



Guía para desarrolladores

Amazon Simple Storage Service



Versión de API 2006-03-01

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

Amazon Simple Storage Service: Guía para desarrolladores

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

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas comerciales que no sean propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es Amazon S3?	1
Recomendaciones de llamada a la API	2
Realización de llamadas a la API de REST directas	2
Permisos	3
Desarrollo con Amazon S3	4
Realizar solicitudes	4
Acerca de las claves de acceso	5
Puntos de enlace de solicitud	7
Realizar solicitudes mediante IPv6	7
Realización de solicitudes con los SDK de AWS	18
Realizar solicitudes con la API REST	57
Mediante AWS CLI	70
Más información sobre el AWS CLI	71
Desarrollo con AWS SDK	71
Interfaces de programación del SDK	72
Especificación de Signature Version en la autenticación de solicitudes	73
Obtener los ID de las solicitudes de Amazon S3 para Soporte	83
Uso de AWS CLI para obtener ID de solicitudes	83
Uso de Windows PowerShell para obtener ID de solicitudes	84
Uso de eventos de datos de AWS CloudTrail para obtener ID de solicitudes	84
Uso del registro de acceso al servidor de S3 para obtener los ID de solicitudes	84
Utilización de HTTP para obtener los ID de las solicitudes	84
Utilización de un navegador web para obtener ID de solicitudes	85
Uso de los SDK de AWS para obtener los ID de solicitudes	85
Operaciones de API por objetos de S3 compatibles con las tablas de S3	90
Ejemplos de código	96
Amazon S3	102
Conceptos básicos	108
Escenarios	808
Ejemplos sin servidor	1265
Amazon S3 Control	1277
Conceptos básicos	1277
S3 Directory Buckets	1358
Conceptos básicos	1360

Escenarios	1485
Autenticación de solicitudes (AWS Signature Version 4)	1489
Métodos de autenticación	1490
Introducción a la firma de solicitudes	1491
Uso de un encabezado de Authorization	1492
Descripción general	1493
Cálculo de firma: transferencia de la carga útil en un solo fragmento	1498
Cálculo de firma: transferencia de la carga útil en varios fragmentos	1518
Cálculo de firmas: incluye los encabezados finales	1531
Uso de parámetros de consulta	1538
Cálculo de una firma	1541
Un ejemplo	1546
Ejemplo 2	1548
Ejemplos: Cálculos de firma	1548
Ejemplos de cálculos de firma mediante Java	1549
Ejemplos de cálculos de firma mediante C#	1550
Autenticación de solicitudes HTTP POST	1551
Cálculo de una firma	1554
Claves de política específica de autenticación de Signature Version 4 de Amazon S3	1555
Ejemplos de políticas de bucket con claves de condición relacionadas con Signature Version 4	1558
Cargas basadas en el navegador con POST	1562
POST Object	1563
Descripción	1563
Control de versiones	1564
Solicitudes	1564
Ejemplos	1584
Activos relacionados	1585
POST Object restore	1586
Descripción	1586
Consulta de archivos con solicitudes Select	1586
Restauración de archivos	1588
Solicitudes	1590
Respuestas	1605
Ejemplos	1607
Más información	1609

Cargas basadas en el navegador con HTTP POST	1609
Cálculo de una firma	1611
Creación de formularios HTML	1612
Declaración de formulario HTML	1613
Campos de formulario HTML	1614
Política POST	1620
Expiration	1621
Coincidencia de condiciones	1622
Condiciones	1623
Secuencia de escape de caracteres	1628
Ejemplo de carga POST	1630
Carga de un archivo en Amazon S3 con HTTP POST	1630
Cargas basadas en el navegador con AWS Amplify	1633
Uso de la biblioteca de AWS Amplify JavaScript para cargar archivos en Amazon S3	1633
Más información	1634
Encabezados de solicitudes comunes	1635
Encabezados de respuesta comunes	1640
Respuestas de error	1645
Respuestas de error de REST	1646
Lista de códigos de error	1648
Lista de códigos de error de contenido de objeto SELECT	1678
Lista de códigos de error relacionados con la replicación	1692
Lista de códigos de error relacionados con el etiquetado	1696
Lista de códigos de error relacionados con las anotaciones de Amazon S3	1697
Lista de códigos de error de Amazon S3 en Outposts	1701
Lista de códigos de error de Lente de almacenamiento de Amazon S3	1703
Lista de códigos de error de Amazon S3 Object Lambda	1713
Lista de códigos de error asíncronos de Amazon S3	1717
Lista de códigos de error de concesiones de acceso a Amazon S3	1719
Lista de códigos de error relacionados con Amazon FSx	1722
Lista de códigos de error de tablas de Amazon S3	1723
Prácticas recomendadas para los errores de Amazon S3	1724
Reintente en caso de recibir una respuesta de InternalErrors	1725
Ajustar la aplicación para errores SlowDown repetidos	1725
Aislar los errores	1725
AWSGlosario de	1727

Recursos	1728
Apéndice	1730
Apéndice: respuesta de SelectObjectContent	1731
Descripción	1731
Respuestas	1731
Activos relacionados	1741
Apéndice: objeto OPTIONS	1743
Descripción	1743
Solicitudes	1743
Respuestas	1744
Ejemplos	1746
Activos relacionados	1746
Apéndice: API de SOAP	1747
Operaciones en el servicio (API de SOAP)	1747
Operaciones en buckets (API de SOAP)	1749
Operaciones en objetos (API de SOAP)	1763
Autenticar solicitudes SOAP	1788
Configurar políticas de acceso con SOAP	1789
Elementos comunes	1791
Respuestas de error de SOAP	1791
Apéndice: autenticación de solicitudes (AWS Signature Version 2)	1793
Autenticación de solicitudes con la API de REST (AWS Signature Version 2)	1795
Firma y autenticación de solicitudes de REST (AWS Signature Version 2)	1798
Cargas basadas en el navegador con POST (AWS Signature Version 2)	1814
Apéndice: API de configuración del ciclo de vida (obsoleto)	1838
Ciclo de vida de bucket PUT (obsoleto)	1840
Ciclo de vida de bucket GET (obsoleto)	1858
Historial de revisión	1870

¿Qué es Amazon S3?

Amazon Simple Storage Service (Amazon S3) es un servicio de almacenamiento de objetos que ofrece escalabilidad, disponibilidad de datos, seguridad y rendimiento líderes del sector. Los clientes de todos los tamaños y sectores pueden utilizar Amazon S3 para almacenar y proteger cualquier cantidad de datos para diversos casos de uso, tales como lagos de datos, sitios web, aplicaciones móviles, copia de seguridad y restauración, archivado, aplicaciones empresariales, dispositivos IoT y análisis de big data. Amazon S3 proporciona funciones de gestión para que pueda optimizar, organizar y configurar el acceso a sus datos para satisfacer sus requisitos empresariales, organizativos y de conformidad específicos.

Esta guía proporciona información centrada en los desarrolladores sobre el uso de la API de REST de Amazon S3 para crear aplicaciones que almacenen y recuperen datos en la nube.

Puede utilizar cualquier conjunto de herramientas que admita HTTP para utilizar la API REST. Incluso puede utilizar un navegador para recuperar objetos, siempre y cuando se puedan leer de forma anónima.

La API REST utiliza códigos de estado y encabezados HTTP estándar, para que los conjuntos de herramientas y los navegadores estándar funcionen según lo previsto. En algunas áreas, hemos añadido una funcionalidad al HTTP (por ejemplo, añadimos encabezados para admitir el control de acceso). En estos casos, hicimos todo lo posible para añadir la nueva funcionalidad de manera que coincidiera con el estilo del uso de HTTP estándar.

Note

La compatibilidad con SOAP por HTTP está obsoleta, pero aún se encuentra disponible con HTTPS. Sin embargo, las nuevas características de Amazon S3 no serán compatibles con SOAP. Recomendamos que utilice la API de REST o los AWS SDK en el siguiente enlace:

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

Esta guía para desarrolladores incluye:

- [Encabezados de solicitudes comunes](#) y [Encabezados de respuesta comunes](#)
- [Respuestas de error](#)
- [Cargas basadas en el navegador con POST \(AWS Signature Version 4\)](#)

 Important

Lea lo siguiente sobre la autenticación y el control de acceso antes de pasar a temas específicos de la API.

Las solicitudes a Amazon S3 se pueden autenticar o pueden ser anónimas. El acceso autenticado requiere credenciales que AWS puede utilizar para autenticar las solicitudes.

Recomendaciones de llamada a la API

La realización de llamadas a la API de REST directamente desde el código puede ser complicada. Debe escribir el código necesario para calcular una firma válida para autenticar las solicitudes. En su lugar, recomendamos las siguientes alternativas:

- Utilice los [AWS SDK](#) para enviar las solicitudes.

Además, consulte el [código y las bibliotecas de ejemplo](#).

Si utiliza los SDK, no necesita escribir código para calcular una firma para la autenticación de las solicitudes, ya que los clientes del SDK autentican las solicitudes mediante las claves de acceso que usted proporciona. A menos que tenga una buena razón para no hacerlo, debería usar siempre los AWS SDK.

- Utilice la CLI de AWS para realizar llamadas a la API de Amazon S3. Para obtener información acerca de cómo configurar la CLI de AWS y ejemplos de comandos de Amazon S3, consulte los siguientes temas:

[Configure la CLI de AWS](#) en la Guía del usuario de Amazon Simple Storage Service.

[Uso de Amazon S3 con la CLI de AWS](#) en la Guía del usuario de AWS Command Line Interface.

Realización de llamadas a la API de REST directas

 Note

El encabezado de las solicitudes PUT está limitado a un tamaño de 8 KB. En el encabezado de la solicitud PUT, los metadatos definidos por el sistema están limitados a un tamaño de

2 KB. El tamaño de los metadatos definidos por el sistema se mide con la suma del número de bytes de la codificación US-ASCII en cada clave y valor.

Si desea realizar sus propias llamadas a la API de REST en lugar de utilizar una de las alternativas anteriores, debe tener en cuenta algunos aspectos.

- Para realizar llamadas directas a la API de REST desde el código, cree una firma con credenciales válidas e inclúyala en la solicitud. Para obtener información sobre varios métodos de autenticación y cálculos de firma, consulte [Autenticación de solicitudes \(AWS Signature Version 4\)](#).
- La API de REST utiliza códigos de estado y encabezados HTTP estándar, por lo que los conjuntos de herramientas y los navegadores estándar funcionan según lo previsto. En algunas áreas, hemos añadido una funcionalidad al HTTP (por ejemplo: añadimos encabezados para admitir el control de acceso). En estos casos, hicimos todo lo posible para añadir la nueva funcionalidad de manera que coincida con el estilo del uso de HTTP estándar. Para obtener más información acerca de la realización de solicitudes, consulte [Realizar solicitudes](#).

Permisos

Puede tener credenciales válidas para autenticar las solicitudes, pero a menos que tenga permisos de S3 del propietario de la cuenta o del propietario del bucket no puede crear ni acceder a recursos de Amazon S3. Por lo general, estos permisos se conceden mediante una [política](#) de AWS Identity and Access Management (IAM), como una política de bucket. Por ejemplo, debe tener permisos para crear un bucket de S3 u obtener un objeto de un bucket. Para obtener una lista de permisos de S3, consulte [Acciones, recursos y claves de condición de Amazon S3](#).

Para obtener más información sobre los permisos para las operaciones de la API de S3 por tipos de recursos de S3, consulte [Permisos necesarios para las operaciones de la API de Amazon S3](#) en la Guía del usuario de Amazon Simple Storage Service.

Si utiliza las credenciales de usuario raíz de la Cuenta de AWS, tiene todos los permisos. Sin embargo, no se recomienda el uso de las credenciales del usuario raíz. En su lugar, recomendamos que se creen roles de AWS Identity and Access Management (IAM) en la cuenta y se administren los permisos de usuario. Para obtener más información, consulte [Administración de acceso](#) en la Guía del usuario de Amazon Simple Storage Service.

Desarrollo con Amazon S3

En esta sección se tratan temas relacionados con los desarrolladores en torno al uso de Amazon S3. Para obtener más información, consulte los siguientes temas.

Temas

- [Realizar solicitudes](#)
- [Desarrollo con Amazon S3 mediante la AWS CLI](#)
- [Desarrollo con Amazon S3 mediante los SDK de AWS](#)
- [Obtención de los ID de las solicitudes de Amazon S3 para AWS Support](#)
- [Operaciones de API por objetos de Amazon S3 compatibles con las tablas de S3](#)

Realizar solicitudes

Amazon S3 es un servicio de REST. Puede enviar solicitudes a Amazon S3 con la API REST o las bibliotecas de encapsulamiento de los SDK de AWS (consulte [Código de muestra y bibliotecas](#)) que incluyen la API REST de Amazon S3 subyacente, lo que simplifica sus tareas de programación.

Toda interacción con Amazon S3 es o autenticada o anónima. La autenticación es un proceso que permite verificar la identidad del solicitante que intenta acceder a un producto de Amazon Web Services (AWS). Las solicitudes autenticadas deben incluir un valor de firma que autentique al remitente de la solicitud. El valor de firma, en parte, se genera a partir de las claves de acceso de AWS del solicitante (ID de clave de acceso y clave de acceso secreta). Para obtener más información acerca de cómo obtener claves de acceso, consulte [¿Cómo obtengo credenciales de seguridad?](#) en la Referencia general de AWS

Si utiliza el SDK de AWS, las bibliotecas computan la firma a partir de las claves que usted proporciona. Sin embargo, si realiza llamadas directas a la API REST en su aplicación, debe escribir el código para computar la firma y añadirla a la solicitud.

Temas

- [Acerca de las claves de acceso](#)
- [Puntos de enlace de solicitud](#)
- [Realización de solicitudes a Amazon S3 mediante IPv6](#)

- [Realización de solicitudes con los SDK de AWS](#)
- [Realizar solicitudes con la API REST](#)

Acerca de las claves de acceso

En las siguientes secciones se revisan los tipos de claves de acceso que puede utilizar para realizar solicitudes autenticadas.

Cuenta de AWS Claves de acceso de la

Las claves de acceso de la cuenta proporcionan acceso completo a los recursos de AWS que son propiedad de la cuenta. A continuación, se proporcionan ejemplos de claves de acceso:

- ID de clave de acceso (una cadena alfanumérica de 20 caracteres). Por ejemplo: `AKIAIOSFODNN7EXAMPLE`
- Clave de acceso secreta (una cadena de 40 caracteres). Por ejemplo: `wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY`

El ID de clave de acceso identifica una de forma exclusiva Cuenta de AWS. Puede utilizar estas claves de acceso para enviar solicitudes autenticadas a Amazon S3.

Claves de acceso de usuario de IAM

Puede crear una Cuenta de AWS para su empresa. Sin embargo, puede haber varios empleados en la organización que necesitan acceso a los recursos de AWS de la organización. Compartir sus claves de acceso de la Cuenta de AWS reduce la seguridad y crear Cuentas de AWS individuales para cada empleado puede no ser conveniente. Además, no puede compartir fácilmente recursos como buckets y objetos ya que estos pertenecen a diferentes cuentas. Para compartir recursos, debe conceder permisos, lo que implica un trabajo adicional.

En dichos casos, puede utilizar AWS Identity and Access Management (IAM) para crear usuarios en la Cuenta de AWS con sus propias claves de acceso y adjuntar políticas de usuario de IAM que concedan los permisos de acceso a recursos adecuados a esos usuarios. Para administrar mejor estos usuarios, IAM le permite crear grupos de usuarios y conceder permisos a nivel grupal que se aplican a todos los usuarios de ese grupo.

Estos usuarios se denominan usuarios de IAM, y usted los crea y administra en AWS. La cuenta principal controla la capacidad de un usuario para acceder a AWS. Cualquier recurso creado por

un usuario de IAM se encuentra bajo control de la principal, que paga dicho recurso Cuenta de AWS. Estos usuarios de IAM pueden enviar solicitudes autenticadas a Amazon S3 con sus propias credenciales de seguridad. Para obtener más información acerca de la creación y la administración de usuarios en su Cuenta de AWS, consulte la [página de detalles del producto de AWS Identity and Access Management](#).

Credenciales de seguridad temporales

Además de crear usuarios de IAM con sus propias claves de acceso, IAM también permite otorgar credenciales de seguridad temporales (claves de acceso temporales y un token de seguridad) a cualquier usuario de IAM para que tenga acceso a los servicios y recursos de AWS. También puede administrar usuarios en su sistema que no pertenecen a AWS. Estos se conocen como usuarios federados. Además, los usuarios pueden ser aplicaciones que usted crea para acceder a sus recursos de AWS.

IAM proporciona la API de AWS Security Token Service para que pueda solicitar credenciales de seguridad temporales. Para solicitar estas credenciales, puede utilizar la API de STS de AWS o el SDK de AWS. La API devuelve credenciales de seguridad temporales (ID de clave de acceso y clave de acceso secreta) y un token de seguridad. Estas credenciales son válidas solo durante el tiempo que usted especifica cuando las solicita. Usted utiliza el ID de clave de acceso y la clave secreta de la misma manera que los utiliza cuando envía solicitudes con sus claves de acceso de usuario de IAM o su Cuenta de AWS. Además, debe incluir el token en cada solicitud que envía a Amazon S3.

Un usuario de IAM puede solicitar estas credenciales de seguridad temporales para uso propio o puede entregarlas a usuarios federados o aplicaciones. Cuando solicita credenciales de seguridad temporales para usuarios federados, debe proporcionar un nombre de usuario y una política de IAM que defina los permisos que desea asociar a estas credenciales de seguridad temporales. El usuario federado no puede tener más permisos que el usuario de la cuenta principal de IAM que solicitó las credenciales temporales.

Puede utilizar estas credenciales de seguridad temporales para realizar solicitudes a Amazon S3. Las bibliotecas de la API computan el valor de firma necesario con esas credenciales para autenticar su solicitud. Si envía las solicitudes con las credenciales vencidas, Amazon S3 deniega la solicitud.

Para obtener información acerca de cómo firmar solicitudes con credenciales de seguridad temporales en sus solicitudes de API REST, consulte [Firma y autenticación de solicitudes de REST \(AWS Signature Version 2\)](#). Para obtener más información acerca del envío de solicitudes con los SDK de AWS, consulte [Realización de solicitudes con los SDK de AWS](#).

Para obtener más información acerca de la compatibilidad de IAM con las credenciales de seguridad temporales, consulte [Credenciales de seguridad temporales](#) en la guía de usuario de IAM.

Para más seguridad, puede solicitar la Multifactor Authentication (MFA, Autenticación multifactor) cuando accede a sus recursos de Amazon S3 mediante la configuración de una política de bucket. Para obtener más información, consulte [Políticas de bucket de ejemplo: requerir MFA](#). Después de solicitar la MFA para acceder a sus recursos de Amazon S3, la única manera de acceder a estos recursos es proporcionando credenciales temporales que se crean con una clave de MFA. Para obtener más información, consulte la página de detalles de [Autenticación multifactor de AWS](#) y [Configuración del acceso a una API protegida por MFA](#) en la Guía del usuario de IAM.

Puntos de enlace de solicitud

Usted envía solicitudes REST al punto de enlace predefinido del servicio. Para ver una lista de todos los servicios de AWS y los puntos conexión correspondientes, consulte [Regiones y puntos de conexión](#) en la Referencia general de AWS.

Realización de solicitudes a Amazon S3 mediante IPv6

Amazon Simple Storage Service (Amazon S3) admite la capacidad de acceder a los buckets de S3 mediante el Protocolo de Internet, versión 6 (IPv6), además del protocolo IPv4. Los puntos de enlace de doble pila de Amazon S3; admiten solicitudes a buckets de S3 a través de IPv6 y de IPv4. No hay cargos adicionales por acceder a Amazon S3 mediante IPv6. Para obtener más información acerca de los precios, consulte [Precios de Amazon S3](#).

Note

La siguiente información corresponde a los puntos de conexión IPv6 públicos de Amazon S3. Para obtener información sobre los puntos de conexión de la nube privada virtual (VPC) de AWS, consulte [AWS PrivateLink para Amazon S3](#).

Temas

- [Introducción a la realización de solicitudes en IPv6](#)
- [Uso de direcciones IPv6 en políticas de IAM](#)
- [Probar la compatibilidad de dirección IP](#)
- [Uso de puntos de conexión de doble pila en Amazon S3](#)

Introducción a la realización de solicitudes en IPv6

Para hacer una solicitud a un bucket de S3 mediante IPv6, debe utilizar un punto de enlace de doble pila. En la siguiente sección se describe cómo hacer solicitudes mediante IPv6 con los puntos de enlace de doble pila.

A continuación se describen algunos puntos que debe tener en cuenta antes de intentar acceder a un bucket mediante IPv6:

- El cliente y la red que acceden al bucket deben estar autorizados para utilizar IPv6.
- Se admiten tanto solicitudes de estilo alojamiento virtual como de tipo ruta para el acceso a IPv6. Para obtener más información, consulte [Puntos de conexión de doble pila en Amazon S3](#).
- Si utiliza el filtrado de dirección IP de origen en sus políticas de bucket o de usuario de AWS Identity and Access Management (IAM), debe actualizar las políticas para incluir los rangos de direcciones IPv6. Para obtener más información, consulte [Uso de direcciones IPv6 en políticas de IAM](#).
- Cuando utiliza IPv6, los archivos de registro de acceso al servidor producen direcciones IP en un formato de IPv6. Debe actualizar el software, las herramientas y los scripts existentes que utiliza para analizar archivos de registro de Amazon S3 para que puedan analizar las direcciones Remote IP con formato de IPv6. Para obtener más información, consulte [Registro de solicitudes con registro de acceso al servidor](#).

Note

Si experimenta problemas relacionados con la presencia de direcciones IPv6 en archivos de registro, contacte con [AWS Support](#).

Realización de solicitudes mediante IPv6 con puntos de enlace de doble pila

Puede realizar solicitudes con llamadas a la API de Amazon S3 mediante IPv6 con los puntos de enlace de doble pila. Las operaciones de la API de Amazon S3 funcionan de la misma manera que cuando accede a Amazon S3 mediante IPv6 o IPv4. El desempeño también debe ser el mismo.

Cuando utiliza la API REST, accede directamente al punto de enlace de doble stack. Para obtener más información, consulte [Puntos de enlace de doble pila](#).

Al usar AWS Command Line Interface (AWS CLI) y los SDK de AWS, puede utilizar un parámetro o una marca para cambiar a un punto de enlace de doble pila. También puede especificar el punto de enlace de doble pila directamente como una anulación del punto de enlace de Amazon S3 en el archivo de configuración.

Puede utilizar el punto de enlace de doble pila para acceder a un bucket mediante IPv6 desde cualquiera de las siguientes opciones:

- La AWS CLI, consulte [Usar puntos de enlace de doble pila desde la AWS CLI](#).
- Para los SDK de AWS, consulte [Uso de los puntos de enlace de doble pila de los SDK de AWS](#).
- La API REST, consulte [Realizar solicitudes a los puntos de enlace de doble pila con la API REST](#).

Características no disponibles con IPv6

Actualmente no se admite la siguiente característica cuando se accede a un bucket de S3 a través de IPv6: alojamiento de sitios web estáticos desde un bucket de S3.

Uso de direcciones IPv6 en políticas de IAM

Antes de intentar acceder a un bucket mediante IPv6, debe asegurarse de que las políticas de usuario de IAM o de bucket de S3 utilizadas para el filtrado de direcciones IP estén actualizadas para incluir los rangos de direcciones IPv6. Si no se actualizan la políticas de filtrado de direcciones IP para gestionar direcciones IPv6, los clientes pueden perder u obtener incorrectamente el acceso al bucket cuando comienzan a utilizar IPv6. Para obtener más información sobre administración de permisos de acceso con IAM, consulte [Administración de identidades y accesos para Amazon S3](#).

Las políticas de IAM que filtran direcciones IP utilizan [operadores de condición de dirección IP](#). La siguiente política de buckets identifica el rango 54.240.143.* de direcciones IPv4 permitidas mediante operadores de condición de dirección IP. Cualquier dirección IP fuera de este rango no podrá acceder al bucket (examplebucket). Dado que todas las direcciones IPv6 están fuera del rango permitido, esta política evita que las direcciones IPv6 puedan acceder a 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"}
    }
  }
]
}

```

Puede modificar el elemento `Condition` de la política de bucket para permitir los rangos de direcciones IPv4 (54.240.143.0/24) e IPv6 (2001:DB8:1234:5678::/64) como se muestra en el siguiente ejemplo. Puede utilizar el mismo tipo de bloque `Condition` que se muestra en el ejemplo para actualizar las políticas de bucket y de usuario de IAM.

```

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

```

Antes de utilizar IPv6, debe actualizar todas las políticas de bucket y de usuario de IAM; relevantes que utilizan filtrado de direcciones IP. No recomendamos utilizar el filtrado de direcciones IP en políticas de bucket.

Puede revisar sus políticas de usuario de IAM en la consola de IAM en <https://console.aws.amazon.com/iam/>. Para obtener más información acerca de IAM, consulte la [guía del usuario de IAM](#). Para obtener información acerca de cómo editar políticas de bucket de S3, consulte [Adición de una política de bucket](#).

Probar la compatibilidad de dirección IP

Si utiliza Linux/Unix o Mac OS X, puede probar si tiene acceso a un punto de enlace de doble pila mediante IPv6 con el comando `curl` como se muestra en el siguiente ejemplo:

Example

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

Usted obtiene información similar a la del siguiente ejemplo. Si está conectado mediante IPv6, la dirección IP conectada será una dirección 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 utiliza Microsoft 7 o 10, puede probar si tiene acceso a un punto de conexión de doble pila mediante IPv6 o IPv4 con el comando ping como se muestra en el siguiente ejemplo.

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

Uso de puntos de conexión de doble pila en Amazon S3

Los puntos de enlace de doble pila de Amazon S3; admiten solicitudes a buckets de S3 a través de IPv6 y de IPv4. En esta sección se describe cómo utilizar los puntos de enlace de doble pila.

Temas

- [Puntos de conexión de doble pila en Amazon S3](#)
- [Usar puntos de enlace de doble pila desde la AWS CLI](#)
- [Uso de los puntos de enlace de doble pila de los SDK de AWS](#)
- [Usar los puntos de enlace de doble pila desde la API REST](#)

Puntos de conexión de doble pila en Amazon S3

Cuando realiza una solicitud a un punto de enlace de doble pila, la URL del bucket resuelve a una dirección IPv6 o IPv4. Para obtener más información acerca de cómo obtener acceso a un bucket mediante IPv6, consulte [Realización de solicitudes a Amazon S3 mediante IPv6](#).

Cuando utiliza la API REST, accede directamente a un punto de enlace de Amazon S3 mediante el nombre del punto de enlace (URI). Puede obtener acceso a un bucket de S3 mediante un punto de conexión de doble pila con un nombre de punto de conexión de estilo alojado virtual o de estilo ruta. Amazon S3 solo es compatible con nombres de puntos de enlace de doble pila regionales, lo que implica que debe especificar la región como parte del nombre.

Use las siguientes convenciones de nomenclatura para los nombres de punto de conexión de doble pila de estilo alojado virtual y de estilo de ruta

- punto de conexión de doble pila de estilo alojado virtual:

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

- punto de conexión de doble pila de estilo de ruta:

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

Para obtener más información acerca de los estilos de denominación de los puntos de conexión, consulte [Acceso y muestra de un bucket de Amazon S3](#). Para ver una lista de los puntos de conexión de Amazon S3, consulte [Regiones y puntos de conexión](#) en la Referencia general de AWS.

Important

Puede usar transfer acceleration con los puntos de enlace de doble pila. Para obtener más información, consulte [Introducción a Aceleración de transferencia de Amazon S3](#).

Note

Los dos tipos de puntos de conexión de nube privada virtual (VPC) que acceden a Amazon S3 (Puntos de conexión de VPC de interfaz y Puntos de conexión de VPC de puerta de enlace) ahora son compatibles con la doble pila. Para obtener más información acerca de los puntos de conexión de VPC de Amazon S3, consulte [AWS PrivateLink para Amazon S3](#).

Al usar AWS Command Line Interface (AWS CLI) y los SDK de AWS, puede utilizar un parámetro o una marca para cambiar a un punto de enlace de doble pila. También puede especificar el punto

de enlace de doble pila directamente como una anulación del punto de enlace de Amazon S3 en el archivo de configuración. En las secciones siguientes, se describe cómo utilizar los puntos de enlace de doble pila desde la AWS CLI y los SDK de AWS.

Usar puntos de enlace de doble pila desde la AWS CLI

Esta sección proporciona ejemplos de comandos de la AWS CLI, que se usan para realizar solicitudes a un punto de conexión de doble pila. Para obtener instrucciones acerca de cómo configurar la AWS CLI, consulte [Desarrollo con Amazon S3 mediante la AWS CLI](#).

Puede establecer el valor de configuración `use_dualstack_endpoint` a `true` en un perfil de su archivo de AWS Config para dirigir todas las solicitudes de Amazon S3 que realicen los comandos `s3` y `s3api` de la AWS CLI al punto de enlace de doble pila para la región especificada. Puede especificar la región en el archivo de configuración o en un comando utilizando la opción `--region`.

Cuando utilice puntos de enlace de doble pila con la AWS CLI, puede utilizar los estilos de direccionamiento `path` y `virtual`. El estilo de direccionamiento configurado en el archivo de configuración controla si el nombre del bucket está en el hostname o es parte de la URL. La CLI intentará, de manera predeterminada, usar el estilo virtual siempre que sea posible, pero si es necesario recurrirá al estilo de ruta. Para obtener más información, consulte [Configuración de la AWS CLI de Amazon S3](#).

También puede realizar cambios a la configuración mediante un comando, como se muestra en el siguiente ejemplo, en el que se configura `use_dualstack_endpoint` como `true` y `addressing_style` como `virtual` en el perfil predeterminado.

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

Si quiere utilizar un punto de conexión de doble pila exclusivamente para comandos de la AWS CLI específicos (no para todos los comandos), puede usar uno de los métodos siguientes:

- Puede utilizar el punto de conexión de doble pila por cada comando configurando el parámetro `--endpoint-url` como `https://s3.dualstack.aws-region.amazonaws.com` o `http://s3.dualstack.aws-region.amazonaws.com` para cualquier comando `s3` o `s3api`.

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

- Puede configurar perfiles separados en su archivo de AWS Config. Por ejemplo, puede crear un perfil que configure `use_dualstack_endpoint` como `true` y un perfil que no configure `use_dualstack_endpoint`. Al ejecutar un comando, especifique qué perfil quiere usar, en función de si quiere usar el punto de conexión de doble pila o no.

Note

Por el momento, al usar la AWS CLI no podrá utilizar la transfer acceleration con puntos de enlace de doble pila. Sin embargo, pronto ofreceremos compatibilidad con la AWS CLI. Para obtener más información, consulte [Habilitación y uso de Aceleración de transferencias de S3](#).

Uso de los puntos de enlace de doble pila de los SDK de AWS

En esta sección, se proporcionan ejemplos de cómo obtener acceso a un punto de enlace de doble pila con los SDK de AWS.

AWS SDK para JavaEjemplo de punto de enlace de doble pila con

En el siguiente ejemplo, se muestra cómo habilitar puntos de enlace de doble pila al crear un cliente de Amazon S3 mediante AWS SDK para Java.

Para obtener instrucciones sobre cómo crear y probar una muestra funcional de Java, consulte [Introducción](#) en la Guía del desarrollador de AWS SDK para Java.

```
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 está utilizando AWS SDK para Java en Windows, puede que tenga que configurar adecuadamente la siguiente propiedad de la máquina virtual Java (JVM):

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

AWSEjemplo de punto de enlace de doble pila con el SDK para .NET

Cuando utiliza el SDK de AWS para .NET, puede usar la clase `AmazonS3Config` para habilitar el uso de un punto de enlace de doble pila, como se muestra en el siguiente ejemplo.

```

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());
        }
    }
}
```

Para obtener información acerca de cómo configurar y ejecutar ejemplos de código, consulte [Introducción al SDK de AWS para .NET](#) en la Guía para desarrolladores del SDK de AWS para .NET.

Usar los puntos de enlace de doble pila desde la API REST

Para obtener información sobre cómo realizar solicitudes a puntos de enlace de doble pila con la API REST, consulte [Realizar solicitudes a los puntos de enlace de doble pila con la API REST](#).

Realización de solicitudes con los SDK de AWS

Temas

- [Realización de solicitudes con las credenciales de usuario de IAM o Cuenta de AWS](#)
- [Realización de solicitudes con las credenciales temporales de usuario de IAM](#)
- [Realizar solicitudes con credenciales temporales de usuario federado](#)

Puede enviar solicitudes autenticadas a Amazon S3 con los SDK de AWS o mediante las llamadas a la API REST directamente en su aplicación. La API del SDK de AWS utiliza las credenciales que proporciona para computar la firma para autenticación. Si usa la API REST directamente en su aplicación, debe escribir el código necesario para computar la firma a fin de autenticar su solicitud. Para ver una lista de los SDK de AWS disponibles, consulte [Código de muestra y bibliotecas](#).

Realización de solicitudes con las credenciales de usuario de IAM o Cuenta de AWS

Puede usar sus credenciales de seguridad de usuario de IAM o su Cuenta de AWS para enviar solicitudes autenticadas a Amazon S3. En esta sección se proporcionan ejemplos de cómo enviar solicitudes autenticadas con AWS SDK para Java, AWS SDK para .NET y AWS SDK para PHP. Para ver una lista de los SDK de AWS disponibles, consulte [Código de muestra y bibliotecas](#).

Cada uno de estos SDK de AWS utiliza una cadena de proveedores de credenciales específica del SDK para encontrar y utilizar las credenciales, y realizar acciones en nombre del propietario de las credenciales. Todas estas cadenas de proveedores de credenciales tienen en común que todas buscan el archivo de credenciales local de AWS.

Para obtener más información, consulte los siguientes temas:

Temas

- [Para crear un archivo local de credenciales de AWS](#)
- [Envío de solicitudes autenticadas mediante los SDK de AWS](#)
- [Recursos relacionados](#)

Para crear un archivo local de credenciales de AWS

La forma más sencilla de configurar credenciales para los SDK de AWS es utilizar un archivo de credenciales de AWS. Si utiliza AWS Command Line Interface (AWS CLI), es posible que ya

tenga configurado un archivo local de credenciales de AWS. De lo contrario, utilice el siguiente procedimiento para configurar un archivo de credenciales:

1. Inicie sesión en la Consola de administración de AWS y abra la consola de IAM en <https://console.aws.amazon.com/iam/>.
2. Cree un usuario con permisos limitados a los servicios y acciones a los que desea que tenga acceso el código. Para obtener más información acerca de cómo crear un usuario nuevo, consulte [Creación de usuarios de IAM \(consola\)](#) y siga las instrucciones a través del paso 8.
3. Elija Download .csv (Descargar .csv) para guardar una copia local de sus credenciales de AWS.
4. En su equipo, vaya al directorio principal y cree un directorio `.aws`. En sistemas basados en Unix, como Linux u OS X, se encuentra en la siguiente ubicación:

```
~/ .aws
```

En Windows, se encuentra en la siguiente ubicación:

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

5. En el directorio `.aws`, cree un archivo denominado `credentials`.
6. Abra el archivo `.csv` de credenciales que descargó de la consola de IAM y copie su contenido en el archivo de `credentials` con el siguiente formato:

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

7. Guarde el archivo `credentials` y elimine el archivo `.csv` que descargó en el paso 3.

El archivo de credenciales compartidas ahora está configurado en el equipo local y está listo para su uso con los SDK de AWS.

Envío de solicitudes autenticadas mediante los SDK de AWS

Utilice los SDK de AWS para enviar solicitudes autenticadas. Para obtener más información sobre el envío de solicitudes autenticadas, consulte [Credenciales de seguridad de AWS](#) o [IAM Identity Center Authentication](#) (Autenticación del Centro de identidades de IAM).

Java

Para obtener información sobre la autenticación de solicitudes mediante el AWS SDK para Java, consulte [Uso de archivos de configuración y credenciales compartidos para configurar globalmente los AWS SDK y las herramientas](#) y [Autenticación y acceso mediante AWS SDK y herramientas](#) en la Guía de referencia de AWS SDK y herramientas.

.NET

Para enviar solicitudes autenticadas mediante sus credenciales de usuario de IAM o su Cuenta de AWS:

- Cree una instancia de la clase `AmazonS3Client`.
- Ejecute uno de los métodos de `AmazonS3Client` para enviar solicitudes a Amazon S3. El cliente generará la firma necesaria a partir de las credenciales que se le proporcionan y la incluirá en la solicitud que envíe a Amazon S3.

Para obtener más información, consulte [Realización de solicitudes con las credenciales de usuario de IAM o de una cuenta de AWS](#).

Note

- Puede crear el cliente `AmazonS3Client` sin facilitar sus credenciales de seguridad. Las solicitudes enviadas con este cliente son solicitudes anónimas y no tienen firma. Amazon S3 devuelve un error si envía solicitudes anónimas para un recurso que no esté disponible públicamente.
- Puede crear una Cuenta de AWS y crear los usuarios necesarios. También puede administrar las credenciales para esos usuarios. Necesita estas credenciales para realizar la tarea del siguiente ejemplo. Para obtener más información, consulte [Configuración de credenciales de AWS](#) en la Guía para desarrolladores de SDK para .NET.

A continuación, también puede configurar la aplicación para recuperar perfiles y credenciales de forma activa y luego usar explícitamente dichas credenciales al crear un cliente del servicio de AWS. Para obtener más información, consulte [Acceso a credenciales y perfiles en una aplicación](#) en la Guía para desarrolladores de SDK para .NET.

En el siguiente ejemplo de código C# se muestra cómo realizar las tareas anteriores. Para obtener información acerca de cómo configurar y ejecutar ejemplos de código, consulte [Introducción al SDK de AWS para .NET](#) en la Guía para desarrolladores del SDK de AWS para .NET.

Example

```
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

En esta sección, se explica el uso de una clase de la versión 3 de AWS SDK para PHP para enviar solicitudes autenticadas con las credenciales de usuario de IAM o su Cuenta de AWS. Para obtener más información acerca de la API del SDK de AWS para Ruby, consulte [SDK de AWS para Ruby, versión 2](#).

El siguiente ejemplo de PHP muestra cómo el cliente realiza una solicitud con las credenciales de seguridad para enumerar todos los buckets de la cuenta.

Example

```
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

Puede crear el cliente `S3Client` sin facilitar sus credenciales de seguridad. Las solicitudes enviadas con este cliente son solicitudes anónimas y no tienen firma.

Amazon S3 devuelve un error si envía solicitudes anónimas para un recurso que no esté disponible públicamente. Para obtener más información, consulte [Creación de clientes anónimos](#) en la [Documentación de AWS SDK para PHP](#).

Ruby

A fin de poder utilizar la versión 3 de AWS SDK para Ruby para hacer llamadas a Amazon S3, debe establecer las credenciales de acceso de AWS que utiliza el SDK para comprobar el acceso a los buckets y los objetos. Si ha configurado las credenciales compartidas en el perfil de credenciales de AWS de su sistema local, la versión 3 del SDK para Ruby puede utilizarlas, lo que le evitará tener que declararlas en el código. Para obtener más información sobre la configuración de credenciales compartidas, consulte [Realización de solicitudes con las credenciales de usuario de IAM o de una cuenta de AWS](#).

En el siguiente fragmento de código de Ruby, se utilizan las credenciales de un archivo de credenciales de AWS compartido en un equipo local para autenticar una solicitud para obtener todos los nombres de claves de objeto de un bucket específico. Hace lo siguiente:

1. Crea una instancia de la clase `Aws::S3::Client`.
2. Realiza una solicitud a Amazon S3 enumerando los objetos de un bucket con el método `list_objects_v2` de `Aws::S3::Client`. El cliente genera el valor de firma necesario a partir de las credenciales del archivo de credenciales de AWS del equipo y lo incluye en la solicitud que envía a Amazon S3.
3. Imprime la matriz de nombres de claves de objeto en el 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__

```

Aunque no tenga un archivo de credenciales de AWS local, puede crear el recurso `Aws::S3::Client` y ejecutar código en los buckets y los objetos de Amazon S3. Las solicitudes que se envían utilizando la versión 3 del SDK para Ruby son anónimas; de forma predeterminada, no tienen firma. Amazon S3 devuelve un error si envía solicitudes anónimas para un recurso que no esté disponible públicamente.

Puede utilizar y ampliar el fragmento de código anterior para aplicaciones del SDK para Ruby como se muestra en el siguiente ejemplo, que es más robusto.

```
# 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

En el siguiente ejemplo, se utilizan credenciales de AWS cargadas automáticamente por el SDK para Go desde el archivo de credenciales compartido.

```
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)
    }
}
}
```

Recursos relacionados

- [Desarrollo con Amazon S3 mediante los SDK de AWS](#)
- [AWS SDK para PHP para la clase Aws\S3\S3Client de Amazon S](#)
- [AWS SDK para PHP Documentación](#)

Realización de solicitudes con las credenciales temporales de usuario de IAM

Una Cuenta de AWS o un usuario de IAM puede solicitar las credenciales de seguridad temporales y utilizarlas para enviar solicitudes autenticadas a Amazon S3. En esta sección, se proporcionan ejemplos sobre cómo utilizar AWS SDK para Java, .NET y PHP para obtener las credenciales de seguridad temporales y utilizarlas con el fin de autenticar las solicitudes para Amazon S3.

Java

Un usuario de IAM o una Cuenta de AWS pueden solicitar credenciales de seguridad temporales (consulte [Realizar solicitudes](#)) mediante AWS SDK para Java y utilizarlas para obtener acceso a Amazon S3. Estas credenciales caducan cuando termine la sesión especificada.

De forma predeterminada, la sesión durará una hora. Si utiliza credenciales de usuario de IAM, puede especificar la duración al solicitar las credenciales de seguridad temporales desde 15 minutos hasta la duración máxima de sesión del rol. Para obtener más información acerca de las credenciales de seguridad temporales, consulte [Credenciales de seguridad temporales](#) en la guía del usuario de IAM. Para obtener más información acerca de la realización de solicitudes, consulte [Realizar solicitudes](#).

Para obtener credenciales de seguridad temporales y acceso a Amazon S3

1. Cree una instancia de la clase `AWSecurityTokenService`.
2. Recupere las credenciales de seguridad temporales para el rol deseado al llamar al método `assumeRole()` del cliente de Security Token Service (STS).
3. Empaque las credenciales de seguridad temporales en un objeto `BasicSessionCredentials`. Utilice este objeto para proporcionar las credenciales de seguridad temporales al cliente de Amazon S3.
4. Cree una instancia de la clase `AmazonS3Client` usando las credenciales de seguridad temporales. Envíe las solicitudes a Amazon S3 con este cliente. Si envía las solicitudes con las credenciales vencidas, Amazon S3 devolverá un error.

Note

Si obtiene las credenciales de seguridad temporales con las credenciales de seguridad de su Cuenta de AWS, las credenciales de seguridad temporales tendrán validez durante

una hora únicamente. Puede especificar la duración de la sesión solo si utiliza las credenciales del usuario de IAM para solicitar una sesión.

En el siguiente ejemplo se muestra un conjunto de claves de objeto del bucket especificado. En el ejemplo se obtienen credenciales de seguridad temporales para una sesión y se utilizan para enviar una solicitud autenticada a Amazon S3.

Si desea probar el ejemplo con las credenciales de usuario de IAM, debe crear un usuario de IAM en la Cuenta de AWS. Para obtener más información acerca de cómo crear un usuario de IAM, consulte [Creación del primer grupo de usuarios y administradores de IAM](#) en la Guía del usuario de IAM.

Para obtener instrucciones sobre cómo crear y probar una muestra funcional, consulte [Introducción](#) en la Guía del desarrollador de AWS SDK para Java.

```
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 usuario de IAM o una Cuenta de AWS pueden solicitar credenciales de seguridad temporales mediante AWS SDK para .NET y utilizarlas para obtener acceso a Amazon S3. Estas credenciales caducan cuando termine la sesión.

De forma predeterminada, la sesión durará una hora. Si utiliza credenciales de usuario de IAM, puede especificar la duración al solicitar las credenciales de seguridad temporales desde 15 minutos hasta la duración máxima de sesión del rol. Para obtener más información acerca de las credenciales de seguridad temporales, consulte [Credenciales de seguridad temporales](#) en la guía del usuario de IAM. Para obtener más información acerca de la realización de solicitudes, consulte [Realizar solicitudes](#).

Para obtener credenciales de seguridad temporales y acceso a Amazon S3

1. Cree una instancia del cliente de AWS Security Token Service.
`AmazonSecurityTokenServiceClient`.
2. Inicie una sesión utilizando el método `GetSessionToken` del cliente de STS que creó en el paso anterior. Proporcione información acerca de la sesión para este método con un objeto `GetSessionTokenRequest`.

El método devuelve las credenciales de seguridad temporales.

3. Empaquete las credenciales de seguridad temporales en una instancia del objeto `SessionAWSCredentials`. Utilice este objeto para proporcionar las credenciales de seguridad temporales al cliente de Amazon S3.
4. Cree una instancia de la clase `AmazonS3Client` al proporcionar las credenciales de seguridad temporales. Envíe las solicitudes a Amazon S3 con este cliente. Si envía las solicitudes con las credenciales vencidas, Amazon S3 devolverá un error.

 Note

Si obtiene las credenciales de seguridad temporales con las credenciales de seguridad de su Cuenta de AWS, esas credenciales tendrán validez durante una hora únicamente. Puede especificar la duración de una sesión solo si utiliza las credenciales de usuario de IAM para solicitar una sesión.

En el siguiente ejemplo de código C# se muestran las claves de objeto del bucket especificado. A modo de ilustración, en el ejemplo se obtienen las credenciales de seguridad temporales para una sesión predeterminada de una hora y se utilizan para enviar una solicitud autenticada a Amazon S3.

Si desea probar el ejemplo con las credenciales de usuario de IAM, debe crear un usuario de IAM en la Cuenta de AWS. Para obtener más información acerca de cómo crear un usuario de IAM, consulte [Creación del primer grupo de usuarios y administradores de IAM](#) en la Guía del usuario de IAM. Para obtener más información acerca de la realización de solicitudes, consulte [Realizar solicitudes](#).

Para obtener información acerca de cómo configurar y ejecutar ejemplos de código, consulte [Introducción al SDK de AWS para .NET](#) en la Guía para desarrolladores del SDK de AWS para .NET.

```
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

Para obtener más información acerca de la API del SDK de AWS para Ruby, consulte [SDK de AWS para Ruby, versión 2](#).

Tanto un usuario de IAM como una Cuenta de AWS pueden solicitar credenciales de seguridad temporales mediante la versión 3 de AWS SDK para PHP. A continuación, puede usar las credenciales temporales para acceder a Amazon S3. Las credenciales caducan cuando finaliza la duración de la sesión.

De forma predeterminada, la sesión durará una hora. Si utiliza credenciales de usuario de IAM, puede especificar la duración al solicitar las credenciales de seguridad temporales desde 15 minutos hasta la duración máxima de sesión del rol. Para obtener más información acerca de las credenciales de seguridad temporales, consulte [Credenciales de seguridad temporales](#) en la guía del usuario de IAM. Para obtener más información acerca de la realización de solicitudes, consulte [Realizar solicitudes](#).

 Note

Si obtiene las credenciales de seguridad temporales con las credenciales de seguridad de su Cuenta de AWS, estas tendrán validez durante una hora únicamente. Puede especificar la duración de la sesión solo si utiliza las credenciales del usuario de IAM para solicitar una sesión.

Example

En el siguiente ejemplo de PHP se muestran las claves de objeto del bucket especificado con las credenciales de seguridad temporales. En el ejemplo se obtienen credenciales de seguridad temporales para una sesión predeterminada de una hora y se utilizan para enviar una solicitud autenticada a Amazon S3. Para obtener más información acerca de la API del SDK de AWS para Ruby, consulte [SDK de AWS para Ruby, versión 2](#).

Si desea probar el ejemplo con las credenciales de usuario de IAM, debe crear un usuario de IAM en la Cuenta de AWS. Para obtener información acerca de cómo crear un usuario de IAM, consulte [Creación del primer grupo de usuarios y administradores de IAM](#) en la Guía del usuario de IAM. Para obtener ejemplos de configuración de la duración de la sesión con las credenciales de usuario de IAM para solicitar una sesión, consulte [Realización de solicitudes con las credenciales temporales de usuario de 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 usuario de IAM o una Cuenta de AWS pueden solicitar credenciales de seguridad temporales mediante AWS SDK para Ruby y utilizarlas para obtener acceso a Amazon S3. Estas credenciales caducan cuando termine la sesión.

De forma predeterminada, la sesión durará una hora. Si utiliza credenciales de usuario de IAM, puede especificar la duración al solicitar las credenciales de seguridad temporales desde 15 minutos hasta la duración máxima de sesión del rol. Para obtener más información acerca de las credenciales de seguridad temporales, consulte [Credenciales de seguridad temporales](#) en la guía del usuario de IAM. Para obtener más información acerca de la realización de solicitudes, consulte [Realizar solicitudes](#).

 Note

Si obtiene las credenciales de seguridad temporales con las credenciales de seguridad de su Cuenta de AWS, estas tendrán validez durante una hora únicamente. Puede especificar la duración de la sesión solo si utiliza las credenciales del usuario de IAM para solicitar una sesión.

El siguiente ejemplo de código Ruby crea un usuario temporal para mostrar los elementos en un bucket especificado durante una hora. Para utilizar este ejemplo, debe tener credenciales de AWS con los permisos necesarios para crear nuevos clientes de AWS Security Token Service (AWS STS) y mostrar buckets de 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
```

Realizar solicitudes con credenciales temporales de usuario federado

Puede solicitar credenciales de seguridad temporales y proporcionarlas a sus usuarios federados o aplicaciones que necesitan acceso a sus recursos de AWS. En esta sección, se proporcionan ejemplos de cómo puede utilizar el SDK de AWS para obtener credenciales de seguridad temporales para sus usuarios federados o aplicaciones y enviar solicitudes autenticadas a Amazon S3 con esas credenciales. Para ver una lista de los SDK de AWS disponibles, consulte [Código de muestra y bibliotecas](#).

Note

Tanto la Cuenta de AWS como un usuario de IAM pueden solicitar credenciales de seguridad temporales para usuarios federados. Sin embargo, para mayor seguridad, solo un usuario de IAM con los permisos necesarios debe solicitar estas credenciales temporales para asegurarse de que el usuario federado obtenga al menos los permisos del usuario de IAM que realiza la solicitud. En algunas aplicaciones, puede resultarle útil crear un usuario de IAM con permisos específicos con el único fin de proporcionar credenciales de seguridad temporales a sus usuarios federados y aplicaciones.

Java

Puede proporcionar credenciales de seguridad temporales para sus usuarios federados y aplicaciones, de modo que estos puedan enviar solicitudes autenticadas a fin de obtener acceso a sus recursos de AWS. Al solicitar estas credenciales temporales, debe proporcionar un nombre de usuario y una política de IAM que describa los permisos de recursos que desea otorgar. De forma predeterminada, la sesión durará una hora. Puede solicitar las credenciales de seguridad temporales para usuarios federados y aplicaciones para establecer un valor de duración distinto de forma explícita.

Note

Para mayor seguridad, cuando solicite las credenciales de seguridad temporales para usuarios federados y aplicaciones, recomendamos que use un usuario de IAM específico que solo tenga los permisos de acceso necesarios. El usuario temporal creado nunca puede obtener más permisos que el usuario de IAM que solicitó las credenciales de seguridad temporales. Para obtener más información, consulte [Preguntas frecuentes de AWS Identity and Access Management](#).

Para proporcionar credenciales de seguridad y utilizar una solicitud autenticada para obtener acceso a recursos, haga lo siguiente:

- Cree una instancia de la clase `AWSSecurityTokenServiceClient`.
- Inicie una sesión utilizando el método `getFederationToken()` del cliente Security Token Service (STS). Proporcione información acerca de la sesión, como el nombre de usuario y una política de IAM que desee adjuntar a las credenciales temporales. Puede proporcionar una duración de sesión opcional. Este método devuelve sus credenciales de seguridad temporales.
- Empaque las credenciales de seguridad temporales en una instancia del objeto `BasicSessionCredentials`. Utilice este objeto para proporcionar las credenciales de seguridad temporales al cliente de Amazon S3.
- Cree una instancia de la clase `AmazonS3Client` usando las credenciales de seguridad temporales. Envíe las solicitudes a Amazon S3 con este cliente. Si envía las solicitudes con las credenciales vencidas, Amazon S3 devolverá un error.

Example

En el siguiente ejemplo se enumeran las claves del bucket de S3 especificado. En el ejemplo obtiene las credenciales de seguridad temporales para una sesión de dos horas para su usuario federado y utiliza las credenciales para enviar solicitudes autenticadas a Amazon S3. Para ejecutar el ejemplo, debe crear un usuario de IAM con una política adjuntada que permita al usuario solicitar credenciales de seguridad temporales y generar una lista de sus recursos de AWS. La siguiente política lo cumple:

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

Para obtener más información acerca de cómo crear un usuario de IAM, consulte [Creación del primer grupo de usuarios y administradores de IAM](#) en la Guía del usuario de IAM.

Después de crear un usuario de IAM y asociarle la política anterior, puede ejecutar el ejemplo siguiente. Para obtener instrucciones sobre cómo crear y probar una muestra funcional, consulte [Introducción](#) en la Guía para desarrolladores de AWS SDK para Java.

```
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

Puede proporcionar credenciales de seguridad temporales para sus usuarios federados y aplicaciones, de modo que estos puedan enviar solicitudes autenticadas a fin de obtener acceso a sus recursos de AWS. Al solicitar estas credenciales temporales, debe proporcionar un nombre de usuario y una política de IAM que describa los permisos de recursos que desea otorgar. De forma predeterminada, la sesión dura una hora. Puede solicitar las credenciales de seguridad temporales para usuarios federados y aplicaciones para establecer un valor de duración distinto de forma explícita. Para obtener información acerca del envío de solicitudes autenticadas, consulte [Realizar solicitudes](#).

Note

Cuando solicite las credenciales de seguridad temporales para usuarios federados y aplicaciones, para mayor seguridad le sugerimos que use un usuario de IAM específico que solo tenga los permisos de acceso necesarios. El usuario temporal creado nunca puede obtener más permisos que el usuario de IAM que solicitó las credenciales de seguridad temporales. Para obtener más información, consulte [Preguntas frecuentes de AWS Identity and Access Management](#).

Siga estas instrucciones:

- Cree una instancia del cliente de AWS Security Token Service, clase `AmazonSecurityTokenServiceClient`.
- Inicie sesión utilizando el método `GetFederationToken` del cliente STS. Tiene que proporcionar información acerca de la sesión, como el nombre de usuario y una política de IAM que desee adjuntar a las credenciales temporales. También tiene la posibilidad de proporcionar una duración de sesión. Este método devuelve sus credenciales de seguridad temporales.
- Empaquete las credenciales de seguridad temporales en una instancia del objeto `SessionAWSCredentials`. Utilice este objeto para proporcionar las credenciales de seguridad temporales al cliente de Amazon S3.

- Cree una instancia de la clase `AmazonS3Client` proporcionando las credenciales de seguridad temporales. Use este cliente para enviar solicitudes a Amazon S3. Si envía las solicitudes con las credenciales vencidas, Amazon S3 devolverá un error.

Example

En el siguiente ejemplo de C# se enumeran las claves del bucket especificado. En el ejemplo se obtienen las credenciales de seguridad temporales para una sesión de dos horas para el usuario federado (User1) y se utilizan las credenciales para enviar solicitudes autenticadas a Amazon S3.

- Para este ejercicio cree un usuario de IAM con permisos mínimos. Con las credenciales de este usuario de IAM, solicite credenciales temporales para otros usuarios. Este ejemplo solo enumera los objetos de un bucket específico. Cree un usuario de IAM con la siguiente política asociada:

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

La política permite al usuario de IAM solicitar las credenciales de seguridad temporales y el permiso de acceso solo para mostrar sus recursos de AWS. Para obtener más información acerca de cómo crear un usuario de IAM, consulte [Creación del grupo de usuarios y administradores de IAM](#) en la Guía del usuario de IAM.

- Use las credenciales de seguridad del usuario de IAM para probar el siguiente ejemplo. En el ejemplo se envía una solicitud autenticada a Amazon S3 con las credenciales de seguridad temporales. En el ejemplo se especifica la siguiente política cuando se solicitan las credenciales de seguridad temporales para el usuario federado (User1), que restringe el acceso para mostrar los objetos de un bucket específico (YourBucketName). Debe actualizar la política y proporcionar su propio nombre de bucket existente.

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

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

- Example

Actualice la siguiente muestra y proporcione el nombre de bucket que especificó en la política de acceso del usuario federado anterior. Para obtener información acerca de cómo configurar y ejecutar ejemplos de código, consulte [Introducción al SDK de AWS para .NET](#) en la Guía para desarrolladores del SDK de AWS para .NET.

```
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"":""Stmt1311212314284"",
                    ""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

En este tema, se explica cómo usar clases de la versión 3 de AWS SDK para PHP a fin de solicitar credenciales de seguridad temporales para usuarios federados y aplicaciones, y utilizarlas para obtener acceso a recursos almacenados en Amazon S3. Para obtener más información acerca de la API del SDK de AWS para Ruby, consulte [SDK de AWS para Ruby, versión 2](#).

Puede proporcionar credenciales de seguridad temporales a sus usuarios federados y aplicaciones para que puedan enviar solicitudes autenticadas a fin de obtener acceso a sus recursos de AWS. Al solicitar estas credenciales temporales, debe proporcionar un nombre de usuario y una política de IAM que describa los permisos de recursos que desea otorgar. Estas credenciales caducan cuando finaliza la duración de la sesión. De forma predeterminada, la sesión durará una hora. Puede solicitar las credenciales de seguridad temporales para usuarios federados y aplicaciones para establecer explícitamente otro valor para la duración. Para obtener más información acerca de las credenciales de seguridad temporales, consulte [Credenciales de seguridad temporales](#) en la guía del usuario de IAM. Para obtener más información sobre cómo proporcionar credenciales de seguridad temporales a sus usuarios federados y sus aplicaciones, consulte [Realizar solicitudes](#).

Para mayor seguridad, cuando solicite las credenciales de seguridad temporales para usuarios federados y aplicaciones, recomendamos que use un usuario de IAM específico que solo tenga los permisos de acceso necesarios. El usuario temporal creado nunca puede obtener más permisos que el usuario de IAM que solicitó las credenciales de seguridad temporales. Para obtener más información sobre la federación de identidad, consulte [Preguntas frecuentes de AWS Identity and Access Management](#).

Para obtener más información acerca de la API del SDK de AWS para Ruby, consulte [SDK de AWS para Ruby, versión 2](#).

Example

En el siguiente ejemplo de PHP se enumeran las claves del bucket especificado. En el ejemplo se obtienen las credenciales de seguridad temporales para una sesión de una hora para el usuario federado (User1). A continuación, use las credenciales de seguridad temporales para enviar solicitudes autenticadas a Amazon S3.

Para mayor seguridad, cuando solicite las credenciales temporales para otros usuarios, utilice las credenciales de seguridad de un usuario de IAM que tenga permisos para solicitar credenciales de seguridad temporales. Para asegurarse de que el usuario de IAM otorga solo los permisos específicos de la aplicación mínimos al usuario federado, también puede limitar los permisos de acceso a este usuario de IAM. En este ejemplo solo se enumera objetos de un bucket específico. Cree un usuario de IAM con la siguiente política asociada:

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

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

La política permite al usuario de IAM solicitar las credenciales de seguridad temporales y el permiso de acceso solo para mostrar sus recursos de AWS. Para obtener más información acerca de cómo crear un usuario de IAM, consulte [Creación del primer grupo de usuarios y administradores de IAM](#) en la Guía del usuario de IAM.

Ahora puede utilizar las credenciales de seguridad del usuario de IAM para probar el siguiente ejemplo. En el ejemplo se envía una solicitud autenticada a Amazon S3 con las credenciales de seguridad temporales. Cuando solicite las credenciales de seguridad temporales para el usuario federado (User1), el ejemplo especifica la política siguiente, que restringe el acceso para enumerar los objetos de un bucket específico. Actualice la política con el nombre de su bucket.

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

En el siguiente ejemplo, al especificar el recurso de política, reemplace `YourBucketName` por el nombre del bucket.

```
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

Puede proporcionar credenciales de seguridad temporales para sus usuarios federados y aplicaciones, de modo que estos puedan enviar solicitudes autenticadas a fin de obtener acceso a sus recursos de AWS. Al solicitar estas credenciales temporales al servicio de IAM, debe proporcionar un nombre de usuario y una política de IAM que describa los permisos de recursos que desea conceder. De forma predeterminada, la sesión durará una hora. Sin embargo, si solicita las credenciales temporales con las credenciales de usuario de IAM, puede solicitar las credenciales de seguridad temporales para usuarios federados y aplicaciones para establecer un valor de duración distinto de forma explícita. Para obtener más información sobre cómo proporcionar credenciales de seguridad temporales para sus usuarios federados y sus aplicaciones, consulte [Realizar solicitudes](#).

Note

Para mayor seguridad, cuando solicita las credenciales de seguridad temporales para usuarios federados y aplicaciones, se recomienda utilizar un usuario de IAM específico que solo tenga los permisos de acceso necesarios. El usuario temporal creado nunca puede obtener más permisos que el usuario de IAM que solicitó las credenciales de seguridad temporales. Para obtener más información, consulte [Preguntas frecuentes de AWS Identity and Access Management](#).

Example

En el siguiente ejemplo de código Ruby se permite que un usuario federado con un conjunto limitado de permisos enumere las clave en el bucket especificado.

```
# 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__

```

Realizar solicitudes con la API REST

Esta sección incluye información acerca de cómo realizar solicitudes a los puntos de enlace de Amazon S3 con la API REST. Para ver una lista de los puntos de conexión de Amazon S3, consulte [Regiones y puntos de conexión](#) en la Referencia general de AWS.

Crear nombres de host de S3 para solicitudes de la API REST

Los puntos de enlace de Amazon S3 siguen la estructura que se muestra a continuación:

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

Los puntos de enlace de puntos de acceso de Amazon S3 y los puntos de enlace de doble pila también siguen la estructura estándar:

- Puntos de acceso de Amazon S -s3-accesspoint.*Region*.amazonaws.com
- Doble pila - s3.dualstack.*Region*.amazonaws.com

Para obtener una lista completa de las regiones y los puntos de conexión de Amazon S3, consulte [Puntos de conexión y cuotas de Amazon S3](#) en la Referencia general de Amazon Web Services.

Solicitudes de tipo alojamiento virtual y de tipo ruta

Cuando realiza solicitudes con la API REST, puede utilizar los URI de tipo alojamiento virtual o tipo ruta para los puntos de enlace de Amazon S3. Para obtener más información, consulte [Solicitudes de tipo ruta](#).

Example Solicitud de tipo alojamiento virtual

A continuación, se muestra un ejemplo de una solicitud de tipo alojamiento virtual para eliminar el archivo puppy.jpg del bucket denominado examplebucket en la región EE. UU. Oeste (Oregón). Para obtener más información acerca de las solicitudes de tipo alojamiento virtual, consulte [Solicitudes de tipo ruta](#).

```
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
```

Example Solicitud de tipo ruta

A continuación se muestra un ejemplo de una versión tipo ruta de la misma solicitud.

```
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
```

Actualmente, Amazon S3 admite URL tanto de tipo host virtual como de tipo ruta en todas las Regiones de AWS. Sin embargo, las URL de tipo ruta dejarán de usarse en el futuro. Para obtener más información, consulte la siguiente [nota Importante](#).

Para obtener más información acerca de las solicitudes de tipo ruta, consulte [Solicitudes de tipo ruta](#).

Important

Actualización (23 de septiembre de 2020): para garantizar que los clientes tienen el tiempo necesario para pasar a las URL de tipo host virtual, hemos decidido retrasar la obsolescencia de las URL de tipo ruta. Para obtener más información, consulte [Amazon S3 Path Deprecation Plan – The Rest of the Story](#) en el Blog de noticias de AWS.

Realizar solicitudes a los puntos de enlace de doble pila con la API REST

Cuando utiliza la API REST, puede acceder directamente al punto de enlace de doble pila con un nombre de punto de enlace de tipo alojamiento virtual o tipo ruta (URI). Todos los nombres de punto de enlace de doble pila de Amazon S3 incluyen la región en el nombre. A diferencia de los puntos de enlace estándar que solo admiten IPv4, tanto los puntos de enlace de tipo alojamiento virtual como los de tipo ruta utilizan nombres de puntos de enlace específicos de la región.

Example Solicitud de punto de enlace de doble pila de tipo alojamiento virtual

Puede utilizar un punto de enlace de tipo alojamiento virtual en su solicitud REST como se muestra en el siguiente ejemplo, que recupera el objeto puppy.jpg del bucket denominado examplebucket en la región EE. UU. Oeste (Oregón).

```
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
```

Example Solicitud de punto de enlace de doble pila de tipo ruta

O bien, puede utilizar un punto de enlace de tipo ruta en su solicitud como se muestra en el siguiente ejemplo.

```
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
```

Para obtener más información acerca de los puntos de enlace de doble pila, consulte [Uso de puntos de conexión de doble pila en Amazon S3](#).

Para obtener más información acerca de cómo realizar solicitudes mediante la API de REST, consulte los temas siguientes.

Temas

- [Solicitar redireccionamiento y la API REST](#)
- [Enrutamiento de solicitudes](#)

Solicitar redireccionamiento y la API REST

Temas

- [Redireccionamientos y agentes de usuario HTTP](#)
- [Redireccionamientos y 100-continue](#)
- [Ejemplo de redireccionamiento](#)

En esta sección se describe cómo administrar el redireccionamiento HTTP mediante la API REST de Amazon S3. Para obtener información general sobre los redireccionamientos de Amazon S3, consulte [Realizar solicitudes](#) en la Referencia de la API de Amazon Simple Storage Service.

Redireccionamientos y agentes de usuario HTTP

Los programas que usan la API REST de Amazon S3 deberían poder ocuparse de los redireccionamientos o bien en la capa de aplicación o bien en la capa HTTP. Muchas bibliotecas clientes y agentes de usuario HTTP pueden configurarse para administrar los redireccionamientos de forma correcta y automática. Sin embargo, hay muchas otras que tienen implementaciones de los redireccionamientos incorrectas o incompletas.

Antes de depender de una biblioteca para cumplir el requisito de redireccionamiento, pruebe lo siguiente:

- Compruebe que todos los encabezados de solicitudes HTTP estén incluidos correctamente en la solicitud redirigida (la segunda solicitud tras haber recibido un redireccionamiento) incluidos los estándares HTTP como la autorización y la fecha.
- Compruebe que los redireccionamientos que no sean GET, como PUT o DELETE, funcionen correctamente.
- Compruebe que las solicitudes PUT grandes sigan los redireccionamientos correctamente.
- Compruebe que las solicitudes PUT sigan los redireccionamientos correctamente si la respuesta 100-continue tarda demasiado en llegar.

