`/awsexamplebucket1/photos/puppy.jpg`». A ce stade, la CanonicalizedResource est identique pour la demande d'hébergement virtuel et de type chemin.

Pour une demande qui ne fait pas référence à un compartiment, comme [GET Service](#), ajoutez «`/`».

4. Si la demande fait référence à une sous-ressource, comme `?versioning`, `?location`, `?acl`, `?lifecycle`, `?versionid`, ou, ajoutez la sous-ressource, sa valeur le cas échéant et le point d'interrogation. Notez que dans le cas de plusieurs sous-ressources, celles-ci doivent être triées lexicographiquement par nom de sous-ressource et séparées par «`&`», par exemple `? ACL&ID de version =value`.

Les sous-ressources qui doivent être incluses lors de la création de l' CanonicalizedResource élément sont `acl`, `lifecycle`, `location`, `logging`, `notification`, `PartNumber`, `policy`, `RequestPayment`, `UploadId`, `uploadId`, `uploads`, `VersionId`, `VersionId`, `Versioning`, `versions` et `site web`.

Si la demande spécifie des paramètres de la chaîne d'interrogation qui ignorent les valeurs de l'en-tête de la réponse (veuillez consulter [GetObject](#)), ajoutez les paramètres de la chaîne d'interrogation et leurs valeurs. Lors de la signature, vous n'encodez pas ces valeurs ; toutefois, lorsque vous faites la demande, vous devez encoder les valeurs de ces paramètres. Les paramètres de la chaîne d'interrogation dans une demande GET incluent `response-content-type`, `response-content-language`, `response-expires`, `response-cache-control`, `response-content-disposition` et `response-content-encoding`.

Le paramètre de chaîne de delete requête doit être inclus lorsque vous créez la demande de suppression CanonicalizedResource pour une requête de suppression multi-objets.

Les éléments provenant du CanonicalizedResource HTTP Request-URI doivent être signés littéralement tels qu'ils apparaissent dans la requête HTTP, y compris les URL-Encoding méta-caractères.

`CanonicalizedResource` Cela peut être différent du HTTP Request-URI. En particulier, si votre demande utilise l'Host en-tête HTTP pour spécifier un compartiment, celui-ci n'apparaît pas dans le protocole HTTP Request-URI. Toutefois, la `CanonicalizedResource` continue d'inclure le compartiment. Les paramètres de chaîne de requête peuvent également apparaître dans le Request-URI mais ne sont pas inclus dans `CanonicalizedResource`. Pour plus d'informations, consultez la section Hébergement [virtuel de buckets](#).

Construction de l' `CanonicalizedAmzHeaders` élément

Pour créer la `CanonicalizedAmzHeaders` partie de `StringToSign`, sélectionnez tous les en-têtes de requêtes HTTP qui commencent par « x-amz- » (en utilisant une comparaison sans distinction majuscules/minuscules), puis procédez comme suit.

1. Convertissez chaque nom d'en-tête HTTP en minuscule. Par exemple, « X-Amz-Date » devient « x-amz-date ».
2. Triez la sélection d'en-tête de manière lexicographique par nom d'en-tête.
3. Combinez les champs d'en-tête avec le même nom dans une paire « nom d'en-tête:liste de valeurs séparées par des virgules » tel que recommandé par RFC 2616, section 4.2, sans aucun espace entre les valeurs. Par exemple, les deux en-têtes de métadonnées « x-amz-meta-username: fred » et « x-amz-meta-username: barney » seraient combinés en un seul en-tête « x-amz-meta-username: fred,barney ».
4. « Séparez » les longs en-têtes qui s'étendent sur plusieurs lignes (comme le permet RFC 2616, section 4.2) en remplaçant le retour à la ligne (notamment la nouvelle ligne) par un espace simple.
5. Coupez les espaces autour des deux-points dans l'en-tête. Par exemple, l'en-tête « x-amz-meta-username: fred,barney » deviendrait « x-amz-meta-username:fred,barney ».
6. Enfin, ajoutez un caractère de nouvelle ligne (U+000A) à chaque en-tête converti sous forme canonique dans la liste obtenue. Construisez l' `CanonicalizedResource` élément en concaténant tous les en-têtes de cette liste en une seule chaîne.

Éléments d'en-tête HTTP positionnels et éléments d'en-tête `StringToSign` HTTP nommés

Les premiers éléments d'en-tête de `StringToSign` (Content-Type, Date et Content-MD5) sont de nature positionnelle. `StringToSign` n'inclut pas les noms de ces en-têtes, mais uniquement leurs valeurs issues de la demande. En revanche, les éléments « x-amz- » sont nommés. Les noms et les valeurs de l'en-tête apparaissent dans `StringToSign`.

Si un en-tête positionnel demandé pour une définition de `StringToSign` n'est pas présent dans la demande (par exemple, les en-têtes `Content-Type` ou `Content-MD5` sont facultatifs pour les demandes `PUT` et insignifiants pour les demandes `GET`), substituez la chaîne de caractères vide ("") pour cette position.

Exigence d'horodatage

Un horodatage valide (qui utilise l'en-tête `Date` HTTP ou une alternative `x-amz-date`) est obligatoire pour les demandes authentifiées. De plus, l'horodatage du client inclus dans une demande authentifiée doit se faire dans les 15 minutes de l'heure du système Amazon S3 à la réception de la demande. Dans le cas contraire, la demande échoue avec le code d'erreur `RequestTimeTooSkewed`. L'objectif de ces restrictions est de limiter la possibilité que des demandes interceptées soient réutilisées par un adversaire. Pour une meilleure protection contre l'écoute, utilisez le transport HTTPS pour des demandes authentifiées.

Note

La contrainte de validation sur la date de la demande s'applique uniquement aux demandes authentifiées qui n'utilisent pas d'authentification par chaîne d'interrogation. Pour de plus amples informations, veuillez consulter [Alternative à l'authentification d'une demande par chaîne d'interrogation](#).

Certaines bibliothèques client HTTP n'offrent pas la possibilité de configurer l'en-tête `Date` pour une demande. Si vous avez des problèmes pour inclure la valeur de l'en-tête « `Date` » dans les en-têtes convertis sous forme canonique, vous pouvez à la place configurer l'horodatage pour la demande grâce à un en-tête « `x-amz-date` ». La valeur de l'en-tête `x-amz-date` doit être dans l'un des formats RFC 2616 (<http://www.ietf.org/rfc/rfc2616.txt>). Lorsqu'un en-tête `x-amz-date` est présent dans une demande, le système ignore tous les en-têtes `Date` lors du calcul de la signature de la demande. Par conséquent, si vous incluez l'en-tête `x-amz-date`, utilisez la chaîne de caractères vide pour la `Date` lors de la création de l'élément `StringToSign`. consultez suivante pour obtenir un exemple.

Exemples d'authentification

Les exemples de cette section utilisent les informations d'identification (qui ne fonctionnent pas) dans le tableau suivant.

Paramètre	Value
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

Dans l'exemple `StringToSign`, le formatage n'est pas significatif, et `\n` signifie le point de code Unicode U+000A, communément appelé nouvelle ligne. De plus, les exemples utilisent « +0000 » pour désigner le fuseau horaire. Vous pouvez utiliser « GMT » à la place pour désigner le fuseau horaire, mais les signatures illustrées dans les exemples sont différentes.

Objet GET

Cet exemple obtient un objet à partir du compartiment `awsexamplebucket1`.

Demande	StringToSign
<pre>GET /photos/puppy.jpg HTTP/1.1 Host: awsexamplebucket1.us- west-1.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:36:42 +0000 Authorization: AWS AKIAIOSFO DNN7EXAMPLE: qgk2+6Sv9/oM7G3qLEjTH1a1l1g=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:36:42 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Notez que le nom du compartiment est `CanonicalizedResource` inclus, mais `Request-URI` pas le HTTP. (Le compartiment est spécifié par l'en-tête hôte.)

Note

Le script Python suivant calcule la signature précédente à l'aide des paramètres fournis. Vous pouvez utiliser ce script pour créer vos propres signatures, en remplaçant les clés et le cas `StringToSign` échéant.

```

import base64
import hmac
from hashlib import sha1

access_key = 'AKIAIOSFODNN7EXAMPLE'.encode("UTF-8")
secret_key = 'wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY'.encode("UTF-8")

string_to_sign = 'GET\n\n\nTue, 27 Mar 2007 19:36:42 +0000\n/awsexamplebucket1/
photos/puppy.jpg'.encode("UTF-8")
signature = base64.b64encode(
    hmac.new(
        secret_key, string_to_sign, sha1
    ).digest()
).strip()

print(f"AWS {access_key.decode()}:{signature.decode()}")

```

Objet PUT

Cet exemple place un objet dans le compartiment awsexamplebucket1.

Demande	StringToSign
<pre> PUT /photos/puppy.jpg HTTP/1.1 Content-Type: image/jpeg Content-Length: 94328 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:15:45 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: iqRzw+ileNPu1fhspnRs8n0jjIA= </pre>	<pre> PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /awsexamplebucket1/photos/puppy.jpg </pre>

Notez l' Content-Type en-tête de la demande et du StringToSign. Notez également que le champ Content-MD5 est laissé vide dans le StringToSign, car il n'est pas présent dans la demande.

List

Cet exemple répertorie le contenu du compartiment `awsexamplebucket1`.

Demande	StringToSign
<pre>GET /?prefix=photos&max-keys=50&marker=puppy HTTP/1.1 User-Agent: Mozilla/5.0 Host: awsexamplebucket1.s3.us-west-1.amazo naws.com Date: Tue, 27 Mar 2007 19:42:41 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: m0WP8eCtspQl5Ahe6L1SozdX9YA=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:42:41 +0000\n /awsexamplebucket1/</pre>

Notez la barre oblique finale `CanonicalizedResource` et l'absence de paramètres de chaîne de requête.

Fetch

Cet exemple récupère la stratégie de contrôle d'accès pour le compartiment « `awsexamplebucket1` ».

Demande	StringToSign
<pre>GET /?acl HTTP/1.1 Host: awsexamplebucket1.s3.us-west-1.amazo naws.com Date: Tue, 27 Mar 2007 19:44:46 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: 82ZHiFIjc+WbcwFKGUVeQspPn+0=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:44:46 +0000\n /awsexamplebucket1/?acl</pre>

Remarquez comment le paramètre de chaîne de requête de sous-ressource est inclus dans le `CanonicalizedResource`.

Suppression

Cet exemple supprime un objet du compartiment « awsexamplebucket1 » grâce à l'alternative de type chemin et Date.

Demande	StringToSign
<pre>DELETE /awsexamplebucket1/photos/puppy.jpg HTTP/1.1 User-Agent: dotnet Host: s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:20:27 +0000 x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE:XbyTlbQdu9Xw5o8P4iMwPktxQd8=</pre>	<pre>DELETE\n \n \n Tue, 27 Mar 2007 21:20:26 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Notez la méthode alternative « x-amz-date » pour spécifier la date (parce que notre bibliothèque cliente nous empêchait de définir la date, par exemple). Dans ce cas, le code x-amz-date prime sur l'en-tête Date. Par conséquent, l'entrée de la date dans la signature doit contenir la valeur de l'en-tête x-amz-date.

Charger

Cet exemple charge un objet dans un compartiment d'hébergement virtuel CNAME avec des métadonnées.

Demande	StringToSign
<pre>PUT /db-backup.dat.gz HTTP/1.1 User-Agent: curl/7.15.5 Host: static.example.com:8080 Date: Tue, 27 Mar 2007 21:06:08 +0000 x-amz-acl: public-read content-type: application/x-download Content-MD5: 4gJE4saaMU4BqNR0kLY+lw== X-Amz-Meta-ReviewedBy: joe@example.com</pre>	<pre>PUT\n 4gJE4saaMU4BqNR0kLY+lw==\n application/x-download\n Tue, 27 Mar 2007 21:06:08 +0000\n x-amz-acl:public-read\n x-amz-meta-checksumalgorithm:crc32\n</pre>

Demande	StringToSign
<pre>X-Amz-Meta-ReviewedBy: jane@example.com X-Amz-Meta-FileChecksum: 0x02661779 X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment; filename=database.dat Content-Encoding: gzip Content-Length: 5913339 Authorization: AWS AKIAIOSFODNN7EXAMPLE LE: jtBQa0Aq+DkULFI8qrpwIjGEx0E=</pre>	<pre>x-amz-meta-filechecksum:0x026 61779\n x-amz-meta-reviewedby: joe@example.com,jane@example.com \n /static.example.com/db-backup.dat .gz</pre>

Notez la façon dont les en-têtes sont triés, débarrassés de leurs espaces et convertis en minuscules. Notez également que plusieurs en-têtes avec le même nom ont été réunis grâce aux virgules pour séparer les valeurs.

Notez comme seuls les en-têtes d'entité HTTP Content-Type et Content-MD5 apparaissent dans l'élément StringToSign. Les autres en-têtes d'entité Content-* n'apparaissent pas.

Encore une fois, notez que le nom du bucket est CanonicalizedResource inclus, mais Request-URI pas le HTTP. (Le compartiment est spécifié par l'en-tête hôte.)

Lister tous les compartiments

Demande	StringToSign
<pre>GET / HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:29:59 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdE RIC03wnaRNKh60qZehG9s=</pre>	<pre>GET\n \n \n Wed, 28 Mar 2007 01:29:59 +0000\n /</pre>

Clés Unicode

Demande	StringToSign
<pre>GET /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:49:49 +0000 Authorization: AWS AKIAIOSFODNN7EXAMP LE:DNEZGsoieTZ92F3bUfSPQcbGmLM=</pre>	<pre>GET\n \n \n Wed, 28 Mar 2007 01:49:49 +0000\n /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re</pre>

Note

Les éléments StringToSign qui en sont dérivés Request-URI sont pris littéralement, y compris en URL-Encoding majuscules.

Problèmes de signature de la demande REST

En cas d'échec de l'authentification de la demande REST, le système répond à la demande avec un rapport d'erreur XML. Les informations contenues dans ce rapport d'erreur aident les développeurs à identifier le problème. L'élément StringToSign du rapport d'erreur SignatureDoesNotMatch vous indique notamment la conversion sous forme canonique de la demande utilisée par le système.

Certaines boîtes à outils insèrent en silence et au préalable des en-têtes comme l'en-tête Content-Type lors d'une opération PUT. Dans la plupart des cas, la valeur de l'en-tête inséré reste constante, ce qui vous permet d'identifier les en-têtes manquants grâce à des outils comme Ethereal ou tcpmon.

Alternative à l'authentification d'une demande par chaîne d'interrogation

Vous pouvez authentifier certains types de demandes en transférant les informations requises en tant que paramètres de la chaîne d'interrogation au lieu d'utiliser l'en-tête HTTP Authorization. Cette méthode est utile pour activer l'accès direct d'un navigateur tiers vers les données Amazon S3 privées sans rediriger la demande. L'idée est de créer une demande « pré-signée » et de l'encoder comme une URL récupérable par le navigateur d'un utilisateur final. De plus, vous pouvez limiter une demande pré-signée en spécifiant un délai d'expiration.

Pour plus d'informations sur l'utilisation des paramètres de requête pour authentifier les demandes, consultez [Authentification des demandes : utilisation des paramètres de requête \(AWS Signature version 4\)](#) dans le manuel Amazon Simple Storage Service API Reference. Pour des exemples d'utilisation des AWS SDK pour générer des URL présignées, consultez la section [Partage d'objets avec des URL présignées](#).

Création d'une signature

Voici un exemple de requête REST Amazon S3 authentifiée par la chaîne d'interrogation.

```
GET /photos/puppy.jpg
?AWSAccessKeyId=AKIAIOSFODNN7EXAMPLE&Expires=1141889120&Signature=vjbyPxybdZaNmGa%2ByT272YEAiv4%3D HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
Date: Mon, 26 Mar 2007 19:37:58 +0000
```

La méthode d'authentification d'une demande par chaîne d'interrogation n'exige aucun en-tête HTTP spécial. A la place, les éléments d'authentification requis sont spécifiés en tant que paramètres de la chaîne d'interrogation :

Nom du paramètre de la chaîne d'interrogation	Exemple de valeur	Description
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE	ID de votre clé d' AWS accès. Spécifie la clé d'accès AWS secrète utilisée pour signer la demande et, indirectement, l'identité du développeur à l'origine de la demande.
Expires	1141889120	Le délai d'expiration de la signature , indiqué en nombre de secondes depuis l'époque (00:00:00 UTC le 1er janvier 1970). Passé ce délai (selon le serveur), toute demande reçue est rejetée.

Nom du paramètre de la chaîne d'interrogation	Exemple de valeur	Description
Signature	vjbyPxybdZaNmGa%2B yT272YEAiv4%3D	L'encodage URL du codage Base64 du HMAC-SHA1 of StringToSign.

La méthode d'authentification de la demande par chaîne d'interrogation diffère légèrement du paramètre de la demande `Signature` et de l'élément `StringToSign`. Voici une pseudo-grammaire qui illustre la construction de la méthode d'authentification de la demande par chaîne d'interrogation.

```
Signature = URL-Encode( Base64( HMAC-SHA1( YourSecretAccessKey, UTF-8-Encoding-Of( StringToSign ) ) ) );
```

```
StringToSign = HTTP-VERB + "\n" +  
    Content-MD5 + "\n" +  
    Content-Type + "\n" +  
    Expires + "\n" +  
    CanonicalizedAmzHeaders +  
    CanonicalizedResource;
```

`YourSecretAccessKey` est l'identifiant de clé d'accès AWS secrète qu'Amazon vous attribue lorsque vous vous inscrivez en tant que développeur Amazon Web Service. Remarquez `Signature` comment il convient de le placer dans la chaîne de requête. URL-Encoded Notez également que dans l'élément `StringToSign`, l'élément HTTP positionnel `Date` a été remplacé par `Expires`. Les `CanonicalizedAmzHeaders` et la `CanonicalizedResource` sont identiques.

Note

Dans la méthode d'authentification par chaîne d'interrogation, vous n'utilisez pas le `Date` ou l'en-tête `x-amz-date request` lors du calcul de l'élément `StringToSign`.

Authentification d'une demande par chaîne d'interrogation

Demande	StringToSign
<pre>GET /photos/puppy.jpg?AWSAccess KeyId=AKIAIOSFODNN7EXAMPLE& Signature=NpgCjnDzrM%2BWFzo ENXmpNDUsSn8%3D& Expires=1175139620 HTTP/1.1 Host: awsexamplebucket1.s3.us-wes t-1.amazonaws.com</pre>	<pre>GET\n \n \n 1175139620\n /awsexamplebucket1/photos/puppy.jpg</pre>

Cet exemple suppose que lorsqu'un navigateur effectue la requête GET, il ne fournit pas d' Content-Type en-tête Content-MD5 ou, et il ne définit pas non plus d'en-têtes x-amz-, de sorte que ces parties StringToSign sont laissées vides.

Utilisation de l'encodage en Base64

Les signatures de la demande HMAC doivent être encodées en Base64. L'encodage Base64 convertit la signature en une simple chaîne de caractères ASCII qui peut être attachée à la demande. Les caractères qui peuvent apparaître dans la chaîne de signature comme plus (+), barre oblique (/) et égal (=) doivent être encodés s'ils sont utilisés dans une URI. Par exemple, si le code d'authentification inclut un signe plus (+), utilisez %2B dans la demande. Pour une barre oblique, utilisez %2F et pour le signe égal, utilisez %3D.

Pour obtenir des exemples d'encodage en Base64, consultez les Amazon S3 [Exemples d'authentification](#).