Los agentes de usuario HTTP que se ajusten estrictamente a RFC 2616 podrían necesitar una confirmación explícita antes de seguir un redireccionamiento si el método de solicitud HTTP no es GET o HEAD. Normalmente, es seguro seguir los redireccionamientos generados por Amazon S3 automáticamente, ya que el sistema generará redireccionamientos solamente a los hosts del dominio amazonaws.com, y el efecto de la solicitud redirigida será el mismo que el de la solicitud original.

Redireccionamientos y 100-continue

Para administrar el redireccionamiento de forma más sencilla, mejorar la eficacia y evitar los costos asociados con el envío duplicado del cuerpo de una solicitud redirigida, configure su aplicación para usar 100-continues en operaciones PUT. Cuando su aplicación usa 100-continue, no envía el cuerpo de la solicitud hasta que recibe una confirmación. Si el mensaje se rechaza en función de los encabezados, el cuerpo del mismo no se llega a enviar. Para obtener más información sobre 100-continue, visite [RFC 2616 Sección 8.2.3](#)

Note

Según RFC 2616, al usar Expect: Continue sin un servidor HTTP conocido, no debería esperar un periodo indefinido para enviar el cuerpo de la solicitud. Esto se debe a que algunos servidores HTTP no reconocen 100-continue. Sin embargo, Amazon S3 sí que reconoce si su solicitud contiene un Expect: Continue y responderá con un estado 100-continue provisional o un código de estado final. Además, no se producirá ningún error de redireccionamiento tras recibir la autorización provisional del 100-continue. Así, le resultará más fácil evitar recibir una respuesta de redireccionamiento mientras sigue escribiendo el cuerpo de la solicitud.

Ejemplo de redireccionamiento

En esta sección se ofrece un ejemplo de interacción cliente-servidor mediante redireccionamientos HTTP y 100-continue.

A continuación mostramos un ejemplo de PUT al bucket `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 devuelve lo siguiente:

```
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>
```

El cliente sigue la respuesta de redireccionamiento y envía una nueva solicitud al punto de conexión temporal `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 devuelve un 100-continue que indica que el cliente debería continuar con el envío del cuerpo de la solicitud.

```
HTTP/1.1 100 Continue
```

El cliente envía el cuerpo de la solicitud.

```
ha ha\n
```

Amazon S3 devuelve la respuesta final.

```
HTTP/1.1 200 OK
```

```
Date: Mon, 15 Oct 2007 22:18:48 GMT
```

```
ETag: "a2c8d6b872054293afd41061e93bc289"
```

```
Content-Length: 0
```

```
Server: AmazonS3
```

Enrutamiento de solicitudes

Los programas que realizan solicitudes a buckets creados con la API [CreateBucket](#) que incluyen una [CreateBucketConfiguration](#) deben admitir el redireccionamiento. Además, algunos clientes que no respetan los TTL de DNS podrían producir problemas.

En esta sección se describe el direccionamiento y algunas cuestiones sobre DNS que se deben tener en cuenta al usar Amazon S3.

Solicitar redireccionamiento y la API REST

Amazon S3 utiliza el sistema de nombres de dominio (DNS) para dirigir las solicitudes a las ubicaciones que pueden procesarlas. Este sistema es eficaz, pero se pueden producir errores de direccionamiento temporal. Si una solicitud llega a la ubicación incorrecta de Amazon S3, Amazon S3 responde con un redireccionamiento temporal que le indica al solicitante que debe volver a enviar la solicitud a un nuevo punto de enlace. Si una solicitud se realiza incorrectamente, Amazon S3 utiliza

el redireccionamiento permanente para proporcionar instrucciones sobre cómo realizar la solicitud correctamente.

Important

Para utilizar esta característica, debe tener una aplicación que pueda gestionar las respuestas de redirección de Amazon S3. La única excepción es para las aplicaciones que funcionan exclusivamente con buckets creados sin `<CreateBucketConfiguration>`.

Para obtener más información acerca de las restricciones de ubicación, consulte [Acceso y muestra de un bucket de Amazon S3](#).

Para todas las regiones que se lanzaron después del 20 de marzo de 2019, si una solicitud llega a la ubicación de Amazon S3 incorrecta, Amazon S3 devuelve un error de solicitud errónea HTTP 400.

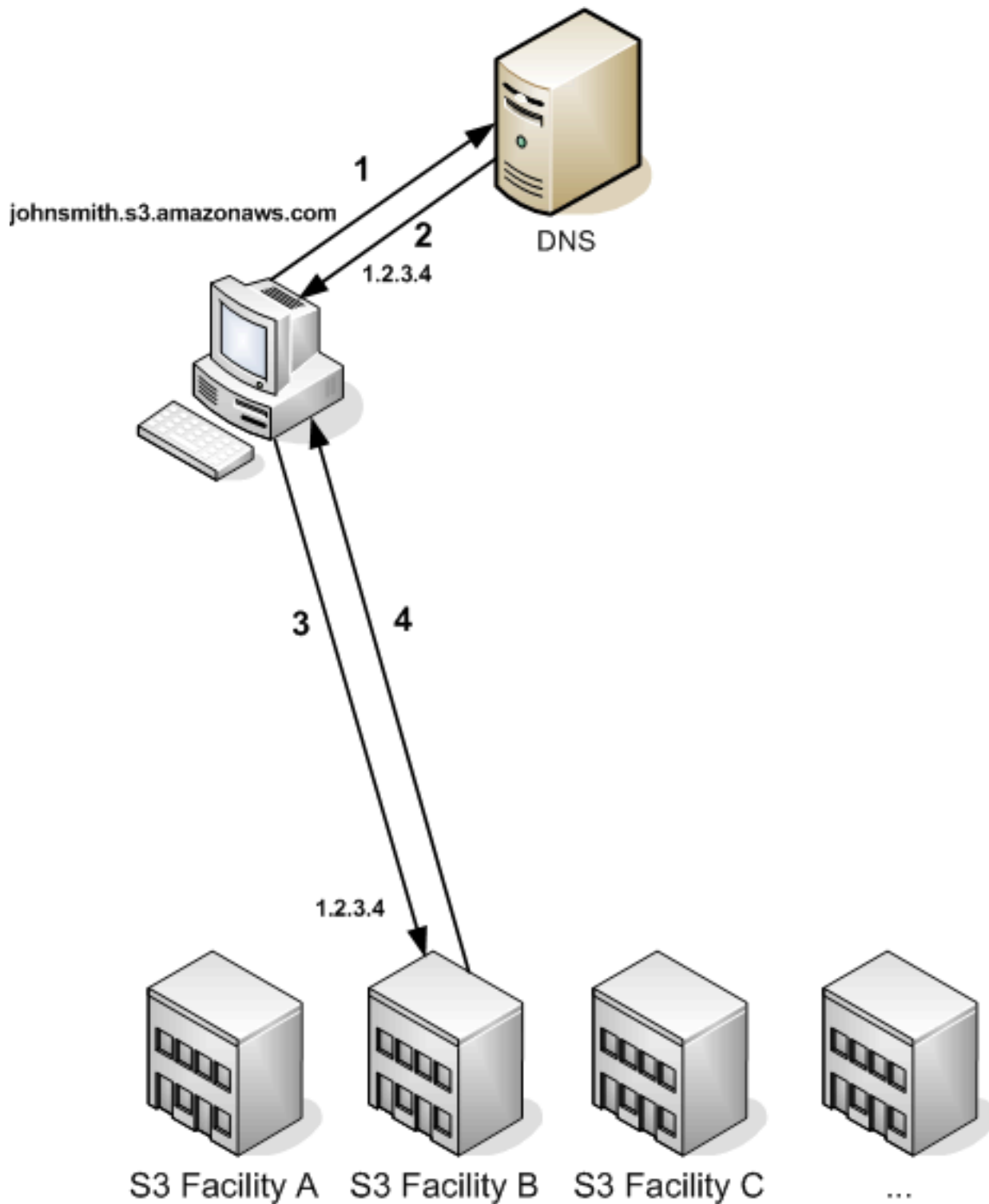
Para obtener más información sobre cómo habilitar o deshabilitar una Región de AWS, consulte [Puntos de conexión y Regiones de AWS](#) en la Referencia general de AWS.

Temas

- [Enrutamiento de DNS](#)
- [Redireccionamiento temporal de solicitud](#)
- [Redireccionamiento permanente de solicitud](#)
- [Ejemplos de redireccionamiento de solicitud](#)

Enrutamiento de DNS

El direccionamiento del sistema de nombres de dominio (DNS) dirige las solicitudes a las ubicaciones correctas de Amazon S3. En la figura y el procedimiento siguientes se muestra un ejemplo de direccionamiento de DNS.



Pasos de la solicitud de direccionamiento de DNS

1. El cliente realiza una solicitud de DNS para obtener un objeto almacenado en Amazon S3.

2. El cliente recibe una o más direcciones IP para las ubicaciones que pueden procesar la solicitud. En este ejemplo, la dirección IP es para la ubicación B.
3. El cliente realiza una solicitud a la ubicación B de Amazon S3.
4. La ubicación B devuelve una copia del objeto al cliente.

Redireccionamiento temporal de solicitud

Un redireccionamiento temporal es un tipo de respuesta de error que indica al solicitante que debe volver a enviar la solicitud a otro punto de enlace. Debido a las características de distribución de Amazon S3, las solicitudes pueden dirigirse temporalmente a la ubicación incorrecta. Es posible que esto ocurra inmediatamente después de crear o eliminar buckets.

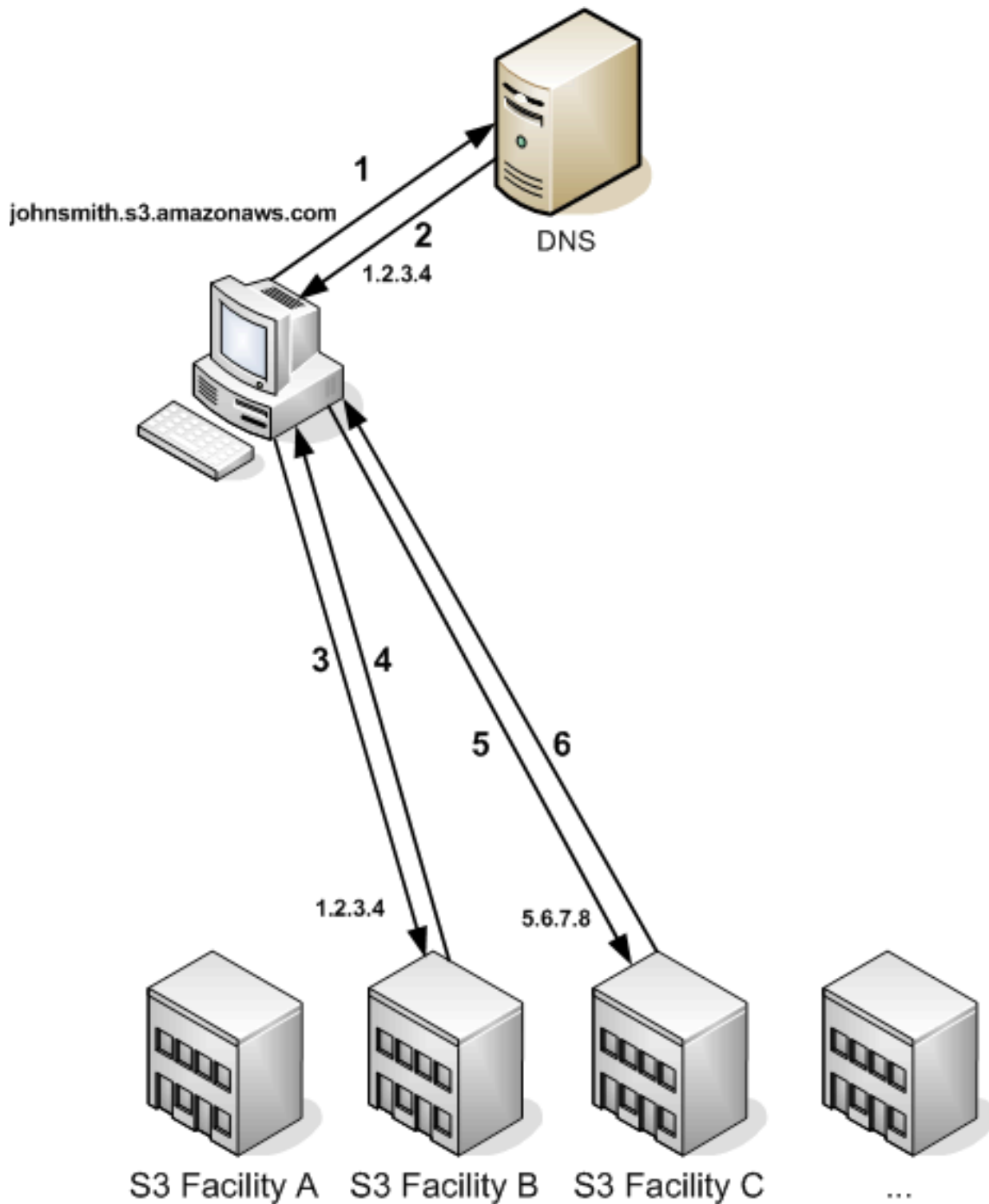
Por ejemplo, si crea un bucket nuevo y realiza una solicitud al bucket de inmediato, es posible que reciba un redireccionamiento temporal, según la restricción de ubicación del bucket. Si creó el bucket en la Región de AWS EE. UU. Este (Norte de Virginia), no verá el redireccionamiento, ya que este también es el punto de enlace predeterminado de Amazon S3.

Sin embargo, si el bucket se creó en otra región, cualquier solicitud para el bucket se dirigirá al punto de enlace predeterminado mientras la entrada de DNS del bucket se propaga. El punto de enlace predeterminado redirige la solicitud al punto de enlace correcto con una respuesta HTTP 302. El redireccionamiento temporal contiene un Uniform Resource Identifier (URI, Identificador de recursos uniforme) en la ubicación correcta, que usted puede utilizar para volver a enviar la solicitud de inmediato.

Important

No vuelva a utilizar un punto de enlace proporcionado por una respuesta de redireccionamiento anterior. Puede parecer que funciona (incluso durante largos períodos de tiempo), pero podría proporcionar resultados impredecibles y fallar sin previo aviso.

En la figura y el procedimiento siguientes se muestra un ejemplo de un redireccionamiento temporal.



Pasos del redireccionamiento de solicitud temporal

1. El cliente realiza una solicitud de DNS para obtener un objeto almacenado en Amazon S3.
2. El cliente recibe una o más direcciones IP para las ubicaciones que pueden procesar la solicitud.

3. El cliente realiza una solicitud a la ubicación B de Amazon S3.
4. La ubicación B devuelve un redireccionamiento que indica que el objeto está disponible en la ubicación C.
5. El cliente vuelve a enviar la solicitud a la ubicación C.
6. La ubicación C devuelve una copia del objeto.

Redireccionamiento permanente de solicitud

Un redireccionamiento permanente indica que su solicitud se dirigió a un recurso de forma incorrecta. Por ejemplo, el redireccionamiento permanente se produce si utiliza una solicitud de tipo ruta para obtener acceso a un bucket creado con `<CreateBucketConfiguration>`. Para obtener más información, consulte [Acceso y muestra de un bucket de Amazon S3](#).

Para ayudar a detectar estos errores durante el desarrollo, este tipo de redireccionamiento no contiene un encabezado HTTP de ubicación que le permita seguir automáticamente la solicitud en la ubicación correcta. Consulte el documento de errores XML resultante para obtener ayuda sobre cómo usar el punto de enlace correcto de Amazon S3.

Ejemplos de redireccionamiento de solicitud

A continuación se proporcionan ejemplos de respuestas de redireccionamiento de solicitud temporal.

Respuesta de redireccionamiento temporal de la 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>
```

Respuesta de redireccionamiento temporal de la API de SOAP

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API REST o los SDK de AWS.

```
<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>
```

Consideraciones sobre DNS

Uno de los requisitos de diseño de Amazon S3 es una disponibilidad extremadamente alta. Una de las maneras de cumplir con este requisito es mediante la actualización de las direcciones IP asociadas con el punto de enlace de Amazon S3 en el DNS según sea necesario. Estos cambios se reflejan automáticamente en los clientes cuya vida útil es corta, pero no en algunos clientes cuya vida útil es larga. Aquellos clientes cuya vida útil es larga deben realizar acciones especiales para volver a resolver el punto de enlace de Amazon S3 periódicamente a fin de beneficiarse de estos cambios. Para obtener más información acerca de las máquinas virtuales (VM), consulte lo siguiente:

- Para Java, la Java Virtual Machine (JVM, Máquina virtual de Java) de Sun almacena en caché las búsquedas del DNS para siempre de forma predeterminada; consulte la sección de “InetAddress Caching” de [la documentación de InetAddress](#) para obtener información acerca de cómo cambiar este comportamiento.
- Para PHP, la VM persistente de PHP que ejecuta las más populares configuraciones de implementación almacena en caché las búsquedas del DNS hasta que la VM se reinicia. Consulte [los documentos de PHP de getHostByName](#).

Desarrollo con Amazon S3 mediante la AWS CLI

Los comandos de la AWS CLI de Amazon S3 se organizan en diferentes conjuntos en la referencia de comandos de la AWS CLI y cada uno tiene sus propios comandos disponibles. Si no encuentra el comando que busca en un conjunto, compruebe uno de los otros conjuntos. Los diferentes conjuntos son:

- `s3`: describe los comandos de alto nivel para trabajar con objetos y buckets. Estos incluyen las acciones de copiar, mostrar y mover. Para obtener una lista completa de comandos en este conjunto, consulte [s3](#) en la referencia de comandos de la AWS CLI.
- `s3api`: describe los comandos de bajo nivel que administran los recursos de S3, como buckets, objetos, sesiones y cargas multiparte. Estos comandos corresponden a las operaciones de la [API de Amazon S3](#). Para obtener una lista completa de comandos de la CLI en este conjunto, consulte [s3api](#) en la referencia de comandos de la AWS CLI.
- `s3control`: describe los comandos de bajo nivel que administran todos los demás recursos de S3, como concesiones de acceso, grupos de lentes de almacenamiento y los buckets de Amazon S3 en Outposts. Estos comandos corresponden a las operaciones de la API [Amazon S3 Control](#). Para obtener una lista completa de comandos de la CLI en este conjunto, consulte [s3control](#) en la referencia de comandos de la AWS CLI.

Si tiene problemas para configurar la AWS CLI, consulte [Cómo resuelvo el error “No se pueden encontrar las credenciales” en Amazon S3](#) en el Centro de conocimiento de AWS re:Post.

Note

Para poder acceder a los servicios de AWS, como Amazon S3, debe proporcionar credenciales. A continuación, el servicio puede determinar si usted tiene permisos para obtener acceso a sus recursos. La consola requiere que especifique la contraseña. Puede crear claves de acceso para su Cuenta de AWS con el fin de tener acceso a la AWS CLI o a la API. Sin embargo, no es recomendable acceder a AWS con las credenciales de su Cuenta de AWS. En su lugar, le recomendamos que utilice AWS Identity and Access Management (IAM). Cree un usuario de IAM, añada el usuario a un grupo de IAM con permisos administrativos y, a continuación, conceda permisos administrativos al usuario de IAM que ha creado. De este modo, podrá tener acceso a AWS mediante una URL especial y las credenciales de ese usuario de IAM. Para obtener instrucciones, consulte [Creación del primer grupo de usuarios y administradores de IAM](#) en la Guía del usuario de IAM.

Más información sobre el AWS CLI

Para obtener más información acerca de la AWS, consulte los siguientes recursos:

- [AWS Command Line Interface](#)
- [AWS Command Line Interface Guía del usuario de la versión 2 de 2](#)

Desarrollo con Amazon S3 mediante los SDK de AWS

Los kits de desarrollo de software (SDK) de se encuentran disponibles en muchos lenguajes de programación populares. Cada SDK proporciona una API, ejemplos de código y documentación que facilitan a los desarrolladores la creación de aplicaciones en su lenguaje preferido.

Note

Puede utilizar AWS Amplify para el desarrollo integral y de pila completa de aplicaciones web y móviles. Amplify Storage integra a la perfección las capacidades de almacenamiento y administración de archivos en aplicaciones frontend web y móviles, basadas en Amazon S3. Para obtener más información, consulte [Almacenamiento](#) en la Guía del usuario de Amplify.

Documentación de SDK	Ejemplos de código
AWS SDK para C++	AWS SDK para C++ Ejemplos de código de la
AWS CLI	AWS CLI Ejemplos de código de la
AWS SDK para Go	AWS SDK para Go Ejemplos de código de la
AWS SDK para Java	AWS SDK para Java Ejemplos de código de la
AWS SDK para JavaScript	AWS SDK para JavaScript Ejemplos de código de la
AWS SDK para Kotlin	AWS SDK para Kotlin Ejemplos de código de la
AWS SDK para .NET	AWS SDK para .NET Ejemplos de código de la

Documentación de SDK	Ejemplos de código
AWS SDK para PHP	AWS SDK para PHP Ejemplos de código de la
Herramientas de AWS para PowerShell	Ejemplos de código de Herramientas para PowerShell
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) Ejemplos de código de la
AWS SDK para Ruby	AWS SDK para Ruby Ejemplos de código de la
AWS SDK para Rust	AWS SDK para Rust Ejemplos de código de la
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP Ejemplos de código de la
AWS SDK para Swift	AWS SDK para Swift Ejemplos de código de la

Para ver ejemplos específicos, consulte [Ejemplos de código de Amazon S3 con SDK de AWS](#).

Interfaces de programación del SDK

Cada SDK de AWS proporciona una o varias interfaces de programación para trabajar con Amazon S3. Cada SDK ofrece una interfaz de bajo nivel para Amazon S3, con métodos que imitan del modo más parecido posible las operaciones de la API. Algunos SDK ofrecen interfaces de alto nivel para Amazon S3, que son abstracciones diseñadas para simplificar los casos de uso frecuentes.

Por ejemplo, cuando realiza una carga de varias partes mediante las operaciones de la API de bajo nivel, utilice una operación para iniciar la carga, otra operación para cargar las partes y una operación final para completar la carga. Una operación de la API de carga multiparte de alto nivel permite realizar todas las operaciones necesarias para la carga en una sola llamada a la API. Para ver ejemplos, consulte [Carga de un objeto mediante la carga multiparte](#) en la Guía del usuario de Amazon S3.

Las operaciones de la API de bajo nivel permiten un mayor control sobre la carga. Le recomendamos que use las operaciones de la API de bajo nivel si necesita detener y reanudar las cargas, variar los tamaños de las partes durante la carga o iniciar las cargas si no conoce de antemano el tamaño de los datos de carga.

Especificación de Signature Version en la autenticación de solicitudes

Amazon S3 solo es compatible con AWS Signature Version 4 en la mayoría de las Regiones de AWS. En algunas de las Regiones de AWS más antiguas, Amazon S3 admite Signature Version 4 y Signature Version 2. Sin embargo, Signature Version 2 se va a desactivar (esta característica quedará obsoleta). Para obtener más información sobre el final del servicio de soporte de Signature Version 2, consulte [AWS Signature Version 2 se va a desactivar \(obsoleta\) para Amazon S3](#).

Para ver una lista de todas las regiones de Amazon S3 y las versiones de Signature que admiten, consulte [Regiones y puntos de enlace](#) en la Referencia general de AWS.

Para todas las Regiones de AWS, los SDK de AWS utilizan Signature Version 4 de forma predeterminada con el fin de autenticar solicitudes. Si utiliza los SDK de AWS lanzados antes de mayo de 2016, es posible que deba solicitar Signature Version 4, como se muestra en la siguiente tabla.

SDK	Solicitud de Signature Version 4 para una autenticación de solicitud
AWS CLI	Para el perfil predeterminado, ejecute el siguiente comando: <pre>\$ aws configure set default.s3.signature_version s3v4</pre> Para un perfil común, ejecute el siguiente comando: <pre>\$ aws configure set profile.your_profile_name.s3.signature_version s3v4</pre>
SDK de Java	Añada lo siguiente en su código: <pre>System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");</pre> O bien, en la línea de comando, especifique lo siguiente: <pre>-Dcom.amazonaws.services.s3.enableV4</pre>

SDK	Solicitud de Signature Version 4 para una autenticación de solicitud
SDK de JavaScript	Configure el parámetro <code>signatureVersion</code> en v4 cuando cree el cliente: <pre>var s3 = new AWS.S3({signatureVersion: 'v4'});</pre>
SDK de PHP	Configure el parámetro <code>signature</code> en v4 cuando cree el cliente del servicio de Amazon S3 para PHP SDK v2: <pre><?php \$client = S3Client::factory(['region' => 'YOUR-REGION', 'version' => 'latest', 'signature' => 'v4']);</pre> Cuando utilice el SDK de PHP v3, establezca el parámetro <code>signature_version</code> en v4 durante la construcción del cliente de servicio de Amazon S3: <pre><?php \$s3 = new Aws\S3\S3Client(['version' => '2006-03-01', 'region' => 'YOUR-REGION', 'signature_version' => 'v4']);</pre>
SDK de Python-Boto	Especifique lo siguiente en el archivo de configuración predeterminado <code>boto</code>: <pre>[s3] use-sigv4 = True</pre>

SDK	Solicitud de Signature Version 4 para una autenticación de solicitud
SDK de Ruby	SDK de Ruby - Version 1: configure el parámetro <code>:s3_signature_version</code> en <code>:v4</code> cuando cree el cliente: <pre>s3 = AWS::S3::Client.new(:s3_signature_version => :v4)</pre> SDK de Ruby - Version 3: configure el parámetro <code>signature_version</code> en <code>v4</code> cuando cree el cliente: <pre>s3 = Aws::S3::Client.new(signature_version: 'v4')</pre>
SDK de .NET	Añada lo siguiente al código antes de crear el cliente de Amazon S3: <pre>AWSConfigsS3.UseSignatureVersion4 = true;</pre> O bien, añada lo siguiente al archivo de configuración: <pre><appSettings> <add key="AWS.S3.UseSignatureVersion4" value="true" /> </appSettings></pre>

AWS Signature Version 2 se va a desactivar (obsoleta) para Amazon S3

Sin embargo, Signature Version 2 se va a desactivar (quedará obsoleta) en Amazon S3. Amazon S3 solo aceptará solicitudes de API que estén firmadas con Signature Version 4.

En esta sección, se incluyen algunas respuestas a preguntas comunes sobre el final del servicio de soporte de Signature Version 2.

¿Qué es Signature Version 2/4 y qué significa la firma de solicitudes?

El proceso de firma Signature Version 2 o Signature Version 4 se utiliza para autenticar las solicitudes de API de Amazon S3. La firma de solicitudes permite que Amazon S3 pueda identificar quién está enviando la solicitud y ayuda a proteger las solicitudes frente a agentes malintencionados.

Para obtener más información sobre la firma de solicitudes de AWS, consulte [Firma de solicitudes de AWS](#) en la Referencia general de AWS.

¿En qué consiste la actualización?

En la actualidad, pueden utilizarse solicitudes de la API de Amazon S3 firmadas con Signature Version 2 y Signature Version 4. Cuando esto suceda, Amazon S3 solamente aceptará solicitudes que estén firmadas con Signature Version 4.

Para obtener más información sobre la firma de solicitudes de AWS, consulte [Cambios de Signature Version 4](#) en la Referencia general de AWS.

¿Por qué se realiza la actualización?

En lugar de utilizar una clave de acceso secreta, Signature Version 4 usa una clave de firma, lo que mejora la seguridad. En la actualidad, Signature Version 4 puede utilizarse en todas las Regiones de AWS, mientras que Signature Version 2 solo se admite en las regiones que se lanzaron antes de enero de 2014. Esta actualización nos permite proporcionar una experiencia más uniforme en todas las regiones.

¿Cómo sé si estoy utilizando Signature Version 4 y qué actualizaciones tengo que hacer?

Normalmente, la versión de Signature que se utiliza para firmar las solicitudes viene determinada por la herramienta o el SDK del lado del cliente. De forma predeterminada, las últimas versiones de los SDK de AWS utilizan Signature Version 4. En el caso del software de terceros, póngase en contacto con el equipo de soporte del software correspondiente para confirmar la versión que necesita. Si envía llamadas REST directas a Amazon S3, debe modificar la aplicación para que utilice el proceso de firma Signature Version 4.

Para obtener información acerca de qué versión de los SDK de AWS debe usarse al pasar a Signature Version 4, consulte [Transición de Signature Version 2 a Signature Version 4](#).

Para obtener información acerca del uso de Signature Version 4 con la API de REST de Amazon S3, consulte [Autenticación de solicitudes \(AWS Signature Version 4\)](#) en la Referencia de la API de Amazon Simple Storage Service.

¿Qué sucede si no realizo la actualización?

Las solicitudes firmadas con Signature Version 2 que se generen cuando esto suceda, no podrán autenticarse con Amazon S3. Los solicitantes recibirán mensajes de error en los que se informará de que la solicitud debe firmarse con Signature Version 4.

¿Debo hacer algún cambio aunque utilice una URL prefirmada que requiera mi firma durante más de siete días?

Si utiliza una URL prefirmada que necesita su firma durante más de siete días, no tiene que hacer nada por el momento. Podrá seguir utilizando AWS Signature Version 2 para firmar y autenticar la URL prefirmada. Seguiremos investigando y le proporcionaremos más detalles sobre la migración a Signature Version 4 con direcciones URL prefirmadas.

Más información

- Para obtener más información acerca del uso de Signature Version 4, consulte [Firma de solicitudes de la API de AWS](#).
- Consulte la lista de cambios entre Signature Version 2 y Signature Version 4 en [Cambios de Signature Version 4](#).
- Consulte la publicación [AWS Signature Version 4 to replace AWS Signature Version 2 for signing Amazon S3 API requests](#) en los foros de AWS.
- Si tiene alguna pregunta o duda, contáctese con [Soporte](#).

Transición de Signature Version 2 a Signature Version 4

Si en la actualidad utiliza Signature Version 2 para autenticar las solicitudes de API de Amazon S3, debe cambiar a Signature Version 4. Tal y como se describe en , el servicio de soporte de Signature Version 2 está a punto de finalizar [AWS Signature Version 2 se va a desactivar \(obsoleta\) para Amazon S3](#).

Para obtener información acerca del uso de Signature Version 4 con la API de REST de Amazon S3, consulte [Autenticación de solicitudes \(AWS Signature Version 4\)](#) en la Referencia de la API de Amazon Simple Storage Service.

En la tabla siguiente, se muestran los SDK con la versión mínima necesaria para utilizar Signature Version 4 (SigV4). Si utiliza URL prefirmadas con los SDK de AWS para Java, JavaScript (Node.js) o Python (Boto/CLI), debe especificar la Región de AWS apropiada y configurar Signature Version 4

para el cliente. Para obtener información acerca de la configuración de SigV4 en el cliente, consulte [Especificación de Signature Version en la autenticación de solicitudes](#).

Si utiliza este SDK o producto	Actualice a esta versión del SDK	¿Es necesario cambiar el código para que el cliente utilice Sigv4?	Enlace a la documentación del SDK
AWS SDK para Java v1	Actualice a Java 1.11.201+ o v2.	Sí	Especificación de Signature Version en la autenticación de solicitudes
AWS SDK para Java v2	No es necesario actualizar el SDK.	No	AWS SDK para Java
AWS SDK para .NET v1	Actualice a 3.1.10 o versiones posteriores.	Sí	AWS SDK para .NET
AWS SDK para .NET v2	Actualice a 3.1.10 o versiones posteriores.	No	AWS SDK para .NET v2
AWS SDK para .NET v3	Actualice a 3.3.0.0 o versiones posteriores.	Sí	AWS SDK para .NET v3
AWS SDK para JavaScript v1	Actualice a 2.68.0 o	Sí	AWS SDK para JavaScript

Si utiliza este SDK o producto	Actualice a esta versión del SDK	¿Es necesario cambiar el código para que el cliente utilice Sigv4?	Enlace a la documentación del SDK
	versiones posteriores.		
AWS SDK para JavaScript v2	Actualice a 2.68.0 o versiones posteriores.	Sí	AWS SDK para JavaScript
AWS SDK para JavaScript v3	No tiene que hacer nada por el momento. Actualice a la versión principal V3 en el tercer trimestre de 2019.	No	AWS SDK para JavaScript

Si utiliza este SDK o producto	Actualice a esta versión del SDK	¿Es necesario cambiar el código para que el cliente utilice Sigv4?	Enlace a la documentación del SDK
AWS SDK para PHP v1	Se recomienda a actualizar a la versión más reciente de PHP o, al menos, a la v2.7.4 con el parámetro de firma establecido en v4 en la configuración del cliente de S3.	Sí	AWS SDK para PHP
AWS SDK para PHP v2	Se recomienda a actualizar a la versión más reciente de PHP o, al menos, a la v2.7.4 con el parámetro de firma establecido en v4 en la configuración del cliente de S3.	No	AWS SDK para PHP

Si utiliza este SDK o producto	Actualice a esta versión del SDK	¿Es necesario cambiar el código para que el cliente utilice Sigv4?	Enlace a la documentación del SDK
AWS SDK para PHP v3	No es necesario actualizar el SDK.	No	AWS SDK para PHP
Boto2	Actualice a Boto2 v2.49.0.	Sí	Actualización a Boto
Boto3	Actualice a 1.5.71 (Botocore), 1.4.6 (Boto3).	Sí	Boto 3: SDK de AWS para Python
AWS CLI	Actualice a 1.11.108.	Sí	AWS Command Line Interface
AWS CLI v2 (vista previa)	No es necesario actualizar el SDK.	No	AWS Command Line Interface Versión 2 de la 2
AWS SDK para Ruby v1	Actualice a Ruby V3.	Sí	Ruby V3 para AWS
AWS SDK para Ruby v2	Actualice a Ruby V3.	Sí	Ruby V3 para AWS
AWS SDK para Ruby v3	No es necesario actualizar el SDK.	No	Ruby V3 para AWS

Si utiliza este SDK o producto	Actualice a esta versión del SDK	¿Es necesario cambiar el código para que el cliente utilice Sigv4?	Enlace a la documentación del SDK
Go	No es necesario actualizar el SDK.	No	AWS SDK para Go
C++	No es necesario actualizar el SDK.	No	AWS SDK para C++

AWS Tools for Windows PowerShell o bien AWS Tools for PowerShell Core

Si utiliza versiones del módulo anteriores a 3.3.0.0, debe actualizar a 3.3.0.0.

Para obtener información sobre la versión, utilice el cmdlet `Get-Module`:

```
Get-Module -Name AWSPowershell
Get-Module -Name AWSPowershell.NetCore
```

Para actualizar a la versión 3.3.0.0, utilice el cmdlet `Update-Module`:

```
Update-Module -Name AWSPowershell
Update-Module -Name AWSPowershell.NetCore
```

Puede utilizar direcciones URL prefiradas que sean válidas durante más siete días en las que el tráfico se envíe con Signature Version 2.

Obtención de los ID de las solicitudes de Amazon S3 para AWS Support

Siempre que contacte con AWS Support porque han surgido errores o se ha encontrado con un comportamiento inesperado en Amazon S3, debe proporcionar los ID de las solicitudes relacionadas con la acción que ha fallado. AWS Support utiliza estos ID de las solicitudes para ayudarle a resolver los problemas que está experimentando.

Las ID de solicitudes vienen en pares, se devuelven en cada respuesta que procesa Amazon S3 (incluso las erróneas) y se pueden obtener a través de registros detallados. Hay una serie de métodos comunes para obtener los ID de solicitud, incluidos los registros de acceso al servidor de S3 y los eventos o eventos de datos de AWS CloudTrail.

Después de recuperar estos registros, copie y conserve esos dos valores porque los necesitará cuando contacte con AWS Support. Para obtener más información sobre cómo contactar con AWS Support, consulte [Contacte con AWS](#) o la [Documentación de AWS Support](#).

Temas

- [Uso de AWS CLI para obtener ID de solicitudes](#)
- [Uso de Windows PowerShell para obtener ID de solicitudes](#)
- [Uso de eventos de datos de AWS CloudTrail para obtener ID de solicitudes](#)
- [Uso del registro de acceso al servidor de S3 para obtener los ID de solicitudes](#)
- [Utilización de HTTP para obtener los ID de las solicitudes](#)
- [Utilización de un navegador web para obtener ID de solicitudes](#)
- [Uso de los SDK de AWS para obtener los ID de solicitudes](#)

Uso de AWS CLI para obtener ID de solicitudes

Para obtener los ID de solicitudes cuando se use la AWS Command Line Interface (AWS CLI), agregue `--debug` al comando. Por ejemplo, debería ver `x-amz-request-id` y `x-amz-id-2` en el registro de depuración como se muestra a continuación:

```
...
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'}  
...
```

Uso de Windows PowerShell para obtener ID de solicitudes

Para obtener información sobre la recuperación de registros con Windows PowerShell, consulte [Registro de respuesta en AWS Tools for Windows PowerShell](#) en el Blog de herramientas para desarrolladores de AWS.

Uso de eventos de datos de AWS CloudTrail para obtener ID de solicitudes

Un bucket de Amazon S3 configurado con los eventos de datos de CloudTrail para registrar operaciones de la API de nivel de objeto de S3 proporciona información detallada sobre las acciones que realiza un usuario, un rol o un servicio de AWS en Amazon S3. Puede identificar los ID de solicitudes de S3 consultando los eventos de CloudTrail con Athena.

Para obtener más información, consulte [Identificación de solicitudes de Amazon S3 mediante CloudTrail](#) en la Guía del usuario de Amazon Simple Storage Service.

Uso del registro de acceso al servidor de S3 para obtener los ID de solicitudes

Un bucket de Amazon S3 configurado para el registro de acceso al servidor S3 proporciona registros detallados para cada solicitud realizada al bucket. Para identificar los ID de solicitudes de S3, consulte los registros de acceso al servidor mediante Athena.

Para obtener más información, consulte [Consulta de los registros de acceso de las solicitudes mediante Amazon Athena](#) en la Guía del usuario de Amazon Simple Storage Service.

Utilización de HTTP para obtener los ID de las solicitudes

Para obtener sus ID de solicitudes, `x-amz-request-id` y `x-amz-id-2` puede registrar los detalles de una solicitud HTTP antes de que llegue a la aplicación de destino. Existen diversas herramientas de terceros que se pueden utilizar para recuperar registros detallados para solicitudes HTTP. Elija una de confianza y ejecútela para escuchar en el puerto por el que circula su tráfico de Amazon S3 al mismo tiempo que envía otra solicitud HTTP de Amazon S3.

Para solicitudes HTTP, el par de ID de las solicitudes se verá de la siguiente manera:

```
x-amz-request-id: 79104EXAMPLEB723
x-amz-id-2: IOwQ4fDEXAMPLEQM+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km
```

 Note

Las solicitudes HTTPS se cifran y ocultan en la mayoría de las capturas de paquetes.

Utilización de un navegador web para obtener ID de solicitudes

La mayoría de los navegadores web tienen herramientas para desarrolladores que le permiten ver encabezados de solicitudes.

Para las solicitudes basadas en navegador web que devuelven un error, el par de ID de solicitudes se verá como en los siguientes ejemplos.

```
<Error><Code>AccessDenied</Code><Message>Access Denied</Message>
<RequestId>79104EXAMPLEB723</RequestId><HostId>IOwQ4fDEXAMPLEQM
+ey7N9WgVhSnQ6JEXAMPLEZb7hSQDASK+Jd1vEXAMPLEa3Km</HostId></Error>
```

Para obtener el par de ID de solicitudes de solicitudes realizadas correctamente, utilice las herramientas para desarrolladores de su navegador para ver los encabezados de respuesta HTTP.

Uso de los SDK de AWS para obtener los ID de solicitudes

En las siguientes secciones, se incluye información para configurar registros con AWS SDK diferentes.

 Note

Si bien puede habilitar registros detallados en cada solicitud y respuesta, no se recomienda habilitar registros en sistemas de producción, porque unas respuestas o solicitudes de gran tamaño pueden causar un retraso significativo en una aplicación.

Para solicitudes de AWS SDK, el par de ID de solicitudes se verá como:

```
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++

El tipo de registrador y el detalle se especifican durante la inicialización del SDK en el argumento `SDKOptions`. El ejemplo siguiente especifica el nivel de detalle como `LogLevel::Debug`.

El registrador predeterminado escribirá en el sistema de archivos y el nombre del archivo seguirá la siguiente convención `aws_sdk_YYYY-MM-DD-HH.log`. El registrador crea un nuevo archivo cada hora.

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

Para obtener más información, consulte [¿Cómo activo el registro?](#) en la wiki del AWS SDK para C++ en GitHub.

Go

Puede configurar el registro mediante SDK para Go. Para obtener más información, consulte [Registro](#) en la Guía para desarrolladores de AWS SDK para Go v2.

Java

Puede habilitar el registro para solicitudes o respuestas específicas para detectar y devolver solo encabezados relevantes. Para ello, importe la clase `com.amazonaws.services.s3.S3ResponseMetadata`. Después, podrá guardar la solicitud en una variable antes de ejecutar la solicitud real. Llame a `getCachedResponseMetadata` para obtener la solicitud o respuesta registrada `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()`.

```
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());
```

Además, puede utilizar registros detallados de cada solicitud y respuesta de Java. Para obtener más información, consulte [Registro detallado en red](#) en la Guía del desarrollador de AWS SDK para Java.

JavaScript

AWS SDK para JavaScript dispone de un registrador integrado, por lo que puede registrar las llamadas a la API que realiza con ello. Para activar el registrador e imprimir las entradas de registro en la consola, agregue la siguiente instrucción al código:

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

Para obtener más información, consulte [Registro de llamadas de AWS SDK para JavaScript](#) en la Guía para desarrolladores de AWS SDK para JavaScript.

Kotlin

Con AWS SDK para Kotlin, puede especificar el modo de registro para los mensajes de red mediante código o la configuración del entorno. Puede configurar el modo de registro para las solicitudes HTTP y las respuestas HTTP.

Para optar por un registro adicional, defina la propiedad `logMode` cuando cree un cliente de servicio:

```
import aws.smithy.kotlin.runtime.client.LogMode  
  
// ...  
  
val client = S3Client {  
    // ...  
    logMode = LogMode.LogRequestWithBody + LogMode.LogResponse  
}
```

Como alternativa, puede configurar el modo de registro mediante una variable de entorno:

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

Para obtener más información, consulte [Registro](#) en la Guía para desarrolladores de AWS SDK para Kotlin.

.NET

Puede configurar registros en SDK para .NET con la herramienta de registro `System.Diagnostics` integrada. Para obtener más información, consulte la publicación [Registro con SDK para .NET](#) del Blog de desarrolladores de AWS.

Note

De forma predeterminada, el registro devuelto solo incluye información de errores. El archivo de configuración debe incluir `AWSLogMetrics` (y, de forma opcional, `AWSResponseLogging`) para obtener los ID de solicitudes.

PHP

Puede obtener información de depuración, incluidos los datos enviados a través de la red, estableciendo la opción de depuración en `true` en un creador de clientes.

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

Para obtener más información, consulte [¿Cómo puedo ver qué datos se envían a través de la red?](#) en la Guía para desarrolladores de AWS SDK para PHP.

Python (Boto3)

AWS SDK para Python (Boto3) le permite registrar respuestas específicas. Puede utilizar esta función para capturar solo los encabezados relevantes. En el siguiente código se muestra cómo registrar partes de la respuesta en un archivo:

```
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'])
```

También puede detectar excepciones y registrar información relevante cuando se produce una excepción. Para obtener más información, consulte el temas sobre cómo [obtener información útil de las respuestas de error](#) en la Referencia de la API de AWS SDK para Python (Boto).

Además, puede configurar Boto3 para generar registros de depuración detallados mediante el siguiente código:

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

Para obtener más información, consulte [set_stream_logger](#) en la Referencia de la API de AWS SDK para Python (Boto).

Ruby

Puede obtener los ID de solicitudes mediante el SDK para Ruby versión 3. Puede habilitar el registro de red HTTP en el cliente mediante el siguiente código:

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

Note

El registro de red puede incluir información confidencial, como el ID de clave de acceso. La información confidencial debe sanearse antes de compartirla con AWS Support.

También puede encontrar el ID de solicitud del objeto de contexto de la solicitud en la respuesta o el error de la solicitud:

```
# 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]
```

Para obtener más información, consulte [Depuración mediante información de rastreo de red de un cliente de AWS SDK para Ruby](#) en la Guía para desarrolladores de AWS SDK para Ruby.

Rust

Para habilitar el registro, agregue la caja `tracing-subscriber` e inicialícela en su aplicación Rust.

Agregue la biblioteca de rastreo al archivo `Cargo.toml`:

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

A continuación, en el código de Rust, inicialice el registrador en la función principal antes de llamar a cualquier operación del SDK:

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

Para obtener más información, consulte [Configuración y uso de registros en AWS SDK para Rust](#) en la Guía para desarrolladores de AWS SDK para Rust.

Operaciones de API por objetos de Amazon S3 compatibles con las tablas de S3

La siguiente tabla incluye las operaciones de API por objetos de S3 y los encabezados correspondientes compatibles con las tablas de S3. Para obtener una lista de las API de las Tablas de Amazon S3, consulte [Tablas de Amazon S3](#). Para obtener más información sobre Tablas de Amazon S3, consulte [Trabajo con Tablas de Amazon S3 y buckets de tablas](#) en la Guía del usuario de Amazon S3.

API por objeto	Encabezados admitidos	Notas	
AbortMultipartUpload	x-amz-expected-bucket-owner	Ninguno	
CompleteMultipartUpload	x-amz-checksum-crc32	Ninguna	
	x-amz-checksum-crc32c	Ninguna	
	x-amz-checksum-sha1	Ninguna	
	x-amz-checksum-sha256	Ninguna	
	x-amz-expected-bucket-owner	Ninguna	
	If-Match	Ninguna	
	If-None-Match	Ninguno	
CreateMultipartUpload	x-amz-acl: ACL	Para las tablas de S3, el valor predeterminado es bucket-owner-full-control y no se puede cambiar.	
	Cache-Control	Ninguno	
	Content-Disposition	Ninguna	
	Content-Encoding	Ninguna	

API por objeto	Encabezados admitidos	Notas	
	Content-Language	Ninguna	
	Content-Type	Ninguna	
	Expires	Ninguna	
	x-amz-checksum-algorithm	Ninguno	
	x-amz-storage-class	Para las tablas de S3, el valor predeterminado es STANDARD y no se puede cambiar.	
	x-amz-server-side-encryption	Para las tablas de S3, el valor predeterminado es (SSE-S3) (AES256) y no se puede cambiar.	
GetObject	If-Match	Ninguno	
	If-Modified-Since	Ninguna	
	If-None-Match	Ninguna	
	If-Unmodified-Since	Ninguna	
	Range	Ninguna	
	x-amz-expected-bucket-owner	Ninguna	
	x-amz-checksum-mode	Ninguna	

API por objeto	Encabezados admitidos	Notas	
HeadObject	If-Match	Ninguna	
	If-Modified-Since	Ninguna	
	If-None-Match	Ninguna	
	If-Unmodified-Since	Ninguna	
	Range	Ninguna	
	x-amz-checksum-mode	Ninguna	
	x-amz-expected-bucket-owner	Ninguna	
ListParts	x-amz-expected-bucket-owner	Ninguno	
PutObject	x-amz-acl: ACL	Para las tablas de S3, el valor predeterminado es bucket-owner-full-control y no se puede cambiar.	
	Cache-Control	Ninguno	
	Content-Disposition	Ninguna	
	Content-Encoding	Ninguna	
	Content-Language	Ninguna	
	Content-Length	Ninguna	

API por objeto	Encabezados admitidos	Notas	
	Content-MD5	Ninguna	
	Content-Type	Ninguna	
	x-amz-sdk-checksum-algorithm	Ninguna	
	x-amz-checksum-crc32	Ninguna	
	x-amz-checksum-crc32c	Ninguna	
	x-amz-checksum-sha1	Ninguna	
	x-amz-checksum-sha256	Ninguna	
	Expires	Ninguna	
	If-None-Match	Ninguna	
	x-amz-expected-bucket-owner	Ninguno	
	x-amz-storage-classes	Para las tablas de S3, el valor predeterminado es STANDARD y no se puede cambiar.	
	x-amz-server-side-encryption	Para las tablas de S3, el valor predeterminado es (SSE-S3) (AES256) y no se puede cambiar.	

API por objeto	Encabezados admitidos	Notas	
UploadPart	Content-Length	Ninguno	
	Content-MD5	Ninguna	
	x-amz-checksum-crc32	Ninguna	
	x-amz-checksum-crc32c	Ninguna	
	x-amz-checksum-sha1	Ninguna	
	x-amz-checksum-sha256	Ninguna	
	x-amz-expected-bucket-owner	Ninguno	

Ejemplos de código de Amazon S3 con SDK de AWS

Los siguientes ejemplos de código muestran cómo utilizar Amazon S3 con un kit de desarrollo de software (SDK) de AWS.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Ejemplos de código

- [Ejemplos de código de Amazon S3 con SDK de AWS](#)
 - [Ejemplos básicos de Amazon S3 con SDK de AWS](#)
 - [Introducción a Amazon S3](#)
 - [Obtención de información de Amazon S3 con un SDK de AWS](#)
 - [Acciones de Amazon S3 con SDK de AWS](#)
 - [Uso de AbortMultipartUpload con un AWS SDK o una CLI](#)
 - [Uso de CompleteMultipartUpload con un SDK de AWS o la CLI](#)
 - [Uso de CopyObject con un SDK de AWS o la CLI](#)
 - [Uso de CreateBucket con un SDK de AWS o la CLI](#)
 - [Uso de CreateMultiRegionAccessPoint con un SDK de AWS](#)
 - [Uso de CreateMultipartUpload con un SDK de AWS o la CLI](#)
 - [Uso de CreatePresignedPost con un SDK de AWS](#)
 - [Uso de DeleteBucket con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketAnalyticsConfiguration con una CLI](#)
 - [Uso de DeleteBucketCors con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketEncryption con una CLI](#)
 - [Utilizar DeleteBucketInventoryConfiguration con una CLI](#)
 - [Uso de DeleteBucketLifecycle con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketMetricsConfiguration con una CLI](#)
 - [Uso de DeleteBucketPolicy con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketReplication con una CLI](#)
 - [Utilizar DeleteBucketTagging con una CLI](#)

- [Uso de DeleteBucketWebsite con un SDK de AWS o la CLI](#)
- [Uso de DeleteObject con un SDK de AWS o la CLI](#)
- [Utilizar DeleteObjectTagging con una CLI](#)
- [Uso de DeleteObjects con un SDK de AWS o la CLI](#)
- [Utilizar DeletePublicAccessBlock con una CLI](#)
- [Utilizar GetBucketAccelerateConfiguration con una CLI](#)
- [Uso de GetBucketAcl con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketAnalyticsConfiguration con una CLI](#)
- [Uso de GetBucketCors con un SDK de AWS o la CLI](#)
- [Uso de GetBucketEncryption con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketInventoryConfiguration con una CLI](#)
- [Uso de GetBucketLifecycleConfiguration con un SDK de AWS o la CLI](#)
- [Uso de GetBucketLocation con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketLogging con una CLI](#)
- [Utilizar GetBucketMetricsConfiguration con una CLI](#)
- [Utilizar GetBucketNotification con una CLI](#)
- [Uso de GetBucketPolicy con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketPolicyStatus con una CLI](#)
- [Uso de GetBucketReplication con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketRequestPayment con una CLI](#)
- [Utilizar GetBucketTagging con una CLI](#)
- [Utilizar GetBucketVersioning con una CLI](#)
- [Uso de GetBucketWebsite con un SDK de AWS o la CLI](#)
- [Uso de GetObject con un SDK de AWS o la CLI](#)
- [Uso de GetObjectAcl con un SDK de AWS o la CLI](#)
- [Uso de GetObjectAttributes con un SDK de AWS o la CLI](#)
- [Uso de GetObjectLegalHold con un SDK de AWS o la CLI](#)
- [Uso de GetObjectLockConfiguration con un SDK de AWS o la CLI](#)
- [Uso de GetObjectRetention con un SDK de AWS o la CLI](#)
- [Utilizar GetObjectTagging con una CLI](#)

- [Utilizar GetPublicAccessBlock con una CLI](#)
- [Uso de HeadBucket con un SDK de AWS o la CLI](#)
- [Uso de HeadObject con un SDK de AWS o la CLI](#)
- [Utilizar ListBucketAnalyticsConfigurations con una CLI](#)
- [Utilizar ListBucketInventoryConfigurations con una CLI](#)
- [Uso de ListBuckets con un SDK de AWS o la CLI](#)
- [Uso de ListMultipartUploads con un SDK de AWS o la CLI](#)
- [Uso de ListObjectVersions con un SDK de AWS o la CLI](#)
- [Utilizar ListObjects con una CLI](#)
- [Uso de ListObjectsV2 con un SDK de AWS o la CLI](#)
- [Uso de PutBucketAccelerateConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketAcl con un SDK de AWS o la CLI](#)
- [Uso de PutBucketCors con un SDK de AWS o la CLI](#)
- [Uso de PutBucketEncryption con un SDK de AWS o la CLI](#)
- [Uso de PutBucketLifecycleConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketLogging con un SDK de AWS o la CLI](#)
- [Utilizar PutBucketNotification con una CLI](#)
- [Uso de PutBucketNotificationConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketPolicy con un SDK de AWS o la CLI](#)
- [Uso de PutBucketReplication con un SDK de AWS o la CLI](#)
- [Utilizar PutBucketRequestPayment con una CLI](#)
- [Utilizar PutBucketTagging con una CLI](#)
- [Uso de PutBucketVersioning con un SDK de AWS o la CLI](#)
- [Uso de PutBucketWebsite con un SDK de AWS o la CLI](#)
- [Uso de PutObject con un SDK de AWS o la CLI](#)
- [Uso de PutObjectAcl con un SDK de AWS o la CLI](#)
- [Uso de PutObjectLegalHold con un SDK de AWS o la CLI](#)
- [Uso de PutObjectLockConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutObjectRetention con un SDK de AWS o la CLI](#)
- [Uso de RestoreObject con un SDK de AWS o la CLI](#)

- [Uso de SelectObjectContent con un SDK de AWS o la CLI](#)
- [Uso de UploadPart con un SDK de AWS o la CLI](#)
- [Uso de UploadPartCopy con un SDK de AWS o la CLI](#)
- [Escenarios de Amazon S3 con SDK de AWS](#)
 - [Comprobación de la existencia de un bucket](#)
 - [Comprobación de si existe un objeto](#)
 - [Convierta texto en voz y de nuevo a texto con un SDK de AWS](#)
 - [Cree una URL prefirmada para Amazon S3 mediante un SDK de AWS](#)
 - [Creación de una aplicación de administración de activos fotográficos que permita a los usuarios administrar las fotos mediante etiquetas](#)
 - [Una página web que indica los objetos de Amazon S3 que usan un SDK de AWS](#)
 - [Creación de una aplicación de exploración de Amazon Textract](#)
 - [Eliminar todos los objetos de un bucket de Amazon S3 dado mediante un SDK de AWS](#)
 - [Eliminación de las cargas multiparte incompletas a Amazon S3 mediante un AWS SDK](#)
 - [Detección de EPI en imágenes con Amazon Rekognition mediante un AWS SDK](#)
 - [Detecte entidades en el texto extraído de una imagen con un SDK de AWS](#)
 - [Detecte rostros en una imagen con un SDK de AWS](#)
 - [Detección de personas y objetos en un vídeo con Amazon Rekognition mediante un AWS SDK](#)
 - [Detección de personas y objetos en un vídeo con Amazon Rekognition mediante un SDK de AWS](#)
 - [Descargar los “directorios” de S3 desde un bucket de Amazon Simple Storage Service \(Amazon S3\)](#)
 - [Descargar todos los objetos de un bucket de Amazon Simple Storage Service \(Amazon S3\) en un directorio local](#)
 - [Descargar un flujo de tamaño desconocido en un objeto de Amazon S3 mediante un SDK de AWS](#)
 - [Obtención de un objeto de Amazon S3 desde un punto de acceso de varias regiones con un SDK de AWS](#)
 - [Obtenga un objeto de un bucket de Amazon S3 con un SDK de AWS al especificar un encabezado If-Modified-Since](#)
- [Introducción al cifrado de objetos de Amazon S3 con un SDK de AWS](#)

- [Introducción a etiquetas de objetos de Amazon S3 con un SDK de AWS](#)
- [Introducción a los clústeres de procesamiento de macrodatos](#)
- [Introducción a la administración de la configuración](#)
- [Introducción a la extracción de texto de documentos](#)
- [Introducción a los almacenes de características de machine learning](#)
- [Introducción al almacenamiento de objetos](#)
- [Introducción al análisis de consultas](#)
- [Trabajo con las características de bloqueo de objetos de Amazon S3 mediante un SDK de AWS](#)
- [Cómo realizar solicitudes condicionales de Amazon S3 mediante un SDK de AWS](#)
- [Administre listas de control de acceso \(ACL\) para buckets de Amazon S3 con un SDK de AWS](#)
- [Administración de mensajes grandes de Amazon SQS mediante Amazon S3 con un AWS SDK](#)
- [Administre objetos de Amazon S3 con control de versiones en lotes con una función de Lambda mediante un SDK de AWS](#)
- [Analizar los URI de Amazon S3 mediante un SDK de AWS](#)
- [Ejecución de una copia multiparte de un objeto de Amazon S3 con un SDK de AWS](#)
- [Recepción y procesamiento de notificaciones de eventos de Amazon S3 mediante un AWS SDK](#)
- [Guarde EXIF y otra información de la imagen con un SDK de AWS](#)
- [Envío de notificaciones de eventos de S3 a Amazon EventBridge mediante un AWS SDK](#)
- [Realización de un seguimiento de la carga o descarga de un objeto de Amazon S3 mediante un AWS SDK](#)
- [Transformación de datos para su aplicación con S3 Object Lambda](#)
- [Ejemplos de enfoques para pruebas unitarias y de integración con un SDK de AWS](#)
- [Cargar de forma recursiva un directorio local en un bucket de Amazon Simple Storage Service \(Amazon S3\)](#)
- [Cargar o descargar archivos de gran tamaño desde y hacia Amazon S3 con un SDK de AWS](#)
- [Carga de un flujo de tamaño desconocido en un objeto de Amazon S3 mediante un SDK de AWS](#)

- [Uso de sumas de comprobación para trabajar con un objeto de Amazon S3 con un SDK de AWS](#)
- [Trabajo con las características de integridad de objetos de Amazon S3 utilizando un AWS SDK](#)
- [Trabajo con objetos con control de versiones de Amazon S3 con un SDK de AWS](#)
- [Ejemplos sin servidor de Amazon S3](#)
 - [Invocación de una función de Lambda desde un desencadenador de Amazon S3](#)
- [Ejemplos de código de Amazon S3 Control con los SDK de AWS](#)
- [Ejemplos básicos de Amazon S3 Control con los SDK de AWS](#)
 - [Hello Amazon S3 Control](#)
 - [Funcionalidades básicas de Amazon S3 Control con un SDK de AWS](#)
 - [Acciones de Amazon S3 Control con los SDK de AWS](#)
 - [Uso de CreateJob con un AWS SDK o una CLI](#)
 - [Uso de DeleteJobTagging con un SDK de AWS](#)
 - [Uso de DescribeJob con un SDK de AWS o la CLI](#)
 - [Uso de GetJobTagging con un SDK de AWS](#)
 - [Uso de PutJobTagging con un SDK de AWS](#)
 - [Uso de UpdateJobPriority con un SDK de AWS o la CLI](#)
 - [Uso de UpdateJobStatus con un SDK de AWS o la CLI](#)
- [Ejemplos de código de S3 Directory Buckets con los SDK de AWS](#)
- [Ejemplos básicos de S3 Directory Buckets con los SDK de AWS](#)
 - [Hello Amazon S3 directory buckets](#)
 - [Funcionalidades básicas de S3 Directory Buckets con un SDK AWS](#)
 - [Acciones de S3 Directory Buckets con los SDK de AWS](#)
 - [Uso de AbortMultipartUpload con un AWS SDK](#)
 - [Uso de CompleteMultipartUpload con un SDK de AWS](#)
 - [Uso de CopyObject con un SDK de AWS](#)
 - [Uso de CreateBucket con un SDK de AWS](#)
 - [Uso de CreateMultipartUpload con un SDK de AWS](#)
 - [Uso de CreateSession con un SDK de AWS](#)
 - [Uso de DeleteBucket con un SDK de AWS](#)

- [Uso de DeleteBucketEncryption con un SDK de AWS](#)
- [Uso de DeleteBucketPolicy con un SDK de AWS](#)
- [Uso de DeleteObject con un SDK de AWS](#)
- [Uso de DeleteObjects con un SDK de AWS](#)
- [Uso de GetBucketEncryption con un SDK de AWS](#)
- [Uso de GetBucketPolicy con un SDK de AWS](#)
- [Uso de GetObject con un SDK de AWS](#)
- [Uso de GetObjectAttributes con un SDK de AWS](#)
- [Uso de HeadBucket con un SDK de AWS](#)
- [Uso de HeadObject con un SDK de AWS](#)
- [Uso de ListDirectoryBuckets con un SDK de AWS](#)
- [Uso de ListMultipartUploads con un SDK de AWS](#)
- [Uso de ListObjectsV2 con un SDK de AWS](#)
- [Uso de ListParts con un SDK de AWS](#)
- [Uso de PutBucketEncryption con un SDK de AWS](#)
- [Uso de PutBucketPolicy con un SDK de AWS](#)
- [Uso de PutObject con un SDK de AWS](#)
- [Uso de UploadPart con un SDK de AWS](#)
- [Uso de UploadPartCopy con un SDK de AWS](#)
- [Escenarios de S3 Directory Buckets con los SDK de AWS](#)
 - [Crear una URL previamente firmada de Amazon S3 Directory Buckets para obtener un objeto con un SDK de AWS](#)

Ejemplos de código de Amazon S3 con SDK de AWS

Los siguientes ejemplos de código muestran cómo utilizar Amazon S3 con un kit de desarrollo de software (SDK) de AWS.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Amazon S3

- [Ejemplos básicos de Amazon S3 con SDK de AWS](#)
 - [Introducción a Amazon S3](#)
 - [Obtención de información de Amazon S3 con un SDK de AWS](#)
 - [Acciones de Amazon S3 con SDK de AWS](#)
 - [Uso de AbortMultipartUpload con un AWS SDK o una CLI](#)
 - [Uso de CompleteMultipartUpload con un SDK de AWS o la CLI](#)
 - [Uso de CopyObject con un SDK de AWS o la CLI](#)
 - [Uso de CreateBucket con un SDK de AWS o la CLI](#)
 - [Uso de CreateMultiRegionAccessPoint con un SDK de AWS](#)
 - [Uso de CreateMultipartUpload con un SDK de AWS o la CLI](#)
 - [Uso de CreatePresignedPost con un SDK de AWS](#)
 - [Uso de DeleteBucket con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketAnalyticsConfiguration con una CLI](#)
 - [Uso de DeleteBucketCors con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketEncryption con una CLI](#)
 - [Utilizar DeleteBucketInventoryConfiguration con una CLI](#)
 - [Uso de DeleteBucketLifecycle con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketMetricsConfiguration con una CLI](#)
 - [Uso de DeleteBucketPolicy con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketReplication con una CLI](#)
 - [Utilizar DeleteBucketTagging con una CLI](#)

- [Uso de DeleteBucketWebsite con un SDK de AWS o la CLI](#)
- [Uso de DeleteObject con un SDK de AWS o la CLI](#)
- [Utilizar DeleteObjectTagging con una CLI](#)
- [Uso de DeleteObjects con un SDK de AWS o la CLI](#)
- [Utilizar DeletePublicAccessBlock con una CLI](#)
- [Utilizar GetBucketAccelerateConfiguration con una CLI](#)
- [Uso de GetBucketAcl con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketAnalyticsConfiguration con una CLI](#)
- [Uso de GetBucketCors con un SDK de AWS o la CLI](#)
- [Uso de GetBucketEncryption con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketInventoryConfiguration con una CLI](#)
- [Uso de GetBucketLifecycleConfiguration con un SDK de AWS o la CLI](#)
- [Uso de GetBucketLocation con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketLogging con una CLI](#)
- [Utilizar GetBucketMetricsConfiguration con una CLI](#)
- [Utilizar GetBucketNotification con una CLI](#)
- [Uso de GetBucketPolicy con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketPolicyStatus con una CLI](#)
- [Uso de GetBucketReplication con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketRequestPayment con una CLI](#)
- [Utilizar GetBucketTagging con una CLI](#)
- [Utilizar GetBucketVersioning con una CLI](#)
- [Uso de GetBucketWebsite con un SDK de AWS o la CLI](#)
- [Uso de GetObject con un SDK de AWS o la CLI](#)
- [Uso de GetObjectAcl con un SDK de AWS o la CLI](#)
- [Uso de GetObjectAttributes con un SDK de AWS o la CLI](#)
- [Uso de GetObjectLegalHold con un SDK de AWS o la CLI](#)
- [Uso de GetObjectLockConfiguration con un SDK de AWS o la CLI](#)
- [Uso de GetObjectRetention con un SDK de AWS o la CLI](#)
- [Utilizar GetObjectTagging con una CLI](#)

- [Utilizar GetPublicAccessBlock con una CLI](#)
- [Uso de HeadBucket con un SDK de AWS o la CLI](#)
- [Uso de HeadObject con un SDK de AWS o la CLI](#)
- [Utilizar ListBucketAnalyticsConfigurations con una CLI](#)
- [Utilizar ListBucketInventoryConfigurations con una CLI](#)
- [Uso de ListBuckets con un SDK de AWS o la CLI](#)
- [Uso de ListMultipartUploads con un SDK de AWS o la CLI](#)
- [Uso de ListObjectVersions con un SDK de AWS o la CLI](#)
- [Utilizar ListObjects con una CLI](#)
- [Uso de ListObjectsV2 con un SDK de AWS o la CLI](#)
- [Uso de PutBucketAccelerateConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketAcl con un SDK de AWS o la CLI](#)
- [Uso de PutBucketCors con un SDK de AWS o la CLI](#)
- [Uso de PutBucketEncryption con un SDK de AWS o la CLI](#)
- [Uso de PutBucketLifecycleConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketLogging con un SDK de AWS o la CLI](#)
- [Utilizar PutBucketNotification con una CLI](#)
- [Uso de PutBucketNotificationConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketPolicy con un SDK de AWS o la CLI](#)
- [Uso de PutBucketReplication con un SDK de AWS o la CLI](#)
- [Utilizar PutBucketRequestPayment con una CLI](#)
- [Utilizar PutBucketTagging con una CLI](#)
- [Uso de PutBucketVersioning con un SDK de AWS o la CLI](#)
- [Uso de PutBucketWebsite con un SDK de AWS o la CLI](#)
- [Uso de PutObject con un SDK de AWS o la CLI](#)
- [Uso de PutObjectAcl con un SDK de AWS o la CLI](#)
- [Uso de PutObjectLegalHold con un SDK de AWS o la CLI](#)
- [Uso de PutObjectLockConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutObjectRetention con un SDK de AWS o la CLI](#)
- [Uso de RestoreObject con un SDK de AWS o la CLI](#)

- [Uso de SelectObjectContent con un SDK de AWS o la CLI](#)
- [Uso de UploadPart con un SDK de AWS o la CLI](#)
- [Uso de UploadPartCopy con un SDK de AWS o la CLI](#)
- [Escenarios de Amazon S3 con SDK de AWS](#)
 - [Comprobación de la existencia de un bucket](#)
 - [Comprobación de si existe un objeto](#)
 - [Convierta texto en voz y de nuevo a texto con un SDK de AWS](#)
 - [Cree una URL prefirmada para Amazon S3 mediante un SDK de AWS](#)
 - [Creación de una aplicación de administración de activos fotográficos que permita a los usuarios administrar las fotos mediante etiquetas](#)
 - [Una página web que indica los objetos de Amazon S3 que usan un SDK de AWS](#)
 - [Creación de una aplicación de exploración de Amazon Textract](#)
 - [Eliminar todos los objetos de un bucket de Amazon S3 dado mediante un SDK de AWS](#)
 - [Eliminación de las cargas multiparte incompletas a Amazon S3 mediante un AWS SDK](#)
 - [Detección de EPI en imágenes con Amazon Rekognition mediante un AWS SDK](#)
 - [Detecte entidades en el texto extraído de una imagen con un SDK de AWS](#)
 - [Detecte rostros en una imagen con un SDK de AWS](#)
 - [Detección de personas y objetos en un vídeo con Amazon Rekognition mediante un AWS SDK](#)
 - [Detección de personas y objetos en un vídeo con Amazon Rekognition mediante un SDK de AWS](#)
 - [Descargar los “directorios” de S3 desde un bucket de Amazon Simple Storage Service \(Amazon S3\)](#)
 - [Descargar todos los objetos de un bucket de Amazon Simple Storage Service \(Amazon S3\) en un directorio local](#)
 - [Descargar un flujo de tamaño desconocido en un objeto de Amazon S3 mediante un SDK de AWS](#)
 - [Obtención de un objeto de Amazon S3 desde un punto de acceso de varias regiones con un SDK de AWS](#)
 - [Obtenga un objeto de un bucket de Amazon S3 con un SDK de AWS al especificar un encabezado If-Modified-Since](#)
 - [Introducción al cifrado de objetos de Amazon S3 con un SDK de AWS](#)

- [Introducción a etiquetas de objetos de Amazon S3 con un SDK de AWS](#)
- [Introducción a los clústeres de procesamiento de macrodatos](#)
- [Introducción a la administración de la configuración](#)
- [Introducción a la extracción de texto de documentos](#)
- [Introducción a los almacenes de características de machine learning](#)
- [Introducción al almacenamiento de objetos](#)
- [Introducción al análisis de consultas](#)
- [Trabajo con las características de bloqueo de objetos de Amazon S3 mediante un SDK de AWS](#)
- [Cómo realizar solicitudes condicionales de Amazon S3 mediante un SDK de AWS](#)
- [Administre listas de control de acceso \(ACL\) para buckets de Amazon S3 con un SDK de AWS](#)
- [Administración de mensajes grandes de Amazon SQS mediante Amazon S3 con un AWS SDK](#)
- [Administre objetos de Amazon S3 con control de versiones en lotes con una función de Lambda mediante un SDK de AWS](#)
- [Analizar los URI de Amazon S3 mediante un SDK de AWS](#)
- [Ejecución de una copia multiparte de un objeto de Amazon S3 con un SDK de AWS](#)
- [Recepción y procesamiento de notificaciones de eventos de Amazon S3 mediante un AWS SDK](#)
- [Guarde EXIF y otra información de la imagen con un SDK de AWS](#)
- [Envío de notificaciones de eventos de S3 a Amazon EventBridge mediante un AWS SDK](#)
- [Realización de un seguimiento de la carga o descarga de un objeto de Amazon S3 mediante un AWS SDK](#)
- [Transformación de datos para su aplicación con S3 Object Lambda](#)
- [Ejemplos de enfoques para pruebas unitarias y de integración con un SDK de AWS](#)
- [Cargar de forma recursiva un directorio local en un bucket de Amazon Simple Storage Service \(Amazon S3\)](#)
- [Cargar o descargar archivos de gran tamaño desde y hacia Amazon S3 con un SDK de AWS](#)
- [Carga de un flujo de tamaño desconocido en un objeto de Amazon S3 mediante un SDK de AWS](#)
- [Uso de sumas de comprobación para trabajar con un objeto de Amazon S3 con un SDK de AWS](#)
- [Trabajo con las características de integridad de objetos de Amazon S3 utilizando un AWS SDK](#)
- [Trabajo con objetos con control de versiones de Amazon S3 con un SDK de AWS](#)
- [Ejemplos sin servidor de Amazon S3](#)

- [Invocación de una función de Lambda desde un desencadenador de Amazon S3](#)

Ejemplos básicos de Amazon S3 con SDK de AWS

Los ejemplos de código siguientes muestran cómo usar los ejemplos básicos de Amazon Simple Storage Service con SDK de AWS.

Ejemplos

- [Introducción a Amazon S3](#)
- [Obtención de información de Amazon S3 con un SDK de AWS](#)
- [Acciones de Amazon S3 con SDK de AWS](#)
 - [Uso de AbortMultipartUpload con un AWS SDK o una CLI](#)
 - [Uso de CompleteMultipartUpload con un SDK de AWS o la CLI](#)
 - [Uso de CopyObject con un SDK de AWS o la CLI](#)
 - [Uso de CreateBucket con un SDK de AWS o la CLI](#)
 - [Uso de CreateMultiRegionAccessPoint con un SDK de AWS](#)
 - [Uso de CreateMultipartUpload con un SDK de AWS o la CLI](#)
 - [Uso de CreatePresignedPost con un SDK de AWS](#)
 - [Uso de DeleteBucket con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketAnalyticsConfiguration con una CLI](#)
 - [Uso de DeleteBucketCors con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketEncryption con una CLI](#)
 - [Utilizar DeleteBucketInventoryConfiguration con una CLI](#)
 - [Uso de DeleteBucketLifecycle con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketMetricsConfiguration con una CLI](#)
 - [Uso de DeleteBucketPolicy con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteBucketReplication con una CLI](#)
 - [Utilizar DeleteBucketTagging con una CLI](#)
 - [Uso de DeleteBucketWebsite con un SDK de AWS o la CLI](#)
 - [Uso de DeleteObject con un SDK de AWS o la CLI](#)
 - [Utilizar DeleteObjectTagging con una CLI](#)
 - [Uso de DeleteObjects con un SDK de AWS o la CLI](#)

- [Utilizar DeletePublicAccessBlock con una CLI](#)
- [Utilizar GetBucketAccelerateConfiguration con una CLI](#)
- [Uso de GetBucketAcl con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketAnalyticsConfiguration con una CLI](#)
- [Uso de GetBucketCors con un SDK de AWS o la CLI](#)
- [Uso de GetBucketEncryption con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketInventoryConfiguration con una CLI](#)
- [Uso de GetBucketLifecycleConfiguration con un SDK de AWS o la CLI](#)
- [Uso de GetBucketLocation con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketLogging con una CLI](#)
- [Utilizar GetBucketMetricsConfiguration con una CLI](#)
- [Utilizar GetBucketNotification con una CLI](#)
- [Uso de GetBucketPolicy con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketPolicyStatus con una CLI](#)
- [Uso de GetBucketReplication con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketRequestPayment con una CLI](#)
- [Utilizar GetBucketTagging con una CLI](#)
- [Utilizar GetBucketVersioning con una CLI](#)
- [Uso de GetBucketWebsite con un SDK de AWS o la CLI](#)
- [Uso de GetObject con un SDK de AWS o la CLI](#)
- [Uso de GetObjectAcl con un SDK de AWS o la CLI](#)
- [Uso de GetObjectAttributes con un SDK de AWS o la CLI](#)
- [Uso de GetObjectLegalHold con un SDK de AWS o la CLI](#)
- [Uso de GetObjectLockConfiguration con un SDK de AWS o la CLI](#)
- [Uso de GetObjectRetention con un SDK de AWS o la CLI](#)
- [Utilizar GetObjectTagging con una CLI](#)
- [Utilizar GetPublicAccessBlock con una CLI](#)
- [Uso de HeadBucket con un SDK de AWS o la CLI](#)
- [Uso de HeadObject con un SDK de AWS o la CLI](#)
- [Utilizar ListBucketAnalyticsConfigurations con una CLI](#)

- [Utilizar ListBucketInventoryConfigurations con una CLI](#)
- [Uso de ListBuckets con un SDK de AWS o la CLI](#)
- [Uso de ListMultipartUploads con un SDK de AWS o la CLI](#)
- [Uso de ListObjectVersions con un SDK de AWS o la CLI](#)
- [Utilizar ListObjects con una CLI](#)
- [Uso de ListObjectsV2 con un SDK de AWS o la CLI](#)
- [Uso de PutBucketAccelerateConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketAcl con un SDK de AWS o la CLI](#)
- [Uso de PutBucketCors con un SDK de AWS o la CLI](#)
- [Uso de PutBucketEncryption con un SDK de AWS o la CLI](#)
- [Uso de PutBucketLifecycleConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketLogging con un SDK de AWS o la CLI](#)
- [Utilizar PutBucketNotification con una CLI](#)
- [Uso de PutBucketNotificationConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketPolicy con un SDK de AWS o la CLI](#)
- [Uso de PutBucketReplication con un SDK de AWS o la CLI](#)
- [Utilizar PutBucketRequestPayment con una CLI](#)
- [Utilizar PutBucketTagging con una CLI](#)
- [Uso de PutBucketVersioning con un SDK de AWS o la CLI](#)
- [Uso de PutBucketWebsite con un SDK de AWS o la CLI](#)
- [Uso de PutObject con un SDK de AWS o la CLI](#)
- [Uso de PutObjectAcl con un SDK de AWS o la CLI](#)
- [Uso de PutObjectLegalHold con un SDK de AWS o la CLI](#)
- [Uso de PutObjectLockConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutObjectRetention con un SDK de AWS o la CLI](#)
- [Uso de RestoreObject con un SDK de AWS o la CLI](#)
- [Uso de SelectObjectContent con un SDK de AWS o la CLI](#)
- [Uso de UploadPart con un SDK de AWS o la CLI](#)
- [Uso de UploadPartCopy con un SDK de AWS o la CLI](#)

Introducción a Amazon S3

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon S3.

.NET

SDK para .NET (v4)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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}");
}
}
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código del archivo de CMake CMakeLists.txt.

```
# 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})

```

Código del archivo de origen 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;
}

```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para C++.

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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%v\n", *bucket.Name)
    }
}
}
}

```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
        }
    }
}
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;  
}  
}  
};
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para JavaScript.

PHP

SDK para PHP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para PHP.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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()
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para Ruby.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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(())
}
```

Utilidades de S3ExampleError.

```
/// 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)
    }
}
```

- Para obtener información acerca de la API, consulte [ListBuckets](#) en la Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Obtención de información de Amazon S3 con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo:

- Crear un bucket y cargar un archivo en el bucket.
- Descargar un objeto desde un bucket.
- Copiar un objeto en una subcarpeta de un bucket.
- Obtención de una lista de los objetos de un bucket.
- Eliminar el bucket y todos los objetos que incluye.

.NET

SDK para .NET (v4)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecutar un escenario interactivo en el que se muestran las características de 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;
    }
}
```


Una clase contenedora para métodos del SDK de 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($"{filePath}\
        \{objectName}", true, CancellationTokens.None);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"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;
    }
}
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para .NET.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Bash

AWS CLI con Bash script

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

Las funciones de Amazon S3 utilizadas en este escenario.

```
#####
# 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
}

```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de comandos de AWS CLI.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

C++

SDK para C++

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
}

```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para C++.
- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Defina una estructura que envuelva las acciones de bucket y objeto utilizadas por el escenario.

```
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
            }
        }
        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
}

```

Ejecutar un escenario interactivo que muestra cómo trabajar con buckets y objetos de 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\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))
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para Go.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Un ejemplo de escenario.

```
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.");
        }
    }
}
}

```

Una clase contenedora que contiene las operaciones.

```

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.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);
        }
    });
}

/**
 * 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)
                .contentType((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;
}
}

```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x.
 - [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Primero, importe todos los módulos necesarios.

```
// 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";
```

Las importaciones anteriores hacen referencia a algunas utilidades auxiliares. Estas utilidades son locales del repositorio de GitHub vinculado al principio de esta sección. Consulte las siguientes implementaciones de esas utilidades a modo de referencia.

```
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`;
};

```

Los objetos se almacenan en “buckets”. Definamos una función para crear un nuevo bucket.

```

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;
};

```

Los buckets contienen «objetos». Esta función carga el contenido de un directorio al bucket en forma de objetos.

```

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.`);
}
};
```

Después de cargar los objetos, confirme que se hayan subido correctamente. Puede usar `ListObjects` para la comprobación. Utilizará la propiedad «Clave», pero también hay otras propiedades útiles en la respuesta.

```
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`);
};
```

A veces, es posible que quiera copiar un objeto de un bucket en otro bucket. Para ello, utilice el comando `CopyObject`.

```
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();
};
```

No existe ningún método de SDK para obtener varios objetos de un bucket. En su lugar, creará una lista de objetos para descargarlos e iterarlos.

```
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");
};
```

Ha llegado el momento de limpiar los recursos. Un bucket debe estar vacío para poder eliminarlo. Estas dos funciones vacían y eliminan el bucket.

```
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 función «principal» reúne todo. Si ejecuta este archivo directamente, se llamará a la función principal.

```
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);
}
};
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para JavaScript.
 - [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
getObjectFromMap(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!")
    }
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API del SDK de AWS para Kotlin.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

PHP

SDK para PHP

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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";
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para PHP.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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"))
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Python (Boto3).
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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__
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para Ruby.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para la caja binaria que ejecuta el escenario.

```
#![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(())
}

```

Acciones comunes utilizadas por el escenario.

```

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))
            }
        }
    }
}
}

```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Rust.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)

- [ListObjectsV2](#)
- [PutObject](#)

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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_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.
```

```

"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.

```

- Para detalles acerca de la API, consulte los siguientes temas en la Referencia de la API del SDK de AWS para SAP ABAP.
 - [CopyObject](#)

- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Swift

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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)
        }
    }
}
}

```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Swift.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Acciones de Amazon S3 con SDK de AWS

Los siguientes ejemplos de código muestran cómo llevar a cabo acciones individuales de Amazon S3 con los SDK de AWS. En cada ejemplo se incluye un enlace a GitHub, con instrucciones de configuración y ejecución del código.

Estos fragmentos llaman a la API de Amazon S3 y son fragmentos de código de programas más grandes que deben ejecutarse en contexto. Puede ver las acciones en contexto en [Escenarios de Amazon S3 con SDK de AWS](#).

Los siguientes ejemplos incluyen solo las acciones que se utilizan con mayor frecuencia. Para obtener una lista completa, consulte la [Referencia de la API de Amazon Simple Storage Service](#).

Ejemplos

- [Uso de AbortMultipartUpload con un AWS SDK o una CLI](#)
- [Uso de CompleteMultipartUpload con un SDK de AWS o la CLI](#)
- [Uso de CopyObject con un SDK de AWS o la CLI](#)
- [Uso de CreateBucket con un SDK de AWS o la CLI](#)
- [Uso de CreateMultiRegionAccessPoint con un SDK de AWS](#)
- [Uso de CreateMultipartUpload con un SDK de AWS o la CLI](#)
- [Uso de CreatePresignedPost con un SDK de AWS](#)

- [Uso de DeleteBucket con un SDK de AWS o la CLI](#)
- [Utilizar DeleteBucketAnalyticsConfiguration con una CLI](#)
- [Uso de DeleteBucketCors con un SDK de AWS o la CLI](#)
- [Utilizar DeleteBucketEncryption con una CLI](#)
- [Utilizar DeleteBucketInventoryConfiguration con una CLI](#)
- [Uso de DeleteBucketLifecycle con un SDK de AWS o la CLI](#)
- [Utilizar DeleteBucketMetricsConfiguration con una CLI](#)
- [Uso de DeleteBucketPolicy con un SDK de AWS o la CLI](#)
- [Utilizar DeleteBucketReplication con una CLI](#)
- [Utilizar DeleteBucketTagging con una CLI](#)
- [Uso de DeleteBucketWebsite con un SDK de AWS o la CLI](#)
- [Uso de DeleteObject con un SDK de AWS o la CLI](#)
- [Utilizar DeleteObjectTagging con una CLI](#)
- [Uso de DeleteObjects con un SDK de AWS o la CLI](#)
- [Utilizar DeletePublicAccessBlock con una CLI](#)
- [Utilizar GetBucketAccelerateConfiguration con una CLI](#)
- [Uso de GetBucketAcl con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketAnalyticsConfiguration con una CLI](#)
- [Uso de GetBucketCors con un SDK de AWS o la CLI](#)
- [Uso de GetBucketEncryption con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketInventoryConfiguration con una CLI](#)
- [Uso de GetBucketLifecycleConfiguration con un SDK de AWS o la CLI](#)
- [Uso de GetBucketLocation con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketLogging con una CLI](#)
- [Utilizar GetBucketMetricsConfiguration con una CLI](#)
- [Utilizar GetBucketNotification con una CLI](#)
- [Uso de GetBucketPolicy con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketPolicyStatus con una CLI](#)
- [Uso de GetBucketReplication con un SDK de AWS o la CLI](#)
- [Utilizar GetBucketRequestPayment con una CLI](#)

- [Utilizar GetBucketTagging con una CLI](#)
- [Utilizar GetBucketVersioning con una CLI](#)
- [Uso de GetBucketWebsite con un SDK de AWS o la CLI](#)
- [Uso de GetObject con un SDK de AWS o la CLI](#)
- [Uso de GetObjectAcl con un SDK de AWS o la CLI](#)
- [Uso de GetObjectAttributes con un SDK de AWS o la CLI](#)
- [Uso de GetObjectLegalHold con un SDK de AWS o la CLI](#)
- [Uso de GetObjectLockConfiguration con un SDK de AWS o la CLI](#)
- [Uso de GetObjectRetention con un SDK de AWS o la CLI](#)
- [Utilizar GetObjectTagging con una CLI](#)
- [Utilizar GetPublicAccessBlock con una CLI](#)
- [Uso de HeadBucket con un SDK de AWS o la CLI](#)
- [Uso de HeadObject con un SDK de AWS o la CLI](#)
- [Utilizar ListBucketAnalyticsConfigurations con una CLI](#)
- [Utilizar ListBucketInventoryConfigurations con una CLI](#)
- [Uso de ListBuckets con un SDK de AWS o la CLI](#)
- [Uso de ListMultipartUploads con un SDK de AWS o la CLI](#)
- [Uso de ListObjectVersions con un SDK de AWS o la CLI](#)
- [Utilizar ListObjects con una CLI](#)
- [Uso de ListObjectsV2 con un SDK de AWS o la CLI](#)
- [Uso de PutBucketAccelerateConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketAcl con un SDK de AWS o la CLI](#)
- [Uso de PutBucketCors con un SDK de AWS o la CLI](#)
- [Uso de PutBucketEncryption con un SDK de AWS o la CLI](#)
- [Uso de PutBucketLifecycleConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketLogging con un SDK de AWS o la CLI](#)
- [Utilizar PutBucketNotification con una CLI](#)
- [Uso de PutBucketNotificationConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutBucketPolicy con un SDK de AWS o la CLI](#)
- [Uso de PutBucketReplication con un SDK de AWS o la CLI](#)

- [Utilizar PutBucketRequestPayment con una CLI](#)
- [Utilizar PutBucketTagging con una CLI](#)
- [Uso de PutBucketVersioning con un SDK de AWS o la CLI](#)
- [Uso de PutBucketWebsite con un SDK de AWS o la CLI](#)
- [Uso de PutObject con un SDK de AWS o la CLI](#)
- [Uso de PutObjectAcl con un SDK de AWS o la CLI](#)
- [Uso de PutObjectLegalHold con un SDK de AWS o la CLI](#)
- [Uso de PutObjectLockConfiguration con un SDK de AWS o la CLI](#)
- [Uso de PutObjectRetention con un SDK de AWS o la CLI](#)
- [Uso de RestoreObject con un SDK de AWS o la CLI](#)
- [Uso de SelectObjectContent con un SDK de AWS o la CLI](#)
- [Uso de UploadPart con un SDK de AWS o la CLI](#)
- [Uso de UploadPartCopy con un SDK de AWS o la CLI](#)

Uso de **AbortMultipartUpload** con un AWS SDK o una CLI

En los siguientes ejemplos de código se muestra cómo se utiliza AbortMultipartUpload.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en su contexto en los siguientes ejemplos de código:

- [Eliminación de cargas multiparte incompletas](#)
- [Trabajo con la integridad de los objetos de Amazon S3](#)

C++

SDK para C++

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
}

```

- Para obtener información sobre la API, consulte [AbortMultipartUpload](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

Anular la carga multiparte especificada

El siguiente comando `abort-multipart-upload` anula una carga multiparte de la clave `multipart/01` en el bucket `amzn-s3-demo-bucket`.

```
aws s3api abort-multipart-upload \  
  --bucket amzn-s3-demo-bucket \  
  --key multipart/01 \  
  --upload-  
id dfRtDYU0WMCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P
```

El ID de carga requerido por este comando se genera mediante `create-multipart-upload` y también se puede recuperar con `list-multipart-uploads`.

- Para obtener información sobre la API, consulte [AbortMultipartUploads](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
        } 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;  
  }  
}
```

- Para obtener información sobre la API, consulte [AbortMultipartUpload](#) en la Referencia de la API de AWS SDK for Java 2.x.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando anula las cargas multiparte creadas hace menos de 5 días.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Ejemplo 2: este comando anula las cargas multiparte creadas antes del 2 de enero de, 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

Ejemplo 3: este comando anula las cargas multiparte creadas antes del 2 de enero de 2014 a las, 2014, 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- Para obtener información sobre la API, consulte [AbortMultipartUpload](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando anula las cargas multiparte creadas hace menos de 5 días.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -DaysBefore 5
```

Ejemplo 2: este comando anula las cargas multiparte creadas antes del 2 de enero de, 2014.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"Thursday, January 02, 2014"
```

Ejemplo 3: este comando anula las cargas multiparte creadas antes del 2 de enero de 2014 a las, 2014, 10:45:37.

```
Remove-S3MultipartUpload -BucketName amzn-s3-demo-bucket -InitiatedDate  
"2014/01/02 10:45:37"
```

- Para obtener información sobre la API, consulte [AbortMultipartUpload](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **CompleteMultipartUpload** con un SDK de AWS o la CLI

En los siguientes ejemplos de código se muestra cómo usar CompleteMultipartUpload.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en su contexto en los siguientes ejemplos de código:

- [Ejecución de una copia multiparte](#)
- [Usar sumas de comprobación](#)
- [Trabajo con la integridad de los objetos de Amazon S3](#)

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}

```

- Para obtener información sobre la API, consulte [CompleteMultipartUpload](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El siguiente comando completa una carga multiparte de la clave `multipart/01` en el bucket `amzn-s3-demo-bucket`:

```
aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-
id dfRtDYU0WMCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6YjlX.cph0gtNBtJ8P
```

El ID de carga requerido por este comando se genera mediante `create-multipart-upload` y también se puede recuperar con `list-multipart-uploads`.

La opción de carga multiparte del comando anterior utiliza una estructura JSON que describe las partes de la carga multiparte que se deben volver a ensamblar en el archivo completo. En este ejemplo, el prefijo `file://` se usa para cargar la estructura JSON desde un archivo de la carpeta local denominada `mpustruct`.

`mpustruct`:

```
{
  "Parts": [
    {
      "ETag": "e868e0f4719e394144ef36531ee6824c",
      "PartNumber": 1
    },
    {
      "ETag": "6bb2b12753d66fe86da4998aa33fffb0",
      "PartNumber": 2
    },
    {
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",
      "PartNumber": 3
    }
  ]
}
```

El valor de ETag de cada parte que se carga aparece cada vez que se carga una parte mediante el comando `upload-part` y también se puede recuperar mediante una llamada a `list-parts` o calcularse mediante la suma de comprobación MD5 de cada parte.

Salida:

```
{
  "ETag": "\"3944a9f7a4faab7f78788ff6210f63f0-3\"",
  "Bucket": "amzn-s3-demo-bucket",
  "Location": "https://amzn-s3-demo-bucket.s3.amazonaws.com/multipart%2F01",
  "Key": "multipart/01"
```

```
}
```

- Para obtener información sobre la API, consulte [CompleteMultipartUpload](#) en la Referencia de comandos de la AWS CLI.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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(),
    );
}
```

- Para ver los detalles de la API, consulte [CompleteMultipartUpload](#) en la Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **CopyObject** con un SDK de AWS o la CLI

En los siguientes ejemplos de código se muestra cómo se utiliza CopyObject.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Conceptos básicos](#)
- [Introducción al cifrado](#)
- [Introducción al almacenamiento de objetos](#)
- [Cómo realizar solicitudes condicionales](#)

.NET

SDK para .NET (v4)

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
        }
    }
}
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para .NET.

SDK para .NET

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Copiar un objeto mediante una solicitud condicional.

```
/// <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;
}
}
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para .NET.

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de comandos de AWS CLI.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
    }
}
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El siguiente comando copia un objeto de bucket-1 a bucket-2:

```
aws s3api copy-object --copy-source bucket-1/test.txt --key test.txt --  
bucket bucket-2
```

Salida:

```
{  
  "CopyObjectResult": {  
    "LastModified": "2015-11-10T01:07:25.000Z",  
    "ETag": "\"589c8b79c230a6ecd5a7e1d040a9a030\""  
  },  
  "VersionId": "YdnYvTCVDqRRFA.NFJjy36p0hxifM1kA"  
}
```

- Para obtener detalles de la API, consulte [CopyObject](#) en la Referencia de comandos de AWS CLI.

Go

SDK para Go V2

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}

```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Copie un objeto con 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);
}

```

Utilice un [S3TransferManager](#) para [copiar un objeto](#) de un bucket a otro. Vea el [archivo completo](#) y [pruébelo](#).

```

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();
    }
}
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Copie el objeto.

```
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;
        }
    }
};

```

Copiar el objeto con la condición de que su ETag no coincida con la proporcionada.

```
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"));  
    }  
}
```

Copiar el objeto con la condición de que su ETag no coincida con la proporcionada.

```
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"));
  }
}

```

Copiar el objeto con la condición de que se haya creado o modificado en un período de tiempo determinado.

```

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"));
  }
}

```

Copiar el objeto con la condición de que no se haya creado o modificado en un período de tiempo determinado.

```

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"));  
  }  
}
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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)
    }
}
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Copia sencilla de un objeto.

```
$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.");
}
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para PHP.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando copia el objeto “sample.txt” del bucket “test-files” al mismo bucket, pero con la nueva clave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey
sample-copy.txt
```

Ejemplo 2: este comando copia el objeto “sample.txt” del bucket “test-files” al mismo bucket “backup-files”, pero con la nueva clave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -  
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Ejemplo 3: este comando descarga el objeto “sample.txt” del bucket “test-files” a un archivo local con el nombre “local-sample.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-  
sample.txt
```

Ejemplo 4: descarga el objeto individual en el archivo especificado. El archivo descargado se encuentra en c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder  
c:\downloads
```

Ejemplo 5: descarga todos los objetos que coinciden con el prefijo de clave especificado en la carpeta local. La jerarquía de claves relativa se conservará como subcarpetas en la ubicación general de descarga.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:  
\downloads
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando copia el objeto “sample.txt” del bucket “test-files” al mismo bucket, pero con la nueva clave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -DestinationKey  
sample-copy.txt
```

Ejemplo 2: este comando copia el objeto “sample.txt” del bucket “test-files” al mismo bucket “backup-files”, pero con la nueva clave de “sample-copy.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-source-bucket -Key sample.txt -  
DestinationKey sample-copy.txt -DestinationBucket amzn-s3-demo-destination-bucket
```

Ejemplo 3: este comando descarga el objeto “sample.txt” del bucket “test-files” a un archivo local con el nombre “local-sample.txt”.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-  
sample.txt
```

Ejemplo 4: descarga el objeto individual en el archivo especificado. El archivo descargado se encuentra en c:\downloads\data\archive.zip

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key data/archive.zip -LocalFolder  
c:\downloads
```

Ejemplo 5: descarga todos los objetos que coinciden con el prefijo de clave especificado en la carpeta local. La jerarquía de claves relativa se conservará como subcarpetas en la ubicación general de descarga.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data -LocalFolder c:  
\downloads
```

Ejemplo 6: descarga un único objeto a un archivo local mediante la descarga paralela multiparte para mejorar el rendimiento en archivos de gran tamaño.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -LocalFile local-  
sample.txt -UseMultipartDownload
```

Ejemplo 7: descarga un objeto grande mediante el modo RANGE con un tamaño de parte personalizado de 64 MB. El modo RANGE utiliza solicitudes HTTP de rango de bytes y funciona para todos los objetos, independientemente de cómo se hayan cargado.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile  
local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -  
PartSize 64MB
```

Ejemplo 8: descarga todos los objetos que coinciden con el prefijo de clave en una carpeta local mediante la descarga multiparte con descargas de archivos simultáneas.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Ejemplo 9: descarga un único objeto mediante una descarga multiparte con 20 conexiones simultáneas para obtener el máximo rendimiento en redes de gran ancho de banda.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -LocalFile local-large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Ejemplo 10: descarga todos los objetos que coinciden con el prefijo de claves en una carpeta local mediante la descarga multiparte y continúa a pesar de los errores individuales de los archivos en lugar de cancelarla.

```
Copy-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -LocalFolder C:\local-data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

```

Copiar un objeto mediante una solicitud condicional.

```

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

```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Copie un objeto.

```

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__

```

Copie un objeto y añada cifrado del lado del servidor al objeto de destino.

```
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
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
end

```

```
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__
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para Ruby.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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(())
}

```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.

```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
    }
}
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK para Swift.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **CreateBucket** con un SDK de AWS o la CLI

En los siguientes ejemplos de código se muestra cómo se utiliza CreateBucket.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Conceptos básicos](#)
- [Introducción a la administración de la configuración](#)
- [Introducción a los almacenes de características de machine learning](#)
- [Introducción al almacenamiento de objetos](#)
- [Administración de mensajes grandes mediante S3](#)
- [Trabajo con objetos con control de versiones](#)

.NET

SDK para .NET (v4)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
    }
}
```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK para .NET.

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree un nuevo bucket con el bloqueo de objetos habilitado.

```
/// <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;
}
}

```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK para .NET.

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

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
}

```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de comandos de AWS CLI.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
}
```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

Ejemplo 1: Creación de un bucket

En los siguientes ejemplos de `create-bucket` se crea un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api create-bucket \
    --bucket amzn-s3-demo-bucket \
    --region us-east-1
```

Salida:

```
{
  "Location": "/amzn-s3-demo-bucket"
}
```

Para obtener más información, consulte [Crear un bucket](#) en la Guía del usuario de Amazon S3.

Ejemplo 2: Creación de un bucket con propietario obligatorio

En el siguiente ejemplo de `create-bucket` se crea un bucket denominado `amzn-s3-demo-bucket` que utiliza la configuración Aplicada al propietario del bucket de S3 Object Ownership.

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region us-east-1 \  
  --object-ownership BucketOwnerEnforced
```

Salida:

```
{  
  "Location": "/amzn-s3-demo-bucket"  
}
```

Para obtener más información, consulte [Control de la propiedad de objetos y desactivación de las ACL](#) en la Guía del usuario de Amazon S3.

Ejemplo 3: creación de un bucket fuera de la región us-east

En el siguiente ejemplo create-bucket, se crea un bucket denominado amzn-s3-demo-bucket en la región eu-west-1. Las regiones situadas fuera de us-east-1 requieren que se especifique el LocationConstraint correspondiente para poder crear el bucket en la región deseada.

```
aws s3api create-bucket \  
  --bucket amzn-s3-demo-bucket \  
  --region eu-west-1 \  
  --create-bucket-configuration LocationConstraint=eu-west-1
```

Salida:

```
{  
  "Location": "http://amzn-s3-demo-bucket.s3.amazonaws.com/"  
}
```

Para obtener más información, consulte [Crear un bucket](#) en la Guía del usuario de Amazon S3.

- Para obtener detalles de la API, consulte [CreateBucket](#) en la Referencia de comandos de AWS CLI.

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree un bucket con la configuración predeterminada.

```
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
}

```

Cree un bucket con el bloqueo de objetos y espere a que aparezca.

```

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
}
```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear un bucket.

```
/**
 * 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);
        }
    });
}

```

Cree un nuevo bucket con el bloqueo de objetos habilitado.

```

// 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");
}

```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear el bucket.

```
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;
        }
    }
};
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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")
    }
}
```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear un bucket.

```
$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.");
}
```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK para PHP.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree un bucket con la configuración predeterminada.

```
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

```

Cree un bucket con control de versiones con una configuración de ciclo de vida.

```

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
```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
end
```

```
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__
```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK para Ruby.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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))
        }
    })
}
```

```
}

```

- Para obtener información sobre la API, consulte [CreaCreateBucket](#) en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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_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.
```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
        }
    }
```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK para Swift.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **CreateMultiRegionAccessPoint** con un SDK de AWS

En el siguiente ejemplo de código se muestra cómo usar `CreateMultiRegionAccessPoint`.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Configure el cliente de control de S3 para enviar la solicitud a la región 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
}
```

Cree el punto de acceso de varias regiones.

```
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
                    },
                    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"
}
```

Espere a que el punto de acceso de varias regiones esté 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)
    }
}
```

- Para obtener información, consulte [Guía para desarrolladores del SDK de AWS SDK para Kotlin](#).
- Para obtener información sobre la API, consulte [CreateMultiRegionAccessPoint](#) en la Referencia de la API del SDK de AWS para Kotlin.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **CreateMultipartUpload** con un SDK de AWS o la CLI

En los siguientes ejemplos de código se muestra cómo se utiliza `CreateMultipartUpload`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en su contexto en los siguientes ejemplos de código:

- [Ejecución de una copia multiparte](#)
- [Usar sumas de comprobación](#)
- [Trabajo con la integridad de los objetos de Amazon S3](#)

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}

```

- Para obtener información acerca de la API, consulte [CreateMultipartUpload](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El siguiente comando crea una carga multiparte en el bucket `amzn-s3-demo-bucket` con la clave `multipart/01`:

```
aws s3api create-multipart-upload --bucket amzn-s3-demo-bucket --key 'multipart/01'
```

Salida:

```

{
    "Bucket": "amzn-s3-demo-bucket",
    "UploadId":
"dFrtdYU0WwCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3
    "Key": "multipart/01"
}

```

El archivo completo se denominará 01 en una carpeta llamada `multipart` en el bucket `amzn-s3-demo-bucket`. Guarde el ID de carga, la clave y el nombre del bucket para usarlos con el comando `upload-part`.

- Para obtener información sobre la API, consulte [CreateMultipartUpload](#) en la Referencia de comandos de la AWS CLI.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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?;

```

- Para obtener detalles sobre la API, consulte [CreateMultipartUpload](#) en la Referencia de la API del SDK AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **CreatePresignedPost** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `CreatePresignedPost`.

.NET

SDK para .NET (v4)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear una URL POST previamente firmada.

```
/// <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);
}
```

- Para obtener información sobre la API, consulte [CreatePresignedPost](#) en la Referencia de la API de AWS SDK para .NET.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteBucket** con un SDK de AWS o la CLI

En los siguientes ejemplos de código se muestra cómo se utiliza DeleteBucket.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Conceptos básicos](#)
- [Introducción a la administración de la configuración](#)
- [Introducción a los almacenes de características de machine learning](#)
- [Introducción al almacenamiento de objetos](#)

.NET

SDK para .NET (v4)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
    }
}
```

```
}
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK para .NET.

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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  
}
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de comandos de AWS CLI.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
}
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El comando siguiente elimina un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket --bucket amzn-s3-demo-bucket --region us-east-1
```

- Para obtener detalles de la API, consulte [DeleteBucket](#) en la Referencia de comandos de AWS CLI.

Go

SDK para Go V2

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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  
}
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);  
}
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine el bucket.

```
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;
  }
}
};
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK para JavaScript.

PHP

SDK para PHP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine un bucket vacío.

```
$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.");
}
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK para PHP.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando elimina todos los objetos y las versiones de los objetos del bucket “test-files” y, a continuación, elimina el bucket. El comando solicitará una confirmación antes de continuar. Añada el conmutador -Force para suprimir la confirmación. Tenga en cuenta que los buckets que no estén vacíos no se pueden eliminar.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando elimina todos los objetos y las versiones de los objetos del bucket “test-files” y, a continuación, elimina el bucket. El comando solicitará una confirmación antes de continuar. Añada el conmutador -Force para suprimir la confirmación. Tenga en cuenta que los buckets que no estén vacíos no se pueden eliminar.

```
Remove-S3Bucket -BucketName amzn-s3-demo-bucket -DeleteBucketContent
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK para Ruby.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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))
            }
        }
    }
}
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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  
    }  
}
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK para Swift.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **DeleteBucketAnalyticsConfiguration** con una CLI

En los siguientes ejemplos de código se muestra cómo se utiliza `DeleteBucketAnalyticsConfiguration`.

CLI

AWS CLI

Eliminar una configuración de análisis de un bucket

En el siguiente ejemplo de `delete-bucket-analytics-configuration`, se elimina la configuración de análisis para el bucket e ID especificados.

```
aws s3api delete-bucket-analytics-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 1
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [DeleteBucketAnalyticsConfiguration](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: el comando elimina el filtro de análisis con el nombre “testfilter” en el bucket de S3 indicado.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -
AnalyticsId 'testfilter'
```

- Para obtener información sobre la API, consulte [DeleteBucketAnalyticsConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: el comando elimina el filtro de análisis con el nombre “testfilter” en el bucket de S3 indicado.

```
Remove-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -
AnalyticsId 'testfilter'
```

- Para obtener información sobre la API, consulte [DeleteBucketAnalyticsConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteBucketCors** con un SDK de AWS o la CLI

En los siguientes ejemplos de código se muestra cómo se utiliza DeleteBucketCors.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
}
```


- Para obtener información sobre la API, consulte [DeleteBucketCors](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

El siguiente comando elimina la configuración de uso compartido de recursos entre orígenes desde un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-cors --bucket amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [DeleteBucketCors](#) en la Referencia de comandos de la AWS CLI.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [DeleteBucketCors](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
end
```

```
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
rescue Aws::Errors::ServiceError => e
  puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
#{e.message}"
  false
end

end
```

- Para obtener información sobre la API, consulte [DeleteBucketCors](#) en la Referencia de la API de AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [DeleteBucketCors](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **DeleteBucketEncryption** con una CLI

En los siguientes ejemplos de código se muestra cómo se utiliza DeleteBucketEncryption.

CLI

AWS CLI

Eliminar la configuración de cifrado del servidor de un bucket

En el siguiente ejemplo de delete-bucket-encryption, se elimina la configuración de cifrado del servidor del bucket especificado.

```
aws s3api delete-bucket-encryption \  
  --bucket amzn-s3-demo-bucket
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [DeleteBucketEncryption](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: esto deshabilita el cifrado habilitado para el bucket de S3 proporcionado.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
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
```

- Para obtener información sobre la API, consulte [DeleteBucketEncryption](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: esto deshabilita el cifrado habilitado para el bucket de S3 proporcionado.

```
Remove-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
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
```

- Para obtener información sobre la API, consulte [DeleteBucketEncryption](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **DeleteBucketInventoryConfiguration** con una CLI

En los siguientes ejemplos de código se muestra cómo se utiliza DeleteBucketInventoryConfiguration.

CLI

AWS CLI

Eliminar la configuración de inventario de un bucket

En el siguiente ejemplo de delete-bucket-inventory-configuration, se elimina la configuración de inventario con el ID 1 del bucket especificado.

```
aws s3api delete-bucket-inventory-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 1
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [DeleteBucketInventoryConfiguration](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando elimina el inventario denominado “testInventoryName” correspondiente al bucket de S3 en cuestión.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -  
InventoryId 'testInventoryName'
```

Salida:

```
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
```

- Para obtener información sobre la API, consulte [DeleteBucketInventoryConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando elimina el inventario denominado “testInventoryName” correspondiente al bucket de S3 en cuestión.

```
Remove-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -  
InventoryId 'testInventoryName'
```

Salida:

```
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
```

- Para obtener información sobre la API, consulte [DeleteBucketInventoryConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteBucketLifecycle** con un SDK de AWS o la CLI

En los siguientes ejemplos de código se muestra cómo se utiliza DeleteBucketLifecycle.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
}
```

```
}
```

- Para obtener información sobre la API, consulte [DeleteBucketLifecycle](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

El comando siguiente elimina una configuración del ciclo de vida de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-lifecycle --bucket amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [DeleteBucketLifecycle](#) en la Referencia de comandos de la AWS CLI.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [DeleteBucketLifecycle](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [DeleteBucketLifecycle](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **DeleteBucketMetricsConfiguration** con una CLI

En los siguientes ejemplos de código se muestra cómo se utiliza DeleteBucketMetricsConfiguration.

CLI

AWS CLI

Eliminar una configuración de métricas de un bucket

En el siguiente ejemplo de delete-bucket-metrics-configuration, se elimina la configuración de métricas para el bucket e ID especificados.

```
aws s3api delete-bucket-metrics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 123
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [DeleteBucketMetricsConfiguration](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: el comando elimina el filtro de métricas con el nombre “testmetrics” en el bucket de S3 indicado.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- Para obtener información sobre la API, consulte [DeleteBucketMetricsConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: el comando elimina el filtro de métricas con el nombre “testmetrics” en el bucket de S3 indicado.

```
Remove-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testmetrics'
```

- Para obtener información sobre la API, consulte [DeleteBucketMetricsConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteBucketPolicy** con un SDK de AWS o la CLI

En los siguientes ejemplos de código se muestra cómo se utiliza DeleteBucketPolicy.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
}
```

- Para obtener información sobre la API, consulte [DeleteBucketPolicy](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El comando siguiente elimina una política de bucket de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-policy --bucket amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [DeleteBucketPolicy](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
    }
}
```

- Para obtener información sobre la API, consulte [DeleteBucketPolicy](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine la política del bucket.

```
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;
    }
  }
};
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).

- Para obtener información sobre la API, consulte [DeleteBucketPolicy](#) en la Referencia de la API de AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {  
    val request =  
        DeleteBucketPolicyRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteBucketPolicy(request)  
        println("Done!")  
    }  
}
```

- Para obtener información sobre la API, consulte [DeleteBucketPolicy](#) en la Referencia de la API de AWS SDK para Kotlin.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: el comando elimina la política de bucket asociada al bucket de S3 indicado.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [DeleteBucketPolicy](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: el comando elimina la política de bucket asociada al bucket de S3 indicado.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [DeleteBucketPolicy](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [DeleteBucketPolicy](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

end
```

- Para obtener información sobre la API, consulte [DeleteBucketPolicy](#) en la Referencia de la API de AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [DeleteBucketPolicy](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **DeleteBucketReplication** con una CLI

En los siguientes ejemplos de código se muestra cómo se utiliza DeleteBucketReplication.

CLI

AWS CLI

El siguiente comando elimina la configuración de replicación de un bucket denominado amzn-s3-demo-bucket:

```
aws s3api delete-bucket-replication --bucket amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [DeleteBucketReplication](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: Elimina la configuración de replicación asociada al bucket denominado “amzn-s3-demo-bucket”. Tenga en cuenta que esta operación necesita permiso para la acción `s3:DeleteReplicationConfiguration`. Se le solicitará la confirmación antes de continuar con la operación; para suprimir la confirmación, utilice el conmutador `-Force`.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [DeleteBucketReplication](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: Elimina la configuración de replicación asociada al bucket denominado “amzn-s3-demo-bucket”. Tenga en cuenta que esta operación necesita permiso para la acción `s3:DeleteReplicationConfiguration`. Se le solicitará la confirmación antes de continuar con la operación; para suprimir la confirmación, utilice el conmutador `-Force`.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [DeleteBucketReplication](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **DeleteBucketTagging** con una CLI

En los siguientes ejemplos de código se muestra cómo se utiliza `DeleteBucketTagging`.

CLI

AWS CLI

El siguiente comando elimina la configuración de etiquetado de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-tagging --bucket amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [DeleteBucketTagging](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando elimina todas las etiquetas asociadas al bucket de S3 indicado.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
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
```

- Para obtener información sobre la API, consulte [DeleteBucketTagging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando elimina todas las etiquetas asociadas al bucket de S3 indicado.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
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
```

- Para obtener información sobre la API, consulte [DeleteBucketTagging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteBucketWebsite** con un SDK de AWS o la CLI

En los siguientes ejemplos de código se muestra cómo se utiliza DeleteBucketWebsite.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();  
}
```

- Para obtener información sobre la API, consulte [DeleteBucketWebsite](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El siguiente comando elimina la configuración de un sitio web de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api delete-bucket-website --bucket amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [DeleteBucketWebsite](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
        }
    }
}
```

- Para obtener información sobre la API, consulte [DeleteBucketWebsite](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine la configuración del sitio web del bucket.

```
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;
    }
  }
};
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).

- Para obtener información sobre la API, consulte [DeleteBucketWebsite](#) en la Referencia de la API de AWS SDK para JavaScript.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando deshabilita la propiedad de alojamiento de sitios web estáticos del bucket de S3 indicado.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
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
```

- Para obtener información sobre la API, consulte [DeleteBucketWebsite](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando deshabilita la propiedad de alojamiento de sitios web estáticos del bucket de S3 indicado.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
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
```

- Para obtener información sobre la API, consulte [DeleteBucketWebsite](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteObject** con un SDK de AWS o la CLI

En los siguientes ejemplos de código se muestra cómo se utiliza DeleteObject.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Trabajo con la integridad de los objetos de Amazon S3](#)
- [Trabajo con objetos con control de versiones](#)

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine un objeto de un bucket de S3 no versionado.

```
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.USWest2.
    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.");
        }
    }
}

```

Elimine un objeto de un bucket de S3 versionado.

```

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 = "verstioned-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.USSouth1;
        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;
    }
}

```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de la API de AWS SDK para .NET.

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de comandos de AWS CLI.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
}
```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El comando siguiente elimina un objeto denominado `test.txt` de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

Si el control de versiones del bucket está activado, el resultado contendrá el ID de versión del marcador de eliminación:

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tND1WlGq",
  "DeleteMarker": true
}
```

Para obtener más información acerca de la eliminación de objetos, consulte [Eliminación de objetos](#) en la Guía para desarrolladores de Amazon S3.

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de comandos de AWS CLI.

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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}
```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
}
```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine un objeto.

```
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;
    }
  }
};
```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de la API de AWS SDK para JavaScript.

PHP

SDK para PHP

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
    }
}
```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de la API de AWS SDK para PHP.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine un objeto.

```
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
```

Revierta un objeto a una versión anterior y elimine versiones posteriores del objeto.

```
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}."
    )

```

Reactive un objeto eliminado quitando el marcador de eliminación activo del objeto.

```

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)

```

Cree un controlador Lambda que elimine un marcador de eliminación de un objeto de S3. Este controlador se puede utilizar para limpiar de forma eficiente marcadores de eliminación extraños en un bucket con control de versiones.

```

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(
        "Got task: remove delete marker %s from object %s.", obj_version_id,
obj_key
    )

    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 = (
            f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
marker."
        )

        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,
}
```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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(() )  
}
```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
    }
}
```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de la API de AWS SDK para Swift.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **DeleteObjectTagging** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar DeleteObjectTagging.

CLI

AWS CLI

Eliminar los conjuntos de etiquetas de un objeto

En el siguiente ejemplo de delete-object-tagging, se elimina del objeto doc1.rtf la etiqueta con la clave especificada.

```
aws s3api delete-object-tagging \
    --bucket amzn-s3-demo-bucket \
    --key doc1.rtf
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [DeleteObjectTagging](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando elimina todas las etiquetas asociadas con el objeto con la clave “testfile.txt” en el bucket de S3 indicado.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

Salida:

```
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
```

- Para obtener información sobre la API, consulte [DeleteObjectTagging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando elimina todas las etiquetas asociadas con el objeto con la clave “testfile.txt” en el bucket de S3 indicado.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

Salida:

```
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
```

- Para obtener información sobre la API, consulte [DeleteObjectTagging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteObjects** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `DeleteObjects`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Conceptos básicos](#)
- [Eliminación de todos los objetos de un bucket](#)
- [Introducción al almacenamiento de objetos](#)

.NET

SDK para .NET (v4)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
    }
}
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para .NET.

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine varios objetos de un bucket de S3 no versionado.

```
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;
}

```

```
}
```

Elimine varios objetos de un bucket de S3 versionado.

```
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.USWest2;
        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>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// 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>

```

```

call    /// <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;
        }
    }
}

```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para .NET.

Bash

AWS CLI con Bash script

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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\"]}"
}
```

```
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
}
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de comandos de AWS CLI.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
}

```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El comando siguiente elimina un objeto denominado de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

`delete.json` es un documento JSON en el directorio actual que especifica el objeto que se va a eliminar:

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

Salida:

```
{
  "Deleted": [
    {
      "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUP01koHwzU",
      "Key": "test1.txt",
      "DeleteMarker": true
    }
  ]
}
```

- Para obtener detalles de la API, consulte [DeleteObjects](#) en la Referencia de comandos de AWS CLI.

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 _, 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",
                *delObjs.Key)
        } else {
            log.Printf("Deleted %s.\n", *delObjs.Key)
        }
    }
}
return err
}
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 putOb;
    ObjectIdentifier objectId;

    for (int i = 0; i < 3; i++) {
        String keyName = "delete object example " + i;
        objectId = ObjectIdentifier.builder()
            .key(keyName)
            .build();

        putOb = 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);
    }
}
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine varios objetos.

```
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;
  }
}
};

```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
suspend fun deleteBucketObjects(
    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")
    }
}
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine un conjunto de objetos de una lista de claves.

```
$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.");
}
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para PHP.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando elimina el objeto “sample.txt” del bucket “test-files”. Antes de ejecutar el comando, se le solicitará que lo confirme; para suprimir el mensaje, utilice el conmutador -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Ejemplo 2: este comando elimina la versión especificada del objeto “sample.txt” del bucket “test-files”, suponiendo que el bucket se haya configurado para habilitar las versiones de los objetos.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId  
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Ejemplo 3: este comando elimina los objetos “sample1.txt”, “sample2.txt” y “sample3.txt” del bucket “test-files” como una sola operación por lotes. La respuesta del servicio mostrará una lista de todas las claves procesadas, independientemente del estado de éxito o error de la eliminación. Para obtener únicamente los errores de las claves que el servicio no ha podido procesar, añada el parámetro -ReportErrorsOnly (este parámetro también se puede especificar con el alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

Ejemplo 4: este ejemplo utiliza una expresión en línea con el parámetro -KeyCollection para obtener las claves de los objetos que se van a eliminar. Get-S3Object devuelve una colección de instancias de Amazon.S3.Model.S3Object, cada una de las cuales tiene un elemento clave de tipo cadena que identifica el objeto.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Ejemplo 5: este ejemplo obtiene todos los objetos que tienen un prefijo de clave “prefijo/subprefijo” en el bucket y los elimina. Tenga en cuenta que los objetos entrantes se procesan de uno en uno. En el caso de colecciones grandes, plantéese la posibilidad de pasarlas al parámetro -InputObject (alias -S3ObjectCollection) del cmdlet para permitir que la eliminación se realice por lotes con una sola llamada al servicio.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

Ejemplo 6: en este ejemplo, se canaliza al cmdlet para su eliminación una colección de instancias de Amazon.S3.Model.S3ObjectVersion que representan marcadores de eliminación. Tenga en cuenta que los objetos entrantes se procesan de uno en uno. En el caso de colecciones grandes, plantéese la posibilidad de pasarlas al parámetro -InputObject (alias -S3ObjectCollection) del cmdlet para permitir que la eliminación se realice por lotes con una sola llamada al servicio.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Ejemplo 7: este script muestra cómo eliminar por lotes un conjunto de objetos (en este caso, marcadores de eliminación) mediante la creación de una matriz de objetos para utilizarlos con el parámetro -KeyAndVersionCollection.

```
$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
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando elimina el objeto “sample.txt” del bucket “test-files”. Antes de ejecutar el comando, se le solicitará que lo confirme; para suprimir el mensaje, utilice el conmutador -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Ejemplo 2: este comando elimina la versión especificada del objeto “sample.txt” del bucket “test-files”, suponiendo que el bucket se haya configurado para habilitar las versiones de los objetos.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId  
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Ejemplo 3: este comando elimina los objetos “sample1.txt”, “sample2.txt” y “sample3.txt” del bucket “test-files” como una sola operación por lotes. La respuesta del servicio mostrará una lista de todas las claves procesadas, independientemente del estado de éxito o error de la eliminación. Para obtener únicamente los errores de las claves que el servicio no ha podido procesar, añada el parámetro -ReportErrorsOnly (este parámetro también se puede especificar con el alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

Ejemplo 4: este ejemplo utiliza una expresión en línea con el parámetro -KeyCollection para obtener las claves de los objetos que se van a eliminar. Get-S3Object devuelve una colección de instancias de Amazon.S3.Model.S3Object, cada una de las cuales tiene un elemento clave de tipo cadena que identifica el objeto.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Ejemplo 5: este ejemplo obtiene todos los objetos que tienen un prefijo de clave “prefijo/subprefijo” en el bucket y los elimina. Tenga en cuenta que los objetos entrantes se procesan de uno en uno. En el caso de colecciones grandes, plantéese la posibilidad de pasarlas al parámetro -InputObject (alias -S3ObjectCollection) del cmdlet para permitir que la eliminación se realice por lotes con una sola llamada al servicio.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

Ejemplo 6: en este ejemplo, se canaliza al cmdlet para su eliminación una colección de instancias de Amazon.S3.Model.S3ObjectVersion que representan marcadores de

eliminación. Tenga en cuenta que los objetos entrantes se procesan de uno en uno. En el caso de colecciones grandes, plantéese la posibilidad de pasarlas al parámetro `-InputObject` (alias `-S3ObjectCollection`) del cmdlet para permitir que la eliminación se realice por lotes con una sola llamada al servicio.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Ejemplo 7: este script muestra cómo eliminar por lotes un conjunto de objetos (en este caso, marcadores de eliminación) mediante la creación de una matriz de objetos para utilizarlos con el parámetro `-KeyAndVersionCollection`.

```
$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
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine un conjunto de objetos mediante una lista de claves de objeto.

```
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
```

Elimine todos los objetos de un bucket.

```
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
```

Elimine de forma permanente un objeto con control de versiones mediante la eliminación de todas sus versiones.

```
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
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
end
```

```
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para Ruby.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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(())
}

```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.

```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
    }
}
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK para Swift.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **DeletePublicAccessBlock** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar DeletePublicAccessBlock.

CLI

AWS CLI

Eliminar la configuración de bloqueo de acceso público de un bucket

En el siguiente ejemplo de delete-public-access-block, se elimina la configuración de bloqueo de acceso público en el bucket especificado.

```
aws s3api delete-public-access-block \
  --bucket amzn-s3-demo-bucket
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [DeletePublicAccessBlock](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando desactiva la configuración de bloqueo de acceso público para el bucket en cuestión.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
'^BucketName'
```

Salida:

```
amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [DeletePublicAccessBlock](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando desactiva la configuración de bloqueo de acceso público para el bucket en cuestión.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

Salida:

```
amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [DeletePublicAccessBlock](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **GetBucketAccelerateConfiguration** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetBucketAccelerateConfiguration`.

CLI

AWS CLI

Recuperar la configuración acelerada de un bucket

En el siguiente ejemplo de `get-bucket-accelerate-configuration`, se recupera la configuración acelerada para el bucket especificado.

```
aws s3api get-bucket-accelerate-configuration \  
--bucket amzn-s3-demo-bucket
```

Salida:

```
{  
  "Status": "Enabled"  
}
```

- Para obtener información sobre la API, consulte [GetBucketAccelerateConfiguration](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve el valor Enabled si la configuración de aceleración de transferencia está habilitada para el bucket especificado.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Value
-----
Enabled
```

- Para obtener información sobre la API, consulte [GetBucketAccelerateConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve el valor Enabled si la configuración de aceleración de transferencia está habilitada para el bucket especificado.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Value
-----
Enabled
```

- Para obtener información sobre la API, consulte [GetBucketAccelerateConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetBucketAc1** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar GetBucketAc1.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Comprobación de la existencia de un bucket](#)
- [Administrar listas de control de acceso \(ACL\)](#)

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}
```


- Para obtener información sobre la API, consulte [GetBucketAcl](#) en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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";
}

```

- Para obtener información sobre la API, consulte [GetBucketAcl](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El siguiente comando recupera la lista de control de acceso de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    }
  ]
}
```

- Para obtener información sobre la API, consulte [GetBucketAcl](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 "";
}
}
```

- Para obtener información sobre la API, consulte [GetBucketAcl](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtenga los permisos de 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;
    }
  }
};
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para obtener información sobre la API, consulte [GetBucketAcl](#) en la Referencia de la API de AWS SDK para JavaScript.

PowerShell

Herramientas para PowerShell V5

Ejemplo 1: El comando obtiene los detalles del propietario del objeto de S.


```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

Salida:

```
DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Para obtener información sobre la API, consulte [GetBucketAcl](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

```

- Para obtener información sobre la API, consulte [GetBucketAcl](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.

```

- Para obtener información sobre la API, consulte [GetBucketAcl](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **GetBucketAnalyticsConfiguration** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetBucketAnalyticsConfiguration`.

CLI

AWS CLI

Recuperar la configuración de análisis de un bucket con un ID específico

En el siguiente ejemplo de `get-bucket-analytics-configuration`, se muestra la configuración de análisis para el bucket e ID especificados.

```
aws s3api get-bucket-analytics-configuration \
  --bucket amzn-s3-demo-bucket \
  --id 1
```

Salida:

```
{
  "AnalyticsConfiguration": {
    "StorageClassAnalysis": {},
    "Id": "1"
  }
}
```

- Para obtener información sobre la API, consulte [GetBucketAnalyticsConfiguration](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve los detalles del filtro de análisis con el nombre “testfilter” en el bucket de S3 indicado.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- Para obtener información sobre la API, consulte [GetBucketAnalyticsConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve los detalles del filtro de análisis con el nombre “testfilter” en el bucket de S3 indicado.

```
Get-S3BucketAnalyticsConfiguration -BucketName 'amzn-s3-demo-bucket' -AnalyticsId  
'testfilter'
```

- Para obtener información sobre la API, consulte [GetBucketAnalyticsConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetBucketCors** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar GetBucketCors.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}
```

- Para ver la información de la API, consulte [GetBucketCors](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

El siguiente comando recupera la configuración de uso compartido de recursos entre orígenes para un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-cors --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "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": [
        "*"
    ]
}
]
```

- Para obtener información sobre la API, consulte [GetBucketCors](#) en la Referencia de comandos de la AWS CLI.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtenga la política de CORS para el bucket.

```
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;
    }
  }
}

```

```
}  
};
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para ver la información de la API, consulte [GetBucketCors](#) en la Referencia de la API de AWS SDK para JavaScript.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre las API, consulte [GetBucketCors](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para ver la información de la API, consulte [GetBucketCors](#) en la Referencia de la API de AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [GetBucketCors](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetBucketEncryption** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetBucketEncryption`.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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}");
        }
    }
```

- Para obtener información sobre la API, consulte [GetBucketEncryption](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

Recuperar la configuración de cifrado del servidor de un bucket

En el siguiente ejemplo de `get-bucket-encryption`, se recupera la configuración de cifrado del lado del servidor del bucket `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-encryption \
    --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
```

- Para obtener información sobre la API, consulte [GetBucketEncryption](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve todas las reglas de cifrado del servidor asociadas al bucket determinado.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketEncryption](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve todas las reglas de cifrado del servidor asociadas al bucket determinado.

```
Get-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketEncryption](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **GetBucketInventoryConfiguration** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar GetBucketInventoryConfiguration.

CLI

AWS CLI

Recuperar la configuración de inventario de un bucket

En el siguiente ejemplo de get-bucket-inventory-configuration, se recupera la configuración de inventario del bucket especificado con el ID 1.

```
aws s3api get-bucket-inventory-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 1
```

--id 1

Salida:

```
{
  "InventoryConfiguration": {
    "IsEnabled": true,
    "Destination": {
      "S3BucketDestination": {
        "Format": "ORC",
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
        "AccountId": "123456789012"
      }
    },
    "IncludedObjectVersions": "Current",
    "Id": "1",
    "Schedule": {
      "Frequency": "Weekly"
    }
  }
}
```

- Para obtener información sobre la API, consulte [GetBucketInventoryConfiguration](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve los detalles del inventario con el nombre “testinventory” del bucket de S3 indicado.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId 'testinventory'
```

- Para obtener información sobre la API, consulte [GetBucketInventoryConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve los detalles del inventario con el nombre “testinventory” del bucket de S3 indicado.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId  
'testinventory'
```

- Para obtener información sobre la API, consulte [GetBucketInventoryConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetBucketLifecycleConfiguration** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetBucketLifecycleConfiguration`.

.NET

SDK para .NET

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
    }
```

- Para obtener información sobre la API, consulte [GetBucketLifecycleConfiguration](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

El siguiente comando recupera la configuración del ciclo de vida de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "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"
    }
]
}

```

- Para obtener información sobre la API, consulte [GetBucketLifecycleConfiguration](#) en la Referencia de comandos de la AWS CLI.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [GetBucketLifecycleConfiguration](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [GetBucketLifecycleConfiguration](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetBucketLocation** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar GetBucketLocation.

CLI

AWS CLI

El siguiente comando recupera la restricción de ubicación de un bucket denominado amzn-s3-demo-bucket, si existe una restricción:

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "LocationConstraint": "us-west-2"
}
```

- Para obtener información sobre la API, consulte [GetBucketLocation](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: Este comando devuelve la restricción de ubicación del bucket “amzn-s3-demo-bucket”, si existe una restricción.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Value
-----
ap-south-1
```

- Para obtener información sobre la API, consulte [GetBucketLocation](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: Este comando devuelve la restricción de ubicación del bucket “amzn-s3-demo-bucket”, si existe una restricción.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Value
-----
ap-south-1
```

- Para obtener información sobre la API, consulte [GetBucketLocation](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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(())
}

```

- Para obtener información sobre la API, consulte [GetBucketLocation](#) en la Referencia de la API de AWS SDK para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **GetBucketLogging** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar GetBucketLogging.

CLI

AWS CLI

Recuperar el estado de registros de un bucket

En el siguiente ejemplo de `get-bucket-logging`, se recupera el estado de registros del bucket especificado.

```
aws s3api get-bucket-logging \
  --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "LoggingEnabled": {
    "TargetPrefix": "",
    "TargetBucket": "amzn-s3-demo-bucket-logs"
  }
}
```

- Para obtener información sobre la API, consulte [GetBucketLogging](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve el estado de registros del bucket especificado.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Salida:

TargetBucketName	Grants	TargetPrefix
testbucket1	{}	testprefix

- Para obtener información sobre la API, consulte [GetBucketLogging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve el estado de registros del bucket especificado.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Salida:

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- Para obtener información sobre la API, consulte [GetBucketLogging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **GetBucketMetricsConfiguration** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar GetBucketMetricsConfiguration.

CLI

AWS CLI

Recuperar la configuración de métricas de un bucket con un ID específico

En el siguiente ejemplo de get-bucket-metrics-configuration, se muestra la configuración de métricas para el bucket e ID especificados.

```
aws s3api get-bucket-metrics-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --id 123
```

Salida:

```
{  
  "MetricsConfiguration": {  
    "Filter": {
```

```
        "Prefix": "logs"
      },
      "Id": "123"
    }
  }
```

- Para obtener información sobre la API, consulte [GetBucketMetricsConfiguration](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve los detalles del filtro de métricas con el nombre “testfilter” del bucket de S3 indicado.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- Para obtener información sobre la API, consulte [GetBucketMetricsConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve los detalles del filtro de métricas con el nombre “testfilter” del bucket de S3 indicado.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- Para obtener información sobre la API, consulte [GetBucketMetricsConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **GetBucketNotification** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetBucketNotification`.

CLI

AWS CLI

El siguiente comando recupera la configuración de notificaciones de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-notification --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "TopicConfiguration": {
    "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
    "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNl",
    "Event": "s3:ObjectCreated:*",
    "Events": [
      "s3:ObjectCreated:*"
    ]
  }
}
```

- Para obtener información sobre la API, consulte [GetBucketNotification](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: en este ejemplo se recupera la configuración de notificaciones del bucket en cuestión

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
TopicConfigurations
```

Salida:

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Para obtener información sobre la API, consulte [GetBucketNotification](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: en este ejemplo se recupera la configuración de notificaciones del bucket en cuestión

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty TopicConfigurations
```

Salida:

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Para obtener información sobre la API, consulte [GetBucketNotification](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetBucketPolicy** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar GetBucketPolicy.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
}

```

- Para obtener información sobre la API, consulte [GetBucketPolicy](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El siguiente comando recupera la política de bucket de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "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/*\\\"}]}"
}
```

Obtención y colocación de una política de bucket En el siguiente ejemplo se muestra cómo se puede descargar una política de bucket de Amazon S3, realizar modificaciones en el archivo y luego usar `put-bucket-policy` para aplicar la política de bucket modificada. Para descargar la política de bucket a un archivo, puede ejecutar:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output text > policy.json
```

A continuación, puede modificar el archivo `policy.json` según sea necesario. Por último, puede volver a aplicar esta política modificada al bucket de S3 ejecutando:

archivo `policy.json` según sea necesario. Por último, puede volver a aplicar esta política modificada al bucket de S3 ejecutando:

archivo según sea necesario. Por último, puede volver a aplicar esta política modificada al bucket de S3 ejecutando:

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://policy.json
```

- Para obtener información sobre la API, consulte [GetBucketPolicy](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 "";
}
}
```

- Para obtener información sobre la API, consulte [GetBucketPolicy](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtenga la política del bucket.