Browser-based téléchargements à l'aide de POST (AWS signature (version 2))

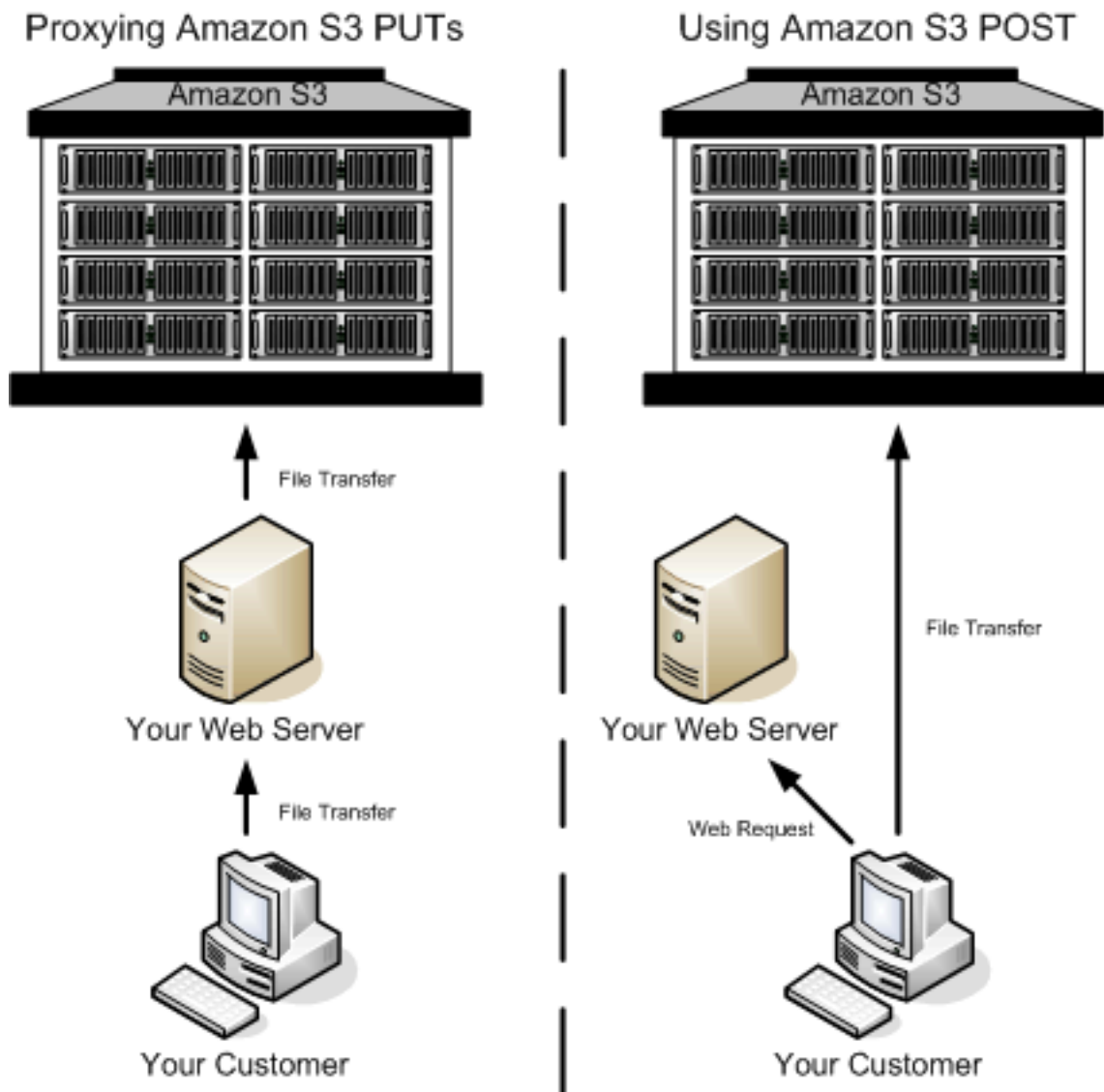
Amazon S3 prend en charge POST, qui permet à vos utilisateurs de télécharger directement du contenu vers Amazon S3. La méthode POST a été conçue pour simplifier les chargements, réduire la latence de chargement et économiser de l'argent sur les applications dans lesquelles les utilisateurs chargent des données à stocker dans Amazon S3.

 Note

L'authentification des demandes abordée dans cette section est basée sur AWS Signature Version 2, un protocole permettant d'authentifier les demandes d'API entrantes destinées aux AWS services.

Amazon S3 prend désormais en charge la version 4 de Signature, un protocole permettant d'authentifier les demandes d'API entrantes AWS destinées aux services. Régions AWSÀ l'heure actuelle, les versions Régions AWS créées avant le 30 janvier 2014 continueront de prendre en charge le protocole précédent, Signature Version 2. Toutes les nouvelles Régions créées après le 30 janvier 2014 prendront en charge uniquement Signature Version 4. Par conséquent, toutes les demandes adressées à ces Régions doivent être effectuées avec Signature Version 4. Pour plus d'informations, consultez [Authentification des demandes dans Browser-Based les téléchargements à l'aide de POST \(AWS Signature Version 4\)](#) dans le manuel Amazon Simple Storage Service API Reference.

La figure suivante illustre un téléchargement à l'aide de la méthode POST Amazon S3.



Chargement à l'aide de POST

- 1 L'utilisateur ouvre un navigateur web et accède à votre page web.
- 2 Votre page web comprend un formulaire HTTP qui contient toutes les informations nécessaires à l'utilisateur pour qu'il puisse télécharger le contenu sur Amazon S3.
- 3 L'utilisateur télécharge directement le contenu sur Amazon S3.

Note

L'authentification par chaîne d'interrogation n'est pas prise en charge pour POST.

Formulaires HTML (AWS signature (version 2))

Rubriques

- [Encodage des formulaires HTML](#)
- [Déclaration de formulaire HTML](#)
- [Champs de formulaire HTML](#)
- [Elaboration de la stratégie](#)
- [Élaboration d'une signature](#)
- [Redirection](#)

Lorsque vous communiquez avec Amazon S3, vous utilisez normalement l'API REST ou SOAP pour effectuer des opérations telles que put, get, delete, etc. Avec POST, les utilisateurs téléchargent les données directement sur Amazon S3 via leur navigateur, qui ne peut pas exécuter l'API SOAP ni créer de demande REST PUT.

Note

Les SOAP API pour Amazon S3 ne sont pas disponibles pour les nouveaux clients. Leur fin de vie est prévue le 31 août 2025. Nous vous recommandons d'utiliser l'API REST ou les kits de AWS développement logiciel (SDK).

Utilisez les formulaires HTML pour autoriser les utilisateurs à télécharger du contenu vers Amazon S3 en utilisant leur navigateur. Les formulaires HTML se composent d'une déclaration de formulaire et de champs de formulaire. La déclaration de formulaire contient des informations de haut niveau sur la demande. Les champs de formulaire contiennent des informations détaillées sur la demande, ainsi que sur la stratégie utilisée pour authentifier cette dernière et garantir qu'elle répond aux conditions que vous spécifiez.

Note

Les limites et les données de formulaire (à l'exclusion du contenu du fichier) ne peuvent pas dépasser 20 Ko.

Cette section explique comment utiliser les formulaires HTML.

Encodage des formulaires HTML

Le formulaire et la politique doivent être UTF-8 codés. Vous pouvez appliquer le UTF-8 codage au formulaire en le spécifiant dans l'en-tête HTML ou en tant qu'en-tête de demande.

Note

La déclaration d'un formulaire HTML n'accepte pas les paramètres d'authentification par chaîne d'interrogation.

Voici un exemple de UTF-8 codage dans le titre HTML :

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

Voici un exemple de UTF-8 codage dans un en-tête de demande :

```
Content-Type: text/html; charset=UTF-8
```

Déclaration de formulaire HTML

La déclaration d'un formulaire possède trois composants : l'action, la méthode et le type d'encadrement. Si l'une quelconque de ces valeurs est définie de façon incorrecte, la demande échoue.

L'action spécifie l'URL qui doit traiter la demande et qu'il convient de définir sur l'URL du compartiment. Par exemple, si le nom de votre compartiment est `awsexamplebucket1` et que la Région est USA Ouest (Californie du Nord), l'URL est `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/`.

Note

Le nom de clé est spécifié dans un champ du formulaire.

La méthode doit être POST.

Le type de pièce jointe (enctype) doit être spécifié et défini sur multipart/form-data pour les téléchargements de fichiers et les téléchargements de zones de texte. Pour de plus amples informations, veuillez consulter la [RFC 1867](#).

Exemple

L'exemple suivant est une déclaration de formulaire pour le compartiment « awsexamplebucket1 ».

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

Champs de formulaire HTML

Le tableau ci-dessous décrit les champs qui peuvent être utilisés au sein d'un formulaire HTML.

Note

La variable `${filename}` est automatiquement remplacée par le nom du fichier fourni par l'utilisateur et est reconnue par l'ensemble des champs de formulaire. Si le navigateur ou le client fournit un chemin d'accès complet ou partiel au fichier, seul le texte suivant la dernière barre oblique (/) ou barre oblique inverse (\) sera utilisé. Par exemple, « C:\Program Files\dossier1\fichier.txt » est interprété comme « fichier.txt ». Si aucun fichier ni nom de fichier n'est fourni, la variable est remplacée par une chaîne vide.

Nom de champ	Description	Obligatoire
AWSAccessKeyId	L'ID de clé d' AWS accès du propriétaire du bucket qui accorde à un utilisateur anonyme l'accès pour une demande qui répond à	Conditionnel

Nom de champ	Description	Obligatoire
	l'ensemble de contraintes de la politique. Ce champ est requis si la demande inclut un document de stratégie.	
acl	Une liste de contrôle d'accès (ACL) Amazon S3 Si une liste de contrôle d'accès non valide est spécifiée, une erreur est générée. Type : Chaîne Valeur par défaut : private Valeurs valides : private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	Non
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	REST-specific en-têtes. Pour de plus amples informations, veuillez consulter PUT Object .	Non
key	Nom de la clé chargée. Pour utiliser le nom de fichier fourni par l'utilisateur, utilisez la variable <code>\${filename}</code>. Par exemple, si l'utilisateur Betty charge le fichier lolcatz.jpg et que vous spécifiez <code>user/betty/\${filename}</code>, le fichier est enregistré sous le nom <code>//lolcatz.jpg</code>. <code>user/betty</code> Pour plus d'informations, consultez Utiliser les métadonnées d'un objet.	Oui

Nom de champ	Description	Obligatoire
<code>policy</code>	Stratégie de sécurité décrivant ce qui est autorisé dans la demande. Les demandes sans stratégie de sécurité sont considérées anonymes et aboutiront uniquement sur des compartiments publiquement accessibles en écriture.	Non
<code>success_action_redirect, redirect</code>	URL vers laquelle le client est redirigé en cas d'échec du chargement. Amazon S3 ajoute à l'URL les valeurs de compartiment, de clé et etag comme paramètres de la chaîne d'interrogation. Si <code>success_action_redirect</code> n'est pas spécifié, Amazon S3 retourne le type de document vide spécifié dans le champ <code>success_action_status</code>. Si Amazon S3 ne peut pas interpréter l'URL, le champ est ignoré. Si le téléchargement échoue, Amazon S3 affiche une erreur et ne redirige pas l'utilisateur vers une URL. Pour de plus amples informations, veuillez consulter Redirection.  Note Le nom du champ de redirection est obsolète et la prise en charge du nom du champ de redirection sera supprimée à l'avenir.	Non