```
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;  
    }  
}  
};
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para obtener información sobre la API, consulte [GetBucketPolicy](#) en la Referencia de la API de AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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  
    }  
}
```


- Para obtener información sobre la API, consulte [GetBucketPolicy](#) en la Referencia de la API de AWS SDK para Kotlin.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando muestra la política de bucket asociada al bucket de S3 indicado.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketPolicy](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando muestra la política de bucket asociada al bucket de S3 indicado.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketPolicy](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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)
```

- Para obtener información sobre la API, consulte [GetBucketPolicy](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

```

- Para obtener información sobre la API, consulte [GetBucketPolicy](#) en la Referencia de la API de AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

TRY.
  oo_result = lo_s3->getbucketpolicy(          " oo_result is returned for
testing purposes. "
  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.
```

- Para obtener información sobre la API, consulte [GetBucketPolicy](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **GetBucketPolicyStatus** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetBucketPolicyStatus`.

CLI

AWS CLI

Recuperar el estado de política de un bucket que indica si el bucket es público

En el siguiente ejemplo de `get-bucket-policy-status`, se recupera el estado de política del bucket `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-policy-status \
  --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "PolicyStatus": {
    "IsPublic": false
  }
}
```

- Para obtener información sobre la API, consulte [GetBucketPolicyStatus](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve el estado de política del bucket específico de S3 e indica si el bucket es público.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketPolicyStatus](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve el estado de política del bucket específico de S3 e indica si el bucket es público.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketPolicyStatus](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetBucketReplication** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar **GetBucketReplication**.

CLI

AWS CLI

El siguiente comando recupera la configuración de replicación de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-replication --bucket amzn-s3-demo-bucket
```

Salida:

```
{
```

```

    "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"
    }
  }
}

```

- Para obtener información sobre la API, consulte [GetBucketReplication](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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());
    }
}
```

- Para obtener información sobre la API, consulte [GetBucketReplication](#) en la Referencia de la API de AWS SDK for Java 2.x.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: Devuelve la información de configuración de replicación establecida en el bucket denominado "amzn-s3-demo-bucket".

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [GetBucketReplication](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: Devuelve la información de configuración de replicación establecida en el bucket denominado "amzn-s3-demo-bucket".

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [GetBucketReplication](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **GetBucketRequestPayment** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar GetBucketRequestPayment.

CLI

AWS CLI

Recuperar la configuración de pagos de solicitudes de un bucket

En el siguiente ejemplo de `get-bucket-request-payment`, se recupera la configuración de pagos por el solicitante para el bucket especificado.

```
aws s3api get-bucket-request-payment \
  --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "Payer": "BucketOwner"
}
```

- Para obtener información sobre la API, consulte [GetBucketRequestPayment](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: Devuelve la configuración de pagos de solicitudes del bucket denominado "amzn-s3-demo-bucket". De forma predeterminada, el propietario del bucket paga las descargas realizadas desde el bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [GetBucketRequestPayment](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: Devuelve la configuración de pagos de solicitudes del bucket denominado "amzn-s3-demo-bucket". De forma predeterminada, el propietario del bucket paga las descargas realizadas desde el bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [GetBucketRequestPayment](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **GetBucketTagging** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar GetBucketTagging.

CLI

AWS CLI

El siguiente comando recupera la configuración de etiquetado de un bucket denominado amzn-s3-demo-bucket:

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "TagSet": [
    {
      "Value": "marketing",
      "Key": "organization"
    }
  ]
}
```

- Para obtener información sobre la API, consulte [GetBucketTagging](#) en la Referencia de comandos de la AWS CLI.

PowerShell**Herramientas para PowerShell V4**

Ejemplo 1: este comando devuelve todas las etiquetas asociadas al bucket indicado.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketTagging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve todas las etiquetas asociadas al bucket indicado.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketTagging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar `GetBucketVersioning` con una CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetBucketVersioning`.

CLI

AWS CLI

El siguiente comando recupera la configuración del control de versiones de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "Status": "Enabled"
}
```

- Para obtener información sobre la API, consulte [GetBucketVersioning](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve el estado del control de versiones con respecto al bucket indicado.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketVersioning](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve el estado del control de versiones con respecto al bucket indicado.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketVersioning](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetBucketWebsite** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar GetBucketWebsite.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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}");
```

- Para obtener información sobre la API, consulte [GetBucketWebsite](#) en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();  
}
```

- Para obtener información sobre la API, consulte [GetBucketWebsite](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El siguiente comando recupera la configuración de sitio web estática de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

Salida:

```
{  
  "IndexDocument": {  
    "Suffix": "index.html"  
  },  
  "ErrorDocument": {  
    "Key": "error.html"  
  }  
}
```

- Para obtener información sobre la API, consulte [GetBucketWebsite](#) en la Referencia de comandos de la AWS CLI.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtenga la configuración de sitio 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;
    }
  }
};
```

- Para obtener información sobre la API, consulte [GetBucketWebsite](#) en la Referencia de la API de AWS SDK para JavaScript.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve los detalles de las configuraciones de sitio web estáticas del bucket de S3 indicado.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketWebsite](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve los detalles de las configuraciones de sitio web estáticas del bucket de S3 indicado.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetBucketWebsite](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetObject** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetObject`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Conceptos básicos](#)

- [Obtenga un objeto de un bucket si se ha modificado](#)
- [Obtención de un objeto desde un punto de acceso de varias regiones](#)
- [Introducción al cifrado](#)
- [Introducción al almacenamiento de objetos](#)
- [Cómo realizar solicitudes condicionales](#)
- [Realización de un seguimiento de cargas y descargas](#)

.NET

SDK para .NET (v4)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}
}
```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK para .NET.

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtener un objeto mediante una solicitud condicional.

```
/// <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;
    }
}

```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK para .NET.

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}

```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de comandos de AWS CLI.

C++

SDK para C++

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
}
```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

En el siguiente ejemplo se utiliza el comando `get-object` para descargar un objeto de Amazon S3.

```
aws s3api get-object --bucket text-content --key dir/  
my_images.tar.bz2 my_images.tar.bz2
```

Tenga en cuenta que el parámetro `outfile` se especifica sin un nombre de opción, como `--outfile`. El nombre del archivo de salida debe ser el último parámetro del comando.

El siguiente ejemplo muestra el uso de `--range` para descargar un intervalo de bytes específico de un objeto. Tenga en cuenta que los intervalos de bytes deben tener el prefijo `"bytes="`:

```
aws s3api get-object --bucket text-content --key dir/my_data --  
range bytes=8888-9999 my_data_range
```

Para obtener más información acerca de la recuperación de objetos, consulte [Obtención de objetos](#) en la Guía para desarrolladores de Amazon S3.

- Para obtener detalles de la API, consulte [GetObject](#) en la Referencia de comandos de AWS CLI.

Go

SDK para Go V2

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Lea datos como una matriz de bytes con un [S3 Client](#).

```
/**
 * 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);
            }
        });
    }
}

```

Utilice un [S3TransferManager](#) para [descargar un objeto](#) de un bucket de S3 en un archivo local. Vea el [archivo completo](#) y [pruébelo](#).

```

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 downloadedFileWithPath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFileWithPath))
            .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }
```

Lea las etiquetas que pertenecen a un objeto con 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);
        }
    }
}

```

Obtenga una URL para un objeto con 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);
    }
}
}

```

Obtenga un objeto mediante el objeto de cliente S3Presigner con 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();
        }
    }
}

```

Obtenga un objeto mediante un objeto `ResponseTransformer` y [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);
    }
}
}
```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Descargue el objeto.

```
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;
    }
  }
};
```

Descargar el objeto con la condición de que su ETag coincida con la proporcionada.

```
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"));  
  }  
}
```

Descargar el objeto con la condición de que su ETag no coincida con la proporcionada.

```
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"));
  }
}

```

Descargar el objeto con la condición de que se haya creado o modificado en un período de tiempo determinado.

```

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"));
    }
}

```

Descargar el objeto con la condición de que no se haya creado o modificado en un período de tiempo determinado.

```

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,
  }),
);
// 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"));  
    }  
}
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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")
        }
    }
}

```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de SDK de AWS para Kotlin.

PHP

SDK para PHP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtenga un objeto.

```

$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.");
}

```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK para PHP.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando recupera el elemento “sample.txt” del bucket “amzn-s3-demo-bucket” y lo guarda en un archivo denominado “local-sample.txt” en la ubicación actual. No es necesario que el archivo “local-sample.txt” exista para poder llamar a este comando.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Ejemplo 2: este comando recupera el directorio virtual “DIR” del bucket “amzn-s3-demo-bucket” y lo guarda en una carpeta denominada “Local-DIR” en la ubicación actual. No es necesario que la carpeta “Local-DIR” exista para poder llamar a este comando.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Ejemplo 3: descarga todos los objetos cuyas claves terminan en “.json” de los buckets con “config” en el nombre del bucket a los archivos de la carpeta especificada. Las claves de objeto se utilizan para establecer los nombres de los archivos.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando recupera el elemento “sample.txt” del bucket “amzn-s3-demo-bucket” y lo guarda en un archivo denominado “local-sample.txt” en la ubicación actual. No es necesario que el archivo “local-sample.txt” exista para poder llamar a este comando.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Ejemplo 2: este comando recupera el directorio virtual “DIR” del bucket “amzn-s3-demo-bucket” y lo guarda en una carpeta denominada “Local-DIR” en la ubicación actual. No es necesario que la carpeta “Local-DIR” exista para poder llamar a este comando.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Ejemplo 3: descarga todos los objetos cuyas claves terminan en “.json” de los buckets con “config” en el nombre del bucket a los archivos de la carpeta especificada. Las claves de objeto se utilizan para establecer los nombres de los archivos.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\Config0bjects
```

Ejemplo 4: descarga un único objeto mediante la descarga paralela multiparte para mejorar el rendimiento en archivos de gran tamaño.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt -UseMultipartDownload
```

Ejemplo 5: descarga un objeto grande mediante el modo RANGE con un tamaño de parte personalizado de 64 MB. El modo RANGE utiliza solicitudes HTTP de rango de bytes y funciona para todos los objetos, independientemente de cómo se hayan cargado.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-large-file.zip -UseMultipartDownload -MultipartDownloadType RANGE -PartSize 64MB
```

Ejemplo 6: descarga todos los objetos con el prefijo de clave “data/” en una carpeta local mediante la descarga multiparte con descargas de archivos simultáneas.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-data -UseMultipartDownload -DownloadFilesConcurrently
```

Ejemplo 7: descarga un único objeto mediante una descarga multiparte con 20 conexiones simultáneas para obtener el máximo rendimiento en redes de gran ancho de banda.


```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key large-file.zip -File local-  
large-file.zip -UseMultipartDownload -ConcurrentServiceRequest 20
```

Ejemplo 8: descarga todos los objetos con el prefijo de claves “data/” en una carpeta local mediante la descarga multiparte y continúa a pesar de los errores individuales de los archivos en lugar de cancelarla.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix data/ -Folder C:\local-  
data -UseMultipartDownload -FailurePolicy ContinueOnFailure
```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

```

Obtener un objeto mediante una solicitud condicional.

```

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

```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtenga un objeto.

```
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))
```

```

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__

```

Obtenga un objeto e informe de su estado de cifrado del lado del servidor.

```

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__

```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK para Ruby.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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)
}

```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
    }
}
```

- Para obtener información acerca de la API, consulte [GetObject](#) en la Referencia de la API del SDK de AWS para Swift.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de `GetObjectAcl` con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetObjectAcl`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en su contexto en el siguiente ejemplo de código:

- [Administrar listas de control de acceso \(ACL\)](#)

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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";
    }
}

```

- Para obtener información sobre la API, consulte [GetObjectAcl](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El siguiente comando recupera la lista de control de acceso de un objeto en un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key index.html
```

Salida:

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

- Para obtener información sobre la API, consulte [GetObjectAcl](#) en la Referencia de comandos de la AWS CLI.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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}")
        }
    }
}
```

- Para obtener información sobre la API, consulte [GetObjectAcl](#) en la Referencia de la API de AWS SDK para Kotlin.

PowerShell

Herramientas para PowerShell V5

Ejemplo 1: El comando obtiene los detalles del propietario del objeto de S.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
*).Owner
```

Salida:

```

DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544

```

- Para obtener información sobre la API, consulte [GetObjectAcl](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python**SDK para Python (Boto3)**** Note**

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

```

- Para obtener información sobre la API, consulte [GetObjectAcl](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.

```


- Para obtener información sobre la API, consulte [GetObjectAcl](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetObjectAttributes** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetObjectAttributes`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en su contexto en el siguiente ejemplo de código:

- [Trabajo con la integridad de los objetos de Amazon S3](#)

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
                case AwsDoc::S3::MD5:
                    break; // MD5 is not supported.
                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;
}

```

- Para obtener información sobre la API, consulte [GetObjectAttributes](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

Para recuperar metadatos de un objeto sin devolver el objeto en sí

En el siguiente ejemplo de `get-object-attributes`, se recuperan los metadatos del objeto `doc1.rtf`.

```

aws s3api get-object-attributes \
    --bucket amzn-s3-demo-bucket \
    --key doc1.rtf \
    --object-attributes "StorageClass" "ETag" "ObjectSize"

```

Salida:

```
{
  "LastModified": "2022-03-15T19:37:31+00:00",
  "VersionId": "IuCPjXTDzHNf1dAuitVBIKJpF2p1fg4P",
  "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",
  "StorageClass": "STANDARD",
  "ObjectSize": 405
}
```

Para obtener más información, consulte [GetObjectAttributes](#) en la Referencia de la API de Amazon S3.

- Para obtener información sobre la API, consulte [GetObjectAttributes](#) en la Referencia de comandos de la AWS CLI.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetObjectLegalHold** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetObjectLegalHold`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Bloqueo de objetos de Amazon S3](#)

.NET

SDK para .NET

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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($"\\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();
        }
    }
}

```

- Para obtener información sobre la API, consulte [GetObjectLegalHold](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

Recupera el estado de retención legal de un objeto

En el siguiente ejemplo de `get-object-legal-hold`, se recupera el estado de retención legal del objeto especificado.

```

aws s3api get-object-legal-hold \
    --bucket amzn-s3-demo-bucket-with-object-lock \

```

```
--key doc1.rtf
```

Salida:

```
{
  "LegalHold": {
    "Status": "ON"
  }
}
```

- Para obtener información sobre la API, consulte [GetObjectLegalHold](#) en la Referencia de comandos de la AWS CLI.

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```


- Para obtener información sobre la API, consulte [GetObjectLegalHold](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}
```

- Para obtener información sobre la API, consulte [GetObjectLegalHold](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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"));
    }
}
```

```
}
```

- Para obtener información sobre la API, consulte [GetObjectLegalHold](#) en la Referencia de la API de AWS SDK para JavaScript.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ponga una retención legal en un objeto.

```
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,
    )

```

- Para obtener información sobre la API, consulte [GetObjectLegalHold](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.

```

- Para obtener información sobre la API, consulte [GetObjectLegalHold](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetObjectLockConfiguration** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetObjectLockConfiguration`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Bloqueo de objetos de Amazon S3](#)

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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}: " +
                        $"  \tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                        $"  \tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");
    }
}
```

```
        return response.ObjectLockConfiguration;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}'");
        return new ObjectLockConfiguration();
    }
}
```

- Para obtener información sobre la API, consulte [GetObjectLockConfiguration](#) en la Referencia de la API de AWS SDK para .NET..

CLI

AWS CLI

Para recuperar una configuración de bloqueo de objetos para un bucket

En el siguiente ejemplo de `get-object-lock-configuration`, se recupera la configuración de bloqueo de objetos para el bucket especificado.

```
aws s3api get-object-lock-configuration \
    --bucket amzn-s3-demo-bucket-with-object-lock
```

Salida:

```
{
  "ObjectLockConfiguration": {
    "ObjectLockEnabled": "Enabled",
    "Rule": {
      "DefaultRetention": {
        "Mode": "COMPLIANCE",
        "Days": 50
      }
    }
  }
}
```

- Para obtener información sobre la API, consulte [GetObjectLockConfiguration](#) en la Referencia de comandos de la AWS CLI.

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

- Para obtener información sobre la API, consulte [GetObjectLockConfiguration](#) en la Referencia de la API de AWS SDK para Go..

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.getObjectLockConfiguration().getObjectLockEnabled());
System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}

```

- Para obtener información sobre la API, consulte [GetObjectLockConfiguration](#) en la Referencia de la API de AWS SDK for Java 2.x..

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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"));  
    }  
}
```

- Para obtener información sobre la API, consulte [GetObjectLockConfiguration](#) en la Referencia de la API de AWS SDK para JavaScript..

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve el valor “Enabled” si la configuración de bloqueo de objetos está habilitada para el bucket de S3 indicado.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Salida:

```
Value  
-----  
Enabled
```

- Para obtener información sobre la API, consulte [GetObjectLockConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve el valor “Enabled” si la configuración de bloqueo de objetos está habilitada para el bucket de S3 indicado.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Salida:

```
Value  
-----  
Enabled
```

- Para obtener información sobre la API, consulte [GetObjectLockConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtenga la configuración de bloqueo de objetos.

```
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

```

- Para obtener información sobre la API, consulte [GetObjectLockConfiguration](#) en la Referencia de la del SDK de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.

```

- Para obtener información sobre la API, consulte [GetObjectLockConfiguration](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetObjectRetention** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetObjectRetention`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Bloqueo de objetos de Amazon S3](#)

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
    }
}
```

- Para ver la información de la API, consulte [GetObjectRetention](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

Para recuperar la configuración de retención de un objeto

En el siguiente ejemplo de `get-object-retention`, se recupera la configuración de retención del objeto especificado.

```
aws s3api get-object-retention \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf
```

Salida:

```
{
  "Retention": {
    "Mode": "GOVERNANCE",
    "RetainUntilDate": "2025-01-01T00:00:00.000Z"
  }
}
```

- Para obtener información sobre la API, consulte [GetObjectRetention](#) en la Referencia de comandos de la AWS CLI.

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

- Para ver la información de la API, consulte [GetObjectRetention](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
    }
}
```

- Para ver la información de la API, consulte [GetObjectRetention](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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"));
  }
}
```

- Para ver la información de la API, consulte [GetObjectRetention](#) en la Referencia de la API de AWS SDK para JavaScript.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: el comando devuelve el modo y la fecha hasta que se retenga el objeto.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Para obtener información sobre la API, consulte [GetObjectRetention](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: el comando devuelve el modo y la fecha hasta que se retenga el objeto.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Para obtener información sobre la API, consulte [GetObjectRetention](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **GetObjectTagging** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar GetObjectTagging.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Comenzar a utilizar etiquetas](#)

CLI

AWS CLI

Recuperar las etiquetas asociadas a un objeto

El siguiente ejemplo de get-object-tagging recupera los valores de la clave especificada del objeto especificado.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc1.rtf
```

Salida:

```
{  
  "TagSet": [  
    {
```

```
        "Value": "confidential",
        "Key": "designation"
    }
]
}
```

El siguiente ejemplo de `get-object-tagging` intenta recuperar los conjuntos de etiquetas del objeto `doc2.rtf`, que no tiene etiquetas.

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc2.rtf
```

Salida:

```
{
  "TagSet": []
}
```

El siguiente ejemplo de `get-object-tagging` recupera los conjuntos de etiquetas del objeto `doc3.rtf`, que tiene varias etiquetas.

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc3.rtf
```

Salida:

```
{
  "TagSet": [
    {
      "Value": "confidential",
      "Key": "designation"
    },
    {
      "Value": "finance",
      "Key": "department"
    },
    {
      "Value": "payroll",
      "Key": "team"
    }
  ]
}
```

```
}  
]  
}
```

- Para obtener información sobre la API, consulte [GetObjectTagging](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: el ejemplo devuelve las etiquetas asociadas al objeto presentes en el bucket de S3 indicado.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Key  Value  
---  -  
test value
```

- Para obtener información sobre la API, consulte [GetObjectTagging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: el ejemplo devuelve las etiquetas asociadas al objeto presentes en el bucket de S3 indicado.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Salida:

```
Key  Value  
---  -  
test value
```

- Para obtener información sobre la API, consulte [GetObjectTagging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **GetPublicAccessBlock** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar `GetPublicAccessBlock`.

CLI

AWS CLI

Establecer o modificar la configuración de bloqueo de acceso público de un bucket

En el siguiente ejemplo de `get-public-access-block`, se elimina la configuración de bloqueo de acceso público del bucket especificado.

```
aws s3api get-public-access-block \
  --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "PublicAccessBlockConfiguration": {
    "IgnorePublicAcls": true,
    "BlockPublicPolicy": true,
    "BlockPublicAcls": true,
    "RestrictPublicBuckets": true
  }
}
```

- Para obtener información sobre la API, consulte [GetPublicAccessBlock](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: el comando devuelve la configuración de bloqueo de acceso público del bucket de S3 indicado.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetPublicAccessBlock](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: el comando devuelve la configuración de bloqueo de acceso público del bucket de S3 indicado.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [GetPublicAccessBlock](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **HeadBucket** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar HeadBucket.

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

- Para obtener información sobre la API, consulte [HeadBucket](#) en la Referencia de comandos de AWS CLI.

CLI

AWS CLI

El siguiente comando verifica el acceso a un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

Si el bucket existe y tiene acceso a él, no se muestra ningún resultado. De lo contrario, se mostrará un mensaje de error. Por ejemplo:

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- Para obtener información sobre la API, consulte [HeadBucket](#) en la Referencia de comandos de AWS CLI.

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

- Para obtener información sobre la API, consulte [HeadBucket](#) en la Referencia de la API de AWS SDK para Go.

PowerShell

Herramientas para PowerShell V5

Ejemplo 1: Este comando devuelve el resultado con el código de estado HTTP 200 OK para el bucket existente cuando el usuario tiene permiso para acceder a él. El parámetro BucketArn solo se admite en los buckets de directorio de S.

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

Salida:

```
AccessPointAlias    : False
BucketArn           :
BucketLocationName  :
BucketLocationType  :
BucketRegion        : us-east-2
ResponseMetadata    : Amazon.Runtime.ResponseMetadata
ContentLength       : 0
HttpStatusCode       : OK
```

Ejemplo 2: Este comando produce un error con el código de estado HTTP NotFound cuando el bucket no existe.

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

Salida:

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status
Code NotFound. No further error information was returned by the service.
```

Ejemplo 3: Este comando produce un error con el código de estado HTTP Forbidden para un bucket existente cuando el usuario no tiene permiso para acceder a él.

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

Salida:

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status
Code Forbidden. No further error information was returned by the service.
```

- Para obtener información sobre la API, consulte [HeadBucket](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [HeadBucket](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [HeadBucket](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **HeadObject** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `HeadObject`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Comprobación de si existe un objeto](#)
- [Introducción al almacenamiento de objetos](#)

CLI

AWS CLI

El siguiente comando recupera los metadatos de un objeto de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

Salida:

```
{
  "AcceptRanges": "bytes",
  "ContentType": "text/html",
  "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
  "ContentLength": 77,
  "VersionId": "null",
  "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
  "Metadata": {}
}
```

- Para obtener información de la API, consulte [HeadObject](#) en la Referencia de comandos de AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Determinar el tipo de contenido de un objeto.

```
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);
        }
    }
}

```

Obtener el estado de restauración de un objeto.

```

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);
        }
    }
}
```

```
}  
}
```

- Para obtener información sobre la API, consulte [HeadObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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  
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__
```

- Para obtener información sobre la API, consulte [HeadObject](#) en la Referencia de la API de AWS SDK para Ruby.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **ListBucketAnalyticsConfigurations** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar **ListBucketAnalyticsConfigurations**.

CLI

AWS CLI

Recuperar una lista de configuraciones de análisis para un bucket

El siguiente `list-bucket-analytics-configurations` recupera una lista de configuraciones de análisis para el bucket especificado.

```
aws s3api list-bucket-analytics-configurations \
  --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "AnalyticsConfigurationList": [
    {
```

```
        "StorageClassAnalysis": {},
        "Id": "1"
    }
],
"IsTruncated": false
}
```

- Para obtener información sobre la API, consulte [ListBucketAnalyticsConfigurations](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve las primeras 100 configuraciones de análisis del bucket de S3 indicado.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [ListBucketAnalyticsConfigurations](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve las primeras 100 configuraciones de análisis del bucket de S3 indicado.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [ListBucketAnalyticsConfigurations](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **ListBucketInventoryConfigurations** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar **ListBucketInventoryConfigurations**.

CLI

AWS CLI

Recuperar una lista de las configuraciones de inventario de un bucket

En el siguiente ejemplo de `list-bucket-inventory-configurations`, se enumeran las configuraciones de inventario del bucket especificado.

```
aws s3api list-bucket-inventory-configurations \
  --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "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
  }
```

- Para obtener información sobre la API, consulte [ListBucketInventoryConfigurations](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve las primeras 100 configuraciones de inventario del bucket de S3 indicado.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [ListBucketInventoryConfigurations](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve las primeras 100 configuraciones de inventario del bucket de S3 indicado.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Para obtener información sobre la API, consulte [ListBucketInventoryConfigurations](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **ListBuckets** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListBuckets`.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
}
}
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El siguiente comando usa el comando `list-buckets` para mostrar los nombres de todos los buckets de Amazon S3 (en todas las regiones):

```
aws s3api list-buckets --query "Buckets[].Name"
```

La opción de consultas filtra la salida de `list-buckets` únicamente a los nombres de los buckets.

Para obtener más información sobre los buckets, consulte Trabajo con buckets de Amazon S3 en la Guía para desarrolladores de Amazon S3.

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de comandos de la AWS CLI.

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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()));
    }
}
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtener una lista de los buckets.

```
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;
  }
}
};
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para JavaScript.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando devuelve todos los buckets de S.

```
Get-S3Bucket
```

Ejemplo 2: este comando devuelve un bucket denominado "test-files"

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando devuelve todos los buckets de S.

```
Get-S3Bucket
```

Ejemplo 2: este comando devuelve un bucket denominado “test-files”

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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__

```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para Ruby.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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(())
}

```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [ListBuckets](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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())  
}
```

- Para obtener información acerca de la API, consulte [ListBuckets](#) en la Referencia de la API del AWS SDK para Swift.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **ListMultipartUploads** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar ListMultipartUploads.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Eliminación de cargas multiparte incompletas](#)

CLI

AWS CLI

El siguiente comando muestra todas las cargas multiparte activas de un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api list-multipart-uploads --bucket amzn-s3-demo-bucket
```

Salida:

```
{  
  "Uploads": [  
    {  
      "Initiator": {  
        "DisplayName": "username",  
        "ID": "arn:aws:iam::0123456789012:user/username"  
      },  
      "Initiated": "2015-06-02T18:01:30.000Z",
```

```

        "UploadId":
        "dfRtDYU0WWCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6YjlX.cph0gtNBtJ8P3",
        "StorageClass": "STANDARD",
        "Key": "multipart/01",
        "Owner": {
            "DisplayName": "aws-account-name",
            "ID":
            "100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
        }
    },
    "CommonPrefixes": []
}

```

Las cargas multiparte en curso conllevan costos de almacenamiento en Amazon S3. Complete o anule una carga multiparte activa para eliminar sus partes de su cuenta.

- Para obtener información sobre la API, consulte [ListMultipartUploads](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
    }
}
```

- Para obtener información sobre la API, consulte [ListMultipartUploads](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **ListObjectVersions** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `ListObjectVersions`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Introducción al almacenamiento de objetos](#)
- [Trabajo con objetos con control de versiones](#)

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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}'");  
    }  
}  
}
```

- Para obtener información sobre la API, consulte [ListObjectVersions](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

El siguiente comando recupera la información de la versión de un objeto en un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix index.html
```

Salida:

```
{  
  "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
}
]
}

```

- Para obtener información sobre la API, consulte [ListObjectVersions](#) en la Referencia de comandos de la AWS CLI.

Go

SDK para Go V2

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

- Para obtener información sobre la API, consulte [ListObjectVersions](#) en la Referencia de la API de AWS SDK para Go.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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(())
}
```

- Para obtener información sobre la API, consulte [ListObjectVersions](#) en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [ListObjectVersions](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **ListObjects** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar ListObjects.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Crear una página web que enumere los objetos de Amazon S3](#)

CLI

AWS CLI

En el siguiente ejemplo se utiliza el comando `list-objects` para mostrar los nombres de todos los objetos del bucket especificado:

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

En el ejemplo se utiliza el argumento `--query` para filtrar la salida de `list-objects` hasta el valor de la clave y el tamaño de cada objeto

Para obtener más información sobre los objetos, consulte Trabajo con objetos de Amazon S3 en la Guía para desarrolladores de Amazon S3.

- Para obtener detalles de la API, consulte [ListObjects](#) en la Referencia de comandos de AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando recupera la información sobre todos los elementos del bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Ejemplo 2: este comando recupera la información sobre el elemento “sample.txt” del bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Ejemplo 3: este comando recupera la información sobre todos los elementos con el prefijo “sample” del bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Para obtener información sobre la API, consulte [ListObjects](#) en la Referencia de Cmdlet de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando recupera la información sobre todos los elementos del bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Ejemplo 2: este comando recupera la información sobre el elemento “sample.txt” del bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Ejemplo 3: este comando recupera la información sobre todos los elementos con el prefijo “sample” del bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Para obtener información sobre la API, consulte [ListObjects](#) en la Referencia de Cmdlet de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **ListObjectsV2** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar ListObjectsV2.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Conceptos básicos](#)
- [Eliminación de todos los objetos de un bucket](#)
- [Introducción al almacenamiento de objetos](#)

.NET

SDK para .NET (v4)

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}
```

```
}
```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK para .NET.

SDK para .NET

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Muestre objetos con un paginador.

```
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}");
            }
        }
    }
}

```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK para .NET.

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```


- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de comandos de AWS CLI.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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());
        }
    } while (continuationToken != "");
}
```

```

        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;
}

```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

Obtención de una lista de objetos en un bucket

En el siguiente ejemplo de `list-objects-v2` se muestran los objetos del bucket especificado.

```
aws s3api list-objects-v2 \
  --bucket amzn-s3-demo-bucket
```

Salida:

```
{
  "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
    }
]
}

```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de comandos de AWS CLI.

Go

SDK para Go V2

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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  
}
```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);  
    });  
}
```

Muestre objetos mediante paginación.

```
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);
    }
}
}
```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Enumere todos los objetos del bucket. Si hay más de un objeto, se usarán `isTruncated` y `NextContinuationToken` para repetir la lista completa.

```
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;
    }
}
};
```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtenga una lista de objetos de un bucket.

```
$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.");
}
```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK para PHP.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
    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__

```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK para Ruby.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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(())
}

```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
}
```

- Para obtener información acerca de la API, consulte [ListObjectsV2](#) en la Referencia de la API del SDK de AWS para Swift.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketAccelerateConfiguration** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `PutBucketAccelerateConfiguration`.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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>
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
the
    /// 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");
    }
}
```

- Para obtener información sobre la API, consulte [PutBucketAccelerateConfiguration](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

Establecer la configuración acelerada de un bucket

En el siguiente ejemplo de `put-bucket-accelerate-configuration`, se habilita la configuración acelerada para el bucket especificado.

```
aws s3api put-bucket-accelerate-configuration \
    --bucket amzn-s3-demo-bucket \
    --accelerate-configuration Status=Enabled
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [PutBucketAccelerateConfiguration](#) en Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando habilita la aceleración de transferencia para el bucket de S3 indicado.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Para obtener información sobre la API, consulte [PutBucketAccelerateConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando habilita la aceleración de transferencia para el bucket de S3 indicado.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Para obtener información sobre la API, consulte [PutBucketAccelerateConfiguration](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketAc1** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutBucketAc1.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Administrar listas de control de acceso \(ACL\)](#)

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;  
}
```

- Para obtener información sobre la API, consulte [PutBucketAcl](#) en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
    }

```

- Para obtener información sobre la API, consulte [PutBucketAcl](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

Este ejemplo otorga `full control` a dos usuarios de AWS (`user1@example.com` y `user2@example.com`) y permiso de `read` a todos los usuarios:

```
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
```

Consulte <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> para obtener más información sobre las ACL personalizadas (los comandos `s3api` de ACL, como `put-bucket-acl`, utilizan la misma notación abreviada para el argumentos).

- Para obtener información sobre la API, consulte [PutBucketAcl](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
        }
    }
}

```

- Para obtener información sobre la API, consulte [PutBucketAcl](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Coloque la ACL del bucket.

```
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;
    }
}
};

```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para obtener información sobre la API, consulte [PutBucketAcl](#) en la Referencia de la API de AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 ownerOb =
        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")
    }
}

```

- Para obtener información sobre la API, consulte [PutBucketAcl](#) en la Referencia de la API de AWS SDK para Kotlin.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [PutBucketAcl](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [PutBucketAcl](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketCors** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutBucketCors.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
}
```

- Para obtener información sobre la API, consulte [PutBucketCors](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

El siguiente ejemplo habilita solicitudes PUT, POST y DELETE desde www.ejemplo.com, y habilita solicitudes GET desde cualquier dominio:

```
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
    }
  ]
}
```

- Para obtener información acerca de la API, consulte [PutBucketCors](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
}
}
```

- Para obtener información sobre la API, consulte [PutBucketCors](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Añada una regla 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;
      }
    }
  };
};
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para obtener información sobre la API, consulte [PutBucketCors](#) en la Referencia de la API de AWS SDK para JavaScript.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

```

- Para obtener información sobre las API, consulte [PutBucketCors](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [PutBucketCors](#) en la Referencia de la API de AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [PutBucketCors](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketEncryption** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutBucketEncryption.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Introducción al almacenamiento de objetos](#)

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}
```

- Para obtener información sobre la API, consulte [PutBucketEncryption](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

Configurar el cifrado del lado del servidor de un bucket

En el siguiente ejemplo de `put-bucket-encryption` se establece el cifrado AES256 como predeterminado para el bucket especificado.

```
aws s3api put-bucket-encryption \
    --bucket amzn-s3-demo-bucket \
    --server-side-encryption-configuration '{"Rules":
[{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [PutBucketEncryption](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando habilita el cifrado AES256 predeterminado del servidor con claves administradas de Amazon S3 (SSE-S3) en el bucket indicado.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Para obtener información sobre la API, consulte [PutBucketEncryption](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando habilita el cifrado AES256 predeterminado del servidor con claves administradas de Amazon S3 (SSE-S3) en el bucket indicado.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Para obtener información sobre la API, consulte [PutBucketEncryption](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketLifecycleConfiguration** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutBucketLifecycleConfiguration.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Eliminación de cargas multiparte incompletas](#)

- [Administración de mensajes grandes mediante S3](#)
- [Trabajo con objetos con control de versiones](#)

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
}
```

- Para obtener información sobre la API, consulte [PutBucketLifecycleConfiguration](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

El comando siguiente aplica una configuración del ciclo de vida a un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --  
lifecycle-configuration file://lifecycle.json
```

El archivo `lifecycle.json` es un documento JSON en la carpeta actual que especifica dos reglas:

```
{  
  "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 primera regla mueve los archivos con el prefijo `rotated` a Glacier en la fecha especificada. La segunda regla mueve las versiones del objeto antiguas a Glacier cuando ya

no están actualizadas. Para obtener más información sobre los formatos de marca temporal permitidos, consulte Especificación de valores de parámetros para la Guía del usuario de la AWS CLI.

- Para obtener información sobre la API, consulte [PutBucketLifecycleConfiguration](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
    }
}
}

```

- Para obtener información sobre la API, consulte [PutBucketLifecycleConfiguration](#) en la Referencia de la API de AWS SDK for Java 2.x.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [PutBucketLifecycleConfiguration](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [PutBucketLifecycleConfiguration](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketLogging** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutBucketLogging.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
    }
}
```

- Para obtener información sobre la API, consulte [PutBucketLogging](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

Ejemplo 1: Configuración del registro de políticas de bucket

En el siguiente ejemplo de `put-bucket-logging`, se establece la política de registro para `amzn-s3-demo-bucket`. En primer lugar, conceda al servicio de registro el permiso de entidad principal en la política de bucket mediante el comando `put-bucket-policy`.

```
aws s3api put-bucket-policy \
  --bucket amzn-s3-demo-bucket \
  --policy file://policy.json
```

Contenido 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"}
      }
    }
  ]
}
```

```
}
```

Para aplicar la política de registro, use `put-bucket-logging`.

```
aws s3api put-bucket-logging \  
  --bucket amzn-s3-demo-bucket \  
  --bucket-logging-status file:///logging.json
```

Contenido de `logging.json`:

```
{  
  "LoggingEnabled": {  
    "TargetBucket": "amzn-s3-demo-bucket",  
    "TargetPrefix": "Logs/"  
  }  
}
```

El comando `put-bucket-policy` es necesario para conceder permisos `s3:PutObject` a la entidad principal del servicio de registro.

Para obtener más información, consulte [Registro de acceso al servidor de Amazon S3](#) en la Guía del usuario de Amazon S3.

Ejemplo 2: Establecimiento de una política de bucket para registrar el acceso a un solo usuario

En el siguiente ejemplo de `put-bucket-logging`, se establece la política de registro para `amzn-s3-demo-bucket`. El usuario de AWS `bob@example.com` tendrá el control total sobre los archivos de registro y nadie más tendrá acceso a ellos. En primer lugar, conceda permiso de S3 mediante `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
```

A continuación, aplique la política de registro mediante `put-bucket-logging`.

```
aws s3api put-bucket-logging \  
  --bucket amzn-s3-demo-bucket \  
  --bucket-logging-status file:///logging.json
```

```
--bucket-logging-status file://logging.json
```

Contenido 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"
      }
    ]
  }
}
```

el comando `put-bucket-acl` es necesario para conceder los permisos necesarios (write y read-acp) al sistema de entrega de registros de S3.

Para obtener más información, consulte [Registro de acceso al servidor de Amazon S3](#) en la Guía para desarrolladores de Amazon S3.

- Para obtener información sobre la API, consulte [PutBucketLogging](#) en la Referencia de comandos de la AWS CLI.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **PutBucketNotification** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar `PutBucketNotification`.

CLI

AWS CLI

Aplica una configuración de notificación a un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-configuration file://notification.json
```

El archivo `notification.json` es un documento JSON en la carpeta actual que especifica un tema de SNS y un tipo de evento para supervisar:

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

El tema de SNS debe tener una política de IAM adjunta que permita a Amazon S3 publicar en él:

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

- Para obtener información sobre la API, consulte [PutBucketNotification](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este ejemplo configura la configuración del tema de SNS para el evento ObjectRemovedDelete de S3 y se habilita la notificación para el bucket de S3 indicado

```
$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
```

Ejemplo 2: este ejemplo habilita las notificaciones de ObjectCreatedAll para el bucket indicado y las envía a la función de 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
```

Ejemplo 3: este ejemplo crea 2 configuraciones de Lambda diferentes sobre la base de un sufijo clave diferente y las configura en un solo comando.

```
#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

```

- Para obtener información sobre la API, consulte [PutBucketNotification](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este ejemplo configura la configuración del tema de SNS para el evento ObjectRemovedDelete de S3 y se habilita la notificación para el bucket de S3 indicado

```

$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

```

Ejemplo 2: este ejemplo habilita las notificaciones de ObjectCreatedAll para el bucket indicado y las envía a la función de 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

```

Ejemplo 3: este ejemplo crea 2 configuraciones de Lambda diferentes sobre la base de un sufijo clave diferente y las configura en un solo comando.

```

#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
```

- Para obtener información sobre la API, consulte [PutBucketNotification](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketNotificationConfiguration** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `PutBucketNotificationConfiguration`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Procese notificaciones de eventos de S3](#)
- [Envío de notificaciones de eventos a EventBridge](#)

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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}");  
    }  
}  
}
```

- Para obtener información sobre la API, consulte [PutBucketNotificationConfiguration](#) en Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

Habilitación de las notificaciones especificadas en un bucket

El siguiente ejemplo de `put-bucket-notification-configuration` se aplica una configuración de notificación a un bucket llamado `amzn-s3-demo-bucket`. El archivo `notification.json` es un documento JSON en la carpeta actual que especifica un tema de SNS y un tipo de evento para supervisar.

```
aws s3api put-bucket-notification-configuration \  
    --bucket amzn-s3-demo-bucket \  
    --notification-configuration file://notification.json
```

Contenido de `notification.json`:

```
{  
  "TopicConfigurations": [  
    {  
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-  
topic",  
      "Events": [  
        "s3:ObjectCreated:*"  
      ]  
    }  
  ]  
}
```

El tema de SNS debe tener una política de IAM adjunta que permita a Amazon S3 publicar en él.

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

- Para obtener información sobre la API, consulte [PutBucketNotificationConfiguration](#) en Referencia de comandos de la AWS CLI.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketPolicy** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutBucketPolicy.

C++

SDK para C++

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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"
        "}";
}

```

- Para obtener información acerca de la API, consulte [PutBucketPolicy](#) en la referencia de la API de AWS SDK para C++.

CLI

AWS CLI

Este ejemplo permite a todos los usuarios recuperar cualquier objeto en amzn-s3-demo-bucket excepto aquellos en MySecretFolder. También concede un permiso put y delete al usuario raíz de la cuenta de AWS 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/*"  
    }  
  ]  
}
```

- Para obtener información sobre la API, consulte [PutBucketPolicy](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
}
}
```

- Para obtener información acerca de la API, consulte [PutBucketPolicy](#) en la referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Añada la política.

```
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;
  }
}
};

```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para obtener información sobre la API, consulte [PutBucketPolicy](#) en la Referencia de la API de AWS SDK para JavaScript.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [PutBucketPolicy](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información acerca de la API, consulte [PutBucketPolicy](#) en la referencia de la API de AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [PutBucketPolicy](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketReplication** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutBucketReplication.

CLI

AWS CLI

Configurar la replicación de un bucket de S

El siguiente ejemplo de `put-bucket-replication` aplica una configuración de replicación al bucket de S3 especificado.

```
aws s3api put-bucket-replication \  
  --bucket amzn-s3-demo-bucket1 \  
  --replication-configuration file://replication.json
```

Contenido 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"  
      }  
    }  
  ]  
}
```

El bucket de destino debe tener habilitado el control de versiones. El rol especificado debe tener permiso para escribir en el bucket de destino y tener una relación de confianza que permita a Amazon S3 asumir el rol.

Ejemplo de política de permisos de roles:

```
{  
  "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/*"
    }
  ]
}

```

Ejemplo de política de relación de confianza:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

Este comando no genera ninguna salida.

Para obtener más información, consulte [Este es el título del tema](#) en la guía del usuario de la consola de Amazon Simple Storage Service:

- Para obtener información sobre la API, consulte [PutBucketReplication](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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());
}
}
```

- Para obtener información sobre la API, consulte [PutBucketReplication](#) en la Referencia de la API de AWS SDK for Java 2.x.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: Este ejemplo establece una configuración de replicación con una sola regla que permite replicar en el bucket “amzn-s3-demo-bucket” cualquier objeto nuevo creado con el prefijo de nombre de clave “TaxDocs” en el bucket “amzn-s3-demo-bucket”.

```
$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
```

Ejemplo 2: Este ejemplo establece una configuración de replicación con varias reglas que permiten replicar en el bucket “amzn-s3-demo-bucket” cualquier objeto nuevo creado con el prefijo de nombre de clave “TaxDocs” u “OtherDocs”. Los prefijos de claves no deben superponerse.

```
$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
```

Ejemplo 3: Este ejemplo actualiza la configuración de replicación en el bucket especificado para inhabilitar la regla que controla la replicación de objetos con el prefijo de nombre de clave "TaxDocs" en el bucket "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
```

- Para obtener información sobre la API, consulte [PutBucketReplication](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: Este ejemplo establece una configuración de replicación con una sola regla que permite replicar en el bucket “amzn-s3-demo-bucket” cualquier objeto nuevo creado con el prefijo de nombre de clave “TaxDocs” en el bucket “amzn-s3-demo-bucket”.

```
$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
```

Ejemplo 2: Este ejemplo establece una configuración de replicación con varias reglas que permiten replicar en el bucket “amzn-s3-demo-bucket” cualquier objeto nuevo creado con el prefijo de nombre de clave “TaxDocs” u “OtherDocs”. Los prefijos de claves no deben superponerse.

```
$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

```

Ejemplo 3: Este ejemplo actualiza la configuración de replicación en el bucket especificado para inhabilitar la regla que controla la replicación de objetos con el prefijo de nombre de clave "TaxDocs" en el bucket "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

```

- Para obtener información sobre la API, consulte [PutBucketReplication](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **PutBucketRequestPayment** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar PutBucketRequestPayment.

CLI

AWS CLI

Ejemplo 1: habilitar la configuración de “el solicitante paga” para un bucket

El siguiente ejemplo de `put-bucket-request-payment` habilita `requester pays` para el bucket especificado.

```
aws s3api put-bucket-request-payment \
  --bucket amzn-s3-demo-bucket \
  --request-payment-configuration '{"Payer":"Requester"}
```

Este comando no genera ninguna salida.

Ejemplo 2: deshabilitar la configuración de “el solicitante paga” para un bucket

El siguiente ejemplo de `put-bucket-request-payment` deshabilita `requester pays` para el bucket especificado.

```
aws s3api put-bucket-request-payment \
  --bucket amzn-s3-demo-bucket \
  --request-payment-configuration '{"Payer":"BucketOwner"}
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [PutBucketRequestPayment](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: Actualiza la configuración de pago de solicitudes del bucket denominado “amzn-s3-demo-bucket”, de modo que se cobre la descarga a la persona que solicita las descargas del bucket. De forma predeterminada, el propietario del bucket paga las descargas. Para volver a establecer el pago de la solicitud al modo predeterminado, use “BucketOwner” para el parámetro `RequestPaymentConfiguration_Payer`.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- Para obtener información sobre la API, consulte [PutBucketRequestPayment](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: Actualiza la configuración de pago de solicitudes del bucket denominado “amzn-s3-demo-bucket”, de modo que se cobre la descarga a la persona que solicita las descargas del bucket. De forma predeterminada, el propietario del bucket paga las descargas. Para volver a establecer el pago de la solicitud al modo predeterminado, use “BucketOwner” para el parámetro `RequestPaymentConfiguration_Payer`.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- Para obtener información sobre la API, consulte [PutBucketRequestPayment](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Utilizar **PutBucketTagging** con una CLI

Los siguientes ejemplos de código muestran cómo utilizar `PutBucketTagging`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Introducción al almacenamiento de objetos](#)

CLI

AWS CLI

El siguiente comando aplica una configuración de etiquetado a un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://  
tagging.json
```

El archivo `tagging.json` es un documento JSON en la carpeta actual que especifica etiquetas:

```
{
  "TagSet": [
    {
      "Key": "organization",
      "Value": "marketing"
    }
  ]
}
```

O aplique una configuración de etiquetado a `amzn-s3-demo-bucket` directamente desde la línea de comandos:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging
'TagSet=[{Key=organization, Value=marketing}]'
```

- Para obtener información sobre la API, consulte [PutBucketTagging](#) en la Referencia de comandos de la AWS CLI.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando aplica dos etiquetas a un bucket denominado **cloudtrail-test-2018**, una etiqueta con una clave de Stage y un valor de Test, y una etiqueta con una clave de Environment y un valor de Alpha. Para comprobar que las etiquetas se han añadido al bucket, ejecute **Get-S3BucketTagging -BucketName bucket_name**. Los resultados deben mostrar las etiquetas que ha aplicado al bucket en el primer comando. Tenga en cuenta que **Write-S3BucketTagging** sobrescribe todo el conjunto de etiquetas existente en un bucket. Para añadir o eliminar etiquetas individuales, ejecute los cmdlets Resource Groups y Tagging API, **Add-RGTResourceTag** y **Remove-RGTResourceTag**. Como alternativa, puede utilizar el editor de etiquetas de la consola de administración de AWS para administrar las etiquetas del bucket de S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
  Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Ejemplo 2: este comando canaliza un bucket denominado **cloudtrail-test-2018** al cmdlet de **Write-S3BucketTagging**. Aplica las etiquetas Stage:Production y Department:Finance al bucket. Tenga en cuenta que **Write-S3BucketTagging** sobrescribe todo el conjunto de etiquetas existente en un bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
Value="Finance" } )
```

- Para obtener información sobre la API, consulte [PutBucketTagging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando aplica dos etiquetas a un bucket denominado **cloudtrail-test-2018**, una etiqueta con una clave de Stage y un valor de Test, y una etiqueta con una clave de Environment y un valor de Alpha. Para comprobar que las etiquetas se han añadido al bucket, ejecute **Get-S3BucketTagging -BucketName bucket_name**. Los resultados deben mostrar las etiquetas que ha aplicado al bucket en el primer comando. Tenga en cuenta que **Write-S3BucketTagging** sobrescribe todo el conjunto de etiquetas existente en un bucket. Para añadir o eliminar etiquetas individuales, ejecute los cmdlets Resource Groups y Tagging API, **Add-RGResourceTag** y **Remove-RGResourceTag**. Como alternativa, puede utilizar el editor de etiquetas de la consola de administración de AWS para administrar las etiquetas del bucket de S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Ejemplo 2: este comando canaliza un bucket denominado **cloudtrail-test-2018** al cmdlet de **Write-S3BucketTagging**. Aplica las etiquetas Stage:Production y Department:Finance al bucket. Tenga en cuenta que **Write-S3BucketTagging** sobrescribe todo el conjunto de etiquetas existente en un bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
Value="Finance" } )
```

- Para obtener información sobre la API, consulte [PutBucketTagging](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketVersioning** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutBucketVersioning.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Introducción al almacenamiento de objetos](#)

CLI

AWS CLI

El siguiente comando habilita el control de versiones en un bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled
```

El siguiente comando habilita el control de versiones y usa un código mfa

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled --mfa "SERIAL 123456"
```

- Para obtener información sobre la API, consulte [PutBucketVersioning](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
}
```

- Para obtener información sobre la API, consulte [PutBucketVersioning](#) en la Referencia de la API de AWS SDK for Java 2.x.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: el comando habilita el control de versiones para el bucket de S3 indicado.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- Para obtener información sobre la API, consulte [PutBucketVersioning](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: el comando habilita el control de versiones para el bucket de S3 indicado.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- Para obtener información sobre la API, consulte [PutBucketVersioning](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [PutBucketVersioning](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketWebsite** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutBucketWebsite.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// Put the website configuration.
PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
{
    BucketName = bucketName,
    WebsiteConfiguration = new WebsiteConfiguration()
    {
        IndexDocumentSuffix = indexDocumentSuffix,
        ErrorDocument = errorDocument,
    },
};
PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- Para obtener información sobre la API, consulte [PutBucketWebsite](#) en la Referencia de la API de AWS SDK para .NET.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
```

```

        const Aws::String &indexPath, const Aws::String
&errorPage,
        const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPath);

    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();
}

```

- Para obtener información sobre la API, consulte [PutBucketWebsite](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

Aplica una configuración de sitio web estática a un bucket llamado `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-configuration file://website.json
```

El archivo `website.json` es un documento JSON en la carpeta actual que especifica las páginas de índice y error del sitio web:

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- Para obtener información sobre la API, consulte [PutBucketWebsite](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
    }
}
```

- Para obtener información sobre la API, consulte [PutBucketWebsite](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Establezca la configuración de sitio 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;  
    }  
}  
};
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para obtener información sobre la API, consulte [PutBucketWebsite](#) en la Referencia de la API de AWS SDK para JavaScript.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: el comando habilita el alojamiento de sitios web para el bucket indicado con el documento de índice como "index.html" y el documento de error como "error.html".

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'  
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Para obtener información sobre la API, consulte [PutBucketWebsite](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: el comando habilita el alojamiento de sitios web para el bucket indicado con el documento de índice como "index.html" y el documento de error como "error.html".

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'  
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -  
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Para obtener información sobre la API, consulte [PutBucketWebsite](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

```
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__
```

- Para obtener información sobre la API, consulte [PutBucketWebsite](#) en la Referencia de la API de AWS SDK para Ruby.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutObject** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutObject.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en los siguientes ejemplos de código:

- [Conceptos básicos](#)
- [Introducción al almacenamiento de objetos](#)
- [Cómo realizar solicitudes condicionales](#)
- [Realización de un seguimiento de cargas y descargas](#)
- [Trabajo con la integridad de los objetos de Amazon S3](#)

.NET

SDK para .NET (v4)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
    }
}
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK para .NET.

SDK para .NET

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cargar un objeto con cifrado del lado del servidor.

```
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");
    }
}

```

Colocar un objeto mediante una solicitud condicional.

```

/// <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;
    }
}

```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK para .NET.

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de comandos de AWS CLI.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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();
}

```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

Ejemplo 1: carga de un objeto en Amazon S3

El ejemplo de comando `put-object` siguiente carga un objeto en Amazon S3.

```
aws s3api put-object \  
  --bucket amzn-s3-demo-bucket \  
  --key my-dir/MySampleImage.png \  
  --body MySampleImage.png
```

Para obtener más información acerca de cómo cargar objetos, consulte [Carga de objetos < http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html>](http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html) en la Guía para desarrolladores de Amazon S3.

Ejemplo 2: carga de un archivo de vídeo en Amazon S3

El ejemplo de comando `put-object` siguiente carga un archivo de vídeo.

```
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
```

Para obtener más información acerca de cómo cargar objetos, consulte [Carga de objetos < http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html>](http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html) en la Guía para desarrolladores de Amazon S3.

- Para obtener detalles de la API, consulte [PutObject](#) en la Referencia de comandos de AWS CLI.

Go

SDK para Go V2

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Coloque un objeto en un bucket con la API de bajo nivel.

```
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
}

```

Cargue un objeto en un bucket con un administrador de transferencias.

```

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
}
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cargue un archivo en un bucket con 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);
        }
    });
}
```

Utilice un [S3TransferManager](#) para [cargar un archivo](#) a un bucket. Vea el [archivo completo](#) y [pruébelo](#).

```
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();
    }
```

Cargue un objeto en un bucket y configure las etiquetas con 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;
    }
}
```

Cargue un objeto en un bucket y configure los metadatos con 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);
    }
}

```

Cargue un objeto en un bucket y configure un valor de retención de objetos con 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 setRentionPeriod(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);
    }
}
}
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cargue el objeto.

```
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;
    }
  }
};

```

Cargar el objeto con la condición de que su ETag coincida con la proporcionada.

```

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"));
  }
}
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).
- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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}")
    }
}
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK para Kotlin.

PHP

SDK para PHP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cargue un objeto en un bucket.

```
$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.");
}
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK para PHP.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: este comando carga el archivo único “local-sample.txt” a Amazon S3 y crea un objeto con la clave “sample.txt” en el bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Ejemplo 2: este comando carga el archivo único “sample.txt” a Amazon S3 y crea un objeto con la clave “sample.txt” en el bucket “test-files”. Si no se proporciona el parámetro -Key, el nombre del archivo se utiliza como clave de objeto de S.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Ejemplo 3: este comando carga el archivo único “local-sample.txt” a Amazon S3 y crea un objeto con la clave “prefix/to/sample.txt” en el bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -  
File .\local-sample.txt
```

Ejemplo 4: este comando carga todos los archivos del subdirectorio “Scripts” al bucket “test-files” y aplica el prefijo de clave común “SampleScripts” a cada objeto. Cada archivo cargado tendrá una clave de “SampleScripts/filename”, donde “filename” varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\
```

Ejemplo 5: este comando carga todos los archivos *.ps1 en el director local “Scripts” al bucket “test-files” y aplica el prefijo de clave común “SampleScripts” a cada objeto. Cada archivo cargado tendrá una clave de “SampleScripts/filename.ps1”, donde “filename” varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Ejemplo 6: este comando crea un nuevo objeto S3 que contiene la cadena de contenido especificada con la clave “sample.txt”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Ejemplo 7: este comando carga el archivo especificado (el nombre del archivo se usa como clave) y aplica las etiquetas especificadas al nuevo objeto.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Ejemplo 8: este comando carga de forma recursiva la carpeta especificada y aplica las etiquetas especificadas a todos los objetos nuevos.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: este comando carga el archivo único “local-sample.txt” a Amazon S3 y crea un objeto con la clave “sample.txt” en el bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Ejemplo 2: este comando carga el archivo único “sample.txt” a Amazon S3 y crea un objeto con la clave “sample.txt” en el bucket “test-files”. Si no se proporciona el parámetro -Key, el nombre del archivo se utiliza como clave de objeto de S.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Ejemplo 3: este comando carga el archivo único “local-sample.txt” a Amazon S3 y crea un objeto con la clave “prefix/to/sample.txt” en el bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -
File .\local-sample.txt
```

Ejemplo 4: este comando carga todos los archivos del subdirectorio “Scripts” al bucket “test-files” y aplica el prefijo de clave común “SampleScripts” a cada objeto. Cada archivo cargado tendrá una clave de “SampleScripts/filename”, donde “filename” varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
SampleScripts\
```

Ejemplo 5: este comando carga todos los archivos *.ps1 en el director local “Scripts” al bucket “test-files” y aplica el prefijo de clave común “SampleScripts” a cada objeto. Cada archivo cargado tendrá una clave de “SampleScripts/filename.ps1”, donde “filename” varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Ejemplo 6: este comando crea un nuevo objeto S3 que contiene la cadena de contenido especificada con la clave "sample.txt".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Ejemplo 7: este comando carga el archivo especificado (el nombre del archivo se usa como clave) y aplica las etiquetas especificadas al nuevo objeto.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Ejemplo 8: este comando carga de forma recursiva la carpeta especificada y aplica las etiquetas especificadas a todos los objetos nuevos.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"  
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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()
```

Cargar un objeto mediante una solicitud condicional.

```
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
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cargue un archivo con un cargador administrado (`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__

```

Cargue un archivo con `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}"
  end
end

```

```

    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__

```

Cargue un archivo con `Object.put` y añada cifrado del lado del servidor.

```

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__
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK para Ruby.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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)
}
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
    }
}
```

- Para obtener información acerca de la API, consulte [PutObject](#) en la Referencia de la API del SDK de AWS para Swift.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutObjectAcl** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar **PutObjectAcl**.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en su contexto en el siguiente ejemplo de código:

- [Administrar listas de control de acceso \(ACL\)](#)

C++

SDK para C++

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}

```

- Para obtener información sobre la API, consulte [PutObjectAcl](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El comando siguiente concede `full control` a dos usuarios de AWS (`user1@example.com` y `user2@example.com`) y permiso de `read` a todos los usuarios:

```
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
```

Consulte <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> para obtener más información sobre las ACL personalizadas (los comandos `s3api` de ACL, como `put-object-acl`, utilizan la misma notación abreviada para el argumentos).

- Para obtener información acerca de la API, consulte [PutObjectAcl](#) en la Referencia de comandos de la AWS CLI.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

```

- Para obtener información sobre la API, consulte [PutObjectAcl](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.

```

- Para obtener información sobre la API, consulte [PutObjectAcl](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutObjectLegalHold** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar **PutObjectLegalHold**.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Bloqueo de objetos de Amazon S3](#)

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}
}
```

- Para obtener información sobre la API, consulte [PutObjectLegalHold](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

Para aplicar una retención legal a un objeto

En el siguiente ejemplo de `put-object-legal-hold`, se establece una retención legal sobre el objeto `doc1.rtf`.

```
aws s3api put-object-legal-hold \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf \
  --legal-hold Status=ON
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [PutObjectLegalHold](#) en la Referencia de comandos de la AWS CLI.

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}

```

- Para obtener información sobre la API, consulte [PutObjectLegalHold](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 +".");
}

```

- Para obtener información sobre la API, consulte [PutObjectLegalHold](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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"));  
  }  
}
```

- Para obtener información sobre la API, consulte [PutObjectLegalHold](#) en la Referencia de la API de AWS SDK para JavaScript.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ponga una retención legal en un objeto.

```
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
        )
```

- Para obtener información sobre la API, consulte [PutObjectLegalHold](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [PutObjectLegalHold](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutObjectLockConfiguration** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutObjectLockConfiguration.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Bloqueo de objetos de Amazon S3](#)

.NET

SDK para .NET

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Establecimiento de la configuración de bloqueo de objetos de un bucket

```
/// <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($"{Environment.NewLine}\tAdded an object lock policy to bucket
{bucketName}.");
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
    return false;
}
}

```

Establecimiento del período de retención predeterminado de un bucket

```

/// <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;
}
}

```

- Para obtener información sobre la API, consulte [PutObjectLockConfiguration](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

Para establecer la configuración de bloqueo de objetos en un bucket

En el siguiente ejemplo de `put-object-lock-configuration`, se establece un bloqueo de objetos de 50 días en el bucket especificado.

```
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 }}}'
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [PutObjectLockConfiguration](#) en la Referencia de comandos de la AWS CLI.

Go

SDK para Go V2

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Establecimiento de la configuración de bloqueo de objetos de un bucket

```
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
}
```

Establecimiento del período de retención predeterminado de un bucket

```
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
}

```

- Para obtener información sobre la API, consulte [PutObjectLockConfiguration](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Establecimiento de la configuración de bloqueo de objetos de un bucket

```

// 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()
+ "'");
}
}
```

Establecimiento del período de retención predeterminado de un bucket

```
// 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
+".");
}
```

- Para obtener información sobre la API, consulte [PutObjectLockConfiguration](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Establecimiento de la configuración de bloqueo de objetos de un bucket

```
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"));
    }
}
```

Establecimiento del período de retención predeterminado de un bucket

```
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),
            },
          },
        },
      })
    );
  } catch (err) {
    if (err instanceof S3ServiceException) {
      console.log("S3ServiceException: ", err.message);
    } else {
      console.log("Unknown error: ", err);
    }
  }
}
```

```

        },
    },
    },
  )),
);
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"));
    }
}
```

- Para obtener información sobre la API, consulte [PutObjectLockConfiguration](#) en la Referencia de la API de AWS SDK para JavaScript.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Configuración de la activación del bloqueo de objetos.

```
s3_client.put_object_lock_configuration(
    Bucket=bucket,
    ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":
{}}
)
```

- Para obtener más información sobre la API, consulte [PutObjectLockConfiguration](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [PutObjectLockConfiguration](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutObjectRetention** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar PutObjectRetention.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Bloqueo de objetos de Amazon S3](#)

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
        }
    }
```

- Para obtener información sobre la API, consulte [PutObjectRetention](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

Para establecer la configuración de retención de un objeto

En el siguiente ejemplo de `put-object-retention`, se establece una configuración de retención del objeto especificado hasta el 1 de enero de 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" }'
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [PutObjectRetention](#) en la Referencia de comandos de la AWS CLI.

Go

SDK para Go V2

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
}
```

- Para obtener información sobre la API, consulte [PutObjectRetention](#) en la Referencia de la API de AWS SDK para Go.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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 +".");
    }

```

- Para obtener información sobre la API, consulte [PutObjectRetention](#) en la Referencia de la API de AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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/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"));
  }
}
```

- Para obtener información sobre la API, consulte [PutObjectRetention](#) en la Referencia de la API de AWS SDK para JavaScript.

PowerShell

Herramientas para PowerShell V4

Ejemplo 1: el comando habilita el modo de retención de gobierno hasta la fecha “31 de diciembre de 2019 a las 00:00:00” para el objeto “testfile.txt” del bucket de S3 indicado.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -  
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Para obtener información sobre la API, consulte [PutObjectRetention](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V4).

Herramientas para PowerShell V5

Ejemplo 1: el comando habilita el modo de retención de gobierno hasta la fecha “31 de diciembre de 2019 a las 00:00:00” para el objeto “testfile.txt” del bucket de S3 indicado.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -  
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Para obtener información sobre la API, consulte [PutObjectRetention](#) en la Referencia de cmdlets de Herramientas de AWS para PowerShell (V5).

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ponga un retención de objeto.

```
s3_client.put_object_retention(  
    Bucket=bucket,  
    Key=key,  
    VersionId=version_id,  
    Retention={"Mode": "GOVERNANCE", "RetainUntilDate":  
far_future_date},  
    BypassGovernanceRetention=True,  
)
```

- Para obtener información sobre la API, consulte [PutObjectRetention](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [PutObjectRetention](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **RestoreObject** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `RestoreObject`.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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}");
    }
}
```

- Para obtener información sobre la API, consulte [RestoreObject](#) en la Referencia de la API de AWS SDK para .NET.

CLI

AWS CLI

Creación de una solicitud de restauración para un objeto

En el siguiente ejemplo de `restore-object` se restaura el objeto de Amazon S3 Glacier especificado para el bucket `my-glacier-bucket` durante 10 días.

```
aws s3api restore-object \
    --bucket my-glacier-bucket \
    --key doc1.rtf \
    --restore-request Days=10
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [RestoreObject](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
    }
}
```

- Para obtener información sobre la API, consulte [RestoreObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **SelectObjectContent** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar SelectObjectContent.

CLI

AWS CLI

Para filtrar el contenido de un objeto de Amazon S3 en función de una instrucción SQL

En el siguiente ejemplo de select-object-content, se filtra el objeto my-data-file.csv con la instrucción SQL especificada y se envía el resultado a un archivo.

```
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"
```

Este comando no genera ninguna salida.

- Para obtener información sobre la API, consulte [SelectObjectContent](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

En el siguiente ejemplo, se muestra una consulta utilizando un objeto JSON. En el [ejemplo completo](#), también se muestra el uso de un objeto 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;
}
}
```

- Para obtener información sobre la API, consulte [SelectObjectContent](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **UploadPart** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar UploadPart.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en su contexto en los siguientes ejemplos de código:

- [Usar sumas de comprobación](#)
- [Trabajo con la integridad de los objetos de Amazon S3](#)

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
}

```

- Para obtener información sobre la API, consulte [UploadPart](#) en la Referencia de la API de AWS SDK para C++.

CLI

AWS CLI

El siguiente comando carga la primera parte de una carga multiparte iniciada con el comando `create-multipart-upload`:

```
aws s3api upload-part --bucket amzn-s3-demo-bucket --key  
'multipart/01' --part-number 1 --body part01 --upload-id  
"dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P"
```

La opción `body` toma el nombre o la ruta de un archivo local para la carga (no utilice el prefijo `file://`). El tamaño mínimo de parte es de 5 MB. El ID de carga lo devuelve `create-multipart-upload` y también se puede recuperar con `list-multipart-uploads`. El bucket y la clave se especifican al crear la carga multiparte.

Salida:

```
{  
  "ETag": "\"e868e0f4719e394144ef36531ee6824c\""  
}
```

Guarde el valor de `ETag` de cada parte para más adelante. Son necesarios para completar la carga multiparte.