Nom de champ	Description	Obligatoire
<code>success_action_status</code>	Code de statut retourné au client lors du succès du chargement si <code>success_action_redirect</code> n'est pas spécifié. Les valeurs valides sont 200, 201 et 204 (par défaut). Si la valeur est définie sur 200 ou 204, Amazon S3 retourne un document vide avec un code de statut égal à 200 ou 204. Si la valeur est définie sur 201, Amazon S3 retourne un document XML avec un code de statut égal à 201. Pour des informations sur le contenu du document XML, veuillez consulter POST Object. Si la valeur n'est pas définie ou si elle est définie sur une valeur non valide, Amazon S3 retourne un document vide avec un code de statut égal à 204.  Note Certaines versions d'Adobe Flash Player ne traitent pas correctement les réponses HTTP dont le corps est vide. Pour prendre en charge les chargements via Adobe Flash, nous vous recommandons de définir <code>success_action_status</code> sur 201.	Non

Nom de champ	Description	Obligatoire
signature	La signature HMAC construite à l'aide de la clé d'accès secrète qui correspond à celle fournieAWSAccessKeyId. Ce champ est requis si un document de stratégie est inclus dans la demande. Pour plus d'informations, consultez la section Gestion des identités et des accès pour Amazon S3.	Conditionnel
x-amz-security-token	Jeton de sécurité utilisé par les informations d'identification de session Si la demande utilise Amazon DevPay , elle nécessite deux champs de x-amz-security-token formulaire : un pour le jeton du produit et un pour le jeton utilisateur. Si la demande utilise les informations d'identification de session, elle requiert un seul formulaire x-amz-security-token . Pour de plus amples informations, veuillez consulter Informations d'identification de sécurité temporaires dans le Guide de l'utilisateur IAM.	Non
Autres noms de champs dotés du préfixe x-amz-meta-	User-specified métadonnées. Amazon S3 ne valide pas et n'utilise pas ces données. Pour de plus amples informations, veuillez consulter PUT Object.	Non

Nom de champ	Description	Obligatoire
dans le fichier	Fichier ou contenu de texte. Le fichier ou le contenu doit être le dernier champ du formulaire. Tout champ situé au-dessous est ignoré. Vous ne pouvez pas charger plus d'un fichier à la fois.	Oui

Elaboration de la stratégie

Rubriques

- [Expiration](#)
- [Conditions](#)
- [Correspondance des conditions](#)
- [Échappement de caractère](#)

La politique est un UTF-8 document Base64-encoded JSON qui spécifie les conditions que la demande doit remplir et qui est utilisé pour authentifier le contenu. En fonction de la manière dont vous concevez vos documents de stratégie, vous pouvez les utiliser par chargement, par utilisateur, pour tous les chargements ou selon d'autres conceptions répondant à vos besoins.

Note

Le document de stratégie est facultatif, mais nous recommandons vivement son utilisation plutôt que de rendre un compartiment publiquement accessible en écriture.

Voici un exemple de document de stratégie :

```
{ "expiration": "2007-12-01T12:00:00.000Z",  
  
  "conditions": [
```

```
{ "acl": "public-read" },  
  
{ "bucket": "awsexamplebucket1" },  
  
[ "starts-with", "$key", "user/eric/" ],  
  
]  
  
}
```

Le document de stratégie contient l'expiration et les conditions.

Expiration

L'élément expiration spécifie la date d'expiration de la stratégie au format de date UTC, conformément à la norme ISO 8601. Par exemple, « 2007-12-01 T12:00:00.000Z » indique que la politique n'est pas valide après minuit UTC le 01/12/2007. Une date d'expiration est requise dans une stratégie.

Conditions

Les conditions figurant dans le document de stratégie valident le contenu de l'objet chargé. Chaque champ de formulaire que vous spécifiez dans le formulaire (à l'exception des noms de signatureAWSAccessKeyId, de fichier, de politique et de champ qui ont un préfixe x-ignore-) doit être inclus dans la liste des conditions.

Note

Si plusieurs champs ont le même nom, les valeurs doivent être séparées par des virgules. Par exemple, si vous avez deux champs nommés « x-amz-meta-tag » et que le premier a la valeur « Ninja » alors que le second a la valeur « Stallman », vous devez définir le document de stratégie sur Ninja, Stallman.

Toutes les variables figurant dans le formulaire sont développées avant la validation de la stratégie. Par conséquent, toutes les correspondances des conditions doivent être effectuées par rapport aux champs développés. Par exemple, si vous définissez le champ clé sur user/betty/\${filename}, la stratégie peut être ["starts-with", "\$key", "user/betty/"]. Ne saisissez pas ["starts-with", "\$key", "user/betty/\${filename}"]. Pour de plus amples informations, veuillez consulter [Correspondance des conditions](#).

Le tableau ci-dessous décrit les conditions d'un document de stratégie.

Nom d'élément	Description
liste acl	Spécifie les conditions que la liste ACL doit respecter. Prend en charge la correspondance exacte et <code>starts-with</code> .
content-length-range	Spécifie les tailles minimale et maximale autorisées pour le contenu chargé. Prend en charge la correspondance de plage.
Cache-Control, Content-Type,, Content-DispositionContent-Encoding, Expire	REST-specific en-têtes. Prend en charge la correspondance exacte et <code>starts-with</code> .
key	Nom de la clé chargée. Prend en charge la correspondance exacte et <code>starts-with</code> .
success_action_redirect, redirect	URL vers laquelle le client est redirigé en cas d'échec du chargement. Prend en charge la correspondance exacte et <code>starts-with</code> .
success_action_status	Code de statut retourné au client lors du succès du chargement si <code>success_action_redirect</code> n'est pas spécifié. Prend en charge la correspondance exacte.
x-amz-security-token	Jeton DevPay de sécurité Amazon.

Nom d'élément	Description
	Chaque demande qui utilise Amazon DevPay nécessite deux champs de <code>x-amz-security-token</code> : un pour le jeton du produit et un pour le jeton utilisateur. Par conséquent, les valeurs doivent être séparées par des virgules. Par exemple, si le jeton utilisateur est <code>eW91dHVIZQ==</code> et le token produit <code>b0hnNVNKWVJIQTA=</code> , vous définissez l'entrée de stratégie sur : <code>{ "x-amz-security-token": "eW91dHVIZQ==,b0hnNVNKWVJIQTA=" }</code> .
Autres noms de champs dotés du préfixe <code>x-amz-meta-</code>	User-specified métadonnées. Prend en charge la correspondance exacte et <code>starts-with</code> .

 Note

Si votre boîte à outils ajoute des champs supplémentaires (p. ex. : Flash ajoute le nom de fichier), vous devez les ajouter dans le document de stratégie. Si vous pouvez contrôler cette fonctionnalité, ajoutez le préfixe `x-ignore-` au champ afin qu'Amazon S3 ignore la fonction et que les versions futures de cette fonction ne soient pas affectées.

Correspondance des conditions

Le tableau ci-dessous décrit les types de correspondance des conditions. Vous devez spécifier une seule condition pour chaque champ de formulaire que vous spécifiez dans le formulaire, mais vous pouvez créer des critères de correspondance plus complexes en spécifiant plusieurs conditions pour un champ de formulaire.

Condition	Description

Condition	Description
Correspondances exactes	Les correspondances exactes vérifient que les champs correspondent à des valeurs spécifiques. Cet exemple indique que la liste ACL doit être définie sur public-read : <pre>{"acl": "public-read" }</pre> Cet exemple représente une méthode alternative pour indiquer que la liste ACL doit être définie sur public-read : <pre>["eq", "\$acl", "public-read"]</pre>
Commence par	Si la valeur doit commencer par une certaine valeur, utilisez le mot clé starts-with. Cet exemple indique que la clé doit commencer par user/betty : <pre>["starts-with", "\$key", "user/betty/"]</pre>
Correspondance avec un contenu quelconque	Pour configurer la stratégie de manière à autoriser un contenu quelconque dans un champ, utilisez starts-with avec une valeur vide. Cet exemple autorise une valeur quelconque pour success_action_redirect : <pre>["starts-with", "\$success_action_redirect", ""]</pre>
Spécification de plages	Pour les champs qui acceptent des plages, séparez les seuils inférieur et supérieur de plage par une virgule. Cet exemple autorise une taille de fichier comprise entre 1 et 10 mégaoctets : <pre>["content-length-range", 1048579, 10485760]</pre>

Échappement de caractère

Le tableau suivant décrit les caractères qui doivent être placés dans une séquence d'échappement au sein d'un document de stratégie.

Séquence d'échappement	Description
\\	Barre oblique inverse
\\$	Symbole dollar
\b	Retour arrière
\f	Saut de page
\n	Nouvelle ligne
\r	Retour chariot
\t	Tabulation horizontale
\v	Tabulation verticale
\ u xxxx	Tous les caractères Unicode

Élaboration d'une signature

Étape	Description
1	Encodez la politique à l'aide UTF-8 de.

Étape	Description
2	Encodez ces UTF-8 octets à l'aide de Base64.
3	Signez la politique à l'aide de votre clé d'accès secrète à l'aide du HMACSHA-1.
4	Encodez la SHA-1 signature à l'aide de Base64.

Pour des informations générales sur l'authentification, consultez la section Gestion des [identités et des accès pour Amazon S3](#).

Redirection

Cette section décrit comment traiter les redirections.

Redirection générale

Une fois la demande POST terminée, l'utilisateur est redirigé à l'emplacement que vous avez spécifié dans le champ `success_action_redirect`. Si Amazon S3 ne peut pas interpréter l'URL, le champ `success_action_redirect` est ignoré.

Si `success_action_redirect` n'est pas spécifié, Amazon S3 retourne le type de document vide spécifié dans le champ `success_action_status`.

Si la demande POST échoue, Amazon S3 affiche une erreur et ne fournit pas de redirection.

Pre-upload redirection

Si votre compartiment a été créé à l'aide de `< CreateBucketConfiguration >`, vos utilisateurs finaux peuvent avoir besoin d'une redirection. Si cela se produit, certains navigateurs peuvent traiter la redirection de façon incorrecte. Cela est relativement rare, mais a le plus de chances de se produire juste après la création d'un compartiment.

Téléchargez des exemples (AWS signature (version 2))

Rubriques

- [Chargement d'un fichier](#)
- [Chargement d'une zone de texte](#)

Note

L'authentification des demandes abordée dans cette section est basée sur AWS Signature Version 2, un protocole permettant d'authentifier les demandes d'API entrantes destinées aux AWS services.