- Para obtener información sobre la API, consulte [UploadPart](#) en la Referencia de comandos de la AWS CLI.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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?;
```

- Para obtener detalles sobre la API, consulte [UploadPart](#) en la Referencia de la API del SDK de AWS para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **UploadPartCopy** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar UploadPartCopy.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Ejecución de una copia multiparte](#)

CLI

AWS CLI

Para cargar parte de un objeto al copiar los datos de un objeto existente como origen de datos

En el siguiente ejemplo de `upload-part-copy`, se carga una parte al copiar los datos de un objeto existente como origen de datos.

```
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"
```

Salida:

```
{  
  "CopyPartResult": {  
    "LastModified": "2019-12-13T23:16:03.000Z",  
    "ETag": "\"711470fc377698c393d94aed6305e245\""  
  }  
}
```

- Para obtener información sobre la API, consulte [UploadPartCopy](#) en la Referencia AWS CLI de comandos.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}

```

- Para obtener información sobre la API, consulte [UploadPartCopy](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Escenarios de Amazon S3 con SDK de AWS

Los siguientes ejemplos de código muestran cómo implementar escenarios comunes en Amazon S3 con el SDK AWS. Estos escenarios muestran cómo llevar a cabo tareas específicas con llamadas a varias funciones dentro de Amazon S3 o en combinación con otros Servicios de AWS. En cada escenario se incluye un enlace al código fuente completo, con instrucciones de configuración y ejecución del código.

Los escenarios requieren un nivel intermedio de experiencia para entender las acciones de servicio en su contexto.

Ejemplos

- [Comprobación de la existencia de un bucket](#)
- [Comprobación de si existe un objeto](#)
- [Convierta texto en voz y de nuevo a texto con un SDK de AWS](#)
- [Cree una URL prefirmada para Amazon S3 mediante un SDK de AWS](#)
- [Creación de una aplicación de administración de activos fotográficos que permita a los usuarios administrar las fotos mediante etiquetas](#)
- [Una página web que indica los objetos de Amazon S3 que usan un SDK de AWS](#)
- [Creación de una aplicación de exploración de Amazon Textract](#)
- [Eliminar todos los objetos de un bucket de Amazon S3 dado mediante un SDK de AWS](#)
- [Eliminación de las cargas multiparte incompletas a Amazon S3 mediante un AWS SDK](#)
- [Detección de EPI en imágenes con Amazon Rekognition mediante un AWS SDK](#)
- [Detecte entidades en el texto extraído de una imagen con un SDK de AWS](#)
- [Detecte rostros en una imagen con un SDK de AWS](#)
- [Detección de personas y objetos en un vídeo con Amazon Rekognition mediante un AWS SDK](#)
- [Detección de personas y objetos en un vídeo con Amazon Rekognition mediante un SDK de AWS](#)
- [Descargar los “directorios” de S3 desde un bucket de Amazon Simple Storage Service \(Amazon S3\)](#)
- [Descargar todos los objetos de un bucket de Amazon Simple Storage Service \(Amazon S3\) en un directorio local](#)

- [Descargar un flujo de tamaño desconocido en un objeto de Amazon S3 mediante un SDK de AWS](#)
- [Obtención de un objeto de Amazon S3 desde un punto de acceso de varias regiones con un SDK de AWS](#)
- [Obtenga un objeto de un bucket de Amazon S3 con un SDK de AWS al especificar un encabezado If-Modified-Since](#)
- [Introducción al cifrado de objetos de Amazon S3 con un SDK de AWS](#)
- [Introducción a etiquetas de objetos de Amazon S3 con un SDK de AWS](#)
- [Introducción a los clústeres de procesamiento de macrodatos](#)
- [Introducción a la administración de la configuración](#)
- [Introducción a la extracción de texto de documentos](#)
- [Introducción a los almacenes de características de machine learning](#)
- [Introducción al almacenamiento de objetos](#)
- [Introducción al análisis de consultas](#)
- [Trabajo con las características de bloqueo de objetos de Amazon S3 mediante un SDK de AWS](#)
- [Cómo realizar solicitudes condicionales de Amazon S3 mediante un SDK de AWS](#)
- [Administre listas de control de acceso \(ACL\) para buckets de Amazon S3 con un SDK de AWS](#)
- [Administración de mensajes grandes de Amazon SQS mediante Amazon S3 con un AWS SDK](#)
- [Administre objetos de Amazon S3 con control de versiones en lotes con una función de Lambda mediante un SDK de AWS](#)
- [Analizar los URI de Amazon S3 mediante un SDK de AWS](#)
- [Ejecución de una copia multiparte de un objeto de Amazon S3 con un SDK de AWS](#)
- [Recepción y procesamiento de notificaciones de eventos de Amazon S3 mediante un AWS SDK](#)
- [Guarde EXIF y otra información de la imagen con un SDK de AWS](#)
- [Envío de notificaciones de eventos de S3 a Amazon EventBridge mediante un AWS SDK](#)
- [Realización de un seguimiento de la carga o descarga de un objeto de Amazon S3 mediante un AWS SDK](#)
- [Transformación de datos para su aplicación con S3 Object Lambda](#)
- [Ejemplos de enfoques para pruebas unitarias y de integración con un SDK de AWS](#)
- [Cargar de forma recursiva un directorio local en un bucket de Amazon Simple Storage Service \(Amazon S3\)](#)
- [Cargar o descargar archivos de gran tamaño desde y hacia Amazon S3 con un SDK de AWS](#)

- [Carga de un flujo de tamaño desconocido en un objeto de Amazon S3 mediante un SDK de AWS](#)
- [Uso de sumas de comprobación para trabajar con un objeto de Amazon S3 con un SDK de AWS](#)
- [Trabajo con las características de integridad de objetos de Amazon S3 utilizando un AWS SDK](#)
- [Trabajo con objetos con control de versiones de Amazon S3 con un SDK de AWS](#)

Comprobación de la existencia de un bucket

El siguiente ejemplo de código muestra cómo comprobar si existe un bucket.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Puede usar el siguiente método `doesBucketExists` en sustitución del método [AmazonS3Client#doesBucketExistV2 \(String\)](#) del SDK para 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.getStatusCode() == HttpStatus.NOT_FOUND) {
        return false;
    }
    throw ase;
}
}
```

- Para obtener información sobre la API, consulte [GetBucketAcl](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Comprobación de si existe un objeto

El ejemplo de código siguiente muestra cómo comprobar si existe un objeto.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Puede usar el siguiente método `doesObjectExist` en sustitución del método [AmazonS3Client#doesObjectExist\(String, String\)](#) del SDK para 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;
        }
    }
}
```

- Para obtener información acerca de la API, consulte [HeadObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Convierta texto en voz y de nuevo a texto con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo:

- Utilice Amazon Polly para sintetizar un archivo de entrada de texto sin formato (UTF-8) en un archivo de audio.
- Cargue el archivo de audio en un bucket de Amazon S3.
- Utilice Amazon Transcribe para convertir el archivo de audio en texto.
- Muestre el texto.

Rust

SDK para Rust

Utilice Amazon Polly para sintetizar un archivo de entrada de texto sin formato (UTF-8) en un archivo de audio, cargue el archivo de audio en un bucket de Amazon S3, utilice Amazon Transcribe para convertir ese archivo de audio en texto y muestre el texto.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cree una URL prefirmada para Amazon S3 mediante un SDK de AWS

En los siguientes ejemplos de código, se muestra cómo crear una URL prefirmada para Amazon S3 y cargar un objeto.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Genere una URL prefirmada que pueda realizar una acción de Amazon S3 durante un tiempo limitado.

```
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;
}
}
```

Genere una URL prefirmada y realice una carga con esa 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 para .NET (v4)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear y utilizar direcciones URL POST previamente firmadas para cargarlas directamente desde el navegador.

```

/// <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 para C++

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Genere una URL prefirmada para descargar un objeto.

```

//! 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);
}

```

Descargue mediante 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;
}

```

Genere una URL prefirmada para cargar un objeto.

```

//! 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);
}

```

Cargue mediante 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

SDK para Go V2

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear funciones que encapsulen acciones prefirma de 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
}
```

Ejecute un ejemplo interactivo que genere y utilice URL prefirmadas para cargar, descargar y eliminar un objeto de 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 := httpReqester.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))
}

```

Defina un contenedor de solicitudes HTTP utilizado en el ejemplo para realizar solicitudes HTTP.

```

// IHttpReqester abstracts HTTP requests into an interface so it can be mocked
// during
// unit testing.
type IHttpReqester 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)
}

// HttpReqester uses the net/http package to make HTTP requests during the
// scenario.
type HttpReqester struct{}

func (httpReq HttpReqester) 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 para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

A continuación, se muestran tres ejemplos de cómo crear direcciones URL previamente firmadas y utilizarlas con bibliotecas cliente HTTP:

- Una solicitud HTTP GET que usa la URL con tres bibliotecas cliente HTTP
- Una solicitud HTTP PUT con metadatos en los encabezados que usa la URL con tres bibliotecas cliente HTTP
- Una solicitud HTTP PUT con parámetros de consulta que usa la URL con una biblioteca cliente HTTP

Genere una URL prefirmada para un objeto y, a continuación, descárguela (solicitud GET).

Importaciones.

```
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;
```

Genere la 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();
    }
}
```

Descargue el objeto mediante uno de los tres enfoques siguientes.

Utilice la clase JDK `URLConnection` (desde la versión 1.1) para realizar la descarga.

```
/* 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();
}
```

Utilice la clase JDK `HttpClient` (desde la versión 11) para realizar la descarga.

```
/* 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();
}
```

```
}
```

Utilice el SDK de AWS para la clase `SdkHttpClient` de Java para realizar la descarga.

```
/* 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();
}
```

Generar una URL previamente firmada con metadatos en los encabezados para una carga y, a continuación, cargar un archivo (solicitud PUT).

Importaciones.

```
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;
```


Genere la 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();
    }
}
```

Cargue un objeto de archivo mediante uno de los tres enfoques siguientes.

Utilice la clase JDK `URLConnection` (desde la versión 1.1) para realizar la carga.

```
/* 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);
    }
}

```

Utilice la clase JDK `HttpClient` (desde la versión 11) para realizar la carga.

```

/* 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);
}
}

```

Utilice la clase `SdkHttpClient` de AWS para Java V2 para realizar la carga.

```

/* 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);
    }
}

```

Generar una URL previamente firmada con parámetros de consulta para una carga y, a continuación, cargar un archivo (solicitud PUT).

Importaciones.

```

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;

```

Genere la 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();
    }
}

```

Utilice la clase `SdkHttpClient` de AWS para Java V2 para realizar la carga.

```

/**
 * 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 para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree una URL prefirmada para cargar un objeto en un bucket.

```
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;
    }
  }
}
```

```

    }
};

```

Cree una URL prefirmada para descargar un objeto de un bucket.

```

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;
  }
}
};
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK para JavaScript](#).

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree una solicitud prefirmada de `GetObject` y utilice la URL para descargar un objeto.

```
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
}
```

Cree una solicitud prefirmada `GetObject` con opciones avanzadas.

```
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
}

```

Cree una solicitud prefirmada de `PutObject` y úsela para subir un objeto.

```

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)
}
```

- Para obtener información, consulte [Guía para desarrolladores del SDK de AWS SDK para Kotlin](#).

PHP

SDK para PHP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Genere una URL prefirmada que pueda realizar una acción de S3 durante un tiempo limitado. Utilice el paquete Requests para realizar una solicitud con la 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()

```

Genere una solicitud POST prefirmada para cargar un archivo.

```
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

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
end
```

```
case response
when Net::HTTPSuccess
  puts 'Content uploaded!'
else
  puts response.value
end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree solicitudes de prefirma para objetos GET S3.

```
/// 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(())  
}
```

Cree solicitudes de prefirma para objetos PUT S3.

```
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

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear solicitudes previamente firmadas para obtener (GET) objetos de S3.

```

" 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( ).

```

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Creación de una aplicación de administración de activos fotográficos que permita a los usuarios administrar las fotos mediante etiquetas

En los siguientes ejemplos de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

.NET

SDK para .NET

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Una página web que indica los objetos de Amazon S3 que usan un SDK de AWS

En los siguientes ejemplos de código se muestra cómo obtener una lista de los objetos de Amazon S3 en una página web.

JavaScript

SDK para JavaScript (v3)

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

El código siguiente es el componente de React relevante que realiza llamadas al AWS SDK. Hay una versión ejecutable de la aplicación que contiene este componente en el enlace anterior de 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;
```

- Para obtener información sobre la API, consulte [ListObjects](#) en la Referencia de la API de AWS SDK para JavaScript.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Creación de una aplicación de exploración de Amazon Textract

Los siguientes ejemplos de código indican cómo explorar la salida de Amazon Textract mediante una aplicación interactiva.

JavaScript

SDK para JavaScript (v3)

Indica cómo utilizar AWS SDK para JavaScript para crear una aplicación React que utilice Amazon Textract para extraer datos de la imagen de un documento y presentarlos en una página web interactiva. Este ejemplo se ejecuta en un navegador web y requiere una identidad autenticada de Amazon Cognito para las credenciales. Para el almacenamiento utiliza Amazon Simple Storage Service (Amazon S3) y para las notificaciones consulta una cola de Amazon Simple Queue Service (Amazon SQS) que está suscrita a un tema de Amazon Simple Notification Service (Amazon SNS).

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK para Python (Boto3)

Indica cómo utilizar AWS SDK para Python (Boto3) con Amazon Textract para detectar elementos de texto, formularios y tablas en la imagen de un documento. La imagen de entrada y la salida de Amazon Textract aparecen en una aplicación Tkinter que permite explorar los elementos detectados.

- Envía la imagen de un documento a Amazon Textract y explora el resultado de los elementos detectados.
- Envía imágenes directamente a Amazon Textract o mediante un bucket de Amazon Simple Storage Service (Amazon S3).
- Utiliza las API asíncronas para iniciar un trabajo que publique una notificación en un tema de Amazon Simple Notification Service (Amazon SNS) cuando finalice el trabajo.
- Consulta una cola de Amazon Simple Queue Service (Amazon SQS) en busca de un mensaje de finalización de trabajo y muestra los resultados.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Para obtener una lista completa de las guías para desarrolladores del SDK de AWS y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Eliminar todos los objetos de un bucket de Amazon S3 dado mediante un SDK de AWS

En los siguientes ejemplos de código, se muestra cómo eliminar todos los objetos de un bucket de Amazon S3.

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Elimine todos los objetos de un bucket de Amazon S3 dado.

```
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);
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para JavaScript.
 - [DeleteObjects](#)
 - [ListObjectsV2](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Eliminación de las cargas multiparte incompletas a Amazon S3 mediante un AWS SDK

En el siguiente ejemplo de código, se muestra cómo eliminar o detener cargas multiparte incompletas de Amazon S3.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Para detener las cargas multiparte que estén en curso o incompletas por cualquier motivo, puede obtener una lista de las cargas y, a continuación, eliminarlas, tal y como se muestra en el siguiente ejemplo.

```
/**
 * 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);
        }
    }
}

```

Para eliminar las cargas multiparte incompletas que se hayan iniciado antes o después de una fecha, puede eliminar de forma selectiva las cargas multiparte en función de un momento dado, como se muestra en el siguiente ejemplo.

```

/**
 * 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 tiene acceso al ID de carga después de iniciar una carga multiparte, puede eliminar la carga en curso utilizando ese 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);
    }
}

```

Para eliminar de forma sistemática las cargas multiparte incompletas que tengan más de un número determinado de días, establezca una configuración de ciclo de vida de bucket para el bucket. En el siguiente ejemplo, se muestra cómo crear una regla para eliminar cargas incompletas que tienen más de 7 días.

```
/**
 * 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());
    }
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x.
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [PutBucketLifecycleConfiguration](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detección de EPI en imágenes con Amazon Rekognition mediante un AWS SDK

El siguiente ejemplo de código muestra cómo crear una aplicación que utiliza Amazon Rekognition para detectar equipos de protección individual (EPI) en imágenes.

Java

SDK para Java 2.x

Muestra cómo crear una función de AWS Lambda que detecte imágenes con equipos de protección individual.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detecte entidades en el texto extraído de una imagen con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar Amazon Comprehend para detectar entidades en el texto extraído por Amazon Textract Textact de una imagen almacenada en Amazon S3.

Python

SDK para Python (Boto3)

Muestra cómo utilizar AWS SDK para Python (Boto3) en un bloc de notas de Jupyter para detectar entidades del texto que se extrae de una imagen. En este ejemplo, se utiliza Amazon Textract para extraer texto de una imagen almacenada en Amazon Simple Storage Service (Amazon S3) y Amazon Comprehend para detectar entidades en el texto extraído.

Este ejemplo es un bloc de notas Jupyter y debe ejecutarse en un entorno que pueda alojar blocs de notas. Para obtener instrucciones sobre cómo ejecutar el ejemplo con Amazon SageMaker AI, consulte las instrucciones en [TextractAndComprehendNotebook.ipynb](#).

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Para obtener una lista completa de las guías para desarrolladores del SDK de AWS y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detecte rostros en una imagen con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo:

- Guarde una imagen en un bucket de Amazon S3.
- Utilice Amazon Rekognition para detectar información faciales, como el rango de edad, el género y las emociones (por ejemplo, una sonrisa).
- Muestre esos detalles.

Rust

SDK para Rust

Guarde la imagen en un bucket de Amazon S3 con el prefijo uploads, use Amazon Rekognition para detectar información faciales, como el rango de edad, el género y las emociones (por ejemplo, una sonrisa) y muestre esos detalles.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detección de personas y objetos en un vídeo con Amazon Rekognition mediante un AWS SDK

Los siguientes ejemplos de código muestran cómo crear una aplicación que utilice Amazon Rekognition para detectar objetos por categoría en imágenes.

.NET

SDK para .NET

Muestra cómo utilizar la API de .NET de Amazon Rekognition para crear una aplicación que utilice Amazon Rekognition para identificar objetos por categoría en imágenes ubicadas en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Rekognition

- Amazon S3
- Amazon SES

Java

SDK para Java 2.x

Muestra cómo utilizar la API de Java de Amazon Rekognition para crear una aplicación que utilice Amazon Rekognition para identificar objetos por categoría en imágenes ubicadas en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Muestra cómo utilizar Amazon Rekognition con AWS SDK para JavaScript para crear una aplicación que utilice Amazon Rekognition para identificar objetos por categoría en imágenes ubicadas en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Aprenda cómo:

- Crear un usuario no autenticado con Amazon Cognito.
- Analizar imágenes en busca de objetos con Amazon Rekognition.
- Verificar una dirección de correo electrónico de Amazon SES.
- Enviar una notificación por correo electrónico con Amazon SES.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK para Kotlin

Muestra cómo utilizar la API de Kotlin de Amazon Rekognition para crear una aplicación que utilice Amazon Rekognition para identificar objetos por categoría en imágenes ubicadas en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK para Python (Boto3)

Le muestra cómo utilizar AWS SDK para Python (Boto3) para crear una aplicación web que le permita hacer lo siguiente:

- Subir fotos en un bucket de Amazon Simple Storage Service (Amazon S3).
- Utilizar Amazon Rekognition para analizar y etiquetar las fotos.
- Utilice Amazon Simple Email Service (Amazon SES) para enviar informes de análisis de imágenes por correo electrónico.

Este ejemplo contiene dos componentes principales: una página web escrita en JavaScript creada con React y un servicio REST escrito en Python creado con Flask-RESTful.

Puede utilizar la página web de React para:

- Mostrar una lista de imágenes almacenadas en el bucket de S3.
- Subir imágenes desde la computadora en el bucket de S3.
- Mostrar imágenes y etiquetas que identifican los elementos detectados en la imagen.
- Obtener un informe de todas las imágenes del bucket de S3 y enviar un correo electrónico del informe.

La página web llama al servicio REST. El servicio envía solicitudes a AWS para llevar a cabo las siguientes acciones:

- Obtener y filtrar la lista de imágenes del bucket de S3.
- Subir fotos en el bucket de S3.
- Utilizar Amazon Rekognition para analizar fotos individuales y obtener una lista de etiquetas que identifican los elementos detectados en la foto.
- Analizar todas las fotos del bucket de S3 y usar Amazon SES para enviar un informe por correo electrónico.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo comenzar a utilizar el SDK y detalles sobre sus versiones anteriores.

Detección de personas y objetos en un vídeo con Amazon Rekognition mediante un SDK de AWS

Los siguientes ejemplos de código indican cómo detectar personas y objetos en un video con Amazon Rekognition.

Java

SDK para Java 2.x

Muestra cómo utilizar la API Java de Amazon Rekognition para crear una aplicación que detecte rostros y objetos en vídeos ubicados en un bucket de Amazon Simple Storage Service (Amazon S3). La aplicación envía al administrador una notificación por correo electrónico con los resultados mediante Amazon Simple Email Service (Amazon SES).

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3)

Utilice Amazon Rekognition para detectar caras, objetos y personas en videos iniciando trabajos de detección asíncronos. Este ejemplo también configura Amazon Rekognition para que notifique un tema de Amazon Simple Notification Service (Amazon SNS) cuando se finalicen los trabajos y suscriba una cola de Amazon Simple Queue Service (Amazon SQS) al tema. Cuando la cola recibe un mensaje sobre un trabajo, se recupera el trabajo y se muestran los resultados

Este ejemplo se puede ver mejor en GitHub. Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS

- Amazon SQS

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Descargar los “directorios” de S3 desde un bucket de Amazon Simple Storage Service (Amazon S3)

El siguiente ejemplo de código muestra cómo descargar y filtrar el contenido de los “directorios” del bucket de Amazon S3.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

En este ejemplo se muestra cómo utilizar el [S3TransferManager](#) AWS SDK for Java 2.x para descargar “directorios” de un bucket de Amazon S3. También muestra cómo usar [DownloadFilters](#) en la solicitud.

```
/**
 * 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;
}
```

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Descargar todos los objetos de un bucket de Amazon Simple Storage Service (Amazon S3) en un directorio local

El siguiente ejemplo de código muestra cómo descargar todos los objetos de un bucket de Amazon Simple Storage Service (Amazon S3) en un directorio local.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Utilice un [S3TransferManager](#) para [descargar todos los objetos de S3](#) en el mismo bucket de S3. Vea el [archivo completo](#) y [pruébelo](#).

```
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();
    }
```

- Para obtener información acerca de la API, consulte [DownloadDirectory](#) en la ReferenciaAWS SDK for Java 2.x de la API de.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Descargar un flujo de tamaño desconocido en un objeto de Amazon S3 mediante un SDK de AWS

Los siguientes ejemplos de código muestran cómo descargar un flujo de tamaño desconocido en un objeto de Amazon S3.

Swift

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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).")
}
```

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Obtención de un objeto de Amazon S3 desde un punto de acceso de varias regiones con un SDK de AWS

En el siguiente ejemplo de código se muestra cómo obtener un objeto desde un punto de acceso de varias regiones.

Kotlin

SDK para Kotlin

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Configure el cliente S3 para que utilice el algoritmo de firma asimétrica 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
}
```

Utilice el ARN del punto de acceso de varias regiones en lugar del nombre de un bucket para obtener el objeto.

```
suspend fun getObjectFromMrap(
    s3: S3Client,
    mrapArn: 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
}
```

- Para obtener información, consulte [Guía para desarrolladores del SDK de AWS SDK para Kotlin](#).
- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de SDK de AWS para Kotlin.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Obtenga un objeto de un bucket de Amazon S3 con un SDK de AWS al especificar un encabezado If-Modified-Since

En el siguiente ejemplo de código se muestra cómo leer datos de un objeto en un bucket de S3, pero solo si ese bucket no se ha modificado desde la última recuperación.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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(())
}
```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK para Rust.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción al cifrado de objetos de Amazon S3 con un SDK de AWS

El siguiente ejemplo de código muestra cómo empezar a cifrar objetos de Amazon S3.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.USWest2.
        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);
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para .NET.
 - [CopyObject](#)
 - [GetObject](#)
 - [GetObjectMetadata](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción a etiquetas de objetos de Amazon S3 con un SDK de AWS

El ejemplo de código siguiente muestra cómo empezar a usar etiquetas para objetos de Amazon S3.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.USSouth1;

        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}'");
    }
}
```

- Para obtener información sobre la API, consulte [GetObjectTagging](#) en la Referencia de la API de AWS SDK para .NET.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción a los clústeres de procesamiento de macrodatos

En el siguiente ejemplo de código, se muestra cómo:

- Creación de un par de claves de EC2
- Configuración del almacenamiento y preparación de la aplicación
- Eliminar recursos

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Encuentre el ejemplo completo y obtenga información sobre cómo configurarlo y ejecutarlo en el repositorio de [Tutoriales para desarrolladores de ejemplo](#).

```
#!/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."
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de comandos de AWS CLI.
 - [AddSteps](#)
 - [Cp](#)
 - [CreateCluster](#)
 - [CreateDefaultRoles](#)
 - [CreateKeyPair](#)
 - [DescribeCluster](#)
 - [DescribeKeyPairs](#)
 - [DescribeStep](#)
 - [Ls](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)
 - [Ssh](#)
 - [TerminateClusters](#)
 - [Wait](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción a la administración de la configuración

En el siguiente ejemplo de código, se muestra cómo:

- Crear un bucket de Amazon S3
- Cree un tema de Amazon SNS.
- Crear un rol de IAM para Config
- Configurar el registrador de configuración de Config

- Iniciar el registrador de configuración
- Verificar la configuración de Config

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Encuentre el ejemplo completo y obtenga información sobre cómo configurarlo y ejecutarlo en el repositorio de [Tutoriales para desarrolladores de ejemplo](#).

```
#!/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..."
        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
    fi
}

```

```

        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
fi
```

```

    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=$(aws configservice describe-configuration-recorders)
check_command "$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!"
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de comandos de AWS CLI.
 - [AttachRolePolicy](#)
 - [CreateBucket](#)
 - [CreateRole](#)
 - [CreateTopic](#)
 - [DeleteBucket](#)
 - [DeleteConfigurationRecorder](#)
 - [DeleteDeliveryChannel](#)
 - [DeleteRole](#)
 - [DeleteRolePolicy](#)
 - [DeleteTopic](#)
 - [DescribeConfigurationRecorderStatus](#)
 - [DescribeConfigurationRecorders](#)
 - [DescribeDeliveryChannels](#)
 - [DetachRolePolicy](#)
 - [GetCallerIdentity](#)
 - [PutConfigurationRecorder](#)
 - [PutDeliveryChannel](#)
 - [PutPublicAccessBlock](#)
 - [PutRolePolicy](#)
 - [Rm](#)
 - [StartConfigurationRecorder](#)
 - [StopConfigurationRecorder](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción a la extracción de texto de documentos

En el siguiente ejemplo de código, se muestra cómo:

- Cree un bucket de S3
- Carga de un documento en S3
- Eliminar recursos

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Encuentre el ejemplo completo y obtenga información sobre cómo configurarlo y ejecutarlo en el repositorio de [Tutoriales para desarrolladores de ejemplo](#).

```
#!/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
}

# 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: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAyAAAFfcSjAAAAADULEQVR42mNk
+M9QDwAdhgGAWjR9awAAAABJRUS5ErkJggg==" | base64 -d > "$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***/g'
    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
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de comandos de AWS CLI.
 - [AnalyzeDocument](#)
 - [Cp](#)
 - [Ayuda](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción a los almacenes de características de machine learning

En el siguiente ejemplo de código, se muestra cómo:

- Configurar permisos de IAM
- Creación de un rol de ejecución de SageMaker

- Creación de un grupo de características
- Eliminar recursos

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Encuentre el ejemplo completo y obtenga información sobre cómo configurarlo y ejecutarlo en el repositorio de [Tutoriales para desarrolladores de ejemplo](#).

```
#!/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"

    # 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
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

    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
    fi

```

```

    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)"

```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de comandos de AWS CLI.
 - [AttachRolePolicy](#)
 - [CreateBucket](#)
 - [CreateFeatureGroup](#)
 - [CreateRole](#)
 - [DeleteBucket](#)
 - [DeleteFeatureGroup](#)

- [DeleteRole](#)
- [DescribeFeatureGroup](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [GetRole](#)
- [ListFeatureGroups](#)
- [PutPublicAccessBlock](#)
- [Rm](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción al almacenamiento de objetos

En el siguiente ejemplo de código, se muestra cómo:

- Creación del primer bucket de S3
- Cargar un objeto
- Habilitación del control de versiones
- Configuración del cifrado predeterminado
- Adición de etiquetas al bucket
- Muestra de objetos y versiones
- Eliminar recursos

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Encuentre el ejemplo completo y obtenga información sobre cómo configurarlo y ejecutarlo en el repositorio de [Tutoriales para desarrolladores de ejemplo](#).

```
#!/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
    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 "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"
            },
            {
                "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."
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de comandos de AWS CLI.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [HeadObject](#)
 - [ListObjectVersions](#)
 - [ListObjectsV2](#)
 - [PutBucketEncryption](#)
 - [PutBucketTagging](#)
 - [PutBucketVersioning](#)
 - [PutObject](#)
 - [PutPublicAccessBlock](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Introducción al análisis de consultas

En el siguiente ejemplo de código, se muestra cómo:

- Creación de un bucket de S3 para los resultados de las consultas
- Crear una base de datos
- Crear una tabla
- Ejecución de una consulta
- Creación y uso de consultas con nombre
- Eliminar recursos

Bash

AWS CLI con Bash script

Note

Hay más en GitHub. Encuentre el ejemplo completo y obtenga información sobre cómo configurarlo y ejecutarlo en el repositorio de [Tutoriales para desarrolladores de ejemplo](#).

```
#!/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_identifier "$DATABASE_NAME"; then
        handle_error "Invalid database name: $DATABASE_NAME"
    fi

    if ! validate_identifier "$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"

# Step 2: Create a table
echo "Step 2: Creating Athena table: $TABLE_NAME"
# Replace the region placeholder in the S3 location
CREATE_TABLE_QUERY="CREATE EXTERNAL TABLE IF NOT EXISTS $DATABASE_NAME.
$TABLE_NAME (
    \`Date\` DATE,
    Time STRING,
    Location STRING,
    Bytes INT,
    RequestIP STRING,
    Method STRING,
    Host STRING,
    Uri STRING,
    Status INT,
    Referrer STRING,
    os STRING,
    Browser STRING,

```


[illegible]

```
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
done

```

```

    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
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
                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)
    fi

```



```
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!"
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de comandos de AWS CLI.
 - [Cp](#)
 - [CreateNamedQuery](#)
 - [DeleteNamedQuery](#)
 - [GetNamedQuery](#)
 - [GetQueryExecution](#)
 - [GetQueryResults](#)
 - [ListDatabases](#)
 - [ListNamedQueries](#)
 - [ListTableMetadata](#)
 - [Mb](#)
 - [Rb](#)

- [Rm](#)
- [StartQueryExecution](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Trabajo con las características de bloqueo de objetos de Amazon S3 mediante un SDK de AWS

El siguiente ejemplo de código muestra cómo trabajar con características de bloqueo de objetos de S3.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute un escenario interactivo en el que se demuestren las características de bloqueo de objetos de 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 =
                        $"Would 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;
    }
}
```

Una clase contenedora para funciones de 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($"\\tAdded an object lock policy to bucket
{bucketName}."");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"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($"{objectKey} in
{bucketName}: " +
            $"{response.LegalHold.Status}");
        return response.LegalHold;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable 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($"Delete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to delete bucket {bucketName}: " +
ex.Message);
        return false;
    }
}

```

```
}  
  
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para .NET.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Go

SDK para Go V2

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute un escenario interactivo en el que se demuestren las características de bloqueo de objetos de 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 (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))
}
```

Defina una estructura que envuelva las acciones de S3 utilizadas en este ejemplo.

```
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 _, 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
}

```

Eliminación de recursos.

```

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)
    }
}
del0bjects := 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)
}
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute un escenario interactivo en el que se demuestren las características de bloqueo de objetos de 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;
    }
}
```

Una clase contenedora para funciones de 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("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;
        }
    }
}
```

```
    }  
}  
  
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().getObjectLockEnabled());  
    System.out.println("\tRule: "+  
response.getObjectLockConfiguration().rule().defaultRetention());  
}  
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)

- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Punto de entrada para el escenario (index.js). Esto organiza todos los pasos. Visite GitHub para ver los detalles de implementación de Scenario, ScenarioInput, ScenarioOutput y 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} escenarios  
 * @param {Record<string, any>} initialState  
 */  
export const getWorkflowStages = (escenarios, initialState = {}) => {  
  const client = new S3Client({});  
  
  return {  
    deploy: new escenarios.Scenario(  
      "S3 Object Locking - Deploy",  
      [  
        welcome(escenarios),  
        welcomeContinue(escenarios),  
        exitOnFalse(escenarios, "welcomeContinue"),  
        getBucketPrefix(escenarios),  
        createBuckets(escenarios),  
        confirmCreateBuckets(escenarios),  
        exitOnFalse(escenarios, "confirmCreateBuckets"),  
        createBucketsAction(escenarios, client),  
        updateRetention(escenarios),  
        confirmUpdateRetention(escenarios),  
        exitOnFalse(escenarios, "confirmUpdateRetention"),  
        updateRetentionAction(escenarios, client),  
        populateBuckets(escenarios),  
        confirmPopulateBuckets(escenarios),  
        exitOnFalse(escenarios, "confirmPopulateBuckets"),  
        populateBucketsAction(escenarios, client),  
        updateLockPolicy(escenarios),  
        confirmUpdateLockPolicy(escenarios),  
        exitOnFalse(escenarios, "confirmUpdateLockPolicy"),  
        updateLockPolicyAction(escenarios, client),  
        confirmSetLegalHoldFileEnabled(escenarios),  
        setLegalHoldFileEnabledAction(escenarios, client),  
        confirmSetRetentionPeriodFileEnabled(escenarios),  
        setRetentionPeriodFileEnabledAction(escenarios, client),  
        confirmSetLegalHoldFileRetention(escenarios),
```

```

        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]",
    });
}

```

Enviar mensajes de bienvenida a la consola (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 };
```

Implementar la configuración de buckets, objetos y archivos (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,
};
```

Ver y eliminar archivos en los buckets (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 escenarios.ScenarioAction(
  "replAction",
  async (state) => {
    const files = await getAllFiles(client, [
      state.noLockBucketName,
      state.lockEnabledBucketName,
      state.retentionBucketName,
    ]);

    const fileInput = new escenarios.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 };

```

Destruir todos los recursos creados (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 };
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para JavaScript.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cómo realizar solicitudes condicionales de Amazon S3 mediante un SDK de AWS

Los siguientes ejemplos de código muestran cómo añadir condiciones previas a las solicitudes de Amazon S3.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecutar un escenario interactivo en el que se muestran las características de solicitud condicional de 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;
    }
}

```

Una clase contenedora para funciones de 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($"{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($"Delete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {

```

```
        Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
ex.Message);
        return false;
    }

}

}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para .NET.
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

JavaScript

SDK para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Punto de entrada para el flujo de trabajo (index.js). Esto organiza todos los pasos. Visite GitHub para ver los detalles de implementación de Scenario, ScenarioInput, ScenarioOutput y 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} escenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (escenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new escenarios.Scenario(
      "S3 Conditional Requests - Deploy",
      [
        welcome(escenarios),
        welcomeContinue(escenarios),
        exitOnFalse(escenarios, "welcomeContinue"),
        getBucketPrefix(escenarios),
        createBuckets(escenarios),
        confirmCreateBuckets(escenarios),
        exitOnFalse(escenarios, "confirmCreateBuckets"),
        createBucketsAction(escenarios, client),
        populateBuckets(escenarios),
        confirmPopulateBuckets(escenarios),
        exitOnFalse(escenarios, "confirmPopulateBuckets"),
        populateBucketsAction(escenarios, client),
        saveState,
      ],
      initialState,
    ),
    demo: new escenarios.Scenario(
      "S3 Conditional Requests - Demo",
      [loadState, welcome(escenarios), replAction(escenarios, client)],
      initialState,
    ),
    clean: new escenarios.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]",
    });
}

```

Enviar mensajes de bienvenida a la consola (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 };

```

Implementar buckets y objetos (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,  
};
```

Obtener, copiar y colocar objetos mediante solicitudes condicionales de 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 };

```

Destruir todos los recursos creados (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 };
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para JavaScript.
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecutar un escenario interactivo en el que se muestran las solicitudes condicionales de 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,
conditions_types[condition_type], etag
                )
            elif condition_type == 2 or condition_type == 3:
                self.conditional_requests.get_object_conditional(
                    object_key,
                    source_bucket,
                    conditions_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_conditions_types[condition_type],
                    etag,
                )
            elif condition_type == 2 or condition_type == 3:
                self.conditional_requests.copy_object_conditional(
                    object_key,
                    dest_key,
                    copy_conditions_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_scenari(source_bucket_name, dest_bucket_name,
object_key)
        self.display_menu(source_bucket_name, dest_bucket_name, object_key,
etag)
        finally:
            self.cleanup_scenari(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_scenari()

```

Una clase contenedora que define las operaciones de solicitud condicional.

```

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

```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Python (Boto3).
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Administre listas de control de acceso (ACL) para buckets de Amazon S3 con un SDK de AWS

En los siguientes ejemplos de código, se muestra cómo administrar listas de control de acceso (ACL) para buckets de Amazon S3.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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>
    /// <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
S3
    /// 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
param>
    /// 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 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,
});
}

}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para .NET.
 - [GetBucketAcl](#)
 - [GetObjectAcl](#)
 - [PutBucketAcl](#)
 - [PutObjectAcl](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Administración de mensajes grandes de Amazon SQS mediante Amazon S3 con un AWS SDK

En el siguiente ejemplo de código, se muestra cómo utilizar la biblioteca de clientes ampliada de Amazon SQS para trabajar con mensajes grandes de Amazon SQS.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
    }
}
```

- Para obtener más información, consulte la [Guía para desarrolladores de AWS SDK for Java 2.x](#).
- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x.
 - [CreateBucket](#)
 - [PutBucketLifecycleConfiguration](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Administre objetos de Amazon S3 con control de versiones en lotes con una función de Lambda mediante un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo administrar objetos de S3 con control de versiones en lotes con una función de Lambda.

Python

SDK para Python (Boto3)

Muestra cómo manipular los objetos con control de versiones de Amazon Simple Storage Service (Amazon S3) en lotes mediante la creación de trabajos que llaman a funciones AWS Lambda para llevar a cabo el procesamiento. En este ejemplo se crea un bucket con control de versiones, se cargan las estrofas del poema *You Are Old, Father William* de Lewis Carroll y se utilizan trabajos por lotes de Amazon S3 para cambiar el poema de varias formas.

Aprenda cómo:

- Crear funciones Lambda que funcionen en objetos con control de versiones.
- Crear un manifiesto de objetos para actualizar.
- Crear trabajos por lotes que invoquen funciones Lambda para actualizar objetos.
- Eliminar funciones Lambda.
- Vaciar y eliminar un bucket con control de versiones.

Este ejemplo se puede ver mejor en GitHub. Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Amazon S3

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Analizar los URI de Amazon S3 mediante un SDK de AWS

En el siguiente ejemplo se muestra cómo analizar los URI de Amazon S3 para extraer componentes importantes como el nombre del bucket y la clave de objeto.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Analice un URI de Amazon S3 mediante la clase [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);  
    }  
}
```

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Ejecución de una copia multiparte de un objeto de Amazon S3 con un SDK de AWS

El siguiente ejemplo de código muestra cómo realizar una copia multiparte de un objeto de Amazon S3.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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");  
        }  
    }  
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para .NET.
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [GetObjectMetadata](#)
 - [UploadPartCopy](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Recepción y procesamiento de notificaciones de eventos de Amazon S3 mediante un AWS SDK

El siguiente ejemplo de código muestra cómo trabajar con notificaciones de eventos de S3 de una forma orientada a objetos.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Este ejemplo muestra cómo procesar un evento de notificación de S3 utilizando 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);
}
}

```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x.
 - [DeleteMessageBatch](#)
 - [GetQueueAttributes](#)
 - [PutBucketNotificationConfiguration](#)
 - [ReceiveMessage](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Guarde EXIF y otra información de la imagen con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo:

- Obtener información EXIF de un archivo JPG, JPEG o PNG.
- Subir el archivo de imagen en un bucket de Amazon S3.
- Usar Amazon Rekognition para identificar los tres atributos principales (etiquetas) en el archivo.
- Agregar la información EXIF y de etiquetas a una tabla de Amazon DynamoDB de la región.

Rust

SDK para Rust

Obtenga información EXIF de un archivo JPG, JPEG o PNG, cargue el archivo de imagen en un bucket de Amazon S3, utilice Amazon Rekognition para identificar los tres atributos principales (etiquetas de Amazon Rekognition) en el archivo y añada la información EXIF y de etiquetas a una tabla de Amazon DynamoDB de la región.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Envío de notificaciones de eventos de S3 a Amazon EventBridge mediante un AWS SDK

El siguiente ejemplo de código muestra cómo habilitar un bucket para enviar notificaciones de eventos de S3 a EventBridge y enrutar las notificaciones a un tema de Amazon SNS y a una cola de Amazon SQS.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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"]
                    }
                }
            }
        """).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;
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x.
 - [PutBucketNotificationConfiguration](#)
 - [PutRule](#)
 - [PutTargets](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Realización de un seguimiento de la carga o descarga de un objeto de Amazon S3 mediante un AWS SDK

En el siguiente ejemplo de código se muestra cómo realizar un seguimiento de cargas o descargas de objetos de Amazon S3.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Realice el seguimiento del progreso de una carga de archivos.

```
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!

```

```
*/
```

```
}
```

Realice el seguimiento del progreso de una descarga de archivos.

```
public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                        String key, String downloadedFileWithPath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .destination(Paths.get(downloadedFileWithPath))
        .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!  
  
    */  
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x.
 - [GetObject](#)
 - [PutObject](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Transformación de datos para su aplicación con S3 Object Lambda

En el siguiente ejemplo de código se muestra cómo transformar datos para su aplicación con S3 Object Lambda.

.NET

SDK para .NET

Muestra cómo agregar código personalizado a las solicitudes GET S3 estándar para modificar el objeto solicitado recuperado de S3, de modo que el objeto se ajuste a las necesidades del cliente o aplicación solicitante.

Para ver el código fuente completo y las instrucciones de configuración y ejecución, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Lambda
- Amazon S3

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Ejemplos de enfoques para pruebas unitarias y de integración con un SDK de AWS

El siguiente ejemplo de código muestra ejemplos de técnicas de prácticas recomendadas a la hora de escribir pruebas unitarias y de integración mediante un SDK de AWS.

Rust

SDK para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cargo.toml para ver ejemplos de pruebas.

```
[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"
```

Ejemplo de pruebas unitarias con automock y un encapsulador de servicios.

```
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);
}
}

```

Ejemplo de pruebas de integración con 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(&[]);
}
}

```

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cargar de forma recursiva un directorio local en un bucket de Amazon Simple Storage Service (Amazon S3)

El siguiente ejemplo de código muestra cómo cargar de forma recursiva un directorio local en un bucket de Amazon Simple Storage Service (Amazon S3).

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Utilice un [S3TransferManager](#) para [cargar un directorio local](#). Vea el [archivo completo](#) y [pruébelo](#).

```
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();
    }
```


- Para obtener información acerca de la API, consulte [UploadDirectory](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Cargar o descargar archivos de gran tamaño desde y hacia Amazon S3 con un SDK de AWS

En los siguientes ejemplos de código se muestra cómo cargar o descargar archivos grandes hacia y desde Amazon S3.

Para obtener información, consulte [Carga de un objeto con carga multiparte](#).

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Llame a funciones que transfieren archivos hacia y desde un bucket de S3 mediante 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\n\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($"{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);
}

```

Cargar un solo archivo.

```

    /// <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;
    }
}

```

Cargar un directorio local completo.

```

/// <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;
    }
}
```

Descargar un solo archivo.

```
/// <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}"));
    }

```

Descargar el contenido de un bucket de 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;
    }
}
```

Realice un seguimiento del progreso de una carga mediante la 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.USWest2 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}");  
    }  
}
```

Cargar un objeto con cifrado.

```
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

SDK para Go V2

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear funciones que utilicen administradores de carga y descarga para dividir los datos en partes y transferirlos simultáneamente.

```

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
}

```

```
}
```

Ejecutar un escenario interactivo en el que se muestra cómo utilizar los administradores de carga y descarga en contexto.

```
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 para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Llame a funciones que transfieran archivos hacia y desde un bucket de S3 mediante 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();
}
```

Cargar un directorio local completo.

```
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();
    }
}
```

Cargar un solo archivo.

```
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();
}
```

En los ejemplos de código se utilizan las siguientes importaciones.

```
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;
```

Utilice el [Gestor de transferencias de Amazon S3](#) situado sobre el [cliente S3 basado en CRT de AWS](#) para realizar de forma transparente una carga multiparte cuando el tamaño del contenido supere un umbral. El umbral de tamaño predeterminado es 8 MB.

```
/**
 * 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();
}
```

Utilice la [API S3Client](#) para realizar una carga multiparte.

```
/**
```



```
* 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()));
}

```

Utilice la [API S3AsyncClient](#) con soporte multipart habilitado para realizar una carga multiparte.

```

/**
 * 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 para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cargue un archivo grande.

```
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;
  }
}
};

```

Descargue un archivo grande.

```

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

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree funciones que transfieran archivos utilizando varios de los ajustes del administrador de transferencias disponibles. Utilice una clase de devolución de llamada para escribir el progreso de la devolución de llamada durante la transferencia de archivos.

```
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

```

Demuestre las funciones del administrador de transferencias e informe de los resultados.

```

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 para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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(()  
}
```

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Carga de un flujo de tamaño desconocido en un objeto de Amazon S3 mediante un SDK de AWS

Los siguientes ejemplos de código muestran cómo cargar un flujo de tamaño desconocido en un objeto de Amazon S3.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Utilice el [Cliente S3 basado en CRT de AWS](#).

```
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;
    }
}

```

Utilice el [cliente de S3 asincrónico estándar con carga multiparte habilitada](#).

```

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;
    }
}
```

Utilice el [Gestor de transferencias de 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

SDK para Swift

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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).")
}
```

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de sumas de comprobación para trabajar con un objeto de Amazon S3 con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar sumas de comprobación para trabajar con un objeto de Amazon S3.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

En los ejemplos de código se utiliza un subconjunto de las siguientes importaciones.

```
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;
```

Especifique un algoritmo de suma de comprobación para el método `putObject` al [crear la `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"));
}
```

Compruebe la suma de comprobación del método `getObject` cuando [cree la `GetObjectRequest`](#).

```
public GetObjectResponse getObjectWithChecksum(String bucketName, String key)
{
    return s3Client.getObject(b -> b
```

```
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
    .response();
}
```

Calcule previamente una suma de comprobación para el método `putObject` cuando [cree la 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)));
}
```

Utilice el [Gestor de transferencias de Amazon S3](#) situado sobre el [cliente S3 basado en CRT de AWS](#) para realizar de forma transparente una carga multiparte cuando el tamaño del contenido supere un umbral. El umbral de tamaño predeterminado es 8 MB.

Puede especificar un algoritmo de suma de comprobación para que lo utilice el SDK. De forma predeterminada, el SDK usa el algoritmo 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();
}
```

```
}
```

Utilice la [API S3Client](#) o (la API S3AsyncClient) para realizar una carga multiparte. Si especifica una suma de comprobación adicional, debe especificar el algoritmo que se utilizará al iniciar la carga. También debe especificar el algoritmo para cada solicitud de parte y proporcionar la suma de comprobación calculada para cada parte una vez cargada.

```
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()));
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x.
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Trabajo con las características de integridad de objetos de Amazon S3 utilizando un AWS SDK

El siguiente ejemplo de código muestra cómo trabajar con las características de integridad de objetos de S3.

C++

SDK para C++

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute un escenario interactivo en el que se demuestren las características de integridad de objetos de 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;
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++.
 - [AbortMultipartUpload](#)
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [DeleteObject](#)
 - [GetObjectAttributes](#)
 - [PutObject](#)
 - [UploadPart](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Trabajo con objetos con control de versiones de Amazon S3 con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo:

- Crear un bucket de S3 con control de versiones.
- Obtener todas las versiones de un objeto.
- Revertir un objeto a una versión anterior.
- Eliminar y restaurar un objeto con control de versiones.
- Eliminar permanentemente todas las versiones de un objeto.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear funciones que encapsulen acciones de 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
```

Cargar la estrofa de un poema en un objeto con control de versiones y realizar una serie de acciones en él.

```
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!")
```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Python (Boto3).
 - [CreateBucket](#)
 - [DeleteObject](#)
 - [ListObjectVersions](#)
 - [PutBucketLifecycleConfiguration](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Ejemplos sin servidor de Amazon S3

Los siguientes ejemplos de código muestran cómo utilizar Amazon S3 con los SDK de AWS.

Ejemplos

- [Invocación de una función de Lambda desde un desencadenador de Amazon S3](#)

Invocación de una función de Lambda desde un desencadenador de Amazon S3

En los siguientes ejemplos de código, se muestra cómo implementar una función de Lambda que recibe un evento desencadenado al cargar un objeto en un bucket de S3. La función recupera el nombre del bucket de S3 y la clave del objeto del parámetro de evento y llama a la API de Amazon S3 para recuperar y registrar el tipo de contenido del objeto.

.NET

SDK para .NET

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de S3 con Lambda mediante .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

SDK para Go V2

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de S3 con Lambda mediante 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 para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de S3 con Lambda mediante Java.

```

// 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 para JavaScript (v3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de S3 con Lambda mediante 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);
  }
};
```


Uso de un evento de S3 con Lambda mediante 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

SDK para PHP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Consumo de un evento de S3 con Lambda mediante 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

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de S3 con Lambda mediante 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

SDK para Ruby

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Consumo de un evento de S3 con Lambda mediante 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 para Rust

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de S3 con Lambda mediante 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(())
}
```

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Ejemplos de código de Amazon S3 Control con los SDK de AWS

Los ejemplos de código siguientes muestran cómo utilizar Amazon S3 Control con un kit de desarrollo de software (SDK) de AWS.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Las acciones muestran cómo llamar a las funciones de servicio individuales y es posible verlas en contexto en los escenarios relacionados.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Amazon S3 Control

- [Ejemplos básicos de Amazon S3 Control con los SDK de AWS](#)
 - [Hello Amazon S3 Control](#)
 - [Funcionalidades básicas de Amazon S3 Control con un SDK de AWS](#)
 - [Acciones de Amazon S3 Control con los SDK de AWS](#)
 - [Uso de CreateJob con un AWS SDK o una CLI](#)
 - [Uso de DeleteJobTagging con un SDK de AWS](#)
 - [Uso de DescribeJob con un SDK de AWS o la CLI](#)
 - [Uso de GetJobTagging con un SDK de AWS](#)
 - [Uso de PutJobTagging con un SDK de AWS](#)
 - [Uso de UpdateJobPriority con un SDK de AWS o la CLI](#)
 - [Uso de UpdateJobStatus con un SDK de AWS o la CLI](#)

Ejemplos básicos de Amazon S3 Control con los SDK de AWS

Los siguientes ejemplos de código muestran cómo utilizar las funcionalidades básicas de Amazon S3 Control con SDK de AWS.

Ejemplos

- [Hello Amazon S3 Control](#)

- [Funcionalidades básicas de Amazon S3 Control con un SDK de AWS](#)
- [Acciones de Amazon S3 Control con los SDK de AWS](#)
 - [Uso de CreateJob con un AWS SDK o una CLI](#)
 - [Uso de DeleteJobTagging con un SDK de AWS](#)
 - [Uso de DescribeJob con un SDK de AWS o la CLI](#)
 - [Uso de GetJobTagging con un SDK de AWS](#)
 - [Uso de PutJobTagging con un SDK de AWS](#)
 - [Uso de UpdateJobPriority con un SDK de AWS o la CLI](#)
 - [Uso de UpdateJobStatus con un SDK de AWS o la CLI](#)

Hello Amazon S3 Control

En los siguientes ejemplos de código se muestra cómo empezar a utilizar Amazon S3 Control.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
        });
    }
}

```

- Para obtener información sobre la API, consulte [ListJobs](#) en la Referencia de la API de AWS SDK for Java 2.x.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [ListJobs](#) en la Referencia de la API del SDK de AWS SDK para Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Funcionalidades básicas de Amazon S3 Control con un SDK de AWS

Los ejemplos de código siguientes muestran cómo aprender a usar las operaciones principales de Amazon S3 Control.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Conocer las operaciones principales.

```
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.");
        }
    }
}
```

```

    }
}

}

```

Crear una clase de acción que contenga operaciones.

```

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();
}
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x.
 - [CreateJob](#)
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)
 - [UpdateJobStatus](#)

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtenga información sobre el escenario básico de lote de S3.

```
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
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para Python (Boto3).
 - [CreateJob](#)
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)

- [ListJobs](#)
- [PutJobTagging](#)
- [UpdateJobPriority](#)
- [UpdateJobStatus](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Acciones de Amazon S3 Control con los SDK de AWS

Los siguientes ejemplos de código muestran cómo realizar acciones individuales de Amazon S3 Control con los SDK de AWS. En cada ejemplo se incluye un enlace a GitHub, con instrucciones de configuración y ejecución del código.

Los siguientes ejemplos incluyen solo las acciones que se utilizan con mayor frecuencia. Para ver una lista completa, consulte la [Referencia de la API de Amazon S3 Control](#).

Ejemplos

- [Uso de CreateJob con un AWS SDK o una CLI](#)
- [Uso de DeleteJobTagging con un SDK de AWS](#)
- [Uso de DescribeJob con un SDK de AWS o la CLI](#)
- [Uso de GetJobTagging con un SDK de AWS](#)
- [Uso de PutJobTagging con un SDK de AWS](#)
- [Uso de UpdateJobPriority con un SDK de AWS o la CLI](#)
- [Uso de UpdateJobStatus con un SDK de AWS o la CLI](#)

Uso de **CreateJob** con un AWS SDK o una CLI

Los siguientes ejemplos de código muestran cómo utilizar CreateJob.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

CLI

AWS CLI

Para crear un trabajo de Operaciones por lotes de Amazon S

En el siguiente ejemplo de `create-job`, se crea un trabajo de operaciones por lotes de Amazon S3 para etiquetar objetos como `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
```

Salida:

```
{  
  "JobId": "93735294-df46-44d5-8638-6356f335324e"  
}
```

- Para obtener más información sobre la API, consulte [CreateJob](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree un trabajo de S3 asincrónico.

```
/**
 * 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;
    });
}

```

Crear un trabajo de retención de conformidad.

```

/**
 * 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();
}

```

Crear un trabajo para quitar la retención legal.

```

/**
 * 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();
}

```

Crear un nuevo trabajo de retención de gobernanza.

```

/**
 * 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();
}
}
```

- Para obtener información sobre la API, consulte [CreateJob](#) en la Referencia de la API de AWS SDK for Java 2.x.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [CreateJob](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.

```

- Para obtener información sobre la API, consulte [CreateJob](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteJobTagging** con un SDK de AWS

Los siguientes ejemplos de código muestran cómo utilizar DeleteJobTagging.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

Java

SDK para Java 2.x

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
        });
}
```

```
}
```

- Para obtener información sobre la API, consulte [DeleteJobTagging](#) en la Referencia de la API de AWS SDK for Java 2.x.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [DeleteJobTagging](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [DeleteJobTagging](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DescribeJob** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar `DescribeJob`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

CLI

AWS CLI

Para describir un trabajo de Operaciones por lotes de Amazon S

En el siguiente ejemplo de `describe-job`, se proporcionan los parámetros de configuración y el estado del trabajo de operaciones por lotes especificado.

```
aws s3control describe-job \  
  --account-id 123456789012 \  
  --job-id 93735294-df46-44d5-8638-6356f335324e
```

Salida:

```
{  
  "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"
}
```

- Para obtener información sobre la API, consulte [DescribeJob](#) en la Referencia de comandos de AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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);
    });
}

```

- Para obtener información sobre la API, consulte [DescribeJob](#) en la Referencia de la API de AWS SDK for Java 2.x.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

```

- Para obtener información sobre la API, consulte [DescribeJob](#) en Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.

```

- Para obtener información sobre la API, consulte [DescribeJob](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetJobTagging** con un SDK de AWS

Los siguientes ejemplos de código muestran cómo utilizar GetJobTagging.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

Java

SDK para Java 2.x

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
        });
}
```



```
    });  
}
```

- Para obtener información sobre la API, consulte [GetJobTagging](#) en la Referencia de la API de AWS SDK for Java 2.x.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [GetJobTagging](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.

```

- Para obtener información sobre la API, consulte [GetJobTagging](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutJobTagging** con un SDK de AWS

Los siguientes ejemplos de código muestran cómo utilizar PutJobTagging.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
            });
    }

```

- Para obtener información sobre la API, consulte [PutJobTagging](#) en la Referencia de la API de AWS SDK for Java 2.x.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

```

- Para obtener información sobre la API, consulte [PutJobTagging](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.

```

- Para obtener información sobre la API, consulte [PutJobTagging](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **UpdateJobPriority** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar UpdateJobPriority.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

CLI

AWS CLI

Para actualizar la prioridad de un trabajo de Operaciones por lotes de Amazon S

En el siguiente ejemplo de `update-job-priority`, se actualiza el trabajo especificado con una prioridad nueva.

```
aws s3control update-job-priority \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
  --priority 52
```

Salida:

```
{
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",
  "Priority": 52
}
```

- Para obtener información sobre la API, consulte [UpdateJobPriority](#) en la Referencia de comandos de la AWS CLI.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
}

```

- Para obtener información sobre la API, consulte [UpdateJobPriority](#) en la Referencia de la API de AWS SDK for Java 2.x.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

```

- Para obtener información sobre la API, consulte [UpdateJobPriority](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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_s3cnofoundexception 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.
```

- Para obtener información sobre la API, consulte [UpdateJobPriority](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **UpdateJobStatus** con un SDK de AWS o la CLI

Los siguientes ejemplos de código muestran cómo utilizar UpdateJobStatus.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

CLI

AWS CLI

Para actualizar el estado de un trabajo de Operaciones por lotes de Amazon S

En el siguiente ejemplo de update-job-status, se cancela el trabajo especificado que espera ser aprobado.

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
  --requested-job-status Cancelled
```

Salida:

```
{
  "Status": "Cancelled",
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"
}
```

En el siguiente ejemplo de `update-job-status`, se confirma y ejecuta el trabajo especificado que espera ser aprobado.

```
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"
}
```

En el siguiente ejemplo de `update-job-status`, se cancela el trabajo especificado que se está ejecutando.

```
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"
}
```

- Para obtener información sobre la API, consulte [UpdateJobStatus](#) en la Referencia de la API de la AWS CLI.

Java

SDK para Java 2.x

 Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
        });
}
```

- Para obtener información sobre la API, consulte [UpdateJobStatus](#) en la Referencia de la API de AWS SDK for Java 2.x.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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

```

- Para obtener información sobre la API, consulte [UpdateJobStatus](#) en la Referencia de la API de AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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.
```

- Para obtener información sobre la API, consulte [UpdateJobStatus](#) en la Referencia de la API de AWS SDK para SAP ABAP.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Ejemplos de código de S3 Directory Buckets con los SDK de AWS

Los ejemplos de código siguientes muestran cómo utilizar los buckets de directorio de S3 con un kit de desarrollo de software (SDK) de AWS.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

S3 Directory Buckets

- [Ejemplos básicos de S3 Directory Buckets con los SDK de AWS](#)
 - [Hello Amazon S3 directory buckets](#)

- [Funcionalidades básicas de S3 Directory Buckets con un SDK AWS](#)
- [Acciones de S3 Directory Buckets con los SDK de AWS](#)
 - [Uso de AbortMultipartUpload con un AWS SDK](#)
 - [Uso de CompleteMultipartUpload con un SDK de AWS](#)
 - [Uso de CopyObject con un SDK de AWS](#)
 - [Uso de CreateBucket con un SDK de AWS](#)
 - [Uso de CreateMultipartUpload con un SDK de AWS](#)
 - [Uso de CreateSession con un SDK de AWS](#)
 - [Uso de DeleteBucket con un SDK de AWS](#)
 - [Uso de DeleteBucketEncryption con un SDK de AWS](#)
 - [Uso de DeleteBucketPolicy con un SDK de AWS](#)
 - [Uso de DeleteObject con un SDK de AWS](#)
 - [Uso de DeleteObjects con un SDK de AWS](#)
 - [Uso de GetBucketEncryption con un SDK de AWS](#)
 - [Uso de GetBucketPolicy con un SDK de AWS](#)
 - [Uso de GetObject con un SDK de AWS](#)
 - [Uso de GetObjectAttributes con un SDK de AWS](#)
 - [Uso de HeadBucket con un SDK de AWS](#)
 - [Uso de HeadObject con un SDK de AWS](#)
 - [Uso de ListDirectoryBuckets con un SDK de AWS](#)
 - [Uso de ListMultipartUploads con un SDK de AWS](#)
 - [Uso de ListObjectsV2 con un SDK de AWS](#)
 - [Uso de ListParts con un SDK de AWS](#)
 - [Uso de PutBucketEncryption con un SDK de AWS](#)
 - [Uso de PutBucketPolicy con un SDK de AWS](#)
 - [Uso de PutObject con un SDK de AWS](#)
 - [Uso de UploadPart con un SDK de AWS](#)
 - [Uso de UploadPartCopy con un SDK de AWS](#)
- [Escenarios de S3 Directory Buckets con los SDK de AWS](#)

- [Crear una URL previamente firmada de Amazon S3 Directory Buckets para obtener un objeto con un SDK de AWS](#)

Ejemplos básicos de S3 Directory Buckets con los SDK de AWS

Los siguientes ejemplos de código muestran cómo usar las funcionalidades básicas de Amazon S3 Directory Buckets con los SDK de AWS.

Ejemplos

- [Hello Amazon S3 directory buckets](#)
- [Funcionalidades básicas de S3 Directory Buckets con un SDK AWS](#)
- [Acciones de S3 Directory Buckets con los SDK de AWS](#)
 - [Uso de AbortMultipartUpload con un AWS SDK](#)
 - [Uso de CompleteMultipartUpload con un SDK de AWS](#)
 - [Uso de CopyObject con un SDK de AWS](#)
 - [Uso de CreateBucket con un SDK de AWS](#)
 - [Uso de CreateMultipartUpload con un SDK de AWS](#)
 - [Uso de CreateSession con un SDK de AWS](#)
 - [Uso de DeleteBucket con un SDK de AWS](#)
 - [Uso de DeleteBucketEncryption con un SDK de AWS](#)
 - [Uso de DeleteBucketPolicy con un SDK de AWS](#)
 - [Uso de DeleteObject con un SDK de AWS](#)
 - [Uso de DeleteObjects con un SDK de AWS](#)
 - [Uso de GetBucketEncryption con un SDK de AWS](#)
 - [Uso de GetBucketPolicy con un SDK de AWS](#)
 - [Uso de GetObject con un SDK de AWS](#)
 - [Uso de GetObjectAttributes con un SDK de AWS](#)
 - [Uso de HeadBucket con un SDK de AWS](#)
 - [Uso de HeadObject con un SDK de AWS](#)
 - [Uso de ListDirectoryBuckets con un SDK de AWS](#)
 - [Uso de ListMultipartUploads con un SDK de AWS](#)
 - [Uso de ListObjectsV2 con un SDK de AWS](#)

- [Uso de ListParts con un SDK de AWS](#)
- [Uso de PutBucketEncryption con un SDK de AWS](#)
- [Uso de PutBucketPolicy con un SDK de AWS](#)
- [Uso de PutObject con un SDK de AWS](#)
- [Uso de UploadPart con un SDK de AWS](#)
- [Uso de UploadPartCopy con un SDK de AWS](#)

Hello Amazon S3 directory buckets

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon S3 Directory Buckets.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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;
    }
}
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK for Java 2.x.
 - [CreateBucket](#)
 - [ListDirectoryBuckets](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Funcionalidades básicas de S3 Directory Buckets con un SDK AWS

En el siguiente ejemplo de código, se muestra cómo:

- Configurar una VPC y un punto de conexión de VPC.
- Configurar las políticas, los roles y el usuario que va a trabajar con S3 Directory Buckets y la clase de almacenamiento S3 Express One Zone.
- Crear dos clientes de S3.
- Cree dos buckets.
- Crear un objeto y copiarlo.
- Mostrar la diferencia de rendimiento.
- Rellenar los buckets para mostrar la diferencia lexicográfica.
- Indicar al usuario si desea limpiar los recursos.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecutar un escenario interactivo en el que se muestran las características de 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 createSessionAddObjects() {
        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.");
        }
    }
}
}
}

```

Una clase contenedora para métodos del SDK de 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.waitUntilVpcAvailable(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);
    });
}
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK for Java 2.x.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

PHP

SDK para PHP

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecutar un escenario en el que se muestren las funcionalidades básicas de S3 Directory Buckets y 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 an 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;
        }
    }
}

```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para PHP.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)

- [ListObjects](#)
- [PutObject](#)

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecutar un escenario en el que se muestren las funcionalidades básicas de S3 Directory Buckets y 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"],
            )

```

Una clase contenedora para funciones del SDK de 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

```

- Para obtener información sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Python (Boto3).
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)

- [PutObject](#)

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Acciones de S3 Directory Buckets con los SDK de AWS

Los siguientes ejemplos de código muestran cómo realizar acciones individuales de S3 Directory Buckets con los SDK de AWS. En cada ejemplo se incluye un enlace a GitHub, con instrucciones de configuración y ejecución del código.

Estos fragmentos llaman a la API de S3 Directory Buckets y son fragmentos de código de programas más grandes que deben ejecutarse en contexto. Puede ver las acciones en contexto en [Escenarios de S3 Directory Buckets con los SDK de AWS](#).

Los siguientes ejemplos incluyen solo las acciones que se utilizan con mayor frecuencia. Para ver una lista completa, consulte la [Referencia de la API de Amazon S3 Directory Buckets](#).

Ejemplos

- [Uso de AbortMultipartUpload con un AWS SDK](#)
- [Uso de CompleteMultipartUpload con un SDK de AWS](#)
- [Uso de CopyObject con un SDK de AWS](#)
- [Uso de CreateBucket con un SDK de AWS](#)
- [Uso de CreateMultipartUpload con un SDK de AWS](#)
- [Uso de CreateSession con un SDK de AWS](#)
- [Uso de DeleteBucket con un SDK de AWS](#)
- [Uso de DeleteBucketEncryption con un SDK de AWS](#)
- [Uso de DeleteBucketPolicy con un SDK de AWS](#)
- [Uso de DeleteObject con un SDK de AWS](#)
- [Uso de DeleteObjects con un SDK de AWS](#)
- [Uso de GetBucketEncryption con un SDK de AWS](#)
- [Uso de GetBucketPolicy con un SDK de AWS](#)
- [Uso de GetObject con un SDK de AWS](#)
- [Uso de GetObjectAttributes con un SDK de AWS](#)

- [Uso de HeadBucket con un SDK de AWS](#)
- [Uso de HeadObject con un SDK de AWS](#)
- [Uso de ListDirectoryBuckets con un SDK de AWS](#)
- [Uso de ListMultipartUploads con un SDK de AWS](#)
- [Uso de ListObjectsV2 con un SDK de AWS](#)
- [Uso de ListParts con un SDK de AWS](#)
- [Uso de PutBucketEncryption con un SDK de AWS](#)
- [Uso de PutBucketPolicy con un SDK de AWS](#)
- [Uso de PutObject con un SDK de AWS](#)
- [Uso de UploadPart con un SDK de AWS](#)
- [Uso de UploadPartCopy con un SDK de AWS](#)

Uso de **AbortMultipartUpload** con un AWS SDK

En el siguiente ejemplo de código, se muestra cómo utilizar `AbortMultipartUpload`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cancelar la carga multiparte de un bucket de directorio.

```
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;
    }
}

```

- Para obtener información sobre la API, consulte [AbortMultipartUpload](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **CompleteMultipartUpload** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar CompleteMultipartUpload.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Completar una carga multiparte de un bucket de directorio.

```
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 =
CompleteMultipartUploadRequest.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;
    }
}

```

- Para obtener información sobre la API, consulte [CompleteMultipartUpload](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **CopyObject** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `CopyObject`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Copiar un objeto de un bucket de directorio a otro bucket de directorio.

```
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;
    }
}
```

- Para obtener información sobre la API, consulte [CopyObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **CreateBucket** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateBucket`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear un bucket de directorio de 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;
    }
}
```

- Para obtener información sobre la API, consulte [CreateBucket](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **CreateMultipartUpload** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateMultipartUpload`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear una carga multiparte de un bucket de directorio.

```
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;
    }
}

```

- Para obtener información acerca de la API, consulte [CreateMultipartUpload](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **CreateSession** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateSession`.

Python

SDK para Python (Boto3)

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de 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
```

- Para obtener información sobre la API, consulte [CreateSession](#) en la Referencia de la API de AWS SDK para Python (Boto3).

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteBucket** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteBucket`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Eliminar un bucket de directorio de 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;  
    }  
}
```

- Para obtener información sobre la API, consulte [DeleteBucket](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteBucketEncryption** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteBucketEncryption`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Eliminar la configuración de cifrado de un bucket de directorio.

```
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;
    }
}
```

- Para obtener información sobre la API, consulte [DeleteBucketEncryption](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteBucketPolicy** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteBucketPolicy.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Eliminar una política de bucket para un bucket de directorio.

```
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;
    }
}
```

- Para obtener información acerca de la API, consulte [DeleteBucketPolicy](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteObject** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteObject`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Eliminar un objeto en un bucket de directorio.

```
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;
    }
}
```

- Para obtener información sobre la API, consulte [DeleteObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **DeleteObjects** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteObjects`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Eliminar varios objetos en un bucket de directorio.

```
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;
    }
}
```

- Para obtener información sobre la API, consulte [DeleteObjects](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetBucketEncryption** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `GetBucketEncryption`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtener la configuración de cifrado de un bucket de directorio.

```
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;
    }
}
```

- Para obtener información sobre la API, consulte [GetBucketEncryption](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetBucketPolicy** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `GetBucketPolicy`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtener la política de un bucket de directorio.

```
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;
    }
}
```

- Para obtener información acerca de la API, consulte [GetBucketPolicy](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetObject** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObject`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en su contexto en los siguientes ejemplos de código:

- [Conceptos básicos](#)
- [Creación de una URL previamente firmada para obtener un objeto](#)

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtener un objeto de un bucket de directorio.

```
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;
    }
}
```

```
}
```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **GetObjectAttributes** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObjectAttributes`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtener los atributos de un objeto de un bucket de directorio.

```
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;
    }
}
```

- Para obtener información sobre la API, consulte [GetObjectAttributes](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **HeadBucket** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar HeadBucket.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Comprobar si el bucket de directorio de S3 especificado existe y es accesible.

```
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;
    }
}
```

- Para obtener información sobre la API, consulte [HeadBucket](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **HeadObject** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `HeadObject`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Obtener los metadatos de un objeto en un bucket de directorio.

```
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;
}
```

- Para obtener información acerca de la API, consulte [HeadObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **ListDirectoryBuckets** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `ListDirectoryBuckets`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Mostrar todos los buckets de directorio.

```
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;
}
}
```

- Para obtener información sobre la API, consulte [ListDirectoryBuckets](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **ListMultipartUploads** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar ListMultipartUploads.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Mostrar las cargas multiparte de un bucket de directorio.

```
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
}
}
```

- Para obtener información acerca de la API, consulte [ListMultipartUploads](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **ListObjectsV2** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `ListObjectsV2`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Mostrar los objetos de un bucket de directorio.

```
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;
    }
}
```

- Para obtener información sobre la API, consulte [ListObjectsV2](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **ListParts** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `ListParts`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Mostrar las partes de una carga multiparte de un bucket de directorio.

```
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
        }
    }
}

```

- Para obtener información sobre la API, consulte [ListParts](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketEncryption** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `PutBucketEncryption`.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Establecer el cifrado de bucket en un bucket de directorio.

```
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;
    }
}
```

- Para obtener información sobre la API, consulte [PutBucketEncryption](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutBucketPolicy** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar PutBucketPolicy.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Aplicar una política de bucket a un bucket de directorio.

```
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;
    }
}

```

```
}  
}
```

- Para obtener información sobre la API, consulte [PutBucketPolicy](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **PutObject** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar `PutObject`.

Los ejemplos de acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Puede ver esta acción en contexto en el siguiente ejemplo de código:

- [Conceptos básicos](#)

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Colocar un objeto en un bucket de directorio.

```
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;
    }
}
```

- Para obtener información sobre la API, consulte [PutObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **UploadPart** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar UploadPart.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cargar parte de una carga multiparte de un bucket de directorio.

```
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;
    }
```

- Para obtener información sobre la API, consulte [UploadPart](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Uso de **UploadPartCopy** con un SDK de AWS

En el siguiente ejemplo de código, se muestra cómo utilizar UploadPartCopy.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear partes de copia en función del tamaño del objeto de origen y copiar las partes individuales en un bucket de directorio.

```
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;  
}
```

- Para obtener información sobre la API, consulte [UploadPartCopy](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Escenarios de S3 Directory Buckets con los SDK de AWS

Los siguientes ejemplos de código muestran cómo implementar escenarios comunes de S3 Directory Buckets con los SDK de AWS. Estos escenarios muestran cómo realizar tareas específicas con llamadas a varias funciones dentro de S3 Directory Buckets o en combinación con otros Servicios de AWS. En cada escenario se incluye un enlace al código fuente completo, con instrucciones de configuración y ejecución del código.

Los escenarios requieren un nivel intermedio de experiencia para entender las acciones de servicio en su contexto.

Ejemplos

- [Crear una URL previamente firmada de Amazon S3 Directory Buckets para obtener un objeto con un SDK de AWS](#)

Crear una URL previamente firmada de Amazon S3 Directory Buckets para obtener un objeto con un SDK de AWS

El siguiente ejemplo de código muestra cómo crear una URL previamente firmada de S3 Directory Buckets y obtener un objeto.

Java

SDK para Java 2.x

Note

Hay más en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Generar una URL GET previamente firmada para acceder a un objeto de un bucket de directorio de 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.presignedGetObject(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;
    }
}
```

```
}
```

- Para obtener información sobre la API, consulte [GetObject](#) en la Referencia de la API de AWS SDK for Java 2.x.

Para obtener una lista completa de las guías para desarrolladores de AWS SDK y ejemplos de código, consulte [Desarrollo con Amazon S3 mediante los SDK de AWS](#). En este tema también se incluye información sobre cómo empezar a utilizar el SDK y detalles sobre sus versiones anteriores.

Autenticación de solicitudes (AWS Signature Version 4)

Temas

- [Métodos de autenticación](#)
- [Introducción a la firma de solicitudes](#)
- [Autenticación de solicitudes: uso del encabezado de Authorization \(AWS Signature Version 4\)](#)
- [Autenticación de solicitudes: uso de parámetros de consulta \(AWS Signature Version 4\)](#)
- [Ejemplos: cálculos de firma en AWS Signature Version 4](#)
- [Autenticación de solicitudes: cargas basadas en el navegador con POST \(AWS Signature Version 4\)](#)
- [Claves de política específica de autenticación de Signature Version 4 de Amazon S3](#)

Toda interacción con Amazon S3 es o autenticada o anónima. En esta sección, se explica la autenticación de solicitudes con el algoritmo de AWS Signature Version 4.

Note

Si utiliza los AWS SDK (consulte [Bibliotecas y código de ejemplo](#)) para enviar las solicitudes, no necesita leer esta sección porque los clientes del SDK autentican las solicitudes mediante las claves de acceso que proporciona. A menos que tenga una buena razón para no hacerlo, debería usar siempre los AWS SDK. En las regiones que admiten ambas versiones de firma, puede solicitar que los AWS SDK usen una versión de firma específica. Para obtener más información, consulte [Envío de solicitudes autenticadas mediante los SDK de AWS](#). Debe leer esta sección solo si está implementando el algoritmo de AWS Signature Version 4 en el cliente personalizado.

La autenticación con AWS Signature Version 4 proporciona algunos o todos los elementos siguientes, según la forma en que elija firmar la solicitud:

- Verificación de la identidad del solicitante: las solicitudes autenticadas requieren una firma que se crea con las claves de acceso (ID de clave de acceso, clave de acceso secreta). Para obtener información acerca de cómo obtener las claves de acceso, consulte [Descripción y obtención de](#)

[las credenciales de seguridad](#) en Referencia general de AWS. Si utiliza credenciales de seguridad temporales, los cálculos de la firma también requieren un token de seguridad. Para obtener más información, consulte [Solicitud de credenciales de seguridad temporales](#) en la Guía del usuario de IAM.

- Protección de datos en tránsito: para evitar la manipulación de una solicitud mientras está en tránsito, se utilizan algunos de los elementos de la solicitud para calcular la firma de la solicitud. Al recibir la solicitud, Amazon S3 calcula la firma con los mismos elementos de la solicitud. Si algún componente de la solicitud recibido por Amazon S3 no coincide con el componente que se utilizó para calcular la firma, Amazon S3 rechazará la solicitud.
- Protección frente a la reutilización de las partes firmadas de la solicitud: las partes firmadas (mediante AWS Signatures) de las solicitudes son válidas en un plazo de 15 minutos a partir de la marca temporal de la solicitud. Una parte no autorizada que tenga acceso a una solicitud firmada puede modificar las partes no firmadas de la solicitud sin que ello afecte a la validez de la solicitud en un plazo de 15 minutos. Por ello, recomendamos que se maximice la protección firmando los encabezados y el cuerpo de las solicitudes, realizando solicitudes HTTPS a Amazon S3 y utilizando la clave de condición `s3:x-amz-content-sha256` (consulte [Claves de política específica de autenticación de Signature Version 4 de Amazon S3](#)) en las políticas de AWS para exigir a los usuarios que firmen los cuerpos de las solicitudes de Amazon S3.

Note

Amazon S3 admite Signature Version 4, un protocolo para autenticar las solicitudes de la API entrantes para los servicios de AWS, en todas las Regiones de AWS. En este momento, las Regiones de AWS creadas antes del 30 de enero de 2014 seguirán admitiendo el protocolo anterior, Signature Version 2. Cualquier región nueva después del 30 de enero de 2014 solo admitirá Signature Version 4, y por lo tanto, todas las solicitudes de esas regiones se deben realizar con Signature Version 4. Para obtener más autorización sobre AWS Signature Version 2, consulte [Firma y autenticación de solicitudes REST](#) en la Guía del usuario de Amazon Simple Storage Service.

Métodos de autenticación

Puede expresar la información de autenticación mediante uno de los siguientes métodos:

- Encabezado de HTTP Authorization: el uso del encabezado de HTTP Authorization es el método más común para autenticar una solicitud de Amazon S3. Todas las operaciones REST de Amazon S3 (excepto las cargas basadas en el navegador mediante solicitudes POST) requieren este encabezado. Para obtener más información sobre el valor del encabezado Authorization y sobre cómo calcular la firma y las opciones relacionadas, consulte [Autenticación de solicitudes: uso del encabezado de Authorization \(AWS Signature Version 4\)](#).
- Parámetros de cadena de consulta: puede utilizar una cadena de consulta para expresar una solicitud en su totalidad en una URL. En este caso, utiliza los parámetros de consulta para proporcionar la información de la solicitud, incluida la información de autenticación. Dado que la firma de la solicitud forma parte de la dirección URL, este tipo de URL suele denominarse como URL prefirmada. Puede utilizar URL prefirmadas para incrustar enlaces en los que se puede hacer clic, que pueden tener una validez de hasta siete días, en HTML. Para obtener más información, consulte [Autenticación de solicitudes: uso de parámetros de consulta \(AWS Signature Version 4\)](#).

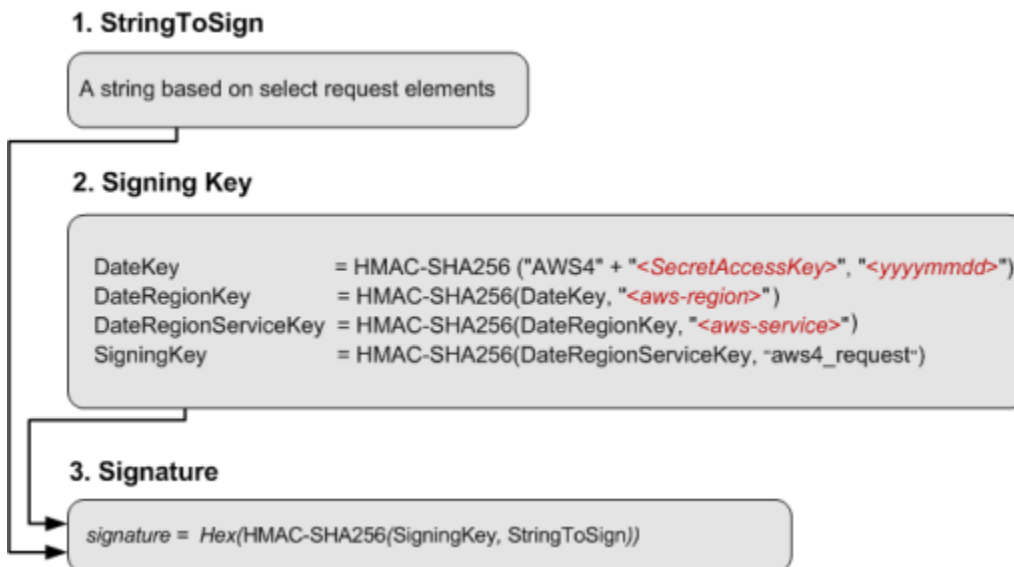
Amazon S3 también admite las cargas basadas en el navegador que utilizan solicitudes HTTP POST. Con una solicitud HTTP POST, puede cargar contenido en Amazon S3 directamente desde el navegador. Para obtener información acerca de la autenticación de solicitudes POST, consulte [Cargas basadas en el navegador con POST \(AWS Signature Version 4\)](#).

Introducción a la firma de solicitudes

La información de autenticación que envíe en una solicitud debe incluir una firma. Para calcular una firma, primero debe concatenar los elementos de la solicitud Select para formar una cadena, denominada cadena para firmar. A continuación, debe utilizar una clave de firma para calcular el código de autenticación de mensajes basado en hash (HMAC) de la cadena para firmar.

En AWS Signature Version 4, no usa su clave de acceso secreta para firmar la solicitud. En cambio, primero usa su clave de acceso secreta para derivar una clave de firma. La clave de firma derivada es específica de la fecha, el servicio y la región. Para obtener más información acerca de cómo generar una clave de firma en diferentes lenguajes de programación, consulte [Ejemplos de cómo generar una clave de firma para Signature Version 4](#).

En el siguiente diagrama, se ilustra el proceso general de cálculo de una firma.



La cadena para firmar depende del tipo de solicitud. Por ejemplo, cuando utiliza el encabezado de autorización HTTP o los parámetros de consulta para la autenticación, utiliza una combinación variable de elementos de solicitud a fin de crear la cadena para firmar. Para una solicitud HTTP POST, la política POST de la solicitud es la cadena que firma. Para obtener más información acerca de cómo calcular la cadena para firmar, siga los enlaces que se proporcionan al final de esta sección.

En clave de firma, el diagrama muestra una serie de cálculos, donde el resultado de cada paso se suma al siguiente paso. El último paso es la clave de firma.

Al recibir una solicitud autenticada, los servidores de Amazon S3 vuelven a crear la firma con la información de autenticación incluida en la solicitud. Si las firmas coinciden, Amazon S3 procesa la solicitud; de lo contrario, se rechaza.

Para obtener más información sobre la autenticación de solicitudes, consulte los siguientes temas:

- [Autenticación de solicitudes: uso del encabezado de Authorization \(AWS Signature Version 4\)](#)
- [Autenticación de solicitudes: uso de parámetros de consulta \(AWS Signature Version 4\)](#)
- [Cargas basadas en el navegador con POST \(AWS Signature Version 4\)](#)

Autenticación de solicitudes: uso del encabezado de Authorization (AWS Signature Version 4)

Temas

- [Descripción general](#)

- [Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en un solo fragmento \(AWS Signature Version 4\)](#)
- [Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en varios fragmentos \(carga fragmentada\) \(AWS Signature Version 4\)](#)
- [Cálculos de firmas para los encabezados finales \(cargas fragmentadas\) \(AWS Signature Version 4\)](#)

Descripción general

El uso del encabezado HTTP `Authorization` es el método más común para proporcionar información de autenticación. A excepción de las [solicitudes POST](#) y las solicitudes que se firman mediante parámetros de consulta, todas las operaciones de Amazon S3 utilizan el encabezado de solicitud `Authorization` para proporcionar información de autenticación.

A continuación, se muestra un ejemplo de un valor de encabezado de `Authorization`. Se agregan saltos de línea a este ejemplo para facilitar la lectura:

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

En la siguiente tabla, se describen los distintos componentes del valor del encabezado `Authorization` del ejemplo anterior:

Componente	Descripción
AWS4-HMAC-SHA256	El algoritmo que se utilizó para calcular la firma. Debe proporcionar este valor cuando utilice AWS Signature Version 4 para la autenticación. La cadena específica AWS Signature Version 4 (AWS4) y el algoritmo de firma (HMAC-SHA256).
Credential	

Componente	Descripción
	El ID de clave de acceso y la información sobre el alcance, que incluye la fecha, la región y el servicio que se utilizaron para calcular la firma. Esta cadena tiene el siguiente formato: <your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request Donde: - El valor de <date> se especifica mediante el formato YYYYMMDD. - El valor de <aws-service> es s3 cuando se envía una solicitud a Amazon S3.
SignedHeaders	Una lista de encabezados de solicitud separados por punto y coma que utilizó para calcular Signature . La lista solo incluye los nombres de los encabezados y estos nombres deben estar en minúsculas. Por ejemplo: host;range;x-amz-date

Componente	Descripción
Signature	La firma de 256 bits expresada en 64 caracteres hexadecimales en minúsculas. Por ejemplo: <pre>fe5f80f77d5fa3beca038a248ff027d0445342fe2855d dc963176630326f1024</pre> Tenga en cuenta que los cálculos de la firma varían según la opción que elija para transferir la carga.

Los cálculos de la firma varían según el método que elija para transferir la carga útil de la solicitud. S3 admite las siguientes opciones:

- Transfiera la carga útil en un solo fragmento: en este caso, tiene las siguientes opciones de cálculo de firmas:
 - Opción de carga útil firmada: si lo desea, puede calcular la suma de comprobación completa de la carga útil e incluirla en el cálculo de firmas. Esto proporciona mayor seguridad, pero es necesario leer la carga útil dos veces o almacenarla en un búfer en la memoria.

Por ejemplo, para cargar un archivo, primero debe leerlo para calcular un hash de carga útil para el cálculo de la firma y, de nuevo, para la transmisión al crear la solicitud. Para cargas útiles más pequeñas, este enfoque podría ser preferible. Sin embargo, en el caso de archivos grandes, leer el archivo dos veces puede resultar ineficiente, por lo que puede que prefiera cargar los datos por partes.

Recomendamos que se incluya la suma de comprobación de la carga útil para mayor seguridad.

- Opción de carga útil sin firmar: no incluya la suma de comprobación de la carga útil en el cálculo de la firma.

Para ver instrucciones paso a paso acerca de cómo calcular la firma y crear el valor del encabezado de Authorization, consulte [Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en un solo fragmento \(AWS Signature Version 4\)](#).

- Transfiera la carga útil en varios fragmentos (carga fragmentada): en este caso, se transfiere la carga útil en fragmentos. Puede transferir una carga útil en fragmentos independientemente del tamaño de la carga útil.

Puede dividir la carga útil en fragmentos. Pueden ser fragmentos de tamaño fijo o variable. Al cargar los datos en fragmentos, se evita leer toda la carga útil para calcular la firma. En su lugar, para el primer fragmento, se calcula una firma inicial que utiliza solo los encabezados de las solicitudes. El segundo fragmento contiene la firma del primer fragmento y cada fragmento subsiguiente contiene la firma del fragmento que lo precede. Al final de la carga, se envía un último fragmento con 0 bytes de datos que contiene la firma del último fragmento de la carga útil. Para obtener más información, consulte [Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en varios fragmentos \(carga fragmentada\) \(AWS Signature Version 4\)](#).

Al firmar sus solicitudes, puede utilizar cualquiera de las siguientes opciones: AWS Signature Version 4 o AWS Signature Version 4A. La diferencia clave entre las dos se determina por la forma en que se calcula la firma. Con AWS Signature Version 4A, la firma no incluye información específica de la región y se calcula mediante el algoritmo AWS4-ECDSA-P256-SHA256.

Además de estas opciones, tiene la opción de incluir un tráiler con la solicitud. Para incluir un tráiler en la solicitud, debe especificarlo en el encabezado configurando `x-amz-content-sha256` en el valor adecuado. Si utiliza un encabezado final, debe incluir `x-amz-trailer` en el encabezado y especificar los nombres de los encabezados finales como una cadena en una lista separada por comas. Todos los encabezados finales se escriben después del último fragmento. Si va a cargar los datos en varios fragmentos, debe enviar un último fragmento con 0 bytes de datos antes de enviar el encabezado final.

Al enviar una solicitud, debe indicar a Amazon S3 cuál de las opciones anteriores ha elegido en el cálculo de la firma agregando el encabezado `x-amz-content-sha256` con uno de los siguientes valores:

Valor del encabezado	Descripción
Valor real de la suma de comprobación de la carga útil	Este valor es la suma de comprobación real del objeto y solo es posible cuando se cargan los datos en un único fragmento.
UNSIGNED-PAYLOAD	Use esto cuando cargue el objeto como un único fragmento sin firmar.

Valor del encabezado	Descripción
STREAMING-UNSIGNED-PAYLOAD-TRAILER	Use esto cuando envíe una carga útil sin firmar en varios fragmentos. En este caso, también tendrá un encabezado final después de cargar el fragmento.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD	Use esto cuando envíe una carga útil en varios fragmentos y los fragmentos se firmen con AWS4-HMAC-SHA256 . Esto produce una firma SigV4.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER	Use esto cuando envíe una carga útil en varios fragmentos y los fragmentos se firmen con AWS4-HMAC-SHA256 . Esto produce una firma SigV4. Además, el resumen de los fragmentos se incluye como encabezado final.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD	Use esto cuando envíe una carga útil en varios fragmentos y los fragmentos se firmen con AWS4-ECDSA-P256-SHA256 . Esto produce una firma SigV4A.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD-TRAILER	Use esto cuando envíe una carga útil en varios fragmentos y los fragmentos se firmen con AWS4-ECDSA-P256-SHA256 . Esto produce una firma SigV4A. Además, el resumen de los fragmentos se incluye como encabezado final.

Al recibir la solicitud, Amazon S3 vuelve a crear la cadena para firmarla con la información del encabezado `Authorization` y del encabezado `date`. A continuación, verifica con el servicio de autenticación que las firmas coinciden. La fecha de la solicitud se puede especificar mediante el encabezado `HTTP Date` o el encabezado `x-amz-date`. Si ambos encabezados están presentes, `x-amz-date` tiene prioridad.

Si las firmas coinciden, Amazon S3 procesa la solicitud; de lo contrario, la solicitud producirá un error.

Para obtener más información, consulte los temas siguientes:

[Cálculos de firma para el encabezado de `Authorization`: transferencia de la carga útil en un solo fragmento \(AWS Signature Version 4\)](#)

[Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en varios fragmentos \(carga fragmentada\) \(AWS Signature Version 4\)](#)

[Cálculos de firmas para los encabezados finales \(cargas fragmentadas\) \(AWS Signature Version 4\)](#)

Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en un solo fragmento (AWS Signature Version 4)

Cuando se utiliza el encabezado `Authorization` para autenticar las solicitudes, el valor del encabezado incluye, entre otras cosas, una firma. Los cálculos de la firma varían según lo que elija para transferir la carga útil ([Descripción general](#)). En esta sección, se explican los cálculos de la firma cuando se elige transferir la carga útil en un solo fragmento. La sección de ejemplos (consulte [Ejemplos: Cálculos de firma](#)) muestra los cálculos de firma y los encabezados `Authorization` resultantes que puede utilizar como conjunto de pruebas para verificar el código.

Important

Al transferir la carga útil en un único fragmento, si lo desea, puede optar por incluir el hash de la carga útil en los cálculos de firma, lo que se denomina carga útil firmada (si no la incluye, la carga útil se considera no firmada). El procedimiento de firma analizado en la siguiente sección se aplica a ambos, pero hay que tener en cuenta las siguientes diferencias:

- Opción de carga firmada: se incluye el hash de carga útil al crear la solicitud canónica (que luego pasa a formar parte de `StringToSign`, como se explica en la sección de cálculo de firmas). También se especifica el mismo valor que el valor del encabezado `x-amz-content-sha256` al enviar la solicitud a S3.
- Opción de carga útil sin firmar: se incluye la cadena literal `UNSIGNED-PAYLOAD` al crear una solicitud canónica y se establece el mismo valor que el valor del encabezado `x-amz-content-sha256` al enviar la solicitud a Amazon S3.

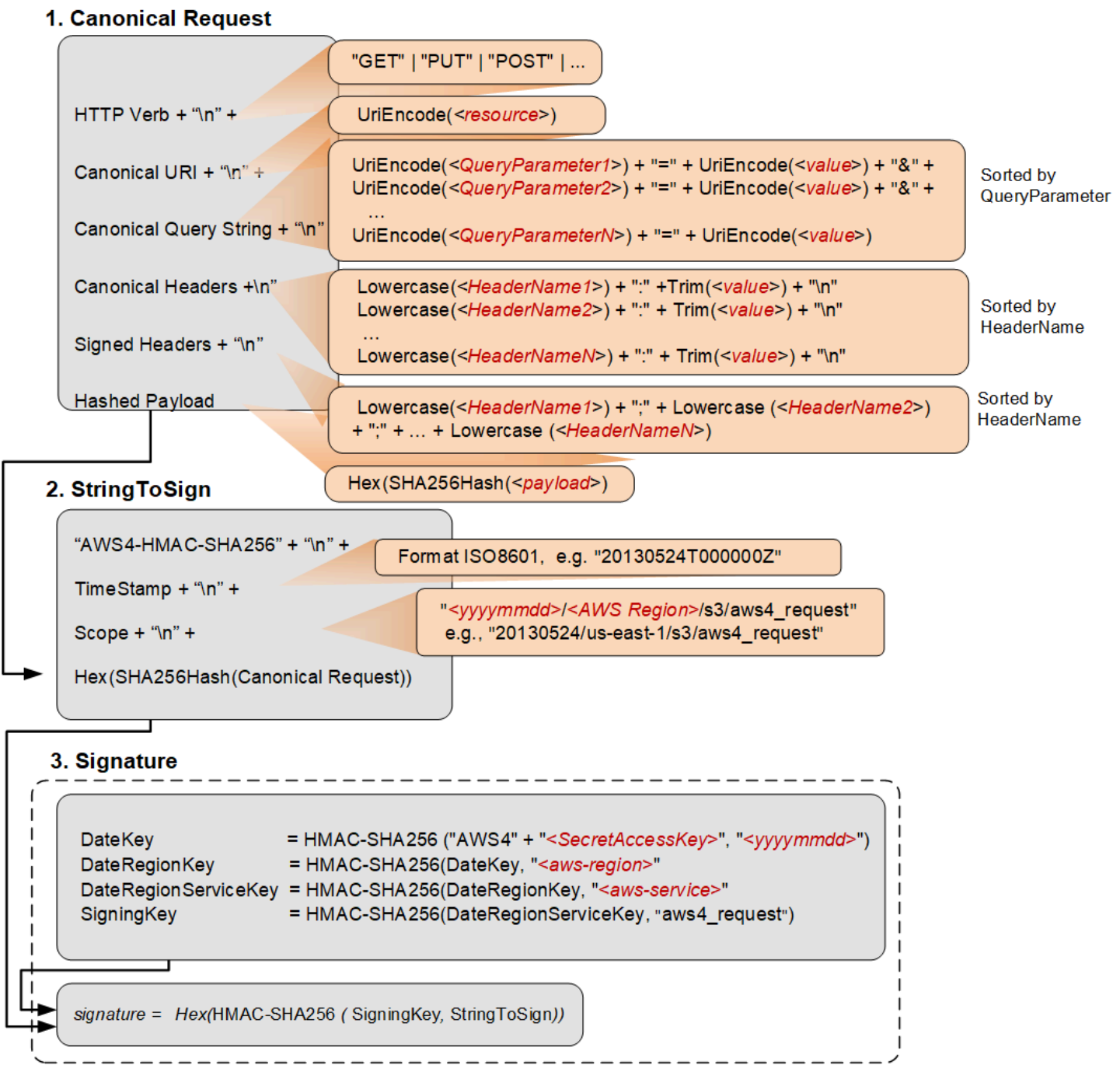
Al enviar la solicitud a Amazon S3, el valor del encabezado `x-amz-content-sha256` informa a Amazon S3 de si la carga útil está firmada o no. Amazon S3 puede entonces crear la firma correspondiente para su verificación.

En ambos casos, dado que el valor del encabezado `x-amz-content-sha256` ya forma parte de `HashedPayload`, no es necesario que incluya el encabezado `x-amz-content-sha256` como canónico.

Cálculo de una firma

A fin de calcular una firma, primero necesita una cadena para firmar. Luego, se calcula un hash HMAC-SHA256 de la cadena que se va a firmar mediante una clave de firma. En el siguiente diagrama, se ilustra el proceso, incluidos los distintos componentes de la cadena que se crea para la firma

Cuando Amazon S3 recibe una solicitud autenticada, calcula la firma y luego la compara con la firma proporcionada en la solicitud. Por este motivo, se debe calcular la firma utilizando el mismo método que use Amazon S3. El proceso de poner una solicitud en una forma acordada para la firma se llama canonicalización.



Función	Descripción
Hex()	Codificación en minúsculas en base 16.
SHA256Hash()	Función de hash criptográfico del algoritmo de hash seguro (SHA).
HMAC-SHA256()	Calcula el HMAC mediante el algoritmo SHA256 con la clave de firma proporcionada. Esta es la firma final.
Trim()	Elimine cualquier espacio en blanco inicial o final.

Función	Descripción
UriEncode()	El URI codifica cada byte. UriEncode() debe aplicar las siguientes reglas: • El URI codifica todos los bytes excepto los caracteres no reservados: “A”-“Z”, “a”-“z”, “0”-“9”, “-”, “.”, “_” y “~”. • El carácter de espacio es un carácter reservado y debe codificarse como “%20” (y no como “+”). • Cada byte codificado en URI se encuentra formado por un “%” y el valor hexadecimal de dos dígitos del byte. • Las letras del valor hexadecimal deben estar en mayúsculas, por ejemplo, “%1A”. • Codifique el carácter de barra diagonal, “/”, en todas partes excepto en el nombre de la clave del objeto. Por ejemplo, si el nombre de la clave del objeto es photos/Jan/sample.jpg , la barra diagonal del nombre de la clave no se encuentra codificada.  Important Es posible que las funciones estándar de UriEncode que proporciona su plataforma de desarrollo no funcionen debido a las diferencias en la implementación y a la ambigüedad relacionada en las RFC subyacentes. Le recomendamos que escriba su propia función UriEncode personalizada para asegurarse de que la codificación funcione. A continuación, se muestra una función UriEncode() de ejemplo en Java. <pre data-bbox="617 1738 1412 1812">public static String UriEncode(CharSequence input, boolean encodeSlash) {</pre>

Función	Descripción
	<pre> 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>

Tarea 1: Crear una solicitud canónica

En esta sección, se proporciona información general sobre la creación de una solicitud canónica.

A continuación, se muestra el formato de solicitud canónica que Amazon S3 utiliza para calcular una firma. Para que las firmas coincidan, debe crear una solicitud canónica en este formato:

```

<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>

```

Donde:

- *HTTPMethod* es uno de los métodos HTTP, por ejemplo GET, PUT, HEAD y DELETE.
- *CanonicalURI* es la versión codificada del URI del componente de ruta absoluta del URI, todo lo que comienza con la "/" que sigue al nombre de dominio y continúa hasta el final de la cadena o del signo de interrogación ("?",) si tiene parámetros de cadena de consulta. El URI del siguiente

ejemplo, `/examplebucket/myphoto.jpg`, es la ruta absoluta y no se codifica la `/` en la ruta absoluta:

```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

 Note

No se normalizan las rutas de URI para las solicitudes a Amazon S3. Por ejemplo, puede tener un bucket con un objeto denominado `my-object//example//photo.user`. Al normalizar la ruta, el nombre del objeto de la solicitud pasa a ser `my-object/example/photo.user`. Esta es una ruta incorrecta para ese objeto.

- ***CanonicalQueryString*** especifica los parámetros de la cadena de consulta codificados en URI. Codifique en URI el nombre y los valores de forma individual. También debe ordenar los parámetros de la cadena de consulta canónica alfabéticamente por nombre de clave. La clasificación se produce después de la codificación. La cadena de consulta del siguiente ejemplo de URI es `prefix=somePrefix&marker=someMarker&max-keys=20`:

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

La cadena de consulta canónica es la siguiente (se agregan saltos de línea a este ejemplo para facilitar la lectura):

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&"+  
UriEncode("max-keys")+"="+UriEncode("20") + "&" +  
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

Cuando una solicitud se dirige a un subrecurso, el valor del parámetro de consulta correspondiente será una cadena vacía (`""`). Por ejemplo, el siguiente URI identifica el subrecurso ACL en el bucket `examplebucket`:

```
http://s3.amazonaws.com/examplebucket?acl
```

En este caso, la cadena ***CanonicalQueryString*** es la siguiente:

```
UriEncode("acl") + "=" + ""
```

Si el URI no incluye un "?", no hay ninguna cadena de consulta en la solicitud y establece la cadena de consulta canónica en una cadena vacía (""). Aún tendrá que incluir "\n".

- *CanonicalHeaders* es una lista de los encabezados de las solicitudes con sus valores. Los pares individuales de nombre y valor del encabezado se encuentran separados por el carácter de nueva línea ("\n"). Los nombres de los encabezados deben estar en minúsculas. Debe ordenar alfabéticamente los nombres de encabezado para crear la cadena, como se muestra en el siguiente ejemplo:

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"  
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"  
...  
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

Las funciones Lowercase() y Trim() que se utilizan en este ejemplo se describen en la sección anterior.

La lista *CanonicalHeaders* debe incluir lo siguiente:

- Encabezado de host HTTP.
- Si el encabezado Content-MD5 se encuentra en la solicitud, debe agregarlo a la lista *CanonicalHeaders*.
- También debe agregar cualquier encabezado x-amz-* que planea incluir en su solicitud. Por ejemplo, si utiliza credenciales de seguridad temporales, debe incluir x-amz-security-token en su solicitud. Debe agregar este encabezado a la lista de *CanonicalHeaders*.

Note

- El encabezado x-amz-content-sha256 es obligatorio para todas las solicitudes de Signature Version 4 de AWS. Proporciona un hash de la carga de solicitud. Sin embargo, no es necesario que incluya x-amz-content-sha256 como encabezado canónico porque S3 utilizará automáticamente su valor al calcular el hash de carga útil enviado en la solicitud.
- Si no hay ninguna carga, debe proporcionar el hash de una cadena vacía.

- Si no desea que S3 compruebe el hash de la solicitud, puede usar la cadena literal "UNSIGNED-PAYLOAD" en su lugar.

A continuación, se muestran cadenas de CanonicalHeaders de ejemplo. Los nombres de encabezado se encuentran en minúsculas y ordenados.

Example Ejemplo 1

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

Example Ejemplo 2

```
host:s3.amazonaws.com
x-amz-content-sha256:UNSIGNED-PAYLOAD
x-amz-date:20130708T220855Z
```

Example Ejemplo 3

```
host:s3.amazonaws.com
x-amz-date:20130708T220855Z
```

Note

Para calcular una firma de autorización, se requieren el encabezado del host y todos los encabezados x-amz-*, excepto x-amz-content-sha256. Sin embargo, a fin de evitar la manipulación de datos, debería considerar la posibilidad de incluir todos los encabezados en el cálculo de la firma.

Firmar el encabezado x-amz-content-sha256 es opcional porque S3 usará su valor al calcular el hash de carga útil de la solicitud recibida.

- **SignedHeaders** es una lista ordenada alfabéticamente y separada por punto y coma de nombres de encabezados de solicitudes en minúsculas. Los encabezados de las solicitudes de la lista son los mismos que incluyó en la cadena de CanonicalHeaders. En los ejemplos anteriores, el valor de **SignedHeaders** sería el siguiente:

Example Ejemplos 1 y 2

```
host;x-amz-content-sha256;x-amz-date
```

Example Ejemplo 3

```
host;x-amz-date
```

- *HashedPayload* es el valor hexadecimal del hash SHA256 de la carga útil de la solicitud.

```
Hex(SHA256Hash(<payload>))
```

Si no hay ninguna carga en la solicitud, se calcula un hash de la cadena vacía de la siguiente manera:

```
Hex(SHA256Hash(""))
```

El hash devuelve los siguientes valores:

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Por ejemplo, cuando carga un objeto mediante una solicitud PUT, proporciona los datos del objeto en el cuerpo. Al recuperar un objeto mediante una solicitud GET, se calcula el hash de la cadena vacía.

Tarea 2: Crear una cadena para firmar

En esta sección, se proporciona información general sobre la creación de una cadena para firmar. Para obtener instrucciones paso a paso, consulte [Tarea 2: Crear una cadena para firmar](#) en Referencia general de AWS.

La cadena para firmar es una concatenación de las cadenas siguientes:

```
"AWS4-HMAC-SHA256" + "\n" +  
timestampISO8601Format + "\n" +
```

```
<Scope> + "\n" +  
Hex(SHA256Hash(<CanonicalRequest>))
```

La cadena constante AWS4-HMAC-SHA256 especifica el algoritmo hash que está utilizando, HMAC-SHA256. timeStamp es la hora UTC actual en formato ISO 8601 (por ejemplo, 20130524T000000Z).

Scope vincula la firma resultante a una fecha, una Región de AWS y un servicio específicos. Por lo tanto, la firma resultante solo funcionará en la región específica y para un servicio específico. La firma es válida durante siete días a partir de la fecha especificada.

```
date.Format(<YYYYMMDD>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

Para Amazon S3, la cadena del servicio es s3. Para obtener una lista de cadenas de *regiones*, consulte [Regiones y puntos de conexión](#) en Referencia general de AWS. La columna Región de esta tabla proporciona la lista de cadenas de región válidas.

El siguiente alcance restringe la firma resultante a la región us-east-1 y Amazon S3.

```
20130606/us-east-1/s3/aws4_request
```

Note

Scope debe utilizar la misma fecha que utiliza para calcular la clave de firma, tal y como se explica en la siguiente sección.

Tarea 3: Cálculo de la firma

En AWS Signature Version 4, en lugar de utilizar las claves de acceso de AWS para firmar una solicitud, se crea primero una clave de firma que se limita a una región y un servicio específicos. Para obtener más información sobre cómo firmar claves, consulte [Introducción a la firma de solicitudes](#).

```
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

Algunos casos de uso pueden procesar claves de firma durante un máximo de 7 días. Para obtener más información, consulte [Compartir un objeto con otros](#).

Para obtener una lista de cadenas de regiones, consulte [Regiones y puntos de conexión](#) en Referencia general de AWS.

El uso de una clave de firma permite guardar las credenciales de AWS en un lugar seguro. Por ejemplo, si tiene varios servidores que se comunican con Amazon S3, comparte la clave de firma con esos servidores; no tiene que guardar una copia de la clave de acceso secreta en cada servidor. La clave de firma es válida durante un máximo de siete días. Por lo tanto, cada vez que calcule la clave de firma, tendrá que compartirla con los servidores. Para obtener más información, consulte [Autenticación de solicitudes \(AWS Signature Version 4\)](#).

La firma final es el hash HMAC-SHA256 de la cadena a firmar, mediante la clave de firma como clave.

```
HMAC-SHA256(SigningKey, StringToSign)
```

Para obtener instrucciones paso a paso sobre cómo crear una firma, consulte [Tarea 3: Crear una firma](#) en Referencia general de AWS.

Ejemplos: Cálculos de firma

Puede utilizar los ejemplos en esta sección como referencia para comprobar los cálculos de firma en el código. Los cálculos que se muestran en los ejemplos utilizan los siguientes datos:

- Ejemplo de claves de acceso.

Parámetro	Valor
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- Marca temporal de solicitud de 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT).
- Nombre del bucket: `examplebucket`.
- Se supone que el bucket está en la región Este de EE. UU. (Norte de Virginia). La credencial Scope y los cálculos de Signing Key utilizan `us-east-1` como especificador de región. Para obtener información sobre otras regiones, consulte [Regiones y puntos de conexión](#) en Referencia general de AWS.
- Puede utilizar solicitudes de tipo ruta o de alojamiento virtual. Los ejemplos siguientes muestran cómo firmar una solicitud de tipo de alojamiento virtual, por ejemplo:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Para obtener más información, consulte [Alojamiento virtual de buckets](#) en la Guía del usuario de Amazon Simple Storage Service.

Ejemplo: objeto GET

El ejemplo siguiente obtiene los primeros 10 bytes de un objeto (`test.txt`) de `examplebucket`. Para obtener más información sobre la acción de la API, consulte [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
```

Como esta solicitud GET no proporciona ningún contenido en el cuerpo, el valor `x-amz-content-sha256` puede ser el hash del cuerpo vacío de la solicitud o la cadena literal "UNSIGNED-PAYLOAD". Los pasos siguientes muestran los cálculos de firma y la construcción del encabezado `Authorization` mediante el hash de una cadena vacía.

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
```

En la cadena de solicitud canónica, la última línea es el hash del cuerpo de la solicitud vacío. La tercera línea se encuentra vacía porque no hay parámetros de consulta en la solicitud.

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. Encabezado Authorization

El encabezado de Authorization resultante es el siguiente:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;range;x-amz-content-sha256;x-amz-
date,Signature=f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

Ejemplo: objeto PUT

En este ejemplo de solicitud PUT se crea un objeto (`test$file.text`) en `examplebucket`. El ejemplo supone lo siguiente:

- Para solicitar `REDUCED_REDUNDANCY` como clase de almacenamiento, agregue el encabezado de solicitud `x-amz-storage-class`. Para obtener información sobre las clases de almacenamiento, consulte [Clases de almacenamiento](#) en la Guía del usuario de Amazon Simple Storage Service.
- El contenido del archivo cargado es una cadena, "Welcome to Amazon S3." El valor de `x-amz-content-sha256` en la solicitud se basa en esta cadena.

Para obtener información acerca de la acción de la API, consulte [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>
```

Los pasos siguientes muestran cálculos de firma.

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

```

En la solicitud canónica, la tercera línea se encuentra vacía porque no hay parámetros de consulta en la solicitud. El encabezado canónico `x-amz-content-sha256` puede firmarse opcionalmente, ya que su hash de carga útil ya está incluido en la parte inferior de la solicitud. La última línea es el hash del cuerpo, que debe ser igual al valor `x-amz-content-sha256` header enviado a S3 en la solicitud 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. Encabezado Authorization

El encabezado de Authorization resultante es el siguiente:

```
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
```

Ejemplo: ciclo de vida de bucket GET

La solicitud GET siguiente recupera la configuración del ciclo de vida de `examplebucket`. Para obtener información sobre la acción de la API, consulte [GetBucketLifecycleConfiguration](#).

```
GET ?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Como la solicitud no proporciona ningún contenido en el cuerpo, el valor del encabezado `x-amz-content-sha256` es el hash del cuerpo vacío de la solicitud. Los pasos siguientes muestran cálculos de firma.

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
```

En la solicitud canónica, la última línea es el hash del cuerpo de la solicitud vacío.

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. Encabezado Authorization

El encabezado de Authorization resultante es el siguiente:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

Ejemplo: obtener Bucket (Mostrar objetos)

En el ejemplo siguiente, se recupera una lista de objetos del bucket `examplebucket`. Para obtener información acerca de la acción de la API, consulte [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
```

Como la solicitud no proporciona un cuerpo, el valor de `x-amz-content-sha256` es el hash del cuerpo vacío de la solicitud. Los pasos siguientes muestran cálculos de firma.

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
```

En la cadena canónica, la última línea es el hash del cuerpo de la solicitud vacío.

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. Encabezado Authorization

El encabezado de Authorization resultante es el siguiente:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-
date,Signature=34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en varios fragmentos (carga fragmentada) (AWS Signature Version 4)

Como se describe en [Descripción general](#), al autenticar las solicitudes mediante el encabezado Authorization, tiene una opción de cargar la carga útil en fragmentos. Puede enviar datos en fragmentos de tamaño fijo o de tamaño variable. En esta sección, se describe el proceso de cálculo de firmas en la carga fragmentada, cómo se crea el cuerpo del fragmento y cómo funciona la firma diferida cuando se carga el fragmento por primera vez y se envía su firma en el fragmento siguiente. La sección de ejemplos (consulte [Ejemplo: objeto PUT](#)) muestra los cálculos de firma y los encabezados Authorization resultantes que puede utilizar como conjunto de pruebas para verificar el código.

Note

Al transferir datos en una serie de fragmentos, debe realizar una de estas acciones:

- Especifique de forma explícita la longitud total del contenido (longitud del objeto en bytes más los metadatos de cada fragmento) mediante el encabezado HTTP Content-Length. Para ello, debe calcular previamente la longitud total de la carga útil, incluidos los metadatos que envíe en cada fragmento, antes de iniciar la solicitud.
- Especifique el encabezado HTTP Transfer-Encoding. Si incluye el encabezado Transfer-Encoding y especifica cualquier otro valor que no sea identity, debe omitir el encabezado Content-Length.

Para todas las solicitudes, debe incluir el encabezado x-amz-decoded-content-length, especificando el tamaño del objeto en bytes.

El cálculo de la firma de cada fragmento incluye la firma del fragmento anterior. Para empezar, debe crear una firma inicial utilizando solo los encabezados. La firma inicial se utiliza en el cálculo de la firma del primer fragmento. Para cada fragmento subsiguiente, se crea una firma de fragmento que incluye la firma del fragmento anterior. Por lo tanto, las firmas de los fragmentos se encadenan entre sí; es decir, la firma del fragmento n es una función $F(\text{fragmento } n, \text{firma}[\text{fragmento } n-1])$. El encadenamiento garantiza que envíe los fragmentos en el orden correcto.

Para realizar una carga fragmentada, haga lo siguiente:

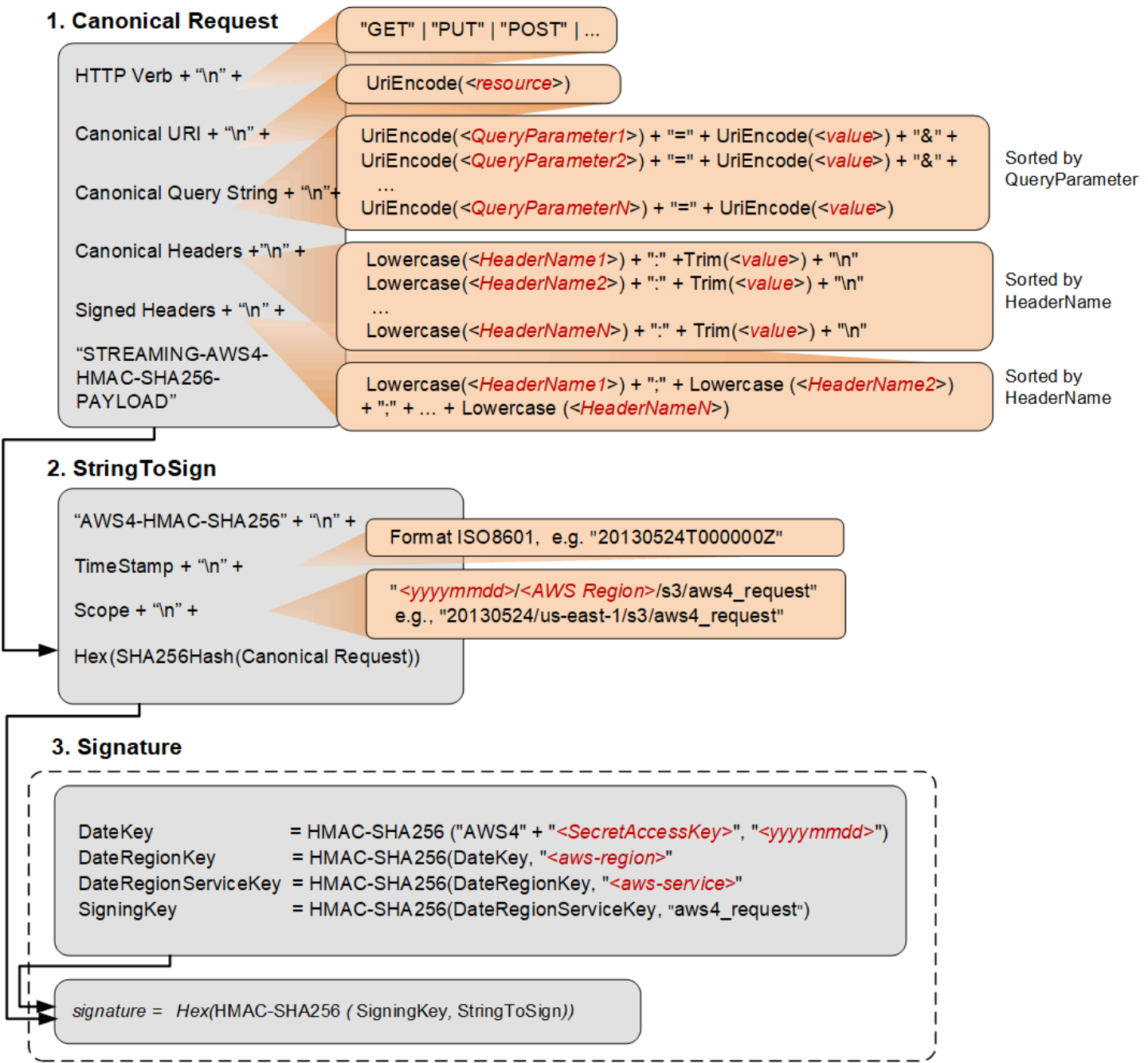
1. Decida el tamaño del fragmento de carga útil. Lo necesita cuando escriba el código.

El tamaño del fragmento debe ser de al menos 8 KB. Recomendamos un tamaño de fragmento de al menos 64 KB para obtener un mejor rendimiento. Este tamaño de fragmento se aplica a todos los fragmentos excepto al último. El último fragmento que envíe puede tener un tamaño inferior a 8 KB. Si la carga útil es pequeña y cabe en un solo fragmento, puede ser inferior a los 8 KB.

2. Cree la firma inicial para incluirla en el primer fragmento. Para obtener más información, consulte [Cálculo de la firma inicial](#).
3. Cree el primer fragmento y transmítalo. Para obtener más información, consulte [Definición del cuerpo del fragmento](#).
4. Para cada fragmento subsiguiente, calcule la firma del fragmento que incluye la firma anterior en la cadena que firme, cree el fragmento y envíelo. Para obtener más información, consulte [Definición del cuerpo del fragmento](#).
5. Envíe el último fragmento adicional, que es el mismo que los demás fragmentos de la creación, pero tiene cero bytes de datos. Para obtener más información, consulte [Definición del cuerpo del fragmento](#).

Cálculo de la firma inicial

En el diagrama siguiente, se ilustra el proceso de cálculo de la firma inicial.



En la siguiente tabla, se describen las funciones que se muestran en el diagrama. Debe implementar código para estas funciones.

Función	Descripción
Lowercase()	Convierta la cadena de caracteres en minúsculas.
Hex()	Codificación en minúsculas en base 16.

Función	Descripción
SHA256Hash()	Función de hash criptográfico del algoritmo de hash seguro (SHA).
HMAC-SHA256()	Calcula el HMAC mediante el algoritmo SHA256 con la clave de firma proporcionada. Esta es la firma final.
Trim()	Elimine cualquier espacio en blanco inicial o final.

Función	Descripción
<code>UriEncode()</code>	El URI codifica cada byte. <code>UriEncode()</code> debe aplicar las siguientes reglas: • El URI codifica todos los bytes excepto los caracteres no reservados: “A”-“Z”, “a”-“z”, “0”-“9”, “-”, “.”, “_” y “~”. • El carácter de espacio es un carácter reservado y debe codificarse como “%20” (y no como “+”). • Cada byte codificado en URI se encuentra formado por un “%” y el valor hexadecimal de dos dígitos del byte. • Las letras del valor hexadecimal deben estar en mayúsculas, por ejemplo, “%1A”. • Codifique el carácter de barra diagonal, “/”, en todas partes excepto en el nombre de la clave del objeto. Por ejemplo, si el nombre de la clave del objeto es <code>photos/Jan/sample.jpg</code>, la barra diagonal del nombre de la clave no se encuentra codificada.  Important Es posible que las funciones estándar de <code>UriEncode</code> que proporciona su plataforma de desarrollo no funcionen debido a las diferencias en la implementación y a la ambigüedad relacionada en las RFC subyacentes. Le recomendamos que escriba su propia función <code>UriEncode</code> personalizada para asegurarse de que la codificación funcione. A continuación, se muestra una función <code>UriEncode()</code> de ejemplo en Java. <pre data-bbox="617 1738 1412 1812">public static String UriEncode(CharSequence input, boolean encodeSlash) {</pre>

Función	Descripción
	<pre> 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>

Para obtener información sobre el proceso de firma, consulte [Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en un solo fragmento \(AWS Signature Version 4\)](#).

El proceso es el mismo, excepto que la creación de CanonicalRequest difiere de la siguiente manera:

- Además de los encabezados de solicitud que planea agregar, debe incluir los siguientes encabezados:

Encabezado	Descripción
x-amz-content-sha256	Este encabezado es obligatorio para todas las solicitudes de AWS Signature Version 4. Establezca el valor en STREAMING-AWS4-HMAC-SHA256-PAYLOAD para indicar que la firma solo cubre los encabezados y que no hay carga útil.

Encabezado	Descripción
Content-Encoding	Establezca el valor en <code>aws-chunked</code> . Amazon S3 admite varios valores de codificación de contenido. Puede especificar la codificación de contenido personalizada cuando utilice la API de streaming de Signature Version 4. Por ejemplo: Content-Encoding : aws-chunked,gzip Amazon S3 almacena el objeto resultante sin el valor <code>aws-chunked</code> en el encabezado <code>content-encoding</code> . Si <code>aws-chunked</code> es el único valor que se pasa en el encabezado <code>content-encoding</code> , S3 considera el encabezado <code>content-encoding</code> vacío y no lo devuelve cuando se recupera el objeto.
x-amz-decoded-content-length	Establezca el valor en la longitud, en bytes, de los datos que se van a fragmentar, sin contar ningún metadato. Por ejemplo, si va a cargar un archivo de 4 GB, establezca el valor en 4294967296. Este es el tamaño original del objeto que se va a cargar (datos que desea almacenar en Amazon S3).
Content-Length	Establezca el valor en el tamaño real del cuerpo HTTP transmitido, que incluye la longitud de los datos (valor establecido para <code>x-amz-decoded-content-length</code>) y los metadatos de los fragmentos. Cada fragmento tiene metadatos, como la firma del fragmento anterior. Los cálculos de fragmentos se analizan en la siguiente sección. Si incluye el encabezado <code>Transfer-Encoding</code> y especifica cualquier otro valor que no sea <code>identity</code>, no debe incluir el encabezado <code>Content-Length</code> .

Se envía el primer fragmento con la firma inicial. Debe crear el fragmento como se describe en la siguiente sección.

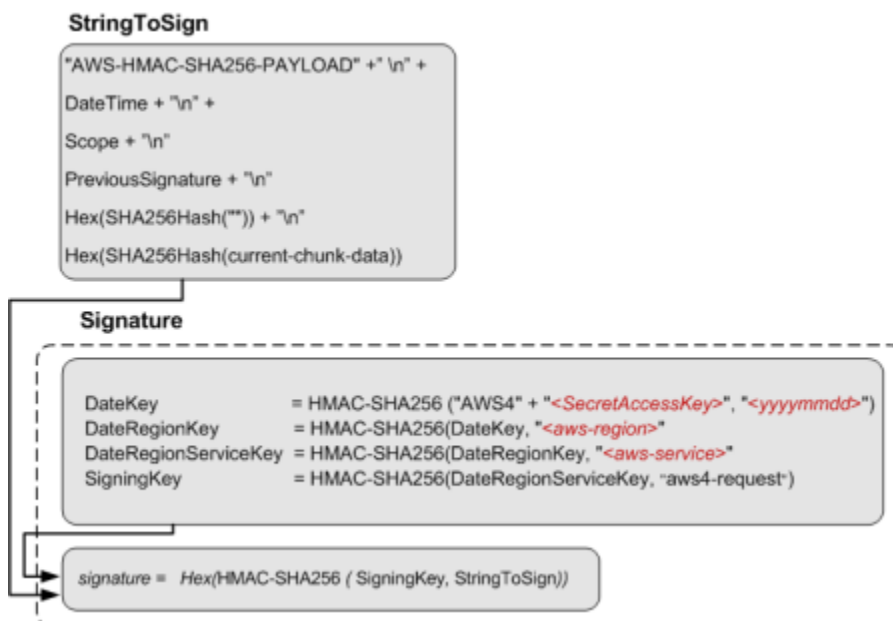
Definición del cuerpo del fragmento

Todos los fragmentos incluyen algunos metadatos. Cada fragmento debe cumplir la siguiente estructura:

```
string(IntHexBase(chunk-size)) + ";chunk-signature=" + signature + \r\n + chunk-data + \r\n
```

Donde:

- `IntHexBase()` es una función que se escribe para convertir el tamaño de un fragmento entero en hexadecimal. Por ejemplo, si el tamaño del fragmento es 65536, la cadena hexadecimal es "10 000".
- *Tamaño de fragmento* es el tamaño, en bytes, de los datos del fragmento, sin metadatos. Por ejemplo, si carga un objeto de 65 KB y utiliza un tamaño de fragmento de 64 KB, carga los datos en tres fragmentos: el primero sería de 64 KB, el segundo de 1 KB y el último fragmento de 0 bytes.
- *Firma* Para cada fragmento, se calcula la firma con la siguiente cadena para firmar. Para el primer fragmento, se utiliza la firma inicial como la firma anterior.



El tamaño de los últimos datos de fragmento que envíe es 0, aunque el cuerpo del fragmento aún contiene metadatos, incluida la firma del fragmento anterior.

Ejemplo: objeto PUT

Puede utilizar los ejemplos en esta sección como referencia para comprobar los cálculos de firma en el código. Antes de revisar los ejemplos, tenga en cuenta lo siguiente:

- Los cálculos de firma de estos ejemplos utilizan las siguientes credenciales de seguridad de ejemplo.

Parámetro	Valor
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- Todos los ejemplos utilizan la marca temporal de la solicitud 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT).
- Todos los ejemplos utilizan `examplebucket` como nombre del bucket.
- Se supone que el bucket está en la región Este de EE. UU. (Norte de Virginia) y la credencial Scope y los cálculos de Signing Key utilizan `us-east-1` como especificador de región. Para obtener más información, consulte [Regiones y puntos de enlace](#) en la Referencia general de Amazon Web Services.
- Puede utilizar solicitudes de tipo ruta o de alojamiento virtual. Los ejemplos siguientes usan solicitudes de tipo de alojamiento virtual, por ejemplo:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Para obtener más información, consulte [Alojamiento virtual de buckets](#) en la Guía del usuario de Amazon Simple Storage Service.

En el siguiente ejemplo, se envía una solicitud PUT para cargar un objeto. Los cálculos de firma suponen lo siguiente:

- Está cargando un archivo de texto de 65 KB y el contenido del archivo es una cadena de un carácter compuesta por la letra “a”.
- El tamaño del fragmento es 64 KB. Como resultado, la carga útil se carga en tres fragmentos: 64 KB, 1 KB y el fragmento final con 0 bytes de datos de fragmento.
- El objeto resultante tiene el nombre de clave `chunkObject.txt`.
- Para solicitar `REDUCED_REDUNDANCY` como clase de almacenamiento, agregue el encabezado de solicitud `x-amz-storage-class`.

Para obtener información acerca de la acción de la API, consulte [PutObject](#). La sintaxis de solicitud general es la siguiente:

```
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>
```

Los pasos siguientes muestran cálculos de firma.

1. Firma inicial: cree una cadena para firmar
 - 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
```

En la solicitud canónica, la tercera línea se encuentra vacía porque no hay parámetros de consulta en la solicitud. La última línea es la cadena constante proporcionada como valor de la carga útil con hash, que debe ser igual al valor de `x-amz-content-sha256` header.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
cee3fed04b70f867d036f722359b0b1f2f0e5dc0efadbc082b76c4c60e316455
```

Note

Para obtener información sobre cada una de las líneas de la cadena que se va a firmar, consulte el diagrama que explica el cálculo de la firma inicial.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Firma inicial

```
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

4. Encabezado Authorization

El encabezado de Authorization resultante es el siguiente:

```
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. Fragmento 1: (65 536 bytes, con un valor de 97 para la letra “a”)

a. Cadena de fragmentos para firmar:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

Para obtener información sobre cada línea de la cadena que se va a firmar, consulte el diagrama anterior, que muestra los distintos componentes de la cadena que se va a firmar (por ejemplo, las tres últimas líneas son `previous-signature`, `hash("")` y `hash(current-chunk-data)`).

b. Firma de fragmentos:

```
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
```

c. Datos de fragmentos enviados:

```
10000;chunk-
signature=ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
<65536-bytes>
```

6. Fragmento 2: (1024 bytes, con un valor de 97 para la letra “a”)

a. Cadena de fragmentos para firmar:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

```
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Firma de fragmentos:

```
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
```

c. Datos de fragmentos enviados:

```
400; chunk-  
signature=0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
<1024 bytes>
```

7. Fragmento 3: (datos de 0 bytes)

a. Cadena de fragmentos para firmar:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Firma de fragmentos:

```
b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

c. Datos de fragmentos enviados:

```
0; chunk-  
signature=b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

Cálculos de firmas para los encabezados finales (cargas fragmentadas) (AWS Signature Version 4)

Al autenticar las solicitudes mediante el encabezado `Authorization`, también puede cargar la carga útil en fragmentos. Cuando envía los datos del objeto en fragmentos, también tiene la opción de incluir los encabezados finales. (Para obtener más información, consulte). [Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en varios fragmentos \(carga fragmentada\) \(AWS Signature Version 4\).](#) En esta sección, se describen los pasos que debe seguir si quiere incluir un encabezado final al final de la carga de varios fragmentos.

Important

Cuando incluya los encabezados finales, debe enviar lo siguiente en el encabezado inicial:

- Debe establecer `x-amz-content-sha256` en un valor adecuado que indica que se incluirá un tráiler. Para ver los valores aceptables para `x-amz-content-sha256`, consulte [Autenticación de solicitudes: uso del encabezado de Authorization \(AWS Signature Version 4\)](#).
- Debe configurar `x-amz-trailer` para indicar el contenido que va a incluir en el encabezado final.

Los encabezados finales solo se envían después de cargar los fragmentos. Los fragmentos anteriores se envían de forma normal y se firman como se describe en las secciones anteriores, incluido el envío del último fragmento con una carga útil de 0 bytes. Los encabezados finales se incluyen como su propio fragmento y se envían después del último fragmento con una carga útil de 0 bytes. Por ejemplo, si los datos terminan en un fragmento de 100 KB, debe enviar lo siguiente:

- Fragmentos de datos anteriores
- Fragmento final de 100 KB del objeto
- Un fragmento de 0 bytes que indica el final del objeto
- Fragmento de encabezados finales

Ejemplos: comprobación de los cálculos de firmas

Puede utilizar los ejemplos en esta sección como referencia para comprobar los cálculos de firma en el código. Antes de revisar los ejemplos, tenga en cuenta lo siguiente:

- Los cálculos de firma de estos ejemplos utilizan las siguientes credenciales de seguridad de ejemplo.

Parámetro	Valor
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- Todos los ejemplos utilizan la marca temporal de la solicitud 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT).
- Todos los ejemplos utilizan `examplebucket` como nombre del bucket.
- Se supone que el bucket está en la región Este de EE. UU. (Norte de Virginia) y la credencial Scope y los cálculos de Signing Key utilizan `us-east-1` como especificador de región. Para obtener más información, consulte [Regiones y puntos de enlace](#) en la Referencia general de Amazon Web Services.
- Puede utilizar solicitudes de tipo ruta o de alojamiento virtual. Los ejemplos siguientes usan solicitudes de tipo de alojamiento virtual, por ejemplo:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Para obtener más información, consulte [Alojamiento virtual de buckets](#) en la Guía del usuario de Amazon Simple Storage Service.

En el siguiente ejemplo, se envía una solicitud PUT para cargar un objeto. Los cálculos de firma suponen lo siguiente:

- Está cargando un archivo de texto de 65 KB y el contenido del archivo es una cadena de un carácter compuesta por la letra “a”.

- El tamaño del fragmento es 64 KB. Como resultado, la carga útil se carga en tres fragmentos: 64 KB, 1 KB y el fragmento final con 0 bytes de datos de fragmento.
- El objeto resultante tiene el nombre de clave `chunkObject.txt`.
- Para solicitar REDUCED_REDUNDANCY como clase de almacenamiento, agregue el encabezado de solicitud `x-amz-storage-class`.
- La transferencia incluye un valor de suma de comprobación CRC32C como encabezado final.

Para obtener información acerca de la acción de la API, consulte [PutObject](#). La sintaxis de solicitud general es la siguiente:

```
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>
```

Los pasos siguientes muestran cálculos de firma.