Amazon S3 prend désormais en charge la version 4 de Signature, un protocole permettant d'authentifier les demandes d'API entrantes AWS destinées aux services. Régions AWS À l'heure actuelle, les versions Régions AWS créées avant le 30 janvier 2014 continueront de prendre en charge le protocole précédent, Signature Version 2. Toutes les nouvelles Régions créées après le 30 janvier 2014 prendront en charge uniquement Signature Version 4.

Par conséquent, toutes les demandes adressées à ces Régions doivent être effectuées avec Signature Version 4. Pour plus d'informations, consultez [Exemples : Browser-Based téléchargement à l'aide de HTTP POST \(à l'aide de AWS Signature version 4\)](#) dans le manuel Amazon Simple Storage Service API Reference.

Chargement d'un fichier

Cet exemple illustre le processus complet d'élaboration d'une stratégie et d'un formulaire pouvant être utilisés pour charger un fichier joint.

Élaboration d'une stratégie et d'un formulaire

La stratégie suivante prend en charge les chargements vers Amazon S3 pour le compartiment `awsexamplebucket1`.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

Cette stratégie exige le respect des conditions suivantes :

- Le chargement doit se produire avant 12:00, heure UTC, le 1er décembre 2007.
- Le contenu doit être chargé dans le compartiment `awsexamplebucket1`.
- La touche doit commencer par «`user/eric/`».
- La liste ACL est définie sur `public-read`.
- Le fichier `success_action_redirect` est défini sur `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html`.
- L'objet est un fichier image.
- La balise `x-amz-meta-uuid` doit être définie sur `14365123651274`.
- Le champ `x-amz-meta-tag` peut contenir une valeur quelconque.

Voici une Base64-encoded version de cette politique.

```
eyJhZlZxhwaXJhdGlvbiI6IClYMDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
```

Créez une signature à l'aide de vos informations d'identification. Par exemple, `0RavWzkygo6QX9caELEqKi9kDbU=` est la signature pour le document de stratégie précédent.

Le formulaire suivant permet d'envoyer une requête POST au `amzn-s3-demo-bucket` bucket qui utilise cette politique.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
    ...
    <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
    enctype="multipart/form-data">
      Key to upload: <input type="input" name="key" value="user/eric/" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="https://
      awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html" />
      Content-Type: <input type="input" name="Content-Type" value="image/jpeg" /><br />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
```

```



```

Exemple de demande

Cette demande suppose que l'image chargée a une taille de 117 108 octets. Les données de l'image ne sont pas incluses.

```

POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
  Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=9431149156168
Content-Length: 118698

--9431149156168
Content-Disposition: form-data; name="key"

user/eric/MyPicture.jpg
--9431149156168
Content-Disposition: form-data; name="acl"

public-read
--9431149156168
Content-Disposition: form-data; name="success_action_redirect"

https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html
--9431149156168
Content-Disposition: form-data; name="Content-Type"

```

```
image/jpeg
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--9431149156168
Content-Disposition: form-data; name="x-amz-meta-tag"

Some, Tag, For, Picture
--9431149156168
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--9431149156168
Content-Disposition: form-data; name="Policy"

eyJhZXBwaXJhdGlvbiI6ICIyMDA3LTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
--9431149156168
Content-Disposition: form-data; name="Signature"

0RavWzkygo6QX9caELEqKi9kDbU=
--9431149156168
Content-Disposition: form-data; name="file"; filename="MyFilename.jpg"
Content-Type: image/jpeg

...file content...
--9431149156168
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--9431149156168--
```

Exemple de réponse

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
```

```
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
successful_upload.html?bucket=awsexamplebucket1&key=user/eric/
MyPicture.jpg&etag="39d459dfbc0faabbb5e179358dfb94c3"
Server: AmazonS3
```

Chargement d'une zone de texte

Rubriques

- [Élaboration d'une stratégie et d'un formulaire](#)
- [Exemple de demande](#)
- [Exemple de réponse](#)

L'exemple suivant illustre le processus complet d'élaboration d'une stratégie et d'un formulaire pour charger une zone de texte. Le chargement d'une zone de texte est utile pour soumettre du contenu créé par l'utilisateur, tel que des billets de blog.

Élaboration d'une stratégie et d'un formulaire

La stratégie suivante prend en charge les chargements de zones de texte vers Amazon S3 pour le compartiment `awsexamplebucket1`.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/
new_post.html"},
    ["eq", "$Content-Type", "text/html"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

Cette stratégie exige le respect des conditions suivantes :

- Le chargement doit se produire avant 12:00, heure GMT, le 1er décembre 2007.
- Le contenu doit être chargé dans le compartiment `awsexamplebucket1`.
- La touche doit commencer par «user/eric/».

- La liste ACL est définie sur public-read.
- Le fichier success_action_redirect est défini sur https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html.
- L'objet est un texte HTML.
- La balise x-amz-meta-uuid doit être définie sur 14365123651274.
- Le champ x-amz-meta-tag peut contenir une valeur quelconque.

Vous trouverez ci-dessous une Base64-encoded version de cette politique.

```
eyJhZGhwaXJhdGlvbiI6IClYMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgYjJdWnrZXQiOiAiam9obnNtaXR0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCaidXNlciLAogICAgYjY2wi0iAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2pC5zMy5hbWwF6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSI6ICJ0ZXh0L2h0CAGIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LWFtei1tIsIClIiXQogIF0KfQo=
```

Créez une signature en utilisant vos informations d'identification. Par exemple, qA7FWXKq6VvU681I9KdveT1cWgF= est la signature pour le document de stratégie précédent.

Le formulaire suivant permet d'envoyer une requête POST au amzn-s3-demo-bucket bucket qui utilise cette politique.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
    ...
    <form action="https://amzn-s3-demo-bucket.s3.us-west-1.amazonaws.com/" method="post"
    enctype="multipart/form-data">
      Key to upload: <input type="input" name="key" value="user/eric/" /><br />
      <input type="hidden" name="acl" value="public-read" />
      <input type="hidden" name="success_action_redirect" value="https://
      awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html" />
      <input type="hidden" name="Content-Type" value="text/html" />
      <input type="hidden" name="x-amz-meta-uuid" value="14365123651274" />
      Tags for File: <input type="input" name="x-amz-meta-tag" value="" /><br />
      <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
```

```



```

Your blog post goes here.

```

</textarea><br />
<!-- The elements after this will be ignored -->


```

Exemple de demande

Cette demande suppose que l'image chargée a une taille de 117 108 octets. Les données de l'image ne sont pas incluses.

```

POST / HTTP/1.1
Host: awsexamplebucket1.s3.us-west-1.amazonaws.com
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.8.1.10) Gecko/20071115
  Firefox/2.0.0.10
Accept: text/xml,application/xml,application/xhtml+xml,text/html;q=0.9,text/
plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip,deflate
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
Keep-Alive: 300
Connection: keep-alive
Content-Type: multipart/form-data; boundary=178521717625888
Content-Length: 118635

-178521717625888
Content-Disposition: form-data; name="key"

ser/eric/NewEntry.html
--178521717625888
Content-Disposition: form-data; name="acl"

public-read
--178521717625888
Content-Disposition: form-data; name="success_action_redirect"

```



```
https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html
--178521717625888
Content-Disposition: form-data; name="Content-Type"

text/html
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-uuid"

14365123651274
--178521717625888
Content-Disposition: form-data; name="x-amz-meta-tag"

Interesting Post
--178521717625888
Content-Disposition: form-data; name="AWSAccessKeyId"

AKIAIOSFODNN7EXAMPLE
--178521717625888
Content-Disposition: form-data; name="Policy"
eyJhZiZxhwaXJhdGlvbiI6IClIyMDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgZjIdWm
--178521717625888
Content-Disposition: form-data; name="Signature"

qA7FWXKq6VvU681I9KdveT1cWgF=
--178521717625888
Content-Disposition: form-data; name="file"

...content goes here...
--178521717625888
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--178521717625888--
```

Exemple de réponse

```
HTTP/1.1 303 Redirect
x-amz-request-id: 1AEE782442F35865
x-amz-id-2: cxzFLJRatFHy+NGtaDFRR8YvI9BHmgLxjvJzNiGGICARZ/mVXHj7T+qQKhdpzHFh
Content-Type: application/xml
Date: Wed, 14 Nov 2007 21:21:33 GMT
Connection: close
```

```
Location: https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html?
bucket=awsexamplebucket1&key=user/eric/
NewEntry.html&etag=40c3271af26b7f1672e41b8a274d28d4
Server: AmazonS3
```

POST avec Adobe Flash (AWS signature (version 2))

Cette section décrit comment utiliser POST avec Adobe Flash.

Sécurité d'Adobe Flash Player

Par défaut, le modèle de sécurité d'Adobe Flash Player interdit aux lecteurs Adobe Flash Player d'établir des connexions réseau à des serveurs situés hors du domaine qui assure le service du fichier SWF.

Pour remplacer le paramètre par défaut, vous devez charger un fichier `crossdomain.xml` publiquement accessible en lecture dans le compartiment qui acceptera les chargements POST. Vous trouverez ci-dessous un exemple de fichier `crossdomain.xml`.

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```

Note

Pour plus d'informations sur le modèle de sécurité d'Adobe Flash, accédez au site web d'Adobe.

L'ajout du fichier `crossdomain.xml` dans votre compartiment autorise tout lecteur Adobe Flash Player à se connecter au fichier `crossdomain.xml` au sein de votre compartiment. Toutefois, cela n'accorde pas l'accès au compartiment Amazon S3 lui-même.

Considérations sur Adobe Flash

L' `FileReference` API d'Adobe Flash ajoute le champ de `Filename` formulaire à la requête POST. Lorsque vous créez des applications Adobe Flash qui sont chargées sur Amazon S3 à l'aide de l'action d' `FileReference` API, incluez la condition suivante dans votre politique :