1. Firma inicial: cree una cadena para firmar
 - 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
```

En la solicitud canónica, la tercera línea se encuentra vacía porque no hay parámetros de consulta en la solicitud. La última línea es la cadena constante proporcionada como valor de la carga útil con hash, que debe ser igual al valor de `x-amz-content-sha256` header.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
44d48b8c2f70eae815a0198cc73d7a546a73a93359c070abbaa5e6c7de112559
```

Note

Para obtener información sobre cada una de las líneas de la cadena que se va a firmar, consulte el diagrama que explica el cálculo de la firma inicial.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Firma inicial

```
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

4. Encabezado Authorization

El encabezado de Authorization resultante es el siguiente:

```
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. Fragmento 1: (65 536 bytes, con un valor de 97 para la letra “a”)

a. Cadena de fragmentos para firmar:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

Para obtener información sobre cada línea de la cadena que se va a firmar, consulte el diagrama en el tema anterior ([Cálculo de la firma inicial](#)) que muestra los distintos componentes de la cadena que se va a firmar. Por ejemplo, las tres últimas líneas constan de lo siguiente:

- *previous-signature*
- `hash("")`
- `hash(current-chunk-data)`

b. Firma de fragmentos:

```
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
```

c. Datos de fragmentos enviados:

```
10000;chunk-
signature=b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
<65536-bytes>
```

6. Fragmento 2: (1024 bytes, con un valor de 97 para la letra “a”)

a. Cadena de fragmentos para firmar:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Firma de fragmentos:

```
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
```

c. Datos de fragmentos enviados:

```
400;chunk-
signature=1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
<1024-bytes>
```

7. Fragmento 3: (datos de 0 bytes)

a. Cadena de fragmentos para firmar:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Firma de fragmentos:

```
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

c. Datos de fragmentos enviados:

```
0;chunk-
signature=2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

8. Fragmento 4: encabezados finales

a. Cadena de fragmento final que se va a firmar:

```

AWS4-HMAC-SHA256-TRAILER
20130524T000000Z
20130524/us-east-1/s3/aws4_request
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
1e376db7e1a34a8ef1c4bcee131a2d60a1cb62503747488624e10995f448d774

```

Note

Las dos últimas líneas son `previous-signature` (la firma del fragmento de datos de 0 bytes), `hash(trailing-checksum-header-name:base64-encoded-trailing-checksum-value\n)`.

El hash se calcula de la siguiente manera sin espacios en blanco:

- El nombre del encabezado de la suma de comprobación final
- Un símbolo de dos puntos (:)
- El valor final de suma de comprobación codificado en base64
- Un carácter de nueva línea (\n).

En este ejemplo, donde utilizamos el algoritmo hash `crc32c`, con el valor de la suma de comprobación codificado en base64 `s008/Q==`, podemos representar el cálculo de la siguiente manera: `hash('x-amz-checksum-crc32c:s008/Q==\n')`.

b. Firma de fragmentos:

```
d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

c. Datos de fragmentos enviados:

```

x-amz-checksum-crc32c:s008/Q==
x-amz-trailer-
signature:d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435

```

Autenticación de solicitudes: uso de parámetros de consulta (AWS Signature Version 4)

Como se describe en la información general de la autenticación (consulte [Métodos de autenticación](#)), puede proporcionar información de autenticación mediante los parámetros de la cadena de consulta. El uso de los parámetros de consulta para autenticar solicitudes es útil cuando desea expresar una solicitud por completo en una URL. Este método también se denomina prefirmar una URL.

Un caso de uso de las URL prefirmadas es que puede conceder acceso temporal a los recursos de Amazon S3. Por ejemplo, puede incrustar una URL prefirmada en el sitio web o utilizarla en un cliente de línea de comandos (como Curl) para descargar objetos.

Note

También puede utilizar la AWS CLI para crear URL prefirmadas. Para obtener más información, consulte [presign](#) en la Referencia de comandos de la AWS CLI.

A continuación, se muestra un ejemplo de una URL prefirmada.

```
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>
```

En la URL de ejemplo, tenga en cuenta lo siguiente:

- Se han agregado saltos de línea para facilitar la lectura.
- El valor X-Amz-Credential de la URL muestra el carácter “/” solo para facilitar la lectura. En la práctica, se debe codificar como %2F. Por ejemplo:

```
&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request
```

En la siguiente tabla, se describen los parámetros de consulta de la URL que proporcionan información de autenticación.

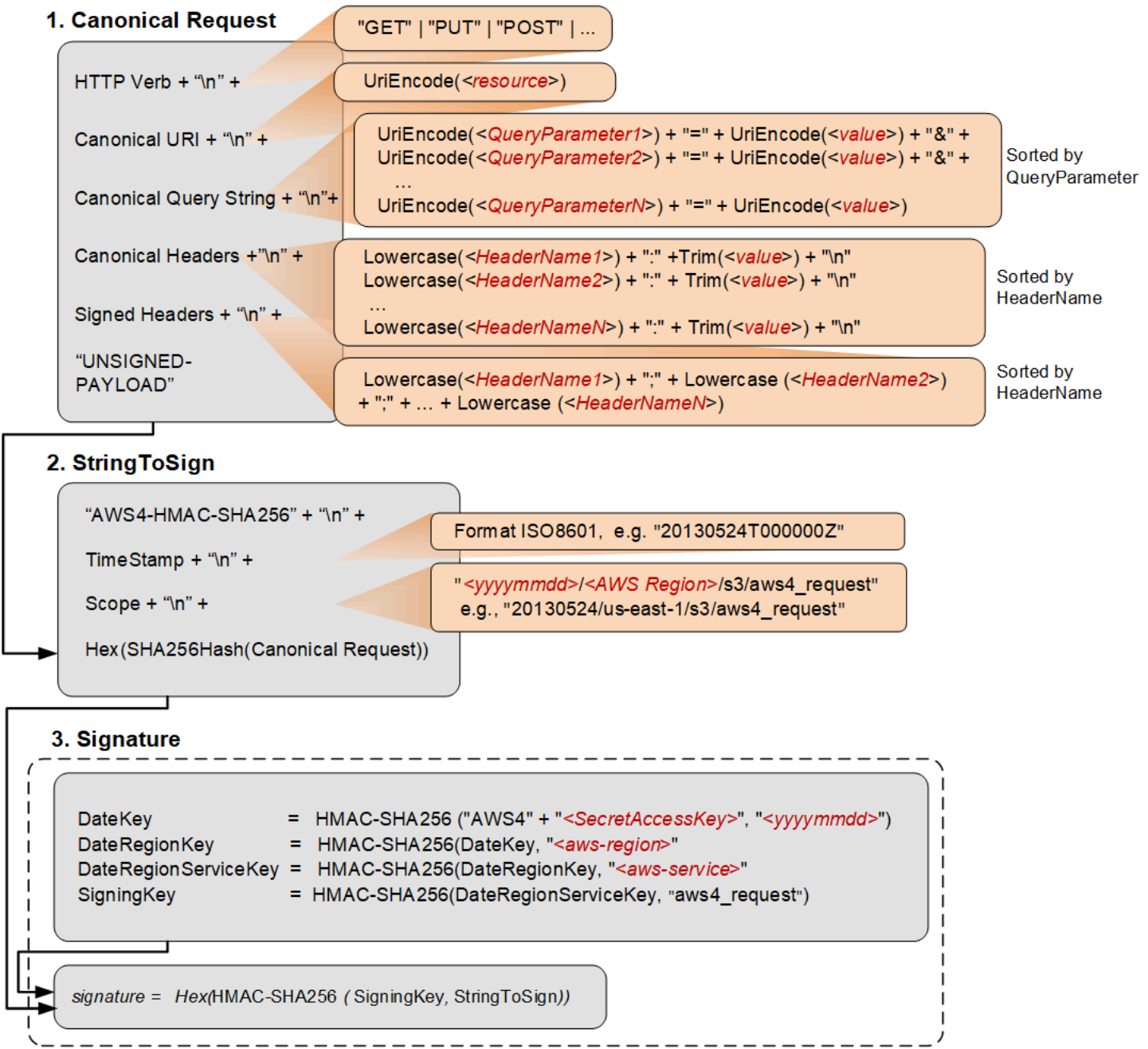
Nombre del parámetro de la cadena de consulta	Ejemplo de valor
X-Amz-Algorithm	Identifica la versión de la firma de AWS y el algoritmo que utilizó para calcular la firma. Para AWS Signature Version 4, defina el valor de este parámetro en <code>AWS4-HMAC-SHA256</code>. Esta cadena identifica AWS Signature Version 4 (AWS 4) y el algoritmo HMAC-SHA256 (HMAC-SHA256).
X-Amz-Credential	Además del ID de clave de acceso, este parámetro también proporciona el alcance (Región de AWS y servicio) para el que es válida la firma. Este valor debe coincidir con el alcance que se utiliza en los cálculos de firmas, como se explica en la siguiente sección. El formato general de este valor de parámetro es el siguiente: <pre><your-access-key-id> /<date>/<Región de AWS>/<AWS-service> /aws4_request</pre> Por ejemplo: <pre>AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request</pre> Para Amazon S3, la cadena del <i>servicio de AWS</i> es <code>s3</code>. Para obtener una lista de las cadenas de <code>AWS-region</code> de S3, consulte Regiones y puntos de conexión en Referencia general de AWS.
X-Amz-Date	El formato de fecha y hora debe seguir la norma ISO 8601 y debe tener el formato “<i>yyyyMMddTHHmmsZ</i>”. Por ejemplo, si

Nombre del parámetro de la cadena de consulta	Ejemplo de valor
	la fecha y la hora son “08/01/2016 15:32:41.982-700”, primero se deben convertir a UTC (hora universal coordinada) y, a continuación, se deben enviar como “20160801T223241Z”.
X-Amz-Expires	Proporciona el periodo, en segundos, durante el que es válida la URL prefirmada generada. Por ejemplo, 86400 (24 horas). Este valor es un entero. El valor mínimo que puede establecer es 1 y el máximo es 604 800 (siete días). Una URL prefirmada puede ser válida durante un máximo de siete días porque la clave de firma que utiliza para calcular la firma es válida durante un máximo de siete días.
X-Amz-SignedHeaders	Muestra los encabezados que utilizó para calcular la firma. Los siguientes encabezados son obligatorios para los cálculos de firmas: • El encabezado HTTP host. • Cualquier encabezado x-amz-* que planea agregar a la solicitud.  Note Para mayor seguridad, debe firmar todos los encabezados de solicitud que planea incluir en su solicitud.

Nombre del parámetro de la cadena de consulta	Ejemplo de valor
X-Amz-Signature	Proporciona la firma para autenticar la solicitud. Esta firma debe coincidir con la firma que calcula Amazon S3; de lo contrario, Amazon S3 deniega la solicitud. Por ejemplo: ., 733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7 Los cálculos de firmas se describen en la siguiente sección.
X-Amz-Security-Token	Parámetro de credenciales opcional si se utilizan credenciales procedentes del servicio STS.

Cálculo de una firma

En el diagrama siguiente se ilustra el proceso de cálculo de firma.



En la siguiente tabla, se describen las funciones que se muestran en el diagrama. Debe implementar código para estas funciones.

Función	Descripción
Lowercase()	Convierta la cadena de caracteres en minúsculas.
Hex()	Codificación en minúsculas en base 16.

Función	Descripción
SHA256Hash()	Función de hash criptográfico del algoritmo de hash seguro (SHA).
HMAC-SHA256()	Calcula el HMAC mediante el algoritmo SHA256 con la clave de firma proporcionada. Esta es la firma final.
Trim()	Elimine cualquier espacio en blanco inicial o final.

Función	Descripción
UriEncode()	El URI codifica cada byte. UriEncode() debe aplicar las siguientes reglas: • El URI codifica todos los bytes excepto los caracteres no reservados: “A”-“Z”, “a”-“z”, “0”-“9”, “-”, “.”, “_” y “~”. • El carácter de espacio es un carácter reservado y debe codificarse como “%20” (y no como “+”). • Cada byte codificado en URI se encuentra formado por un “%” y el valor hexadecimal de dos dígitos del byte. • Las letras del valor hexadecimal deben estar en mayúsculas, por ejemplo, “%1A”. • Codifique el carácter de barra diagonal, “/”, en todas partes excepto en el nombre de la clave del objeto. Por ejemplo, si el nombre de la clave del objeto es photos/Jan/sample.jpg , la barra diagonal del nombre de la clave no se encuentra codificada.  Important Es posible que las funciones estándar de UriEncode que proporciona su plataforma de desarrollo no funcionen debido a las diferencias en la implementación y a la ambigüedad relacionada en las RFC subyacentes. Le recomendamos que escriba su propia función UriEncode personalizada para asegurarse de que la codificación funcione. A continuación, se muestra una función UriEncode() de ejemplo en Java. <pre data-bbox="617 1738 1412 1812">public static String UriEncode(CharSequence input, boolean encodeSlash) {</pre>

Función	Descripción
	<pre> 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>

Para obtener más información sobre el proceso de firma (detalles sobre la creación de una solicitud canónica, cadena para firmar y cálculos de firma), consulte [Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en un solo fragmento \(AWS Signature Version 4\)](#). Por lo general, el proceso es el mismo, excepto en que la creación de CanonicalRequest en una URL prefirmada difiere de la siguiente manera:

- No incluya un hash de carga útil en la Solicitud canónica, ya que cuando crea una URL prefirmada, no conoce el contenido de la carga útil porque la URL se usa para cargar una carga útil arbitraria. En su lugar, utilice una cadena constante UNSIGNED-PAYLOAD.
- La Cadena de consulta canónica debe incluir todos los parámetros de consulta de la tabla anterior excepto X-Amz-Signature.
- Para S3, debe incluir el parámetro de consulta X-Amz-Security-Token en la URL si utiliza credenciales procedentes del servicio STS.
- Los encabezados canónicos deben incluir el encabezado HTTP host. Si planea incluir alguno de los encabezados x-amz-*, también debe agregarlos para calcular la firma. Si lo desea, puede agregar todos los demás encabezados que planea incluir en la solicitud. Para mayor seguridad, debe firmar tantos encabezados como sea posible. Si agrega un encabezado firmado que también

es un parámetro de consulta firmado y su valor es diferente, recibirá un error `InvalidRequest` porque la entrada es contradictoria.

Un ejemplo

Supongamos que tiene un objeto `test.txt` en el bucket de `examplebucket`. Desea compartir este objeto con otras personas durante un periodo de 24 horas (86 400 segundos) mediante la creación de una URL prefirmada.

```
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>
```

Los pasos siguientes ilustran primero los cálculos de firma y, a continuación, la creación de la URL prefirmada. El ejemplo realiza las siguientes suposiciones adicionales:

- La marca temporal de la solicitud es `Fri, 24 May 2013 00:00:00 GMT`.
- El bucket está en la región Este de EE. UU. (Norte de Virginia) y la credencial `Scope` y los cálculos de `Signing Key` utilizan `us-east-1` como especificador de región. Para obtener más información, consulte [Regiones y puntos de enlace](#) en la Referencia general de AWS.

Puede utilizar este ejemplo como caso de prueba para comprobar la firma que calcula el código; sin embargo, debe utilizar el mismo nombre de bucket, la misma clave de objeto, la misma marca temporal y las siguientes credenciales de ejemplo:

Parámetro	Valor
<code>AWSSecretAccessKey</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
```

Ahora tiene toda la información para crear una URL prefirada. La URL resultante de este ejemplo se muestra de la siguiente manera (puede usarla para comparar la URL prefirada):

```
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=aeeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404

Ejemplo 2

El ejemplo siguiente (no relacionado con el ejemplo anterior) muestra una URL prefirmada con el parámetro X-Amz-Security-Token.

```
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request
&X-Amz-Date=20200524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEMv%2F%2F%2F%2F%2F%2F%2F%2F
%2FwEaCXVzLWVhc3QtMSJGMEQCIBSubvdj9YGs2g0HkHs0HFdkw0ozjARSKHL987NhhOC8AiBPepRU1obMvIbGU0T
%2BWphFPgK%2Fqpxaf5Snm5M57XFkCqlAgjz%2F%2F%2F%2F%2F%2F%2F%2F
%2F8BEAAaDDQ3Mjm4NTU0NDY2MCIM83pULBe5%2F
%2BNm1GZBKvkBVslSaJVgwSef7SsoZCJlfJ56weYL3QCWEGr2F4BmCZZyFpmWEYZWhnNK1AnHMj5nkfKLKBx30XAT5PZGVr
%2F3HHMOkpdanMXn%2B4PY8lvM8RgnzSu90jOUUpGXEOAo
%2F6G80qlMim3%2BZmaQmasn4VVRvESEd7072QGZ3%2BvDnDVnss0lSYjl v8PP7IujnvzhZRnj0WoeOyMe1lL0wTG
%2Fa9ush5he52w%2FYUJccOn00aZuyROuvSRV4Q70sbWQHUVUt%2B0tUMKzm8vsFOp4BaNZFqobbjt b36Y92v
%2Bx5ky6i0s8QE886jJtUWMP5ldMziClGxp0mN5dzsYlm3GyiJ
%2F01mwkPQDwg3mtSpOA9oeeuAMPTA7qmQy9RNuTKBDsx9EW27wvpZBUM3SJHEfxv48euaDKgrIX3Z79ruQFSQOc9LUrDJR
%2B4SoWAjqK%2BGX8Q3vPSjsLxhqhemWD6U4TXcm7ku3gxMbzbft8NDg%3D
&X-Amz-Signature=<signature-value>
```

Ejemplos: cálculos de firma en AWS Signature Version 4

Temas

- [Ejemplos de cálculos de firma mediante Java \(AWS Signature Version 4\)](#)
- [Ejemplos de cálculos de firma mediante C# \(AWS Signature Version 4\)](#)

Para las solicitudes autenticadas, a menos que utilice los AWS SDK, debe escribir código para calcular las firmas que brindan información de autenticación en las solicitudes. El cálculo de firmas en AWS Signature Version 4 (consulte [Autenticación de solicitudes \(AWS Signature Version 4\)](#)) puede ser una tarea compleja, por lo que recomendamos que se usen los AWS SDK siempre que sea posible.

En esta sección, se proporcionan ejemplos de cálculos de firmas escritos en Java y C#. Los ejemplos de código envían las siguientes solicitudes y utilizan el encabezado de HTTP Authorization para proporcionar información de autenticación:

- Objeto PUT: ejemplos independientes ilustran la carga de la carga útil completa una sola vez y la carga de la carga útil en fragmentos. Para obtener información sobre el uso del encabezado de Authorization para la autenticación, consulte [Autenticación de solicitudes: uso del encabezado de Authorization \(AWS Signature Version 4\)](#).
- Objeto GET: este ejemplo genera una URL prefirmada para obtener un objeto. Los parámetros de consulta proporcionan la firma y otra información de autenticación. Los usuarios pueden pegar una URL prefirmada en el navegador para recuperar el objeto o puede usar la URL para crear un enlace en el que se pueda hacer clic. Para obtener información acerca del uso de parámetros de consulta para autenticación, consulte [Autenticación de solicitudes: uso de parámetros de consulta \(AWS Signature Version 4\)](#).

En el resto de esta sección, se describen los ejemplos en Java y C#. Los temas incluyen instrucciones para descargar los ejemplos y ejecutarlos.

Ejemplos de cálculos de firma mediante Java (AWS Signature Version 4)

El ejemplo de Java que muestra el cálculo de firmas se puede descargar en <https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AWSS3SigV4JavaSamples.zip>. Tenga en cuenta que en estos ejemplos se usa el AWS SDK para Java v1 y se recomienda usar el [AWS SDK para Java v2](#) para aplicaciones nuevas. En `RunAllSamples.java`, la función `main()` ejecuta solicitudes de muestra para crear un objeto, recuperar un objeto y crear una URL prefirmada para el objeto. En el ejemplo se crea un objeto a partir de la cadena de texto proporcionada en el código:

```
PutS3ObjectSample.putS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
GetS3ObjectSample.getS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
PresignedUrlSample.getPresignedUrlToS3Object(bucketName, regionName, awsAccessKey,
awsSecretKey);
PutS3ObjectChunkedSample.putS3ObjectChunked(bucketName, regionName, awsAccessKey,
awsSecretKey);
```

Prueba de los ejemplos en un ordenador basado en Linux

Las instrucciones siguientes son para el sistema operativo Linux.

1. En un terminal, vaya al directorio que contiene `AWSS3SigV4JavaSamples.zip`.

2. Descomprima el archivo .zip.
3. En un editor de texto, abra el archivo `./com/amazonaws/services/s3/samples/RunAllSamples.java`. Actualice el código con la siguiente información:

- El nombre del bucket en el que se puede crear el nuevo objeto.

 Note

Los ejemplos utilizan una solicitud de tipo alojamiento virtual para acceder al bucket. Para evitar posibles errores, asegúrese de que el nombre del bucket cumpla con las reglas de nomenclatura de buckets, tal como se explica en [Restricciones y limitaciones de los buckets](#) de la Guía del usuario de Amazon Simple Storage Service.

- Región de AWS donde se encuentra el bucket.

Si el bucket está en la región Este de EE. UU. (Norte de Virginia), utilice `us-east-1` para especificar la región. Para ver una lista de otras Regiones de AWS, vaya a [Amazon Simple Storage Service \(S3\)](#), en Referencia general de AWS.

4. Compile el código de origen y guarde las clases compiladas en el directorio `bin/`.

```
javac -d bin -source 6 -verbose com
```

5. Cambie el directorio a `bin/` y, a continuación, ejecute `RunAllSamples`.

```
java com.amazonaws.services.s3.sample.RunAllSamples
```

El código ejecuta todos los métodos en `main()`. Para cada solicitud, el resultado mostrará la solicitud canónica, la cadena a firmar y la firma.

Ejemplos de cálculos de firma mediante C# (AWS Signature Version 4)

El ejemplo de C# que muestra el cálculo de firmas se puede descargar en https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip. Tenga en cuenta que en estos ejemplos se usa el AWS SDK para .Net v1 y se recomienda usar el [AWS SDK para .Net v3](#) para aplicaciones nuevas. En `Program.cs`, la función `main()` ejecuta solicitudes

de muestra para crear un objeto, recuperar un objeto y crear una URL prefirmada para el objeto. El código para el cálculo de la firma se encuentra en la carpeta \Signers.

```
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");
```

Prueba de los ejemplos con Microsoft Visual Studio 2010 o versiones posteriores

1. Descomprima el archivo .zip.
2. Inicie Visual Studio y, a continuación, abra el archivo .sln.
3. Actualice el archivo App.config con credenciales de seguridad válidas.
4. Actualice el código de la siguiente manera:
 - En Program.cs, proporcione el nombre del bucket y la Región de AWS donde se encuentra el bucket. El ejemplo crea un objeto en este bucket.
5. Ejecute el código.
6. Para comprobar que se creó el objeto, copie la URL prefirmada que crea el programa y péguela en una ventana del navegador.

Autenticación de solicitudes: cargas basadas en el navegador con POST (AWS Signature Version 4)

Amazon S3 admite las solicitudes HTTP POST para que los usuarios puedan cargar contenido directamente en Amazon S3. Con HTTP POST para cargar contenido se simplifican las cargas y se reduce la latencia de carga donde los usuarios cargan datos para almacenar en Amazon S3. En esta sección se describe cómo autenticar solicitudes HTTP POST. Para obtener más información sobre las solicitudes HTTP POST, cómo crear un formulario, crear una política POST y un ejemplo, consulte [Cargas basadas en el navegador con POST \(AWS Signature Version 4\)](#).

Para autenticar una solicitud HTTP POST, haga lo siguiente:

1. El formulario debe incluir los siguientes campos para proporcionar la firma y la información relevante que Amazon S3 puede utilizar para volver a calcular la firma al recibir la solicitud:

Nombre del elemento	Descripción
<code>policy</code>	La política de seguridad codificada en base64 que describe lo que se permite en la solicitud. Para el cálculo de la firma, esta política es la cadena que usted firma. Amazon S3 debe contar con esta política para poder volver a calcular la firma.
<code>x-amz-algorithm</code>	El algoritmo de firma usado. Para AWS Signature Version 4, el valor es <code>AWS4-HMAC-SHA256</code> .
<code>x-amz-credential</code>	Además del ID de clave de acceso, proporciona información sobre el alcance que utilizó para calcular la clave de firma para el cálculo de la firma. Es una cadena del formulario siguiente: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Por ejemplo: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .</pre> Para Amazon S3, la cadena <code>aws-service</code> es <code>s3</code>. Para obtener una lista de las cadenas de <code>aws-region</code> de Amazon S3, consulte Regiones y puntos de conexión en Referencia general de AWS.
<code>x-amz-date</code>	

Nombre del elemento	Descripción
	Es el valor de la fecha en formato ISO8601. Por ejemplo, 20130728T000000Z . Es la misma fecha que utilizó al crear la clave de firma. Este debe ser también el mismo valor que proporciona en la política (x-amz-date) que firmó.
x-amz-signature	(AWS Signature Version 4) El hash HMAC-SHA256 de la política de seguridad. Para obtener más información sobre opciones para la firma, consulte Agregar la firma a la solicitud HTTP en Referencia general de AWS.

2. La política POST debe incluir los siguientes elementos:

Nombre del elemento	Descripción
x-amz-algorithm	El algoritmo de firma que utilizó para calcular la firma. Para AWS Signature Version 4, el valor es AWS4-HMAC-SHA256 .
x-amz-credential	Además del ID de clave de acceso, proporciona información sobre el alcance que utilizó para calcular la clave de firma para el cálculo de la firma. Es una cadena del formulario siguiente: <i><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</i> Por ejemplo: AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .

Nombre del elemento	Descripción
x-amz-date	El valor de la fecha especificado en la cadena con formato ISO8601. Por ejemplo, "20130728T000000Z". La fecha debe ser la misma que utilizó al crear la clave de firma para el cálculo de la firma.

3. Para el cálculo de la firma, la política POST es la cadena a firmar.

Cálculo de una firma

En el diagrama siguiente se ilustra el proceso de cálculo de firma.



Cálculo de una firma

1. Cree una política con la codificación UTF-8.
2. Convierta la política codificada en UTF-8 a Base64. El resultado es la cadena que se va a firmar.
3. Cree la firma como un hash HMAC-SHA256 de la cadena que se va a firmar. Proporcionará la clave de firma como clave de la función hash.

4. Codifique la firma mediante la codificación hexadecimal.

Para obtener más información acerca de la creación de formularios HTML, políticas de seguridad y un ejemplo, consulte los siguientes subtemas:

- [Creación de un formulario HTML \(mediante AWS Signature Version 4\)](#)
- [Política POST](#)
- [Ejemplo: carga basada en el navegador con HTTP POST \(con AWS Signature Version 4\)](#)

Claves de política específica de autenticación de Signature Version 4 de Amazon S3

En la siguiente tabla se muestran las claves de política relacionadas con la autenticación de Signature Version 4 de Amazon S3 que pueden estar en las políticas de Amazon S3. En una política de bucket, puede agregar estas condiciones para imponer un comportamiento específico cuando las solicitudes se autentican mediante Signature Version 4. Para ver ejemplos de políticas, consulte [Ejemplos de políticas de bucket con claves de condición relacionadas con Signature Version 4](#).

Claves aplicables	Descripción
s3:signatureversion	Identifica la versión de AWS Signature que desea admitir con las solicitudes autenticadas. Para las solicitudes autenticadas, Amazon S3 admite Signature Version 4 y Signature Version 2. Puede agregar esta condición a la política de bucket para solicitar una versión de firma específica. Valores válidos: "AWS" identifica Signature Version 2 "AWS4-HMAC-SHA256" identifica Signature Version 4

Claves aplicables	Descripción
s3:authType	Amazon S3 admite varios métodos de autenticación (consulte Autenticación de solicitudes (AWS Signature Version 4)). Si lo desea, puede usar esta clave de condición para restringir las solicitudes entrantes para usar un método de autenticación específico. Por ejemplo, puede permitir solo el encabezado HTTP Authorization que se utilizará en la autenticación de solicitudes. Valores válidos: REST-HEADER REST-QUERY-STRING POST

Claves aplicables	Descripción
s3:signatureAge	El periodo, en milisegundos, que una firma es válida en una solicitud autenticada. Esta condición es válida para: • URL prefiradas: en las que gana la condición más restrictiva. Para obtener más información, consulte Trabajo con URL prefiradas. • POST prefirado: cargue archivos directamente a S3 mediante un POST prefirado. Para obtener más información, consulte Política POST de Amazon S3. En Signature Version 2, este valor siempre está establecido en 0. En Signature Version 4, la clave de firma es válida por un plazo máximo de siete días. Por lo tanto, las firmas también son válidas por un plazo máximo de siete días. Puede usar esta condición para limitar aún más la antigüedad de la firma. Para obtener más información, consulte Introducción a la firma de solicitudes. Ejemplo de valor: : 100

Claves aplicables	Descripción
s3:x-amz-content-sha256	Puede utilizar esta clave de condición para no permitir el contenido sin firmar en su bucket. Cuando se utiliza Signature Version 4, para las solicitudes que utilizan el encabezado <code>Authorization</code> , agregue el encabezado <code>x-amz-content-sha256</code> en el cálculo de la firma y luego establezca su valor en la carga de hash. Tenga en cuenta que esta clave de condición no admite el encabezado <code>x-amz-content-sha256</code> como parámetro de cadena de consulta. Puede usar esta clave de condición en la política de bucket para denegar cualquier carga en la que las cargas útiles no estén firmadas. Por ejemplo, puede denegar las cargas que usen el encabezado <code>Authorization</code> para autenticar las solicitudes, pero no firmar la carga útil. Para obtener más información, consulte Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en un solo fragmento (AWS Signature Version 4). Valor válido: UNSIGNED-PAYLOAD

Ejemplos de políticas de bucket con claves de condición relacionadas con Signature Version 4

La política de bucket siguiente deniega cualquier solicitud de URL prefirmada de Amazon S3 en objetos en `examplebucket` si la firma tiene más de 10 minutos.

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 siguiente política de bucket solo permite las solicitudes que utilizan el encabezado `Authorization` para solicitar autenticación. Se rechazará cualquier solicitud POST o de URL prefirmada.

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 política de bucket siguiente deniega las solicitudes que utilizan URL prefirmadas para la autenticación de solicitudes:

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 política de bucket siguiente deniega cualquier carga con cargas útiles sin firmar:

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"
      }
    }
  ]
}
```

Cargas basadas en el navegador con POST (AWS Signature Version 4)

En esta sección se explica cómo cargar archivos directamente en Amazon S3 a través de un navegador mediante solicitudes HTTP POST. También contiene información sobre cómo utilizar la biblioteca AWS Amplify JavaScript para cargar archivos basados en el navegador en Amazon S3.

Temas

- [POST Object](#)
- [POST Object restore](#)
- [Cargas basadas en el navegador con HTTP POST](#)
- [Cálculo de una firma](#)
- [Creación de un formulario HTML \(mediante AWS Signature Version 4\)](#)
- [Política POST](#)
- [Ejemplo: carga basada en el navegador con HTTP POST \(con AWS Signature Version 4\)](#)
- [Cargas basadas en el navegador en Amazon S3 mediante la biblioteca de AWS Amplify](#)

POST Object

Descripción

La operación POST agrega un objeto a un bucket específico mediante formularios HTML. POST es una forma alternativa de PUT que permite las cargas desde el navegador como una forma de colocar objetos en buckets. Los parámetros que se transfieren a PUT a través de encabezados HTTP se transfieren, en su lugar, como campos de formulario a POST en el cuerpo del mensaje codificado en varias partes o con datos del formulario. Para agregar un objeto a un bucket, debe tener el acceso WRITE en el bucket. Amazon S3 nunca almacena objetos parciales. Si recibe una respuesta correcta, puede estar seguro de que se ha almacenado todo el objeto.

Amazon S3 es un sistema distribuido. A menos que habilite el control de versiones para un bucket, si Amazon S3 recibe varias solicitudes de escritura para el mismo objeto simultáneamente, solo se almacena la última versión del objeto escrito.

Para asegurarse de que los datos no se corrompan al atravesar la red, utilice el campo de formulario Content-MD5. Cuando utiliza este campo de formulario, Amazon S3 compara el objeto con el valor MD5 proporcionado. Si no coinciden, Amazon S3 devuelve un error. Además, puede calcular el valor MD5 al publicar un objeto en Amazon S3 y comparar el valor ETag devuelto con el valor MD5 calculado. La ETag solo refleja solo los cambios en el contenido de un objeto, no en los metadatos.

Note

Para configurar la aplicación para que envíe los encabezados de las solicitudes antes de enviar el cuerpo de la solicitud, utilice el código de estado HTTP 100 (Continuar). Para las operaciones POST, el uso de este código de estado ayuda a evitar enviar el cuerpo del mensaje si el mensaje se rechaza en función de los encabezados (por ejemplo, debido a un error de autenticación o a una redirección). Para obtener más información sobre el código de estado HTTP 100 (Continuar), consulte la sección 8.2.3 de <http://www.ietf.org/rfc/rfc2616.txt>.

Amazon S3 cifra automáticamente todos los objetos nuevos que se cargan a un bucket de S3. La configuración de cifrado de un objeto cargado depende de la configuración de cifrado predeterminada del bucket de destino. De forma predeterminada, todos los buckets tienen una configuración de cifrado predeterminada que utiliza el cifrado del servidor con claves administradas de Amazon S3 (SSE-S3).

Si el bucket de destino tiene una configuración de cifrado que utiliza el cifrado del servidor con una clave de AWS Key Management Service (AWS KMS) (SSE-KMS), cifrado del servidor de doble capa con una clave de AWS KMS (DSSE-KMS) o una clave de cifrado proporcionada por el cliente (SSE-C), Amazon S3 utiliza la clave de KMS correspondiente o una clave proporcionada por el cliente para cifrar el objeto cargado. Al cargar un objeto, si desea cambiar la configuración de cifrado del objeto cargado, puede especificar el tipo de cifrado del servidor. Puede configurar SSE-S3, SSE-KMS, DSSE-KMS o SSE-C. Para obtener más información, consulte [Protección de datos mediante el cifrado del servidor](#) en la Guía del usuario de Amazon Simple Storage Service.

Important

Al crear la solicitud, asegúrese de que el campo `file` sea el último campo del formulario.

Control de versiones

Si habilita el control de versiones para un bucket, POST genera automáticamente un ID de versión único para el objeto que se agrega. Amazon S3 devuelve este ID en la respuesta mediante el encabezado de respuesta `x-amz-version-id`.

Si suspende el control de versiones de un bucket, Amazon S3 siempre utiliza `null` como ID de versión del objeto almacenado en un bucket.

Para obtener más información sobre cómo devolver el estado de control de versiones de un bucket, consulte [GetBucketVersioning](#).

Amazon S3 es un sistema distribuido. Si habilita el control de versiones para un bucket y Amazon S3 recibe varias solicitudes de escritura para el mismo objeto simultáneamente, se almacenan todas las versiones del objeto.

Para ver ejemplos de solicitudes que utilizan el control de versiones, consulte [Solicitud de ejemplo](#).

Solicitudes

Sintaxis

```
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--
```

Parámetros de la solicitud

Esta aplicación de la operación no utiliza parámetros de solicitud.

Campos de formulario

Esta operación puede utilizar los siguientes campos de formulario.

Nombre	Descripción	Obligator io
AWSAccessKeyId	El ID de clave de acceso de AWS del propietario del bucket que otorga un acceso de usuario anónimo para una solicitud que cumple con el conjunto de restricciones en la política. Tipo: cadena Valor predeterminado: None Restricciones: obligatorio si se incluye un documento de política con la solicitud.	Condicion al
acl	La lista de control de acceso (ACL) de Amazon S3 especificada. Si la ACL especificada no es válida, se genera un error. Para obtener más información sobre ACL, consulte Información general de la lista de control de acceso (ACL) en la Guía del usuario de Amazon Simple Storage Service.	No

Nombre	Descripción	Obligatorio
	Tipo: cadena Valor predeterminado: privado Valores válidos: 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	Los encabezados específicos de REST. Para obtener más información, consulte PutObject. Tipo: cadena Valor predeterminado: None	No
file	El contenido de texto o archivo. El contenido de texto o archivo debe ser el último campo en el formulario. No puede cargar más de un archivo a la vez. Tipo: el contenido de texto o archivo Valor predeterminado: none	Sí

Nombre	Descripción	Obligatorio
key	El nombre de la clave cargada. Para utilizar el nombre de archivo provisto por el usuario, utilice la variable <code>\${filename}</code> . Por ejemplo, si una usuaria llamada Mary carga el archivo <code>example.jpg</code> y se especifica <code>/user/mary/\${filename}</code> , el nombre de la clave es <code>/user/mary/example.jpg</code> . Para obtener más información, consulte Clave de objeto y metadatos en la Guía del usuario de Amazon Simple Storage Service. Tipo: cadena Valor predeterminado: None	Sí
policy	La política de seguridad que describe lo que se permite en la solicitud. Las solicitudes sin una política de seguridad se consideran anónimas y funcionan solo en buckets que se pueden escribir públicamente. Para obtener más información, consulte Formularios HTML y Ejemplos de carga en la Guía del usuario de Amazon Simple Storage Service. Tipo: cadena Valor predeterminado: None Restricciones: se requiere una política de seguridad si el bucket no se puede escribir públicamente.	Condicional

Nombre	Descripción	Obligatorio
<code>success_action_redirect</code> , <code>redirect</code>	La URL a la que se redirige al cliente después de una carga exitosa. Si no se especifica el campo <code>success_action_redirect</code> , Amazon S3 devuelve el tipo de documento vacío especificado en el campo <code>success_action_status</code> . Si Amazon S3 no puede interpretar la URL, actúa como si el campo no estuviera presente. Si la carga falla, Amazon S3 muestra un error y no redirige al usuario a una URL. Tipo: cadena Valor predeterminado: None  Note El nombre de campo de <code>redirect</code> está obsoleto y el soporte para el nombre de campo de <code>redirect</code> se eliminará en el futuro.	No

Nombre	Descripción	Obligatorio
success_action_status	Si no especifica <code>success_action_redirect</code> , el código de estado se devuelve al cliente cuando la carga se realiza correctamente. Este campo acepta los valores 200, 201 o 204 (el valor predeterminado). Si el valor se establece en 200 o 204, Amazon S3 devuelve un documento vacío con un código de estado 200 o 204. Si el valor se establece en 201, Amazon S3 devuelve un documento XML con un código de estado 201. Si el valor no se establece o si se establece en un valor no válido, Amazon S3 devuelve un documento vacío con un código de estado 204. Tipo: cadena Valor predeterminado: None	No

Nombre	Descripción	Obligatorio
tagging	El conjunto de etiquetas especificado que se agregará al objeto. Para agregar etiquetas, utilice el siguiente esquema de codificación. <pre><Tagging> <TagSet> <Tag> <Key>TagName</Key> <Value>TagValue</Value> </Tag> ... </TagSet> </Tagging></pre> Para obtener más información, consulte Etiquetas o de objetos en la Guía del usuario de Amazon Simple Storage Service. Tipo: cadena Valor predeterminado: None	No

Nombre	Descripción	Obligatorio
<code>x-amz-storage-class</code>	La clase de almacenamiento que se usa para almacenar el objeto. Si no especifica una clase, Amazon S3 utiliza la clase de almacenamiento predeterminada, STANDARD. Amazon S3 admite otras clases de almacenamiento. Para obtener más información, consulte Clases de almacenamiento en la Guía del usuario de Amazon Simple Storage Service. Tipo: cadena Valor predeterminado: STANDARD Valores válidos: REDUCED_REDUNDANCY EXPRESS_ONEZONE DEEP_ARCHIVE GLACIER GLACIER_IR INTELLIGENT_TIERING ONEZONE_IA STANDARD STANDARD_IA	No
<code>x-amz-meta-*</code>	Los encabezados que comienzan con este prefijo son metadatos definidos por el usuario. Cada uno de ellos se almacena y se devuelve como un conjunto de pares clave-valor. Amazon S3 no valida ni interpreta metadatos definidos por el usuario. Para obtener más información, consulte PutObject. Tipo: cadena Valor predeterminado: None	No

Nombre	Descripción	Obligatorio
x-amz-security-token	El token de seguridad de Amazon DevPay. Cada solicitud que utiliza Amazon DevPay requiere dos campos de formulario x-amz-security-token : uno para el token de producto y otro para el token de usuario. Tipo: cadena Valor predeterminado: None	No
x-amz-signature	(AWS Signature Version 4) El hash HMAC-SHA256 de la política de seguridad. Tipo: cadena Valor predeterminado: None	Condicional

Nombre	Descripción	Obligatorio
<code>x-amz-website-redirect-location</code>	Si el bucket está configurado como un sitio web, este campo redirige las solicitudes para este objeto a otro objeto en el mismo bucket o a una URL externa. Amazon S3 almacena el valor de este encabezado en los metadatos del objeto. Para obtener más información sobre los metadatos del objeto, consulte Clave de objeto y metadatos en la Guía del usuario de Amazon Simple Storage Service. En el siguiente ejemplo, el encabezado de la solicitud establece la redirección a un objeto (<code>anotherPage.html</code>) del mismo bucket: <pre>x-amz-website-redirect-location: /anotherPage.html</pre> En el siguiente ejemplo, el encabezado de la solicitud establece la redirección del objeto a otro sitio web: <pre>x-amz-website-redirect-location: http://www.example.com/</pre> Para obtener más información sobre el alojamiento de sitios web en Amazon S3, consulte Alojamiento de sitios web en Amazon S3 y Cómo configurar las redirecciones de página de sitio web en la Guía del usuario de Amazon Simple Storage Service. Tipo: cadena Valor predeterminado: None	No

Nombre	Descripción	Obligatorio
	Restricciones: el valor debe ir precedido de /, http:// o https://. La longitud del valor está limitada a 2 KB.	

Campos del formulario de solicitud de suma de comprobación adicionales

Al cargar un objeto, puede especificar varias sumas de comprobación que desee usar para verificar la integridad de los datos. Puede especificar un algoritmo de suma de comprobación adicional para que lo use Amazon S3. Para obtener más información sobre los valores de suma de comprobación adicionales, consulte [Comprobación de la integridad de los objetos](#) en la Guía del usuario de Amazon Simple Storage Service.

Nombre	Descripción	Obligatorio
x-amz-checksum-algorithm	Indica el algoritmo utilizado para crear la suma de comprobación del objeto. Si se especifica un valor, debe incluir el encabezado de la suma de comprobación correspondiente. De lo contrario, la solicitud generará un error 400. Entre los valores posibles se incluyen: CRC32, CRC32C, SHA1 y SHA256.	No
x-amz-checksum-crc32	Especifica la suma de comprobación CRC32 de 32 bits codificada en base64 del objeto. Este parámetro es obligatorio si el valor de x-amz-checksum-algorithm es CRC32.	Condicional
x-amz-checksum-crc32c	Especifica la suma de comprobación CRC32C de 32 bits codificada en base64 del objeto. Este parámetro es obligatorio si el valor de x-amz-checksum-algorithm es CRC32C.	Condicional

Nombre	Descripción	Obligatorio
x-amz-checksum-sha1	Especifica el resumen SHA-1 de 160 bits codificado en base64 del objeto. Este parámetro es obligatorio si el valor de x-amz-checksum-algorithm es SHA1.	Condicional
x-amz-checksum-sha256	Especifica el resumen SHA-256 de 256 bits codificado en base64 del objeto. Este parámetro es obligatorio si el valor de x-amz-checksum-algorithm es SHA256.	Condicional

Campos del formulario de solicitud específicos del cifrado del servidor

El cifrado del servidor es cifrado de datos en reposo. Amazon S3 cifra los datos mientras los escribe en los discos en los centros de datos de AWS y los descifra cuando accede a ellos. Al cargar un objeto, puede especificar el tipo de cifrado del servidor que desea que Amazon S3 use para cifrar el objeto.

Hay cuatro tipos de cifrado del servidor:

- Cifrado del servidor con claves administradas por Amazon S3 (SSE-S3): desde mayo de 2022, todos los buckets de Amazon S3 tienen el cifrado configurado de forma predeterminada. La opción predeterminada para el cifrado del servidor es con SSE-S3. Cada objeto se cifra con una clave única. Como medida de seguridad adicional, SSE-S3 cifra la propia clave con una clave raíz que cambia periódicamente. SSE-S3 utiliza uno de los cifrados de bloques más seguros disponibles, Advanced Encryption Standard de 256 bits (AES-256), para cifrar sus datos.
- Cifrado del servidor con claves de AWS KMS (SSE-KMS): SSE-KMS se proporciona a través de una integración del servicio de AWS KMS con Amazon S3. Con AWS KMS, tiene más control sobre sus claves. Por ejemplo, puede ver claves distintas, editar las políticas de control y seguir las claves en AWS CloudTrail. Además, puede crear y gestionar claves administradas por el cliente o utilizar Claves administradas por AWS que sean únicas para usted, su servicio y su región.
- Cifrado del servidor de doble capa con claves de AWS KMS (DSSE-KMS): el cifrado del servidor de doble capa con AWS KMS keys (DSSE-KMS) es similar a SSE-KMS, pero aplica dos capas individuales de cifrado por objeto en lugar de una.

- Cifrado del servidor con claves proporcionadas por el cliente (SSE-C): con SSE-C, usted administra las claves de cifrado y Amazon S3 administra el cifrado, al escribir en los discos, y el descifrado, cuando accede a los objetos.

 Note

Si tiene bloqueado el cifrado del servidor con claves proporcionadas por el cliente (SSE-C) para el bucket de uso general, recibirá un error HTTP 403 Acceso denegado cuando especifique los encabezados de solicitud de SSE-C al escribir nuevos datos en el bucket. Para obtener más información, consulte [Bloqueo o desbloqueo de SSE-C para un bucket de uso general](#).

Para obtener más información, consulte [Protección de los datos con el cifrado del servidor](#) en la Guía del usuario de Amazon Simple Storage Service.

En función del tipo de cifrado del servidor que desee usar, especifique los siguientes campos del formulario.

- Use SSE-S3, SSE-KMS o DSSE-KMS: si desea usar estos tipos de cifrado del servidor, especifique los siguientes campos del formulario en la solicitud.

Nombre	Descripción	Obligatorio
<code>x-amz-server-side-encryption</code>	Especifica el algoritmo de cifrado del servidor que se usará cuando Amazon S3 cree un objeto. Para usar SSE-S3, especifique <code>AES256</code>. Para usar SSE-KMS, especifique <code>aws:kms</code>. Para usar DSSE-KMS, especifique <code>aws:kms:dsse</code>. Tipo: cadena Valor válido: <code>aws:kms</code>, <code>AES256</code>, <code>aws:kms:dsse</code>	Sí
<code>x-amz-server-side-encryption</code>	Si el encabezado <code>x-amz-server-side-encryption</code> tiene un valor válido de <code>aws:kms</code> o <code>aws:kms:dsse</code> ,	Sí, si el valor de x-

Nombre	Descripción	Obligatorio
<code>n-aws-kms-key-id</code>	este encabezado especifica el ID de AWS KMS key que se utilizó para cifrar el objeto. Tipo: cadena	<code>amz-server-side-encryption</code> es <code>aws:kms</code> o <code>aws:kms:dsse</code>
<code>x-amz-server-side-encryption-context</code>	Si <code>x-amz-server-side-encryption</code> tiene un valor válido de <code>aws:kms</code> o <code>aws:kms:dsse</code>, este encabezado especifica el contexto de cifrado del objeto. El valor de este encabezado es una cadena UTF-8 codificada en base64 que contiene pares clave-valor con formato JSON para el contexto de cifrado. Tipo: cadena	No
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	Si <code>x-amz-server-side-encryption</code> tiene un valor válido de <code>aws:kms</code> o <code>aws:kms:dsse</code>, este encabezado especifica si Amazon S3 debe usar una clave de bucket de S3 con SSE-KMS o DSSE-KMS. Si se establece este encabezado en <code>true</code>, Amazon S3 utilizará una clave de bucket de S3 para el cifrado de objetos con SSE-KMS o DSSE-KMS. Tipo: booleano	No

 Note

Si especifica `x-amz-server-side-encryption:aws:kms` o `x-amz-server-side-encryption:aws:kms:dsse`, pero no proporciona `x-amz-server-side-`

`encryption-aws-kms-key-id`, Amazon S3 utiliza la Clave administrada de AWS (`aws/S3`) para proteger los datos.

- Utilice SSE-C: si desea administrar sus propias claves de cifrado, debe proporcionar todos los campos del formulario siguientes en la solicitud.

 Note

Si usa SSE-C, el valor de ETag que Amazon S3 devuelve en la respuesta no es el MD5 del objeto.

Nombre	Descripción	Obligatorio
<code>x-amz-server-side-encryption-customer-algorithm</code>	Especifica el algoritmo que se utilizará al cifrar el objeto. Tipo: cadena Valor predeterminado: None Valor válido: AES256 Restricciones: debe ir acompañado de campos <code>x-amz-server-side-encryption-customer-key</code> y <code>x-amz-server-side-encryption-customer-key-MD5</code> válidos.	Sí
<code>x-amz-server-side-encryption-customer-key</code>	Especifica la clave de cifrado proporcionada por el cliente con codificación en base64 para Amazon S3 para usar en el cifrado de datos. Este valor se utiliza para almacenar el objeto y, a continuación, se descarta. Amazon no almacena la clave de cifrado. La clave debe ser adecuada para su uso con el algoritmo especificado en el encabezado <code>x-amz-server-side-encryption-customer-algorithm</code>. Tipo: cadena	Sí

Nombre	Descripción	Obligatorio
	Valor predeterminado: None Restricciones: debe ir acompañado de campos <code>x-amz-server-side-encryption-customer-algorithm</code> y <code>x-amz-server-side-encryption-customer-key-MD5</code> válidos.	
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Especifica el resumen MD5 de 128 bits con codificación en base64 de la clave de cifrado, según RFC 1321. Amazon S3 usa este encabezado para comprobar la integridad del mensaje y garantizar que la clave de cifrado se haya transmitido sin errores. Tipo: cadena Valor predeterminado: None Restricciones: debe ir acompañado de campos <code>x-amz-server-side-encryption-customer-algorithm</code> y <code>x-amz-server-side-encryption-customer-key</code> válidos.	Sí

Respuestas

Encabezados de respuesta

Esta implementación de la operación puede incluir los encabezados de respuesta siguientes además de los encabezados de respuesta comunes a todas las respuestas. Para obtener más información, consulte [Encabezados de respuesta comunes](#).

Nombre	Descripción
<code>x-amz-checksum-crc32</code>	La suma de comprobación CRC32 de 32 bits codificada en base64 del objeto.

Nombre	Descripción
	Tipo: cadena
x-amz-checksum-crc32c	La suma de comprobación CRC32C de 32 bits codificada en base64 del objeto. Tipo: cadena
x-amz-checksum-sha1	El resumen SHA-1 de 160 bits codificado en base64 del objeto. Tipo: cadena
x-amz-checksum-sha256	El resumen SHA-256 de 256 bits codificado en base64 del objeto. Tipo: cadena
x-amz-expiration	Si se configura una acción <code>Expiration</code> para el objeto como parte de la configuración del ciclo de vida del bucket, Amazon S3 devuelve este encabezado. El valor del encabezado incluye un componente <code>expiry-date</code> y un componente <code>rule-id</code> codificado en URL. En el caso de los buckets con versiones habilitadas, este encabezado solo se aplica a las versiones actuales. Amazon S3 no proporciona un encabezado que indique cuándo una versión no actual es apta para su eliminación permanente. Para obtener más información, consulte PutBucketLifecycleConfiguration. Tipo: cadena
success_action_redirect, redirect	La URL a la que se redirige al cliente en una carga exitosa. Tipo: cadena Antecesor: <code>PostResponse</code>

Nombre	Descripción
<code>x-amz-server-side-encryption</code>	Algoritmo de cifrado del servidor que se utilizaba al almacenar este objeto en Amazon S3 (por ejemplo, AES256, <code>aws:kms</code>, <code>aws:kms:dsse</code>). Tipo: cadena
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	Si el encabezado <code>x-amz-server-side-encryption</code> tiene un valor válido de <code>aws:kms</code>, este encabezado especifica el ID de la clave de KMS que se utilizó para cifrar el objeto. Tipo: cadena
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	Si <code>x-amz-server-side-encryption</code> tiene un valor válido de <code>aws:kms</code>, este encabezado indica si el objeto está cifrado con SSE-KMS mediante una clave de bucket de S3. Si este encabezado se establece en <code>true</code>, el objeto usa una clave de bucket de S3 con SSE-KMS. Tipo: booleano
<code>x-amz-server-side-encryption-customer-algorithm</code>	Si se solicitó SSE-C, la respuesta incluye este encabezado, que confirma el algoritmo de cifrado que se utilizó. Tipo: cadena Valores válidos: AES256
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Si se solicitó el SSE-C, la respuesta incluye este encabezado para verificar la integridad del mensaje de ida y vuelta de la clave de cifrado proporcionada por el cliente. Tipo: cadena

Nombre	Descripción
<code>x-amz-version-id</code>	Versión del objeto. Tipo: cadena

Elementos de respuesta

Nombre	Descripción
Bucket	El nombre del bucket en el que se almacenaba el objeto. Tipo: cadena Antecesor: PostResponse
ETag	La etiqueta de entidad (ETag) es un hash MD5 del objeto que se puede utilizar para realizar operaciones GET condicionales mediante el uso de la etiqueta de solicitud <code>If-Modified</code> con la operación de solicitud GET. ETag refleja solo los cambios en el contenido de un objeto, no en sus metadatos. Tipo: cadena Antecesor: PostResponse
Key	El nombre de la clave del objeto. Tipo: cadena Antecesor: PostResponse
Location	El URI del objeto. Tipo: cadena Antecesor: PostResponse

Errores especiales

Esta implementación de la operación no devuelve errores especiales. Para obtener información sobre los errores de Amazon S3 y una lista de códigos de error, consulte [Respuestas de error](#).

Ejemplos

Solicitud de ejemplo

```
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

Respuesta de ejemplo con el control de versiones suspendido

A continuación, se muestra una respuesta de ejemplo cuando el control de versiones del bucket está suspendido:

```
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
```

En esta respuesta, el ID de la versión es null.

Respuesta de ejemplo con el control de versiones habilitado

A continuación, se muestra una respuesta de ejemplo cuando el control de versiones del bucket está habilitado.

```
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
```

Activos relacionados

- [CopyObject](#)
- [POST Object](#)
- [GetObject](#)

POST Object restore

Descripción

Esta operación realiza los siguientes tipos de solicitudes:

- `select`: realización de una consulta Select en un objeto archivado
- `restore an archive`: restauración de un objeto archivado

Para usar esta operación, debe tener permisos para realizar las acciones `s3:RestoreObject` y `s3:GetObject`. El propietario del bucket tiene este permiso de forma predeterminada y puede conceder este permiso a otros. Para obtener más información sobre permisos, consulte [Permisos relacionados con operaciones de subrecursos de bucket](#) y [Administración de permisos de acceso a los recursos de Amazon S3](#) en la Guía del usuario de Amazon Simple Storage Service.

Consulta de archivos con solicitudes Select

Puede utilizar un tipo de solicitud Select para realizar consultas SQL sobre objetos archivados. Los objetos archivados que se consultan mediante solicitud Select deben tener un formato de archivos de valores separados por comas (CSV) sin comprimir. Puede ejecutar consultas y análisis personalizados de los datos archivados sin necesidad de restaurar los datos en un nivel de mayor actividad de Amazon S3. Para obtener información general sobre las solicitudes Select, consulte [Consulta de objetos archivados](#) en la Guía del usuario de Amazon Simple Storage Service.

Al realizar una solicitud Select, realice lo siguiente:

- Defina una ubicación de salida para la salida de la consulta Select. Esta debe ser un bucket de Amazon S3 en la misma Región de AWS que el bucket que contiene el objeto de archivo que se está consultando. La Cuenta de AWS que inicia el trabajo debe tener permisos para escribir en el bucket de S3. Puede especificar la clase de almacenamiento y el cifrado de los objetos de salida almacenados en el bucket. Para obtener más información sobre la salida, consulte [Consulta de objetos archivados](#) en la Guía del usuario de Amazon Simple Storage Service.

Para obtener más información acerca de la estructura de S3 del cuerpo de la solicitud, consulte lo siguiente:

- [PutObject](#)
- [Administración de acceso con ACL](#) en la Guía del usuario de Amazon Simple Storage Service

- [protección de los datos con el cifrado del servidor](#) en la Guía del usuario de Amazon Simple Storage Service
- Defina la expresión SQL para el tipo de restauración SELECT de la consulta en la estructura `SelectParameters` del cuerpo de la solicitud. Puede utilizar expresiones como las siguientes.
- La expresión siguiente devuelve todos los registros del objeto especificado.

```
SELECT * FROM Object
```

- Si no está utilizando encabezados para los datos almacenados en el objeto, puede especificar columnas con encabezados posicionales.

```
SELECT s._1, s._2 FROM Object s WHERE s._3 > 100
```

- Si tiene encabezados y establece `fileHeaderInfo` en la estructura de CSV en el cuerpo de la solicitud como USE, puede especificar encabezados en la consulta. (Si establece el campo `fileHeaderInfo` en IGNORE, la primera fila se omite en la consulta). No se pueden mezclar las posiciones ordinales con los nombres de columna de los encabezados.