```
['starts-with', '$Filename', '']
```

Certaines versions d'Adobe Flash Player ne traitent pas correctement les réponses HTTP dont le corps est vide. Pour configurer POST de manière à retourner une réponse dont le corps n'est pas vide, définissez `success_action_status` sur 201. Amazon S3 retournera alors un document XML avec un code de statut égal à 201. Pour des informations sur le contenu du document XML, veuillez consulter [POST Object](#). Pour obtenir des informations sur les champs de formulaire, consultez [Champs de formulaire HTML](#).

Annexe : API de configuration du cycle de vie (obsolète)

La configuration du cycle de vie des compartiments est mise à jour pour prendre en charge les filtres basés sur les balises d'objets. En d'autres termes, vous pouvez désormais spécifier une règle qui spécifie le préfixe du nom de clé, une ou plusieurs balises d'objet, ou les deux pour sélectionner un sous-ensemble d'objets auquel la règle s'applique. Les API ont été mises à jour en conséquence. Les rubriques suivantes décrivent la version précédente des opérations de cycle de vie des compartiments PUT et GET à des fins de rétrocompatibilité.

Rubriques

- [Cycle de vie du bucket PUT \(obsolète\)](#)
- [Cycle de vie du bucket GET \(obsolète\)](#)

Cycle de vie du bucket PUT (obsolète)

Description

Important

Pour obtenir une version mise à jour de cette API, consultez [PutBucketLifecycleConfiguration](#). Cette version est obsolète. Les configurations de cycle de vie existantes fonctionneront. Pour les nouvelles configurations de cycle de vie, utilisez l'API mise à jour.

Crée une nouvelle configuration de cycle de vie pour le compartiment ou remplace une configuration de cycle de vie existante. Pour plus d'informations sur la configuration du cycle de vie, consultez la section Gestion du cycle de vie des [objets](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Permissions

Par défaut, toutes les ressources Amazon S3, y compris les compartiments, les objets et les sous-ressources associées (par exemple, la configuration du cycle de vie et la configuration du site Web) sont privées. Seul le propriétaire de la ressource, Compte AWS celui qui l'a créée, peut y accéder. Le propriétaire de la ressource peut éventuellement accorder des autorisations d'accès à d'autres personnes en rédigeant une politique d'accès. Pour cette opération, les utilisateurs doivent obtenir l'`s3:PutLifecycleConfiguration` autorisation.

Vous pouvez également refuser explicitement les autorisations. Le refus explicite remplace également toute autre autorisation. Si vous souhaitez empêcher les utilisateurs ou les comptes de retirer ou de supprimer des objets de votre compartiment, vous devez leur refuser l'autorisation d'effectuer les actions suivantes :

- `s3:DeleteObject`
- `s3:DeleteObjectVersion`
- `s3:PutLifecycleConfiguration`

Pour plus d'informations sur les autorisations, consultez [la section Gestion des autorisations d'accès à vos ressources Amazon S3](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Requêtes

Syntaxe

```
PUT /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Content-Length: length
Date: date
Authorization: authorization string
Content-MD5: MD5
```

Lifecycle configuration in the request body

Pour plus de détails sur les chaînes d'autorisation, consultez [Demandes d'authentification \(AWS Signature \(version 4\)\)](#).

Paramètres de demande

Cette implémentation de l'opération n'utilise pas de paramètres de demande.

En-têtes de demande

Nom	Description	Obligatoire
Content-MD5	Le condensé MD5 128 bits codé en base64 des données. Vous devez utiliser cet en-tête pour vérifier l'intégrité du message afin de vérifier que le corps de la demande n'a pas été endommagé pendant le transit. Pour plus d'informations, consultez la RFC 1864. Type : Chaîne Par défaut : aucun	Oui

Corps de la requête

Dans la demande, vous spécifiez la configuration du cycle de vie dans le corps de la demande. La configuration du cycle de vie est spécifiée au format XML. Voici un exemple de configuration de base

du cycle de vie. Il spécifie une règle. Le Prefix contenu de la règle identifie les objets auxquels la règle s'applique. La règle spécifie également deux actions (Transition et Expiration). Chaque action spécifie un calendrier dans lequel Amazon S3 doit effectuer l'action. Status Indique si la règle est activée ou désactivée.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>key-prefix</Prefix>
    <Status>rule-status</Status>
    <Transition>
      <Date>value</Date>
      <StorageClass>storage class</StorageClass>
    </Transition>
    <Expiration>
      <Days>value</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Si l'état de votre compartiment est activé ou suspendu, vous pouvez avoir plusieurs versions du même objet : une version actuelle et zéro ou plusieurs versions non actuelles. La configuration de cycle de vie suivante spécifie les actions (NoncurrentVersionTransition, NoncurrentVersionExpiration) spécifiques aux versions d'objets non actuelles.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>key-prefix</Prefix>
    <Status>rule-status</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>value</NoncurrentDays>
      <StorageClass>storage class</StorageClass>
    </NoncurrentVersionTransition>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>value</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
</LifecycleConfiguration>
```

Vous pouvez utiliser l'API de chargement partitionné pour charger des objets volumineux en plusieurs parties. Pour plus d'informations sur les chargements partitionnés, consultez la section [Présentation des chargements partitionnés](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service. Avec la configuration du cycle de vie, vous pouvez demander à Amazon S3 d'annuler les chargements partitionnés incomplets, qui sont identifiés par le préfixe du nom de clé spécifié dans la règle, s'ils ne sont pas terminés dans un délai de jours spécifié. Lorsqu'Amazon S3 annule un chargement en plusieurs parties, il supprime toutes les parties associées au téléchargement. Cela vous permet d'éviter les chargements partitionnés incomplets dont certaines parties ont été stockées dans Amazon S3. Vous n'avez donc pas à payer de frais de stockage pour ces téléchargements. Voici un exemple de configuration du cycle de vie qui spécifie une règle avec l'action `AbortIncompleteMultipartUpload`. Cette action indique à Amazon S3 d'annuler les chargements partitionnés incomplets sept jours après leur lancement.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>SomeKeyPrefix</Prefix>
    <Status>rule-status</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

Le tableau suivant décrit les éléments XML de la configuration du cycle de vie.

Nom	Description	Obligatoire
<code>AbortIncompleteMultipartUpload</code>	Conteneur permettant de spécifier à quel moment un téléchargement partitionné incomplet peut faire l'objet d'une opération d'abandon. Enfant : <code>DaysAfterInitiation</code> Type : Conteneur Ancêtre : <code>Rule</code>	Oui, si aucune autre action n'est spécifiée pour la règle

Nom	Description	Obligatoire
Date	Date à laquelle vous souhaitez qu'Amazon S3 entreprenne l'action. Pour plus d'informations, consultez Règles de cycle de vie : basées sur une date spécifique dans le guide de l'utilisateur d'Amazon Simple Storage Service. La valeur de date doit être conforme au format ISO 8601. L'heure indique toujours minuit UTC. Type : Chaîne Ancêtre : Expiration ou Transition	Oui, si Days et si vous ExpiredObjectDeleteMarker êtes absents
Days	Spécifie le nombre de jours après la création de l'objet pendant lesquels l'action de règle spécifique prend effet. Type : entier non négatif lorsqu'il est utilisé avec Transition , entier positif lorsqu'il est utilisé avec Expiration Ancêtre : Expiration , Transition	Oui, si Date et si vous ExpiredObjectDeleteMarker êtes absents

Nom	Description	Obligatoire
<code>DaysAfterInitiation</code>	Spécifie le nombre de jours qui suivent le lancement d'un chargement en plusieurs parties pendant lesquels le téléchargement en plusieurs parties doit être terminé. S'il ne se termine pas dans le délai spécifié, il devient éligible à une opération d'abandon et Amazon S3 annule le chargement partitionné incomplet. Type : Entier positif Ancêtre : <code>AbortIncompleteMultipartUpload</code>	Oui, si un tag parent est spécifié

Nom	Description	Obligatoire
Expiration	Cette action spécifie une période de la durée de vie d'un objet pendant laquelle Amazon S3 doit prendre l'action d'expiration appropriée. L'action entreprise par Amazon S3 varie selon que le bucket est activé ou non pour la gestion des versions. • Si la gestion des versions n'a jamais été activée sur le compartiment, Amazon S3 supprime définitivement la seule copie de l'objet. • Si le compartiment est activé pour la gestion des versions (ou si la gestion des versions est suspendue), l'action s'applique uniquement à la version actuelle de l'objet. Un bucket activé pour la gestion des versions peut avoir plusieurs versions du même objet : une version actuelle et zéro ou plusieurs versions non actuelles. Au lieu de supprimer la version actuelle, Amazon S3 en fait une version non actuelle en ajoutant un marqueur de suppression comme nouvelle version actuelle.	Oui, si aucune autre action n'est présente dans leRule.

 **Important**

Si l'état d'un compartiment est suspendu pour cause de version, Amazon S3 crée un marqueur de suppression avec l'ID de version. null Si vous possédez une version

Nom	Description	Obligatoire
ExpiredObjectDeleteMarker	Dans un compartiment versionné (un compartiment activé ou suspendu pour la gestion des versions), vous pouvez ajouter cet élément dans la configuration du cycle de vie pour indiquer à Amazon S3 de supprimer les marqueurs de suppression d'objets expirés. Pour un exemple, consultez l'Exemple 8 : Suppression des marqueurs de suppression d'objets expirés dans le guide de l'utilisateur d'Amazon Simple Storage Service. Ne l'ajoutez pas à un compartiment non versionné, car ce type de compartiment ne peut pas inclure de marqueurs de suppression. Type : Chaîne Valeurs valides : true false (la valeur false est autorisée, mais elle est no-op, ce qui signifie qu'Amazon S3 n'agira pas) Ancêtre : Expiration	Oui, si Date et si vous Days êtes absents

Nom	Description	Obligatoire
<code>NoncurrentDays</code>	Spécifie le nombre de jours depuis lequel un objet est ancien avant qu'Amazon S3 puisse effectuer l'action associée. Pour plus d'informations sur les calculs des jours non courants, consultez Comment Amazon S3 calcule quand un objet est devenu non courant dans le guide de l'utilisateur d'Amazon Simple Storage Service. Type : entier non négatif lorsqu'il est utilisé avec <code>NoncurrentVersionTransition</code>, entier positif lorsqu'il est utilisé avec <code>NoncurrentVersionExpiration</code> Ancêtre : <code>NoncurrentVersionExpiration</code> ou <code>NoncurrentVersionTransition</code>	Oui

Nom	Description	Obligatoire
NoncurrentVersionExpiration	Spécifie quand les versions d'objet anciennes expirent. Lorsqu'elles expirent, Amazon S3 supprime définitivement les versions d'objet anciennes. Définissez cette action de configuration du cycle de vie sur un compartiment dont la gestion des versions est activée (ou suspendue) pour indiquer à Amazon S3 de supprimer les versions non actuelles de l'objet à une période spécifique de la durée de vie de l'objet. Type : Conteneur Enfants : NoncurrentDays Ancêtre : Rule	Oui, si aucune autre action n'est présente dans le Rule

Nom	Description	Obligatoire
NoncurrentVersionTransition	Conteneur pour la règle de transition qui décrit le moment où des objets non courants passent à la classe de stockage STANDARD_IA ONEZONE_IA , ou à la classe de stockage. Si la gestion des versions est activée dans votre compartiment (ou si la gestion des versions est suspendue), vous pouvez définir cette action pour indiquer à Amazon S3 de transférer les versions non actuelles de l'objet à une période spécifique de la durée de vie de l'objet. Type : Conteneur Enfants : NoncurrentDays et StorageClass Ancêtre : Rule	Oui, si aucune autre action n'est présente dans le Rule
Prefix	Préfixe de clé d'objet qui identifie un ou plusieurs objets auxquels la règle s'applique. Type : Chaîne Ancêtre : Rule	Oui
Rule	Conteneur pour une règle de cycle de vie. Une configuration de cycle de vie peut contenir jusqu'à 1 000 règles. Type : Conteneur Ancestor:LifecycleConfiguration	Oui

Nom	Description	Obligatoire
Status	Si cette option est activée, Amazon S3 exécute la règle comme prévu. S'il est désactivé, Amazon S3 ignore la règle. Type : Chaîne Ancêtre : Rule Valeurs valides : Activé, Désactivé	Oui
StorageClass	Spécifie la classe de stockage Amazon S3 vers laquelle vous souhaitez que l'objet fasse la transition. Type : Chaîne Ancêtre : Transition et NoncurrentVersionTransition Valeurs valides : STANDARD_IA ONEZONE_IA GLACIER	Oui Cet élément n'est obligatoire que si vous spécifiez l'un de ses ancêtres ou les deux.

Nom	Description	Obligatoire
Transition	Cette action spécifie une période de la durée de vie des objets au cours de laquelle Amazon S3 doit les faire passer à la classe de stockage STANDARD_IA ONEZONE_IA , ou à la classe de stockage. Lorsque cette action est effective , les actions d'Amazon S3 dépendent du fait que le compartiment est activé ou non pour la gestion des versions. • Si la gestion des versions n'a jamais été activée sur le compartiment, Amazon S3 transfère la seule copie de l'objet vers la classe de stockage spécifiée. • Si la gestion des versions est activée dans votre compartiment (ou si la gestion des versions est suspendue), Amazon S3 transfère uniquement les versions actuelles des objets identifiés dans la règle.  Note Un bucket activé pour la gestion des versions peut contenir de nombreuses versions d'un objet. Cette action n'a aucun effet sur les versions d'objets non actuelles. Pour effectuer la transition d'objets non courants, vous devez utiliser cette NoncurrentVersionTransition action.	Oui, si aucune autre action n'est présente dans le Rule

Nom	Description	Obligatoire
	Type : Conteneur Enfants : jours ou date, et StorageClass Ancêtre : Rule	

Réponses

En-têtes de réponse

Cette implémentation de l'opération utilise uniquement les en-têtes de réponse communs à la plupart des réponses. Pour plus d'informations, consultez la section En-têtes [de réponse](#) communs.

Éléments de réponse

Cette implémentation de l'opération ne renvoie pas d'éléments de réponse.

Erreurs spéciales

Cette implémentation de l'opération ne renvoie pas d'erreurs spéciales. Pour obtenir des informations générales sur les erreurs Amazon S3 et une liste des codes d'erreur, consultez la section Réponses aux [erreurs](#).

Exemples

Exemple 1 : ajouter une configuration de cycle de vie à un compartiment qui ne l'est pas Versioning-enabled

La configuration de cycle de vie suivante spécifie deux règles, chacune comportant une action.

- L'action Transition indique à Amazon S3 de transférer les objets dotés du préfixe « documents/ » vers la classe de stockage 30 jours après leur création.
- L'action Expiration indique à Amazon S3 de supprimer les objets dotés du préfixe « logs/ » 365 jours après leur création.

```
<LifecycleConfiguration>  
  <Rule>
```

```
<ID>id1</ID>
<Prefix>documents/</Prefix>
<Status>Enabled</Status>
<Transition>
  <Days>30</Days>
  <StorageClass>GLACIER</StorageClass>
</Transition>
</Rule>
<Rule>
  <ID>id2</ID>
  <Prefix>logs/</Prefix>
  <Status>Enabled</Status>
  <Expiration>
    <Days>365</Days>
  </Expiration>
</Rule>
</LifecycleConfiguration>
```

Voici un exemple de PUT `/?lifecycle` demande qui ajoute la configuration de cycle de vie précédente au exemplebucket compartiment.

```
PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:11:21 GMT
Content-MD5: q6yJD1IkcBaGGfb3QLY69A==
Authorization: authorization string
Content-Length: 415

<LifecycleConfiguration>
  <Rule>
    <ID>id1</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
  </Rule>
  <Rule>
    <ID>id2</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <Expiration>
```

```
<Days>365</Days>
</Expiration>
</Rule>
</LifecycleConfiguration>
```

Voici un exemple de réponse.

```
HTTP/1.1 200 OK
x-amz-id-2: r+qR7+nhXtJDDIJ0JJYcd+1j5nM/rUFiiiZ/fNbD0sd3JUE8NWMLNHXmvPfwMpdC
x-amz-request-id: 9E26D08072A8EF9E
Date: Wed, 14 May 2014 02:11:22 GMT
Content-Length: 0
Server: AmazonS3
```

Exemple 2 : ajouter une configuration de cycle de vie à un Versioning-enabled bucket

La configuration de cycle de vie suivante spécifie deux règles, chacune comportant une action à effectuer par Amazon S3. Vous spécifiez les actions suivantes lorsque la gestion des versions est activée ou que la gestion des versions est suspendue :

- L'NoncurrentVersionExpirationaction indique à Amazon S3 d'expirer les versions non actuelles des objets avec le préfixe « logs/ » 100 jours après que les objets soient devenus obsolètes.
- L'NoncurrentVersionTransitionaction indique à Amazon S3 de transférer les versions non actuelles des objets avec le préfixe « documents/ » vers la classe de stockage 30 jours après leur mise à jour.

```
<LifecycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>100</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
```

```

<Status>Enabled</Status>
<NoncurrentVersionTransition>
  <NoncurrentDays>30</NoncurrentDays>
  <StorageClass>GLACIER</StorageClass>
</NoncurrentVersionTransition>
</Rule>
</LifecycleConfiguration>

```

Voici un exemple de PUT `/?lifecycle` demande qui ajoute la configuration de cycle de vie précédente au `examplebucket` compartiment.

```

PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:21:48 GMT
Content-MD5: 96rxH9mDqVnKkaZDddgnw==
Authorization: authorization string
Content-Length: 598

<LifecycleConfiguration>
  <Rule>
    <ID>DeleteAfterBecomingNonCurrent</ID>
    <Prefix>logs/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionExpiration>
      <NoncurrentDays>1</NoncurrentDays>
    </NoncurrentVersionExpiration>
  </Rule>
  <Rule>
    <ID>TransitionSoonAfterBecomingNonCurrent</ID>
    <Prefix>documents/</Prefix>
    <Status>Enabled</Status>
    <NoncurrentVersionTransition>
      <NoncurrentDays>0</NoncurrentDays>
      <StorageClass>GLACIER</StorageClass>
    </NoncurrentVersionTransition>
  </Rule>
</LifecycleConfiguration>

```

Voici un exemple de réponse.

```

HTTP/1.1 200 OK
x-amz-id-2: aXQ+KbIrmMmo0//3bMdDTw/CnjArwje+J49Hf+j44yRb/VmbIkgI05A+PT98Cp/6k07hf
+LD2mY=

```

```
x-amz-request-id: 02D7EC4C10381EB1
Date: Wed, 14 May 2014 02:21:50 GMT
Content-Length: 0
Server: AmazonS3
```

Exemples supplémentaires

Pour plus d'exemples de transition d'objets vers des classes de stockage telles que STANDARD_IA ou ONEZONE_IA, consultez la section Exemples de configuration du cycle de vie. <https://docs.aws.amazon.com/AmazonS3/latest/userguide/intro-lifecycle-rules.html#lifecycle-configuration-examples>

Ressources connexes

- [GetBucketLifecycleConfiguration](#)
- [POST Object restore](#)
- Par défaut, le propriétaire d'une ressource, dans ce cas, le propriétaire du compartiment, qui est Compte AWS celui qui a créé le compartiment, peut effectuer n'importe laquelle des opérations. Le propriétaire d'une ressource peut également autoriser d'autres personnes à effectuer l'opération. Pour en savoir plus, consulter les rubriques suivantes dans le Guide de l'utilisateur d'Amazon Simple Storage Service :
 - [Spécifier les autorisations dans une politique](#)
 - [Gestion des autorisations d'accès à vos ressources Amazon S3](#)

Cycle de vie du bucket GET (obsolète)

Description

Important

Pour obtenir une version mise à jour de cette API, consultez [GetBucketLifecycleConfiguration](#). Si vous avez configuré le cycle de vie d'un bucket à l'aide de `<filter>` cet élément, vous devriez voir une version mise à jour de cette rubrique. Cette rubrique est fournie à des fins de rétrocompatibilité.

Renvoie les informations `lifecycle` de configuration définies sur le bucket. Pour plus d'informations sur la configuration du cycle de vie, consultez la [section Gestion du cycle de vie des objets](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Pour utiliser cette opération, vous devez être autorisé à effectuer `s3:GetLifecycleConfiguration` action. Le propriétaire du bucket dispose de cette autorisation par défaut. Le propriétaire du bucket peut accorder cette autorisation à d'autres personnes. Pour plus d'informations sur les autorisations, consultez [la section Gestion des autorisations d'accès à vos ressources Amazon S3](#) dans le guide de l'utilisateur d'Amazon Simple Storage Service.

Requêtes

Syntaxe

```
GET /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Demandes d'authentification \(AWS Signature \(version 4\)\))
```

Paramètres de demande

Cette implémentation de l'opération n'utilise pas de paramètres de demande.

En-têtes de demande

Cette implémentation de l'opération utilise uniquement des en-têtes de requête communs à toutes les opérations. Pour plus d'informations, consultez la section En-têtes [de requête](#) courants.

Elements de la demande

Cette implémentation de l'opération n'utilise pas d'éléments de requête.

Réponses

En-têtes de réponse

Cette implémentation de l'opération utilise uniquement les en-têtes de réponse communs à la plupart des réponses. Pour plus d'informations, consultez la section En-têtes [de réponse](#) communs.

Éléments de réponse

Cette implémentation de GET renvoie les éléments de réponse suivants.

Nom	Description	Obligatoire
<code>AbortIncompleteMultipartUpload</code>	Conteneur permettant de spécifier à quel moment un téléchargement partitionné incomplet peut faire l'objet d'une opération d'abandon. Enfant : <code>DaysAfterInitiation</code> Type : Conteneur Ancêtre : <code>Rule</code>	Oui, si aucune autre action n'est spécifiée pour la règle
<code>Date</code>	Date à laquelle vous souhaitez qu'Amazon S3 entreprenne l'action. Pour plus d'informations, consultez Règles de cycle de vie : basées sur une date spécifique dans le guide de l'utilisateur d'Amazon Simple Storage Service. La valeur de date doit être conforme au format ISO 8601. L'heure indique toujours minuit UTC. Type : Chaîne	Oui, si <code>Days</code> et si vous <code>ExpiredObjectDeleteMarker</code> êtes absents

Nom	Description	Obligatoire
	Ancêtre : <code>Expiration</code> ou <code>Transition</code>	
<code>Days</code>	Spécifie le nombre de jours après la création de l'objet pendant lesquels l'action de règle spécifique prend effet. La durée d'éligibilité de l'objet est calculée comme l'heure de création + le nombre de jours, le temps obtenu étant arrondi à minuit UTC le jour suivant. Type : Non-negative entier lorsqu'il est utilisé avec <code>Transition</code> , entier positif lorsqu'il est utilisé avec <code>Expiration</code> . Ancêtre : <code>Transition</code> ou <code>Expiration</code>	Oui, si <code>Date</code> et si vous <code>ExpiredObjectDeleteMarker</code> êtes absents
<code>DaysAfterInitiation</code>	Spécifie le nombre de jours après le lancement d'un chargement en plusieurs parties pendant lesquels le téléchargement en plusieurs parties doit être terminé. S'il ne se termine pas dans le délai spécifié, il devient éligible à une opération d'abandon et Amazon S3 annule le chargement partitionné incomplet. Type : entier positif Ancêtre : <code>AbortIncompleteMultipartUpload</code>	Oui, en cas <code>Date</code> d'absence

Nom	Description	Obligatoire
Expiration	Cette action spécifie une période de la durée de vie de l'objet pendant laquelle Amazon S3 doit prendre l'action d'expiration appropriée. L'action d'expiration se produit uniquement sur les objets éligibles selon la période spécifiée dans l'enfant <code>Date</code> ou l'<code>Days</code>élément. L'action qu'Amazon S3 entreprend dépend de l'activation ou non de la gestion des versions du bucket. • Si la gestion des versions n'a jamais été activée sur le compartiment, Amazon S3 supprime définitivement la seule copie de l'objet. • Sinon, si la gestion des versions est activée dans votre compartiment (ou si la gestion des versions est suspendue), l'action s'applique uniquement à la version actuelle de l'objet. Les compartiments dont la gestion des versions est activée ou suspendue peuvent contenir plusieurs versions du même objet : une version actuelle et zéro ou plusieurs versions non actuelles. Au lieu de supprimer la version actuelle, Amazon S3 en fait une version non actuelle en ajoutant un marqueur de suppression comme nouvelle version actuelle.	Oui, si la balise parent est spécifiée

[illegible]

Nom	Description	Obligatoire
LifecycleConfiguration	Conteneur de règles de cycle de vie. Vous pouvez ajouter jusqu'à 1 000 règles. Type : Conteneur Enfants : règle Ancêtre : Aucun	Oui
ExpiredObjectDeleteMarker	Dans un compartiment versionné (compartiment dont la version est activée ou dont la version est suspendue), cet élément indique si Amazon S3 supprimera les marqueurs de suppression d'objets expirés dans le compartiment. Pour un exemple, consultez l'Exemple 8 : Spécifier une action d'expiration pour supprimer les marqueurs de suppression d'objets expirés dans le Guide de l'utilisateur d'Amazon Simple Storage Service. Type : Chaîne Valeurs valides : true false (la valeur false est autorisée mais elle n'est pas opérationnelle, Amazon S3 n'agit pas si la valeur est fausse) Ancêtre : Expiration	Oui, si Date et si vous Days êtes absents

Nom	Description	Obligatoire
<code>NoncurrentDays</code>	Spécifie le nombre de jours pendant lesquels un objet n'est pas à jour avant qu'Amazon S3 puisse effectuer l'action associée. Pour plus d'informations sur le calcul des jours non courants, consultez la section Règles de cycle de vie basées sur le nombre de jours dans le guide de l'utilisateur d'Amazon Simple Storage Service. Type : entier non négatif lorsqu'il est utilisé avec <code>NoncurrentVersionTransition</code> , entier positif lorsqu'il est utilisé avec <code>NoncurrentVersionExpiration</code> Ancêtre : <code>NoncurrentVersionExpiration</code> ou <code>NoncurrentVersionTransition</code>	Oui, uniquement si l'ancêtre est présent

Nom	Description	Obligatoire
NoncurrentVersionExpiration	Spécifie quand les versions d'objet anciennes expirent. Lorsqu'elles expirent, Amazon S3 supprime définitivement les versions d'objet anciennes. Définissez cette action de configuration du cycle de vie sur un compartiment dont la gestion des versions est activée (ou suspendue) pour demander à Amazon S3 de supprimer les versions non actuelles de l'objet à une période spécifique de la durée de vie de l'objet. Type : Conteneur Enfants : NoncurrentDays Ancêtre : Rule	Oui, si aucune autre action n'est présente dans le Rule

Nom	Description	Obligatoire
NoncurrentVersionTransition	Conteneur pour la règle de transition qui décrit le moment où des objets non courants passent à la classe de stockage STANDARD_IA ONEZONE_IA , ou à la classe de stockage. Si la gestion des versions est activée dans votre compartiment (ou si la gestion des versions est suspendue), vous pouvez configurer cette action pour demander à Amazon S3 de transférer les versions non actuelles de l'objet vers la classe de stockage à une période spécifique de la durée de vie de l'objet. Type : Conteneur Enfants : NoncurrentDays et StorageClass Ancêtre : Rule	Oui, si aucune autre action n'est présente dans le Rule
Prefix	Préfixe de clé d'objet identifiant un ou plusieurs objets auxquels la règle s'applique. Type : Chaîne Ancêtre : Rule	Oui
Rule	Conteneur pour une règle de cycle de vie. Type : Conteneur Ancêtre : LifecycleConfiguration	Oui

Nom	Description	Obligatoire
Status	Si cette option est activée, Amazon S3 exécute la règle comme prévu. Si cette option est désactivée, Amazon S3 ignore la règle. Type : Chaîne Ancêtre : Rule Valeurs valides : Activé ou Désactivé	Oui
StorageClass	Spécifie la classe de stockage Amazon S3 vers laquelle vous souhaitez transférer l'objet. Type : Chaîne Ancêtre : Transition et NoncurrentVersionTransition Valeurs valides : STANDARD_IA ONEZONE_IA	Oui

Nom	Description	Obligatoire
Transition	Cette action spécifie une période de la durée de vie des objets au cours de laquelle Amazon S3 doit les faire passer à la classe de stockage STANDARD_IA ONEZONE_IA , ou à la classe de stockage. Lorsque cette action est effective , les actions d'Amazon S3 dépendent du fait que le compartiment est activé ou non pour la gestion des versions. • Si la gestion des versions n'a jamais été activée sur le compartiment, Amazon S3 transfère la seule copie de l'objet vers la classe de stockage spécifiée. • Lorsque la gestion des versions de votre compartiment est activée (ou que la gestion des versions est suspendue), Amazon S3 transfère uniquement les versions actuelles des objets identifiés dans la règle.  Note Un bucket dont la gestion des versions est activée ou suspendue peut contenir de nombreuses versions d'un objet. Cette action n'a aucun effet sur les versions non actuelles des objets. Pour effectuer la transition d'objets non courants, vous devez utiliser cette NoncurrentVersionTransition action.	Oui, si aucune autre action n'est présente dans le Rule

Nom	Description	Obligatoire
	Type : Conteneur Enfants : jours ou date, et StorageClass Ancêtre : Rule	

Erreurs spéciales

Code d'erreur	Description	HTTP Status Code	Préfixe du code d'erreur SOAP
NoSuchLifecycleConfiguration	La configuration du cycle de vie n'existe pas.	404 – Non trouvé	Client

Pour obtenir des informations générales sur les erreurs Amazon S3 et une liste des codes d'erreur, consultez [Réponses d'erreur](#).

Exemples

Exemple 1 : récupérer une sous-ressource du cycle de vie

Cet exemple est une requête GET pour récupérer la `lifecycle` sous-ressource du compartiment spécifié, et un exemple de réponse avec la configuration du cycle de vie renvoyée.

Exemple de demande

```
GET /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Thu, 15 Nov 2012 00:17:21 GMT
Authorization: signatureValue
```

Exemple de réponse

```
HTTP/1.1 200 OK
```

```
x-amz-id-2: ITnGT1y4RyTmXa3rPi4hk1TXouTf0hccUjo0iCPjz6FnfIutBj3M7fPG1W02SEWp
x-amz-request-id: 51991C342C575321
Date: Thu, 15 Nov 2012 00:17:23 GMT
Server: AmazonS3
Content-Length: 358
```

```
<?xml version="1.0" encoding="UTF-8"?>
<LifecycleConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Rule>
    <ID>Archive and then delete rule</ID>
    <Prefix>projectdocs</Prefix>
    <Status>Enabled</Status>
    <Transition>
      <Days>30</Days>
      <StorageClass>STANDARD_IA</StorageClass>
    </Transition>
    <Transition>
      <Days>365</Days>
      <StorageClass>GLACIER</StorageClass>
    </Transition>
    <Expiration>
      <Days>3650</Days>
    </Expiration>
  </Rule>
</LifecycleConfiguration>
```

Ressources connexes

- [PutBucketLifecycleConfiguration](#)
- [DeleteBucketLifecycle](#)

Historique du document

Le tableau suivant décrit les modifications importantes apportées au guide du développeur Amazon S3. Pour recevoir les notifications de mise à jour de cette documentation, abonnez-vous à un flux RSS.

Modification	Description	Date
Guide du nouveau développeur	Tout le contenu non lié à l'API, tel que le développement avec S3, les exemples de code, les réponses aux erreurs, etc., du guide de référence de l'API Amazon S3 a été transféré dans ce nouveau guide du développeur Amazon S3. Pour plus d'informations, consultez la référence de l'API Amazon S3 .	15 juin 2026

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.