```
SELECT s.Id, s.FirstName, s.SSN FROM S3Object s
```

Para obtener más información acerca del uso de SQL con la restauración de Amazon Glacier Select, consulte [Referencia de SQL para Amazon S3 Select y Amazon Glacier Select](#) en la Guía del usuario de Amazon Simple Storage Service.

Al realizar una solicitud Select, también puede hacer lo siguiente:

- Para acelerar las consultas, especifique el nivel `Expedited`. Para obtener más información acerca de los niveles, consulte “Restauración de archivos” más adelante en este tema.
- Especifique detalles sobre el formato de serialización de los datos del objeto de entrada que se consulta y la serialización de los resultados de la consulta codificados como CSV.

A continuación, se muestran datos adicionales importantes sobre la característica de selección:

- Los resultados de salida son objetos nuevos de Amazon S3. A diferencia de las recuperaciones de archivos, se almacenan hasta que se eliminan de forma explícita, ya sea de forma manual o mediante una política de ciclo de vida.

- Puede emitir más de una solicitud Select en el mismo objeto de Amazon S3. Amazon S3 no deduplica las solicitudes, así que evite emitir solicitudes duplicadas.
- Amazon S3 acepta una solicitud Select incluso si el objeto ya se ha restaurado. Una solicitud Select no devuelve una respuesta de error 409.

Restauración de archivos

Los objetos en las clases de almacenamiento GLACIER y DEEP_ARCHIVE se archivan. Para obtener acceso a un objeto archivado, primero debe iniciar una solicitud de restauración. Esto restaura una copia temporal del objeto archivado. En una solicitud de restauración, se debe especificar el número de días durante los que se quiere que exista la copia restaurada. Después del periodo especificado, Amazon S3 elimina la copia temporal pero el objeto permanece archivado en la clase de almacenamiento GLACIER o DEEP_ARCHIVE desde la que se restauró el objeto.

Para restaurar una versión específica del objeto, puede proporcionar un ID de versión. Si no proporciona un ID de versión, Amazon S3 restaura la versión actual.

El tiempo que tardan los trabajos de restauración en finalizar depende de la clase de almacenamiento desde la que se esté restaurando el objeto y del nivel de acceso a los datos que especifique.

Al restaurar un objeto archivado (o utilizar una solicitud Select), puede especificar una de las siguientes opciones de nivel de acceso a datos en el elemento `Tier` del cuerpo de la solicitud:

- **Expedited** - las recuperaciones rápidas le permiten obtener acceso rápidamente a sus datos almacenados en la clase de almacenamiento GLACIER cuando son necesarias solicitudes urgentes ocasionales para un subconjunto de archivos. En todos los casos salvo para los objetos archivados de mayor tamaño (más de 250 MB), los datos a los que se obtiene acceso con solicitudes rápidas suelen estar disponibles en entre 1 y 5 minutos. La capacidad aprovisionada garantiza que la capacidad que necesitan las recuperaciones Expedited estará disponible cuando lo necesite. Las recuperaciones rápidas y la capacidad aprovisionada no están disponibles para la clase de almacenamiento DEEP_ARCHIVE.
- **Standard** - las recuperaciones estándares le permiten acceder a sus objetos archivados en un plazo de varias horas. Esta es la opción predeterminada para las solicitudes de recuperación de GLACIER y DEEP_ARCHIVE que no especifican la opción de recuperación. Las recuperaciones estándar suelen completarse en un plazo de 3 a 5 horas en el caso de la clase de almacenamiento GLACIER y, en el caso de la clase de almacenamiento DEEP_ARCHIVE, en un plazo de 12 horas.

- **Bulk:** las recuperaciones en bloque son la opción de recuperación menos costosa de Amazon Glacier, lo que permite recuperar grandes cantidades de datos, incluso petabytes, de forma económica en un día. Las recuperaciones en bloque suelen completarse en un plazo de 5 a 12 horas en el caso de la clase de almacenamiento GLACIER y, en el caso de la clase de almacenamiento DEEP_ARCHIVE, en un plazo de 48 horas.

Para obtener más información acerca de las opciones de recuperación de archivos y capacidad aprovisionada para acceso a datos Expedited, consulte [Restauración de objetos archivados](#) en la Guía del usuario de Amazon Simple Storage Service.

Puede usar la actualización de la velocidad de restauración de Amazon S3 para cambiar la velocidad de restauración a una velocidad más rápida mientras está en curso. Para actualizar la velocidad de una restauración en curso, se emite otra solicitud de restauración para el mismo objeto y se establece un nuevo elemento de solicitud Tier. Al emitir una solicitud para actualizar la capa de restauración, debe elegir una capa más rápida que la capa de la restauración en curso que está usando. No debe cambiar ningún otro parámetro (por ejemplo: el elemento de solicitud Days). Para obtener más información, consulte [Actualización de la velocidad de una restauración en curso](#) en la Guía del usuario de Amazon Simple Storage Service.

Para obtener el estado de la restauración de un objeto, puede enviar una solicitud HEAD. Las operaciones devuelven el encabezado x-amz-restore, que proporciona información sobre el estado de la restauración, en la respuesta. Puede usar las notificaciones de eventos de Amazon S3 para notificar cuando se inicie o complete una restauración. Para obtener más información, consulte [Configuración de notificaciones de eventos de Amazon S3](#) en la Guía del usuario de Amazon Simple Storage Service.

Tras restaurar un objeto archivado, puede actualizar el periodo de restauración volviendo a emitir la solicitud con un nuevo periodo. Amazon S3 actualiza el periodo de restauración en relación con el momento actual y solo cobra por la solicitud; no hay cargos por transferencia de datos. No puede actualizar el periodo de restauración cuando Amazon S3 esté procesando activamente la solicitud de restauración actual para el objeto.

Si el bucket tiene una configuración de ciclo de vida con una regla que incluye una acción de caducidad, la caducidad del objeto invalida la vida útil que especifique en la solicitud de restauración. Por ejemplo, si restaura una copia de un objeto durante 10 días, pero está previsto que el objeto caduque en 3 días, Amazon S3 elimina el objeto en 3 días. Para obtener más información sobre la configuración del ciclo de vida, consulte [PutBucketLifecycleConfiguration](#) y [Administración del ciclo de vida del objeto](#) en la Guía del usuario de Amazon Simple Storage Service.

Solicitudes

Sintaxis

```
POST /ObjectName?restore&versionId=VersionID HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Autenticación de solicitudes (AWS Signature Version 4))
Content-MD5: MD5

request body
```

 **Note**

La sintaxis muestra algunos de los encabezados de las solicitudes. Para obtener una lista completa, consulte “Encabezados de solicitudes”, más adelante en este tema.

Parámetros de la solicitud

Esta aplicación de la operación no utiliza parámetros de solicitud.

Encabezados de solicitud

Nombre	Descripción	Obligatorio
Content-MD5	Resumen MD5 de 128 bits de los datos cifrados en base64. Debe utilizar este encabezado como comprobación de la integridad del mensaje para comprobar que el cuerpo de la solicitud no se dañó durante el tránsito. Para obtener más información, consulte RFC 1864. Tipo: cadena Valor predeterminado: None	Sí

Elementos de la solicitud

A continuación, se muestra un ejemplo de XML del cuerpo de una solicitud para restaurar un archivo.

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Bulk</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

En la tabla siguiente, se explica el XML para la restauración del archivo en el cuerpo de la solicitud.

Nombre	Descripción	Obligatorio
RestoreRequest	Contenedor para restaurar información. Tipo: Contenedor	Sí
Days	Duración de la copia restaurada (activa). El número mínimo de días para restaurar un objeto de Amazon Glacier es de 1. Una vez que la copia del objeto alcanza la vida útil especificada, Amazon S3 la elimina del bucket. Si va a restaurar un archivo, este elemento es obligatorio. No utilice este elemento con un tipo de solicitud SELECT. Tipo: entero positivo Antecedentes: RestoreRequest	Sí, si se restaura un archivo
GlacierJobParameters	Contenedor para los parámetros de trabajo de clase de almacenamiento de S3 Glacier. No utilice este elemento con un tipo de solicitud SELECT. Tipo: Contenedor Antecedentes: RestoreRequest	No

Nombre	Descripción	Obligatorio
Tier	El nivel de acceso a los datos que se utilizará al restaurar el archivo. Standard es el predeterminado. Tipo: enumeración Valores válidos: Expedited Standard Bulk Antecesor: GlacierJobParameters	No

El siguiente XML es el cuerpo de la solicitud para una consulta de selección en un objeto archivado:

```
<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>
        <DisplayNmae>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>
    <StorageClass>Storage class</StorageClass>
</S3>
</OutputLocation>
</RestoreRequest>

```

En las tablas siguientes, se explica el XML para un tipo de restauración SELECT en el cuerpo de la solicitud.

Nombre	Descripción	Obligatorio
RestoreRequest	Contenedor para restaurar información. Tipo: Contenedor	Sí
Tier	El nivel de acceso a los datos que se utilizará al restaurar el archivo. Standard es el predeterminado. Tipo: enumeración Valores válidos: Expedited Standard Bulk Antecedentes: RestoreRequest	No
Description	La descripción opcional de la solicitud. Tipo: cadena Antecedentes: RestoreRequest	No
SelectParameters	Describe los parámetros de la solicitud del trabajo de selección. Tipo: Contenedor Antecedentes: RestoreRequest	Sí, si el tipo de solicitud es SELECT
OutputLocation	Describe la ubicación que recibe los resultados de la solicitud de restauración de selección. Tipo: Contenedor para Amazon S3 Antecedentes: RestoreRequest	Sí, si el tipo de solicitud es SELECT

El elemento del contenedor **SelectParameters** contiene los siguientes elementos.

Nombre	Descripción	Obligatorio
Expression	La expresión SQL. Por ejemplo: - La siguiente expresión SQL recupera la primera columna de los datos del objeto almacenado en formato CSV: <pre>SELECT s._1 FROM Object s</pre> - La siguiente expresión SQL devuelve todo del objeto: <pre>SELECT * FROM Object</pre> Tipo: cadena Antecesoros: SelectParameters	Sí
ExpressionType	Identifica el tipo de expresión. Tipo: cadena Valores válidos: SQL Antecesoros: SelectParameters	Sí
InputSerialization	Describe el formato de serialización del objeto. Tipo: contenedor para CSV Antecesoros: SelectParameters	Sí
OutputSerialization	Describe cómo se serializan los resultados del trabajo de selección. Tipo: contenedor para CSV	Sí

Nombre	Descripción	Obligatorio
	Antecesor: SelectParameters	

El elemento de contenedor de CSV del elemento **InputSerialization** contiene los siguientes elementos.

Nombre	Descripción	Obligatorio
RecordDelimiter	Un único carácter usado para separar registros individuales en la entrada. En lugar del valor predeterminado, puede especificar un delimitador arbitrario. Tipo: cadena Valor predeterminado: \n Antecesor: CSV	No
FieldDelimiter	Un carácter único usado para separar los campos individuales de un registro. Puede especificar un delimitador arbitrario. Tipo: cadena Valor predeterminado: , Antecesor: CSV	No
QuoteCharacter	Un carácter único que se utiliza como escape cuando el delimitador del campo forma parte del valor. Considere este ejemplo en un archivo CSV: "a, b" Al incluir el valor entre comillas, este valor se convierte en un solo campo. Si no usa las comillas, la coma es un delimitador.	No

Nombre	Descripción	Obligatorio
	or de campo (lo que las convierte en dos valores de campo independientes, a y b). Tipo: cadena Valor predeterminado: " Antecesoros: CSV	
QuoteEscapeCharacter	Un carácter único utilizado como escape con el carácter de comillas cuando está dentro de un valor que ya contiene caracteres de escape. Por ejemplo, el valor "\"" a , b "\"" se analiza como " a , b ". Tipo: cadena Valor predeterminado: " Antecesoros: CSV	No

Nombre	Descripción	Obligatorio
FileHeaderInfo	Describe la primera línea de los datos de entrada. Es uno de los valores ENUM. • NONE: la primera línea no es un encabezado. • IGNORE: la primera línea es un encabezado, pero no puede usar los valores del encabezado para indicar la columna en una expresión. Puede usar la posición de la columna (como _1, _2...) para indicar la columna (SELECT s._1 FROM OBJECT s). • Use: la primera línea es un encabezado y puede usar los valores del encabezado para identificar una columna en una expresión (SELECT "name" FROM OBJECT). Tipo: enumeración Valores válidos: -: NONE USE IGNORE Antecesor: CSV	No
Comments	Carácter único que se utiliza para indicar que una fila debe omitirse cuando el carácter está presente al inicio de esa fila. Puede especificar cualquier carácter para indicar una línea de comentario. Tipo: cadena Antecesor: CSV	No

El elemento de contenedor de CSV (en los elementos **OutputSerialization**) contiene los siguientes elementos.

Nombre	Descripción	Obligatorio
QuoteFields	Indica si se deben usar comillas en los campos de salida. • ALWAYS: utilice siempre comillas en los campos de salida. • ASNEEDED: utilice comillas en los campos de salida cuando sea necesario. Tipo: enumeración Valores válidos: -: ALWAYS ASNEEDED Valor predeterminado: AsNeeded Antecesor: CSV	No
RecordDelimiter	Un único carácter usado para separar registros individuales en la salida. En lugar del valor predeterminado, puede especificar un delimitador arbitrario. Tipo: cadena Valor predeterminado: \n Antecesor: CSV	No
FieldDelimiter	Un carácter único usado para separar los campos individuales de un registro. Puede especificar un delimitador arbitrario. Tipo: cadena Valor predeterminado: ,	No

Nombre	Descripción	Obligatorio
	Antecedentes: CSV	
QuoteCharacter	Un carácter único que se utiliza como escape cuando el delimitador del campo forma parte del valor. Por ejemplo, si el valor es a , b, Amazon S3 escribe el valor de este campo entre comillas, de la siguiente manera: " a , b " . Tipo: cadena Valor predeterminado: " Antecedentes: CSV	No
QuoteEscapeCharacter	Un carácter único utilizado como escape con el carácter de comillas cuando está dentro de un valor que ya contiene caracteres de escape. Por ejemplo, si el valor es " a , b ", Amazon S3 escribe el valor entre comillas, de la siguiente manera: "" " a , b "" . Tipo: cadena Antecedentes: CSV	No

El elemento de contenedor de S3 (en el elemento **OutputLocation**) contiene los siguientes elementos.

Nombre	Descripción	Obligatorio
AccessControlList	Lista de concesiones que controlan el acceso a los resultados por etapas. Tipo: contenedor para concesión Antecedentes: S3	No
BucketName		Sí

Nombre	Descripción	Obligatorio
	El nombre del bucket de S3 en el que se almacenan los resultados de la restauración de selección. El bucket debe estar en la misma Región de AWS que el bucket que contiene el objeto de archivo de entrada. Tipo: cadena Antecedentes: S3	
CannedACL	Lista de control de acceso (ACL) predefinida que se va a aplicar a los resultados de restauración de selección. Tipo: cadena Valores válidos: <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> Antecedentes: S3	No
Encryption	Contiene información de cifrado de los resultados almacenados. Tipo: contenedor para cifrado Antecedentes: S3	No
Prefix	El prefijo que se antepone a los resultados de restauración de selección. La longitud máxima del prefijo es de 512 bytes. Tipo: cadena Antecedentes: S3	Sí

Nombre	Descripción	Obligatorio
StorageClass	La clase de almacenamiento utilizado para guardar los resultados de solicitud Select. Tipo: cadena Valores válidos: STANDARD REDUCED_REDUNDANCY STANDARD_IA ONEZONE_IA Antecesor: S3	No
Tagging	Contenedor para información de etiquetas. Tipo: estructura de etiquetas Antecesor: S3	No
UserMetadata	Contiene una lista de metadatos para almacenar con los resultados de restauración de selección. Tipo: estructura MetadataEntry Antecesor: S3	No

El elemento de contenedor de beneficiario (en el elemento **AccessControlList**) contiene los siguientes elementos.

Nombre	Descripción	Obligatorio
DisplayName	Nombre del beneficiario que aparece en pantalla. Tipo: cadena Antecesor: Grantee	No
EmailAddress	Dirección de correo electrónico del beneficiario.	No

Nombre	Descripción	Obligatorio
ID	Tipo: cadena	No
	Antecedentes: Grantee	
	ID de usuario canónico del beneficiario.	
	Tipo: cadena	
Type	Antecedentes: Grantee	No
	Tipo de beneficiario.	
	Tipo: cadena	
	Antecedentes: Grantee	
URI	URI del grupo de beneficiarios.	No
	Tipo: cadena	
	Antecedentes: Grantee	
	Permiso concedido.	
Permission	Tipo: cadena	No
	Antecedentes: Grantee	

El elemento del contenedor **Encryption** (en S3) contiene los siguientes elementos.

Nombre	Descripción	Obligatorio
EncryptionType	El algoritmo de cifrado del servidor que se utiliza cuando se almacenan los resultados del trabajo. El valor predeterminado es sin cifrado.	No

Nombre	Descripción	Obligatorio
	Tipo: cadena Valores válidos aws : kms AES256 Antecedentes: Encryption	
KMSContext	Opcional. Si el tipo de cifrado es aws : kms, puede utilizar este valor para especificar el contexto de cifrado de los resultados de restauración de selección. Tipo: cadena Antecedentes: Encryption	No
KMSKeyId	ID de clave de AWS Key Management Service (AWS KMS) que se va a utilizar con el cifrado de objetos. Tipo: cadena Antecedentes: Encryption	No

El elemento de contenedor de **TagSet** (en el elemento **Tagging**) contiene el siguiente elemento.

Nombre	Descripción	Obligatorio
Tag	Contiene etiquetas. Tipo: Contenedor Antecedentes: TagSet	No

El elemento de contenedor de etiqueta (en el elemento **TagSet**) contiene los siguientes elementos.

Nombre	Descripción	Obligatorio
Key		No

Nombre	Descripción	Obligatorio
	Nombre de la etiqueta. Tipo: cadena Antecedentes: Tag	
Value	Valor de la etiqueta. Tipo: cadena Antecedentes: Tag	No

El elemento de contenedor de **MetadataEntry** (en el elemento **UserMetadata**) contiene los siguientes elementos de par clave-valor para almacenarlos junto con un objeto.

Nombre	Descripción	Obligatorio
MetadataKey	La clave de metadatos. Tipo: cadena Antecedentes:	No
MetadataEntry	El valor de los metadatos. Tipo: cadena Antecedentes:	No

Respuestas

Si la operación se realiza correctamente, se devuelve el código de estado 200 OK o 202 Accepted.

- Si la copia del objeto no se ha restaurado previamente, Amazon S3 devuelve 202 Accepted en la respuesta.

- Si la copia del objeto se ha restaurado previamente, Amazon S3 devuelve 200 OK en la respuesta.

Encabezados de respuesta

Esta implementación de la operación solo usa encabezados de respuesta que sean comunes a la mayoría de las respuestas. Para obtener más información, consulte [Encabezados de respuesta comunes](#).

Elementos de respuesta

Esta operación no devuelve elementos de respuesta.

Errores especiales

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
RestoreAlreadyInProgress	La restauración del objeto ya está en curso. (Este error no se aplica a las solicitudes de tipo SELECT).	Conflicto, 409	Cliente
GlacierExpeditedRetrievalNotAvailable	Las recuperaciones aceleradas de la clase de almacenamiento de S3 Glacier no están disponibles actualmente. Inténtelo de nuevo más tarde. (Se devuelve si no hay capacidad suficiente para procesar la solicitud Expedited). Este error solo se aplica a las recuperaciones Expedited y no a las recuperaciones Standard o Bulk).	503	N/A

Ejemplos

Restauración de un objeto durante dos días con la opción de recuperación acelerada

La siguiente solicitud de restauración restaura una copia del objeto photo1.jpg de Amazon Glacier durante un periodo de dos días mediante la opción de recuperación acelerada.

```
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 no tiene una copia restaurada del objeto, Amazon S3 devuelve la respuesta 202 Accepted siguiente.

```
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 ya se ha restaurado una copia del objeto, Amazon S3 devuelve una respuesta 200 OK y actualiza solo el tiempo de caducidad de la copia restaurada.

Consulta de un archivo con una solicitud SELECT

A continuación, se muestra un ejemplo solicitud de restauración de selección.

```
POST /object-one.csv?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
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>
<StorageClass>STANDARD</StorageClass>
</S3>
</OutputLocation>
</RestoreRequest>
```

Amazon S3 devuelve la respuesta 202 Accepted siguiente.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZ1zYQvPiB1ZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
x-amz-restore-output-path: js-test-s3/qE8nk5M0XIj-LuZE2HXNw6empQm3znLkH1MWInRYPS-
0rl2W0uj6LyYm-neTvm1-btz3wbBxfMhPykd3jkl-1vZE7w42/
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

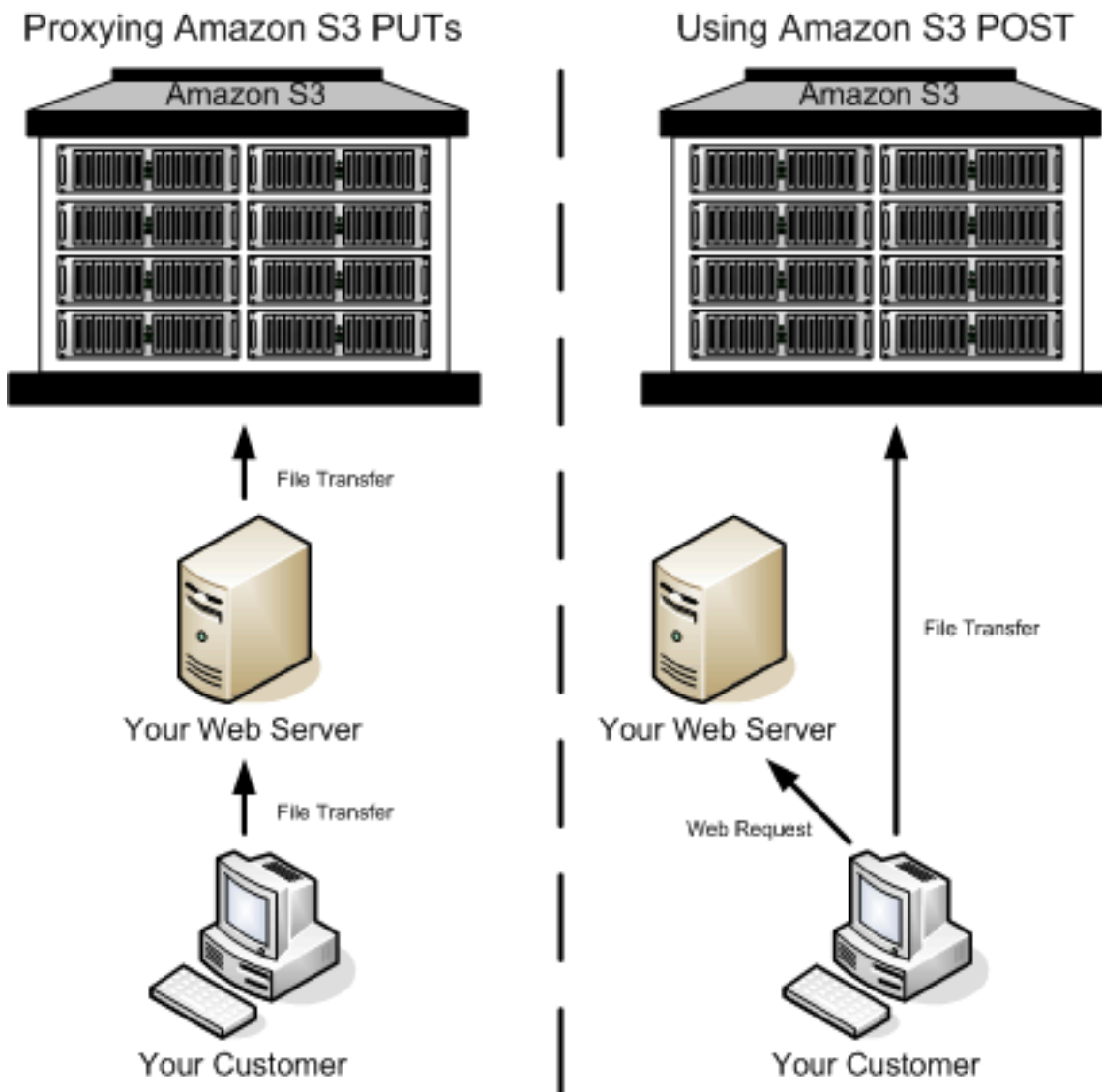
Más información

- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)
- [Referencia de SQL para Amazon S3 Select y Amazon Glacier Select](#) en la Guía del usuario de Amazon Simple Storage Service

Cargas basadas en el navegador con HTTP POST

Amazon S3 admite las solicitudes HTTP POST para que los usuarios puedan cargar contenido directamente en Amazon S3. Al usar POST, los usuarios finales pueden autenticar las solicitudes sin tener que pasar los datos a través de un nodo intermediario seguro que proteja las credenciales. Por lo tanto, HTTP POST tiene el potencial de reducir la latencia.

En la siguiente figura, se muestra una carga de Amazon S3 con una solicitud POST.



1. El usuario accede a la página desde un navegador web.
2. La página web incluye un formulario HTML que contiene toda la información necesaria para que el usuario cargue contenido en Amazon S3.
3. El usuario carga contenido en Amazon S3 mediante el navegador web.

El proceso para enviar solicitudes POST basadas en el navegador es el siguiente:

1. Cree una política de seguridad que especifique las condiciones que restrinjan lo que desea permitir en la solicitud, como el nombre del bucket en el que se pueden cargar los objetos y los prefijos de los nombres de clave que desee permitir para el objeto que se va a crear.

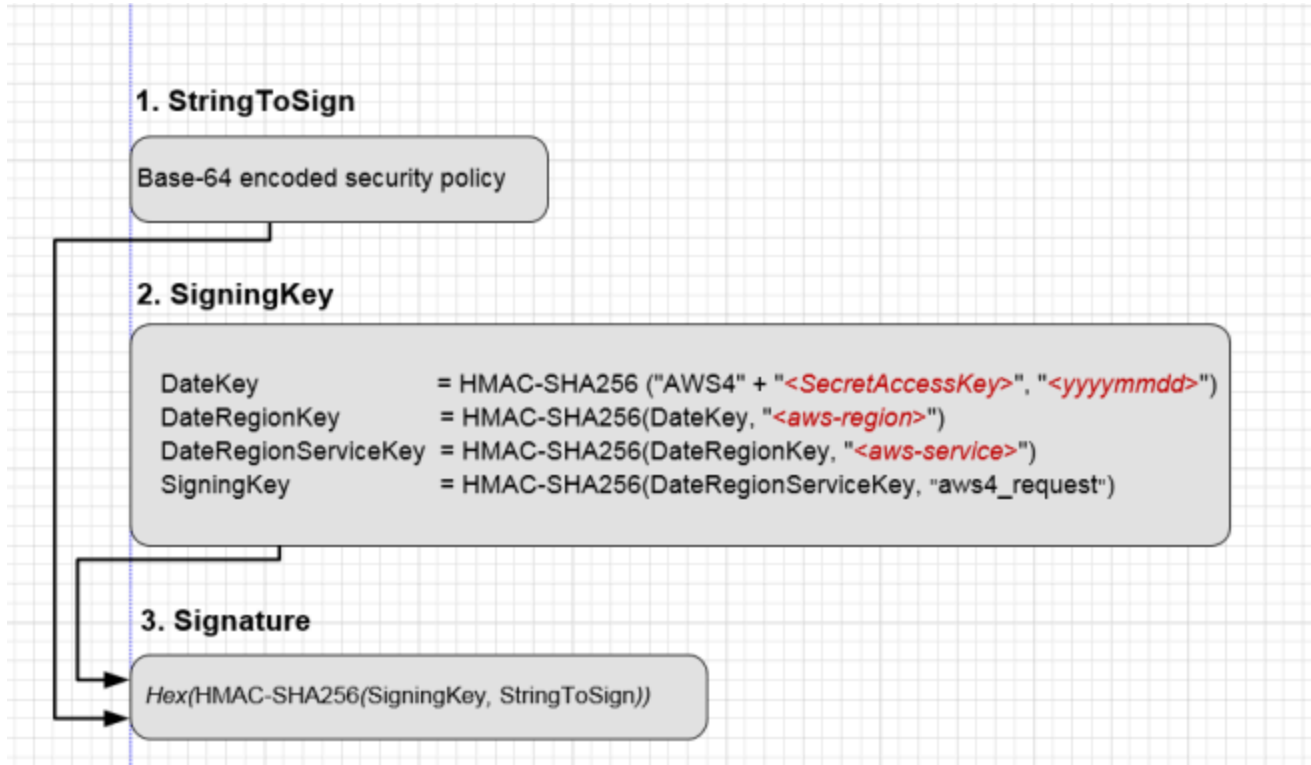
2. Cree una firma basada en la política. Para las solicitudes autenticadas, el formulario debe incluir una firma válida y la política.
3. Cree un formulario HTML al que puedan acceder los usuarios para cargar objetos en el bucket de Amazon S3.

En la siguiente sección, se describe cómo crear una firma para autenticar una solicitud. Para obtener información acerca de la creación de formularios y políticas de IAM, consulte [Creación de un formulario HTML \(mediante AWS Signature Version 4\)](#).

Cálculo de una firma

Para las solicitudes autenticadas, el formulario HTML debe incluir campos para una política y una firma de seguridad.

- Una política de seguridad (consulte [Política POST](#)) controla lo que se permite en la solicitud.
- La política de seguridad es StringToSign (consulte [Introducción a la firma de solicitudes](#)) en el cálculo de la firma.



Cálculo de una firma

1. Cree una política con la codificación UTF-8.
2. Convierta los bytes de la política codificados en UTF-8 a base64. El resultado es `StringToSign`.
3. Cree una clave de firma.
4. Utilice la clave de firma para firmar `StringToSign` mediante el algoritmo de firma HMAC-SHA256.

Para obtener más información acerca de la creación de formularios HTML, políticas de seguridad y un ejemplo, consulte lo siguiente:

- [Creación de un formulario HTML \(mediante AWS Signature Version 4\)](#)
- [Política POST](#)
- [Ejemplo: carga basada en el navegador con HTTP POST \(con AWS Signature Version 4\)](#)

Creación de un formulario HTML (mediante AWS Signature Version 4)

Temas

- [Declaración de formulario HTML](#)
- [Campos de formulario HTML](#)

Para que los usuarios puedan cargar contenido en Amazon S3 con sus navegadores (solicitudes HTTP POST), debe utilizar los formularios HTML. Los formularios HTML constan de una declaración de formulario y campos de formulario. Cada declaración de formulario incluye información de alto nivel acerca de la solicitud. Los campos del formulario contienen información detallada sobre la solicitud.

En esta sección, se describe cómo crear formularios HTML. Para ver un ejemplo práctico de carga basada en un navegador mediante HTTP POST y cálculos de firma relacionados para la autenticación de solicitudes, consulte [Ejemplo: carga basada en el navegador con HTTP POST \(con AWS Signature Version 4\)](#).

El formulario y la política deben estar cifrados con UTF-8. Puede aplicar la codificación UTF-8 al formulario mediante la especificación de `charset=UTF-8` en el atributo `content`. A continuación, se muestra un ejemplo de la codificación UTF-8 en el encabezado HTML.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

A continuación, se muestra un ejemplo de la codificación UTF-8 en un encabezado de solicitud.

```
Content-Type: text/html; charset=UTF-8
```

Note

Los datos y límites del formulario (sin incluir los contenidos del archivo) no pueden exceder los 20 KB.

Declaración de formulario HTML

La declaración del formulario HTML tiene los tres atributos siguientes:

- **action:** la URL que procesa la solicitud, que debe establecerse en la URL del bucket. Por ejemplo, si el nombre del bucket es `examplebucket`, la URL es `http://examplebucket.s3.amazonaws.com/`.

Note

El nombre de clave se especifica en un campo de formulario.

- **method:** el método debe ser POST.
- **enctype:** el tipo de documento adjunto (`enctype`) se debe establecer en datos de formulario o multiparte para cargas de archivos y cargas de área de texto. Para obtener más información acerca de `enctype`, consulte [RFC 1867](#).

Es una declaración de formulario para el bucket `examplebucket`.

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post"
  enctype="multipart/form-data">
```

Campos de formulario HTML

En la siguiente tabla, se describe una lista de los campos que se pueden utilizar en un formulario. Entre otros campos, hay un campo de firma que puede usar para autenticar las solicitudes. Hay campos para especificar el algoritmo de cálculo de firmas (`x-amz-algorithm`), el alcance de las credenciales (`x-amz-credential`) que utilizó para generar la clave de firma y la fecha (`x-amz-date`) utilizada para calcular la firma. Amazon S3 utiliza esta información para volver a crear la firma. Si las firmas coinciden, Amazon S3 procesa la solicitud.

Note

La variable `${filename}` se reemplaza automáticamente con el nombre del archivo provisto por el usuario y es reconocida por todos los campos de formulario. Si el navegador o el cliente proporciona una ruta total o parcial al archivo, solo se utilizará el texto después de la última barra inclinada (/) o barra inversa (\) (por ejemplo, `C:\Program Files\directory1\file.txt` se interpreta como `file.txt`). Si no se brinda ningún archivo o nombre de archivo, la variable se reemplaza con una cadena vacía.

Si no proporciona los elementos necesarios para las solicitudes autenticadas, como el elemento `policy`, se asume que la solicitud es anónima y solo tendrá éxito si ha configurado el bucket para que sea de lectura y escritura públicas.

Nombre del elemento	Descripción	Obligatorio
<code>acl</code>	Una lista de control de acceso (ACL) de Amazon S3. Si se especifica una ACL no válida, Amazon S3 deniega la solicitud. Para obtener más información acerca de las ACL, consulte Uso de las ACL de Amazon S3 . Tipo: cadena	No

Nombre del elemento	Descripción	Obligatorio
	Valor predeterminado: <code>privado</code> Valores válidos: <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>	
Cache-Control Content-Type Content-Disposition Content-Encoding Expires	Encabezados específicos de REST. Para obtener más información, consulte PutObject .	No
key	El nombre de la clave del objeto cargado. Para utilizar el nombre de archivo provisto por el usuario, utilice la variable <code>\${filename}</code>. Por ejemplo, si carga un archivo <code>photo1.jpg</code> y especifica <code>/user/user1/\${filename}</code> como nombre de clave, el archivo se almacena como <code>/user/user1/photo1.jpg</code>. Para obtener más información, consulte Clave de objeto y metadatos en la Guía del usuario de Amazon Simple Storage Service.	Sí

Nombre del elemento	Descripción	Obligatorio
<code>policy</code>	La política de seguridad codificada en base64 que describe lo que se permite en la solicitud. Para las solicitudes autenticadas, se requiere una política. Las solicitudes sin una política de seguridad se consideran anónimas y solo se aceptarán en un bucket que se puede escribir públicamente.	Obligatorio para las solicitudes autenticadas
<code>success_action_redirect</code>	URL al que el cliente es redirigido después de la carga exitosa. Si no se especifica <code>success_action_redirect</code> o Amazon S3 no puede interpretar la URL, Amazon S3 devuelve el tipo de documento vacío especificado en el campo <code>success_action_status</code>. Si la carga produce un error, Amazon S3 devuelve un error y no redirige al usuario a otra URL.	No

Nombre del elemento	Descripción	Obligatorio
success_action_status	El código de estado devuelto al cliente después de la carga exitosa si no se especifica <code>success_action_redirect</code> . Los valores válidos son 200, 201 o 204 (predeterminado). Si el valor se establece en 200 o 204, Amazon S3 devuelve un documento vacío con el código de estado especificado. Si el valor se establece en 201, Amazon S3 devuelve un documento XML con un código de estado 201. Para obtener información acerca del contenido del documento XML, consulte POST Object. Si el valor no se establece o no es válido, Amazon S3 devuelve un documento vacío con un código de estado 204.  Note Algunas versiones de Adobe Flash Player no controlan debidamente las respuestas HTTP con un cuerpo vacío. Para admitir cargas con Adobe Flash, recomendamos establecer <code>success_action_status</code> en 201.	No

Nombre del elemento	Descripción	Obligatorio
x-amz-algorithm	El algoritmo de firma utilizado para autenticar la solicitud. Para AWS Signature Version 4, el valor es AWS4-HMAC-SHA256 . Este campo es obligatorio si se incluye un documento de política en la solicitud.	Obligatorio para las solicitudes autenticadas
x-amz-credential	Además del ID de clave de acceso, este campo también proporciona información de alcance, lo que identifica la región y el servicio para los que es válida la firma. Debe ser el mismo alcance que utilizó al calcular la clave de firma para el cálculo de la firma. Es una cadena del formulario siguiente: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Por ejemplo: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Para Amazon S3, la cadena aws-service es s3. Para obtener una lista de las cadenas de aws-region de Amazon S3, consulte Regiones y puntos de conexión en Referencia general de AWS. Esto es obligatorio si se incluye un documento de política con la solicitud.	Obligatorio para las solicitudes autenticadas

Nombre del elemento	Descripción	Obligatorio
x-amz-date	Es el valor de la fecha en formato ISO8601. Por ejemplo, 20130728T000000Z . Es la misma fecha que utilizó al crear la clave de firma (por ejemplo, 20130728). Este debe ser también el mismo valor que proporciona en la política (x-amz-date) que firmó. Esto es obligatorio si se incluye un documento de política con la solicitud.	Obligatorio para las solicitudes autenticadas
x-amz-security-token	Un token de seguridad utilizado por Amazon DevPay y las credenciales de sesión Si la solicitud utiliza Amazon DevPay, se requieren dos campos de formulario x-amz-security-token : uno para el token de producto y otro para el token de usuario. Para obtener más información, consulte Uso de DevPay en la Guía del usuario de Amazon Simple Storage Service. Si la solicitud utiliza credenciales de sesión, se requiere un formulario x-amz-security-token . Para obtener más información, consulte Solicitud de credenciales de seguridad temporales en la Guía del usuario de IAM.	No
x-amz-signature	(AWS Signature Version 4) El hash HMAC-SHA256 de la política de seguridad. Este campo es obligatorio si se incluye un documento de política en la solicitud.	Obligatorio para las solicitudes autenticadas

Nombre del elemento	Descripción	Obligatorio
x-amz-meta-*	Los nombres de campos que comienzan con este prefijo son metadatos definidos por el usuario. Cada uno de ellos se almacena y se devuelve como un conjunto de pares clave-valor. Amazon S3 no valida ni interpreta metadatos definidos por el usuario. Para obtener más información, consulte PutObject .	No
x-amz-*	Consulte POST Object (POST Object para otros encabezados x-amz-*).	No
file	Contenido de texto o archivo. El archivo o el contenido debe ser el último campo en el formulario. No puede cargar más de un archivo a la vez.	Sí

Los elementos condicionales son obligatorios para las solicitudes autenticadas y son opcionales para las solicitudes anónimas.

Ahora que sabe cómo crear formularios, puede crear una política de seguridad que puede firmar. Para obtener más información, consulte [Política POST](#).

Política POST

Temas

- [Expiration](#)
- [Coincidencia de condiciones](#)
- [Condiciones](#)
- [Secuencia de escape de caracteres](#)

La política necesaria para realizar solicitudes autenticadas mediante HTTP POST es un documento con codificación UTF-8 y base64 escrito en Notación de objetos de JavaScript (JSON) que especifica las condiciones que debe cumplir la solicitud. Según cómo diseñe el documento de la política, puede controlar la granularidad del acceso por carga, por usuario, para todas las cargas o de acuerdo con otros diseños que se ajusten a sus necesidades.

En esta sección, se describe la política POST. Para ver ejemplos de cálculos de firmas mediante la política POST, consulte [Ejemplo: carga basada en el navegador con HTTP POST \(con AWS Signature Version 4\)](#).

Note

Aunque el documento de política es opcional, recomendamos encarecidamente que se utilice para controlar lo que se permite en la solicitud. Si hace que se pueda escribir en el bucket públicamente, no tendrá ningún control sobre qué usuarios pueden escribir en el bucket.

A continuación, se muestra un ejemplo de un documento de política POST.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

La política POST siempre contiene los elementos `expiration` y `conditions`. La política de ejemplo usa dos tipos de coincidencia de condiciones (coincidencia exacta y coincidencia por prefijo). En las secciones siguientes, se describen estos elementos.

Expiration

El elemento de `expiration` especifica la fecha de vencimiento y la hora de la política POST en el formato de fecha ISO8601 GMT. Por ejemplo, `2013-08-01T12:00:00.000Z` especifica que la política POST no es válida después de la medianoche GMT del 1 de agosto de 2013.

Coincidencia de condiciones

A continuación, se incluye una tabla en la que se describen los tipos de coincidencia de condiciones que puede utilizar para especificar las condiciones de la política POST (que se describen en la siguiente sección). Aunque debe especificar al menos una condición para cada campo del formulario que especifica en el formulario, puede crear criterios de coincidencia más complejos especificando varias condiciones para un campo de formulario.

Tipo de coincidencia de condiciones	Descripción
Coincidencias exactas	El valor del campo del formulario debe coincidir con el valor especificado. En este ejemplo se indica que la ACL se debe establecer en public-read: <pre>{"acl": "public-read" }</pre> Este ejemplo es una alternativa para indicar que la ACL se debe establecer en public-read: <pre>["eq", "\$acl", "public-read"]</pre>
Empieza por	El valor debe empezar por el valor especificado. En este ejemplo, se indica que la clave de objeto debe empezar con user/user1: <pre>["starts-with", "\$key", "user/user1/"]</pre>
Tipos de contenido coincidentes en una lista separada por comas	Los valores de los tipos de contenido de una condición <code>starts-with</code> que incluyen comas se interpretan como listas. Cada valor de la lista debe cumplir la condición para que se cumpla toda la condición. Por ejemplo, se proporciona la siguiente condición: <pre>["starts-with", "\$Content-Type", "image/"]</pre> El valor siguiente cumpliría la condición:

Tipo de coincidencia de condiciones	Descripción
	<pre>"image/jpg,image/png,image/gif"</pre> El valor siguiente no cumpliría la condición: <pre>["image/jpg,text/plain"]</pre>  Note Los elementos de datos distintos de Content-Type se tratan como cadenas, independientemente de la presencia de comas.
Coincidencia con cualquier contenido	Para configurar la política POST para permitir cualquier contenido dentro de un campo de formulario, utilice <code>starts-with</code> con un valor vacío (<code>""</code>). Este ejemplo permite cualquier valor para <code>success_action_redirect</code> : <pre>["starts-with", "\$success_action_redirect", ""]</pre>
Especificación de rangos	Para campos de formulario que aceptan rangos, separe el límite superior e inferior con una coma. Este ejemplo permite un tamaño de archivo de 1 a 10 MiB: <pre>["content-length-range", 1048576, 10485760]</pre>

las condiciones específicas admitidas en una política POST se describen en [Condiciones](#).

Condiciones

Las `conditions` en una política POST son una matriz de objetos, cada uno de los cuales se utiliza para validar la solicitud. Puede usar estas condiciones para restringir lo que está permitido en la solicitud. Por ejemplo, las condiciones de política anteriores requieren lo siguiente:

- La solicitud debe especificar el nombre del bucket `johnsmith`.
- El nombre de la clave del objeto debe tener el prefijo `user/eric`.
- La ACL del objeto debe estar configurada en `public-read`.

Cada campo de formulario que especifique en un formulario (excepto `x-amz-signature`, `file`, `policy` y los nombres de campo que tienen un prefijo `x-ignore-`) debe aparecer en la lista de condiciones.

Note

Todas las variables en el formulario se expanden antes de validar la política POST. Por lo tanto, todas las coincidencias de condiciones deben estar contra los campos de formulario expandidos. Suponga que desea restringir el nombre de la clave del objeto a un prefijo específico (`user/user1`). En este caso, defina el campo del formulario de clave en `user/user1/${filename}`. La política POST debería ser [`"starts-with", "$key", "user/user1/"`] (no ingrese [`"starts-with", "$key", "user/user1/${filename}"`]). Para obtener más información, consulte [Coincidencia de condiciones](#).

Las condiciones de los documentos de la política se describen en la tabla siguiente.

Nombre del elemento	Descripción
<code>acl</code>	Especifica el valor de ACL que se debe utilizar en el envío del formulario. Esta condición admite la coincidencia exacta y el tipo de coincidencia de condiciones <code>starts-with</code> que se analizan en la siguiente sección.
<code>bucket</code>	Especifica el nombre del bucket aceptable. Esta condición admite el tipo de coincidencia exacta de condiciones.
<code>content-length-range</code>	

Nombre del elemento	Descripción
	El tamaño mínimo y máximo permitido para el contenido cargado. Esta condición admite el tipo de coincidencia de condiciones <code>content-length-range</code> .
<code>Cache-Control</code> <code>Content-Type</code> <code>Content-Disposition</code> <code>Content-Encoding</code> <code>Expires</code>	Encabezados específicos de REST. Para obtener más información, consulte POST Object. Esta condición admite el tipo de coincidencia exacta y de coincidencia de condiciones <code>starts-with</code> .
<code>key</code>	El nombre de clave aceptable o un prefijo del objeto cargado. Esta condición admite el tipo de coincidencia exacta y de coincidencia de condiciones <code>starts-with</code> .
<code>success_action_redirect</code> <code>redirect</code>	URL al que el cliente es redirigido después de la carga exitosa. Esta condición admite el tipo de coincidencia exacta y de coincidencia de condiciones <code>starts-with</code> .
<code>success_action_status</code>	El código de estado devuelto al cliente después de la carga exitosa si no se especifica <code>success_action_redirect</code> . Esta condición admite la coincidencia exacta.

Nombre del elemento	Descripción
x-amz-algorithm	El algoritmo de firma que se debe utilizar durante el cálculo de la firma. Para AWS Signature Version 4, el valor es <code>AWS4-HMAC-SHA256</code> . Esta condición admite la coincidencia exacta.
x-amz-credential	Las credenciales que utilizó para calcular la firma. Proporciona el ID de clave de acceso y la información de alcance que identifica la región y el servicio para los que es válida la firma. Debe ser el mismo alcance que utilizó al calcular la clave de firma para el cálculo de la firma. Es una cadena del formulario siguiente: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Por ejemplo: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Para Amazon S3, la cadena <code>aws-service</code> es <code>s3</code>. Para obtener una lista de las cadenas de <code>aws-region</code> de Amazon S3, consulte Regiones y puntos de conexión en Referencia general de AWS. Esto es obligatorio si se incluye un documento de política POST con la solicitud. Esta condición admite la coincidencia exacta.

Nombre del elemento	Descripción
x-amz-date	El valor de la fecha especificado en la cadena con formato ISO8601. Por ejemplo, 20130728T000000Z . La fecha debe ser la misma que utilizó al crear la clave de firma para el cálculo de la firma. Esto es obligatorio si se incluye un documento de política POST con la solicitud. Esta condición admite la coincidencia exacta.
x-amz-security-token	Token de seguridad de Amazon DevPay. Cada solicitud que utiliza Amazon DevPay requiere dos campos de formulario x-amz-security-token : uno para el token de producto y otro para el token de usuario. Como consecuencia, los valores deben estar separados por comas. Por ejemplo, si el token de usuario es eW91dHViZQ== y el token del producto es b0hnNVNKWVJIQTA= , se debe establecer la entrada de política POST en: { "x-amz-security-token": "eW91dHViZQ==,b0hnNVNKWVJIQTA=" } . Para obtener más información sobre Amazon DevPay, consulte Uso de DevPay en la Guía del usuario de Amazon Simple Storage Service.
x-amz-meta-*	Metadatos especificados por el usuario. Esta condición admite el tipo de coincidencia exacta y de coincidencia de condiciones starts-with .

Nombre del elemento	Descripción
x-amz-*	Consulte POST Object (POST Object para otros encabezados x-amz-*). Esta condición admite la coincidencia exacta.

Note

Si el conjunto de herramientas agrega más campos de formulario (por ejemplo, Flash agrega filename), debe agregarlos al documento de política POST. Si puede controlar esta funcionalidad, añada el prefijo x-ignore- al campo para que Amazon S3 omita la característica y no afecte futuras versiones de esta característica.

Secuencia de escape de caracteres

Los caracteres a los que se debe aplicar una secuencia de escape en un documento de política POST se describen en la tabla siguiente.

Secuencia de escape	Descripción
\\	Barra inversa
\\\$	Símbolo de dólar
\\b	Retroceso
\\f	Salto de página
\\n	Nueva línea

Secuencia de escape	Descripción
\r	Salto de línea
\t	Tabulador horizontal
\v	Tabulador vertical
\uxxxx	Todos los caracteres Unicode

Ahora que ya conoce los formularios y las políticas y entiende cómo funciona la firma, puede probar un ejemplo de carga de POST. Debe escribir el código para calcular la firma. El ejemplo proporciona un formulario de muestra y una política POST que puede usar para probar los cálculos de firma. Para obtener más información, consulte [Ejemplo: carga basada en el navegador con HTTP POST \(con AWS Signature Version 4\)](#).

Ejemplo: carga basada en el navegador con HTTP POST (con AWS Signature Version 4)

En esta sección, se muestra un ejemplo del uso de una solicitud HTTP POST para cargar contenido directamente en Amazon S3.

Para obtener más información sobre la versión 4 de firma, consulte [Proceso de firma de Signature Version 4](#).

Carga de un archivo en Amazon S3 con HTTP POST

Este ejemplo proporciona una política POST de ejemplo y un formulario que puede utilizar para cargar un archivo. El tema utiliza la política de ejemplo y las credenciales ficticias muestran el flujo de trabajo y el hash de la firma y la política resultantes. Puede utilizar estos datos como conjunto de pruebas para verificar el código de cálculo de la firma.

El ejemplo utiliza las credenciales de ejemplo siguientes de los cálculos de la firma. Puede usar estas credenciales para verificar el código de cálculo de la firma. Sin embargo, a continuación, debe sustituirlas por sus propias credenciales cuando envíe las solicitudes a AWS.

Parámetro	Valor
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

Ejemplo de política y formulario

La política POST siguiente admite cargas en Amazon S3 con condiciones específicas.

```
{ "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" }
]
```

Esta política POST establece las siguientes condiciones en la solicitud:

- La carga se debe realizar antes de las 12:00 UTC el 30 de diciembre de 2015.
- El contenido solo se puede cargar en `sigv4examplebucket`. El bucket debe estar en la región que especificó en el ámbito de las credenciales (parámetro del formulario `x-amz-credential`), ya que la firma que proporcionó solo es válida dentro de este ámbito.
- Puede proporcionar cualquier nombre de clave que comience por `user/user1`. Por ejemplo, `user/user1/MyPhoto.jpg`.
- El ACL debe estar establecido en `public-read`.
- Si la carga se realiza correctamente, se redirige al navegador del usuario a `http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html`.
- El objeto debe ser un archivo de imagen.
- La etiqueta `x-amz-meta-uuid` se debe establecer en `14365123651274`.
- `x-amz-meta-tag` puede incluir cualquier valor.

A continuación, se incluye una versión de esta política POST codificada en Base64. Este valor se utiliza como `StringToSign` en el cálculo de firmas.

```
eyJhZiZlXGhwaXJhdGlvbiI6IClYMDU1LTUyLTUwVDEyOjAwOjAwLjAwMFoiLA0KICAiY29uZG10aW9ucyI6IFsNCiAgICB7ImJ
```

Al copiar y pegar la política anterior, debería incluir retornos de carro y líneas nuevas para que el hash calculado coincida con este valor (es decir, texto ASCII, con terminadores de línea CRLF).

Si utilizamos credenciales de ejemplo para crear una firma, el valor de la firma es el siguiente (en el cálculo de firmas, la fecha es la misma que la `x-amz-date` de la política (20151229):

```
8afdbf4008c03f22c2cd3cdb72e4afbb1f6a588f3255ac628749a66d7f09699e
```

El formulario de ejemplo siguiente especifica la política POST anterior y admite una solicitud POST al `sigv4examplebucket`. Copie o pegue el contenido en un editor de texto y guárdelo como `exampleform.html`. A continuación, puede cargar los archivos de imagen en el bucket específico mediante `exampleform.html`. La solicitud se realizará correctamente si la firma que proporciona coincide con la firma que calcula Amazon S3.

Note

Debe actualizar el nombre del bucket, las fechas, la credencial, la política y la firma con valores válidos para poder cargarlo correctamente en S3.

```
<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>
```

Los parámetros de publicación no distinguen entre mayúsculas y minúsculas. Por ejemplo, puede especificar `x-amz-signature` o `X-Amz-Signature`.

Cargas basadas en el navegador en Amazon S3 mediante la biblioteca de AWS Amplify

En esta sección, se describe cómo cargar archivos en Amazon S3 mediante la biblioteca de AWS Amplify JavaScript.

Para obtener información sobre la configuración de la biblioteca de AWS Amplify, consulte [Instalación y configuración de AWS Amplify](#).

Uso de la biblioteca de AWS Amplify JavaScript para cargar archivos en Amazon S3

El módulo de Storage de la biblioteca de AWS Amplify ofrece un mecanismo de carga sencillo basado en un navegador para administrar el contenido de los usuarios en el almacenamiento público o privado de Amazon S3.

Example: configuración manual de AWS Amplify

En el ejemplo siguiente, se muestra la configuración manual para usar el módulo de AWS Amplify Storage. La implementación predeterminada del módulo de almacenamiento usa 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  
}  
);
```

Example: inclusión de datos en Amazon S3

En el ejemplo siguiente, se muestra cómo put datos públicos en Amazon S3.

```
Storage.put('test.txt', 'Hello')  
  .then (result => console.log(result))  
  .catch(err => console.log(err));
```

En el ejemplo siguiente, se muestra cómo put datos privados en Amazon S3.

```
Storage.put('test.txt', 'Private Content', {  
  level: 'private',  
  contentType: 'text/plain'  
})  
  .then (result => console.log(result))  
  .catch(err => console.log(err));
```

Para obtener más información sobre el uso del módulo de AWS Amplify Storage, consulte [AWS Amplify Storage](#).

Más información

[AWS Amplify Quick Start](#)

Encabezados de solicitudes comunes

En la siguiente tabla se describen los encabezados que se pueden utilizar por varios tipos de solicitudes REST de Amazon S3.

Nombre del encabezado	Descripción
Authorization	Información necesaria para solicitar la autenticación. Para obtener más información, vaya a Encabezado de autenticación en la Guía para desarrolladores de Amazon Simple Storage Service. Para las solicitudes anónimas, este encabezado no es obligatorio.
Access-Control-Request-Method	Una lista de métodos HTTP que se envía como una solicitud de CORS anterior al tránsito. Si la evaluación de CORS previa al tránsito se realiza correctamente, se permite el uso de los métodos especificados en la siguiente solicitud de CORS.
Content-Length	Longitud del mensaje (sin encabezados) de acuerdo con RFC 2616. Este encabezado es necesario para las PUT y las operaciones que cargan XML, como el registro y las ACL.
Content-Type	El tipo de contenido del recurso en caso de que la solicitud tenga contenido en el cuerpo. Ejemplo: text/plain
Content-MD5	Resumen MD5 de 128 bits del mensaje cifrado en base64 (sin los encabezados) según RFC 1864. Este encabezado se puede utilizar como comprobación de la integridad del mensaje para comprobar que los datos son los mismos que se enviaron originalmente. Aunque es opcional, recomendamos utilizar el mecanismo Content-MD5 como comprobación de integridad de principio a fin. Para obtener más información sobre la autenticación de solicitudes REST, vaya a Autenticación de REST en la Guía para desarrolladores de Amazon Simple Storage Service.

Nombre del encabezado	Descripción
Date	Fecha que se puede utilizar para crear la firma incluida en el encabezado <code>Authorization</code> . Si el encabezado <code>Date</code> se va a utilizar para firmar, debe especificarse en el formato básico de la norma ISO 8601. En ese caso, el encabezado <code>x-amz-date</code> no es necesario. Tenga en cuenta que, cuando <code>x-amz-date</code> está presente, anula siempre el valor del encabezado <code>Date</code>. Si el encabezado <code>Date</code> (Fecha) no se utiliza para firmar, puede tener uno de los formatos de fecha completa que se especifican en RFC 2616, sección 3.3. Por ejemplo, el valor de fecha u hora <code>Wed, 01 Mar 2006 12:00:00 GMT</code> es un encabezado de fecha u hora válido para su uso con Amazon S3. Si utiliza el encabezado <code>Date</code> para firmar, debe tener el formato <code>YYYYMMDD'T'HHMMSS'Z'</code> básico de la norma ISO 8601. Si se especifica <code>Date</code> pero no tiene el formato básico de la norma ISO 8601, debe incluirse también el encabezado <code>x-amz-date</code> . Si <code>Date</code> se especifica con el formato básico de la norma ISO 8601, este encabezado es suficiente para firmar las solicitudes y no se necesita el encabezado <code>x-amz-date</code> . Para obtener más información, consulte Control de fechas en Signature Version 4 en el Glosario de Amazon Web Services.
Expect	Cuando la aplicación usa <code>100-continue</code>, no envía el cuerpo de la solicitud hasta que recibe una confirmación. Si el mensaje se rechaza en función de los encabezados, el cuerpo del mismo no se llega a enviar. Este encabezado solo se puede usar si envía un cuerpo. Valores válidos: <code>100-continue</code>

Nombre del encabezado	Descripción
Host	Para las solicitudes con estilo de ruta, el valor es <code>s3.amazonaws.com</code>. Para las solicitudes con estilo virtual, el valor es <code>BucketName.s3.amazonaws.com</code>. Para obtener más información, vaya a Alojamiento virtual en la Guía del usuario de Amazon Simple Storage Service. Este encabezado es obligatorio para HTTP 1.1 (la mayoría de los kits de herramientas agregan este encabezado automáticamente); opcional para las solicitudes HTTP/1.0.
Origin	Un punto de conexión que especifica el nombre del servidor del solicitante inicial.
x-amz-content-sha256	Al usar la versión 4 de la firma para autenticar una solicitud, este encabezado proporciona un hash de la carga útil de la solicitud. Para obtener más información consulte () Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en un solo fragmento (AWS Signature Version 4). Al cargar el objeto en fragmentos, se establece el valor en <code>STREAMING-AWS4-HMAC-SHA256-PAYLOAD</code> para indicar que la firma solo cubre los encabezados y que no hay carga útil. Para obtener más información, consulte Cálculos de firma para el encabezado de Authorization: transferencia de la carga útil en varios fragmentos (carga fragmentada) (AWS Signature Version 4).

Nombre del encabezado	Descripción
x-amz-date	La fecha que se utiliza para crear la firma en el encabezado de autorización. El formato debe ser el formato básico de la norma ISO 8601 YYYYMMDD'T'HHMMSS'Z'. Por ejemplo, el valor de fecha u hora 20170210T120000Z es un encabezado x-amz-date válido que puede utilizarse con Amazon S3. x-amz-date es opcional en todas las solicitudes; se puede utilizar para invalidar la fecha empleada para firmar solicitudes. Si el encabezado Date se especifica en el formato básico de la norma ISO 8601, x-amz-date no es necesario. Cuando x-amz-date está presente, anula siempre el valor del encabezado Date. Para obtener más información, consulte Control de fechas en Signature Version 4 en el Glosario de Amazon Web Services.

Nombre del encabezado	Descripción
x-amz-security-token	Este encabezado se puede utilizar en las siguientes situaciones: • Para proporcionar tokens de seguridad para las operaciones de Amazon DevPay: cada solicitud que utiliza Amazon DevPay requiere dos encabezados <code>x-amz-security-token</code> : uno para el token de producto y otro para el token de usuario. Cuando Amazon S3 recibe una solicitud autenticada, compara la firma calculada con la firma proporcionada. Los encabezados multivalor mal formateados que se utilizan para calcular una firma pueden provocar problemas de autenticación. • Para proporcionar un token de seguridad al usar credenciales de seguridad temporales: al realizar solicitudes con credenciales de seguridad temporales que haya obtenido de IAM, debe proporcionar un token de seguridad mediante este encabezado. Para obtener más información sobre las credenciales de seguridad temporales, consulte Realización de solicitudes. Este encabezado es obligatorio para las solicitudes que utilizan Amazon DevPay y las solicitudes que se firman con credenciales de seguridad temporales.

Encabezados de respuesta comunes

En la siguiente tabla, se describen los encabezados de respuesta que son comunes a la mayoría de las respuestas de Amazon S3.

Nombre	Descripción
Access-Control-Allow-Credentials	Un booleano que determina si el servidor permite que las solicitudes de CORS contengan credenciales. Si el encabezado de la solicitud <code>Access-Control-Allow-Origin</code> está configurado en <code>'*'</code>, se omitirá el encabezado de respuesta <code>Access-Control-Allow-Credentials</code>; de lo contrario, se establecerá en <code>true</code> cuando la evaluación de CORS se realice correctamente. Tipo: booleano Valor predeterminado: none
Access-Control-Allow-Headers	Una lista de los encabezados HTTP permitidos para las solicitudes de CORS. El encabezado de respuesta <code>Access-Control-Allow-Headers</code> se devuelve cuando las evaluaciones de CORS se realizan correctamente y especifica de forma explícita todos los <code>Access-Control-Request-Headers</code> permitidos. Tipo: cadena Valor predeterminado: None
Access-Control-Allow-Methods	Una lista que especifica qué métodos HTTP están permitidos. Amazon S3 solo permitirá las solicitudes de CORS de los métodos de CORS permitidos cuando la evaluación de CORS se haya realizado correctamente. Tipo: cadena Valor predeterminado: None
Access-Control-Allow-Origin	La ubicación del origen permitido. Amazon S3 solo enviará el encabezado de respuesta <code>Access-Control-Allow-Origin</code> cuando la evaluación de CORS se haya realizado correctamente. Si el origen de la solicitud

Nombre	Descripción
	coincide con ' * ' en los orígenes permitidos de la configuración de CORS, ' * ' se devuelve en este encabezado de respuesta en lugar del origen original. Tipo: cadena Valor predeterminado: None
Access-Control-Expose-Headers	Una lista que permite a un servidor identificar un encabezado de respuesta que expone el acceso de las aplicaciones cuando la evaluación de CORS se realiza correctamente. Tipo: cadena Valor predeterminado: None
Access-Control-Max-Age	El tiempo en segundos que el navegador puede almacenar en caché la respuesta de una solicitud previa al tránsito de CORS según la identifica el recurso, el método HTTP y el origen. El encabezado de respuesta Access-Control-Max-Age solo se devuelve cuando la evaluación de CORS se haya realizado correctamente. Tipo: número entero Valor predeterminado: none
Vary	Una lista que indica en qué encabezados de solicitud varía el resultado de la evaluación de CORS. El encabezado de respuesta Vary solo se devuelve cuando la evaluación de CORS se haya realizado correctamente. Tipo: cadena Valor predeterminado: None
Content-Length	Longitud del cuerpo de la respuesta en bytes. Tipo: cadena Valor predeterminado: None

Nombre	Descripción
Content-Type	El tipo de MIME del contenido. Por ejemplo, Content-Type: text/html ; charset=utf-8 . Tipo: cadena Valor predeterminado: None
Connection	Un valor que especifica si la conexión con el servidor está abierta o cerrada. Tipo: enumeración Valores válidos: open cclose Valor predeterminado: none
Date	La fecha y la hora en que Amazon S3 respondió; por ejemplo, miércoles 1 de marzo de 2006 a las 12:00:00 GMT. Tipo: cadena Valor predeterminado: None

Nombre	Descripción
ETag	La etiqueta de entidad (ETag) representa una versión específica del objeto. La ETag solo refleja los cambios en el contenido de un objeto, no en sus metadatos. La ETag puede ser o no un resumen MD5 de los datos del objeto. Esto dependerá de cómo se creó el objeto y cómo está cifrado, de la siguiente manera: • Objetos creados mediante la Consola de administración de AWS o mediante el objeto <code>PUT</code>, <code>POST Object</code> o la operación <code>Copy</code>: • Los objetos que son texto sin formato o cifrados mediante cifrado del servidor con claves administradas por Amazon S3 (SSE-S3) tienen ETags que son un resumen de MD5 de los datos. • Los objetos cifrados mediante cifrado del servidor con claves proporcionadas por el cliente (SSE-C) o claves AWS Key Management Service (AWS KMS) (SSE-KMS) tienen ETags que no son un resumen de MD5 de los datos de objeto. • Los objetos creados mediante la operación de carga multiparte o copia de parte de carga tienen ETags que no son resúmenes de MD5, independientemente del método de cifrado. Tipo: cadena
Server	El nombre del servidor que creó la respuesta. Tipo: cadena Valor predeterminado: AmazonS3
x-amz-delete-marker	Un valor que especifica si el objeto devuelto era (<code>true</code>) o no (<code>false</code>) un marcador de eliminación. Tipo: booleano Valores válidos: <code>true</code> <code>false</code> Predeterminado: <code>false</code>

Nombre	Descripción
<code>x-amz-id-2</code>	Un token especial que se utiliza junto con el encabezado <code>x-amz-request-id</code> para ayudar a solucionar problemas de AWS. Para obtener información sobre Soporte mediante el uso de estos ID de solicitud, consulte Solución de problemas de Amazon S3. Tipo: cadena Valor predeterminado: None
<code>x-amz-request-id</code>	Un valor creado por Amazon S3 que identifica de forma exclusiva la solicitud. Este valor se utiliza junto con el encabezado <code>x-amz-id-2</code> para ayudar a solucionar problemas de AWS. Para obtener información sobre Soporte mediante el uso de estos ID de solicitud, consulte Solución de problemas de Amazon S3. Tipo: cadena Valor predeterminado: None
<code>x-amz-server-side-encryption</code>	Algoritmo de cifrado del servidor que se utiliza al almacenar este objeto en Amazon S3 (por ejemplo, AES256, <code>aws:kms</code>). Valores válidos: AES256 <code>aws:kms</code>
<code>x-amz-version-id</code>	La versión del objeto. Si habilita el control de versiones, Amazon S3 genera un número aleatorio para los objetos agregados a un bucket. El valor está codificado en UTF-8 y preparado para URL. Cuando PUT un objeto en un bucket en el que se ha suspendido el control de versiones, el ID de versión siempre es <code>null</code>. Tipo: cadena Valores válidos: <code>null</code> cualquier cadena codificada en UTF-8 y lista para URL Predeterminado: <code>null</code>

Respuestas de error

En esta sección, se ofrece información de referencia acerca de errores de Amazon S3.

Note

- En general, a los propietarios de los buckets de S3 se les facturan las solicitudes con respuestas correctas HTTP 200 OK y respuestas de error del cliente HTTP 4XX. A los propietarios de los buckets no se les facturan las respuestas a errores del servidor HTTP 5XX, como los errores HTTP 503 Slow Down. Para obtener más información sobre los códigos de error de S3 en los códigos de estado HTTP 3XX y 4XX que no se facturan, consulte [Facturación de las respuestas de error de Amazon S3](#) en la Guía del usuario de Amazon S3. Para obtener más información sobre los cargos de facturación si el bucket está configurado como un bucket de pago por solicitante, consulte [Cómo funcionan los cargos de pagos por solicitante](#) en la Guía del usuario de Amazon S3.
- La compatibilidad con SOAP por HTTP está obsoleta, pero SOAP aún se encuentra disponible con HTTPS. Las características nuevas de Amazon S3 no son compatibles con SOAP. En vez de usar SOAP, le recomendamos que utilice la API de REST o los SDK de AWS.

Temas

- [Respuestas de error de REST](#)
- [Lista de códigos de error](#)
- [Lista de códigos de error de contenido de objeto SELECT](#)
- [Lista de códigos de error relacionados con la replicación](#)
- [Lista de códigos de error relacionados con el etiquetado](#)
- [Lista de códigos de error relacionados con las anotaciones de Amazon S3](#)
- [Lista de códigos de error de Amazon S3 en Outposts](#)
- [Lista de códigos de error de Lente de almacenamiento de Amazon S3](#)
- [Lista de códigos de error de Amazon S3 Object Lambda](#)
- [Lista de códigos de error asíncronos de Amazon S3](#)

- [Lista de códigos de error de concesiones de acceso a Amazon S3](#)
- [Lista de códigos de error relacionados con Amazon FSx](#)
- [Lista de códigos de error de tablas de Amazon S3](#)
- [Prácticas recomendadas para los errores de Amazon S3](#)

Respuestas de error de REST

Cuando se produce un error, la información del encabezado contiene lo siguiente:

- Tipo de contenido: aplicación/xml.
- Un código de estado HTTP apropiado de 3xx, 4xx o 5xx.

El cuerpo de la respuesta también contiene información sobre el error. La respuesta de error de ejemplo siguiente, muestra la estructura de los elementos de respuesta comunes a todas las respuestas de error de 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>
```

En la siguiente tabla, se explican los elementos de respuesta de error de REST.

Nombre	Descripción
Code	El código de error es una cadena que identifica de forma exclusiva una condición de error. Está diseñado para que los programas que detectan y administran errores por tipo puedan leerlo y comprenderlo. Para obtener más información, consulte Lista de códigos de error. Tipo: cadena Antecesor: Error

Nombre	Descripción
Error	Contenedor para todos los elementos de error. Tipo: Contenedor Ancestor: ninguno
Message	El mensaje de error contiene una descripción genérica de la condición de error en inglés. Está destinado a un público humano. Los programas simples muestran el mensaje directamente al usuario final si se detecta una condición de error que no sabe cómo controlar o no le interesa hacerlo. Los programas sofisticados con un control de errores más exhaustivo y una internacionalización adecuada tienen más probabilidades de ignorar el mensaje de error. Tipo: cadena Antecesor: Error
RequestId	El ID de la solicitud asociada al error. Tipo: cadena Antecesor: Error
Resource	El bucket o el objeto que está implicado en el error. Tipo: cadena Antecesor: Error

Muchas respuestas de error contienen datos estructurados adicionales diseñados para ser leídos y comprendidos por un desarrollador que diagnostica errores de programación. Por ejemplo, si envía un encabezado Content-MD5 con una solicitud PUT de REST que no coincide con el resumen calculado en el servidor, recibe un error `BadDigest`. La respuesta de error también incluye como elementos de detalle el resumen que el servidor calculó y el resumen que anticipó al servidor. Durante el desarrollo, puede utilizar esta información para diagnosticar el error. En producción, un programa con buen comportamiento puede incluir esta información en el registro de error.

Para obtener información sobre los elementos de respuesta generales, vaya a [Respuestas a errores](#).

Lista de códigos de error

En la tabla siguiente, se muestran los códigos de error de Amazon S3.

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
AccessControlListNotSupported	El bucket no permite las ACL.	400 Bad Request	Cliente
AccessDenied	Acceso denegado	403 Forbidden	Cliente
AccessPointAlreadyOwnedByYou	Ya existe un punto de acceso con un nombre idéntico en la cuenta.	409 Conflict	Cliente
AccountProblem	Hay un problema con la Cuenta de AWS que evita que la operación se complete de forma satisfactoria. Para obtener más ayuda, consulte Contacte con nosotros .	403 Forbidden	Cliente
AllAccessDisabled	Se ha desactivado todo el acceso a este recurso de Amazon S3. Para obtener más ayuda, consulte Contacte con nosotros .	403 Forbidden	Cliente
AmbiguousGrantByEmailAddress	La dirección de correo electrónico que proporcionó está asociada a más de una cuenta.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
AuthorizationHeaderMalformed	El encabezado de autorización que ha proporcionado no es válido.	400 Bad Request	N/A
AuthorizationQueryParametersError	Los parámetros de consulta de autorización que ha proporcionado no son válidos.	400 Bad Request	N/A
BadDigest	El valor de Content-MD5 o de la suma de comprobación que especificó no coincidía con el que recibió el servidor.	400 Bad Request	Cliente
BucketAlreadyExists	El nombre del bucket solicitado no está disponible. Todos los usuarios del sistema comparten el espacio de nombres del bucket. Especifique un nombre diferente e inténtelo de nuevo.	409 Conflict	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
BucketAlreadyOwnedByYou	El bucket que intentó crear ya existe y es de su propiedad. Amazon S3 devuelve este error en todas las Regiones de AWS excepto en la región Este de EE. UU. (Norte de Virginia) (us-east-1). Para garantizar la compatibilidad con versiones anteriores, si vuelve a crear un bucket existente que ya posee en us-east-1, Amazon S3 devuelve 200 OK y restablece las listas de control de acceso (ACL) del bucket. Para Amazon S3 en Outposts, el bucket que intentó crear ya existe en el Outpost y es de su propiedad.	409 Conflict (en todas las regiones excepto us-east-1)	Cliente
BucketHasAccessPointsAttached	El bucket que intentó eliminar tiene puntos de acceso adjuntos. Elimine los puntos de acceso antes de eliminar el bucket.	400 Bad Request	Cliente
BucketNotEmpty	El bucket que ha intentado eliminar no está vacío.	409 Conflict	Cliente
ClientTokenConflict	El token de idempotencia del punto de acceso multirregional ya se utilizó para otra solicitud.	409 Conflict	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
<code>ConnectionClosedByRequester</code>	Se devuelve al autor de la llamada original cuando se detecta un error al leer el cuerpo de <code>WriteGetObjectResponse</code> .	400 Bad Request	Cliente
<code>ConditionalRequestConflict</code>	Se ha producido una operación conflictiva. Si se usa <code>PutObject</code> , puede volver a intentar la solicitud. Si se usa la carga multiparte, debe iniciar otra solicitud <code>CreateMultipartUpload</code> y volver a cargar cada parte.	409 Conflict	Cliente
<code>CredentialsNotSupported</code>	Esta solicitud no admite credenciales.	400 Bad Request	Cliente
<code>CrossLocationLoggingProhibited</code>	No se permite el registro entre regiones. Los buckets de una Región de AWS no pueden registrar información en un bucket de otra región.	403 Forbidden	Cliente
<code>DeviceNotActiveError</code>	El dispositivo no está activo actualmente.	400 Bad Request	Cliente
<code>EndpointNotFound</code>	Dirija las solicitudes al punto de conexión correcto.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
EntityTooSmall	La carga propuesta es inferior al tamaño de objeto mínimo permitido.	400 Bad Request	Cliente
EntityTooLarge	La carga propuesta supera el tamaño de objeto máximo permitido . Para obtener más información, consulte Puntos de conexión y cuotas de Amazon Simple Storage Service en Referencia general de AWS.	400 Bad Request	Cliente
	La descarga propuesta supera el tamaño máximo permitido.	405 Method Not Allowed	Cliente
ExpiredToken	El token que ha proporcionado ha caducado.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
<code>IllegalLocationConstraintException</code>	Este error puede producirse por las siguientes razones: • Está intentando acceder a un bucket desde una región diferente a la de donde se encuentra el bucket. • Se intenta crear un bucket con una restricción de ubicación que corresponde a una región diferente a la del punto de conexión regional al que se envió la solicitud.	400 Bad Request	Cliente
<code>IllegalVersioningConfigurationException</code>	La configuración de control de versiones especificada en la solicitud no es válida.	400 Bad Request	Cliente
<code>IncompleteBody</code>	No proporcionó el número de bytes especificado en el encabezado HTTP de longitud del contenido.	400 Bad Request	Cliente
<code>IncorrectEndpoint</code>	El bucket especificado existe en otra región. Dirija las solicitudes al punto de conexión correcto.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
<code>IncorrectNumberOfFilesInPostRequest</code>	POST requiere que se cargue exactamente un archivo por solicitud.	400 Bad Request	Cliente
<code>InlineDataTooLarge</code>	Los datos insertados superan el tamaño máximo permitido.	400 Bad Request	Cliente
<code>InternalError</code>	Se ha producido un error interno. Inténtelo de nuevo.	500 Internal Server Error	Server
<code>InvalidAccessKeyId</code>	El ID de clave de acceso de AWS proporcionado no existe en nuestros registros.	403 Forbidden	Cliente
<code>InvalidAccessPoint</code>	El nombre o la cuenta del punto de acceso especificado no es válido.	400 Bad Request	Cliente
<code>InvalidAccessPointAliasError</code>	El nombre del alias del punto de acceso especificado no es válido.	400 Bad Request	Cliente
<code>InvalidAddressingHeader</code>	Debe especificar el rol anónimo.	N/A	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidArgument	Este error puede producirse por las siguientes razones: • Una solicitud ListBuckets se realiza a un punto de conexión regional que es diferente de la región especificada en el parámetro <code>bucket-region</code>. • El argumento especificado no era válido. • A la solicitud le faltaba un encabezado obligatorio. • El argumento especificado estaba incompleto o tenía un formato incorrecto. • El argumento especificado debe tener una longitud superior o igual a 3.	400 Bad Request	Cliente
InvalidBucketAclWithObjectOwnership	El bucket no puede tener ACL establecidas con la configuración <code>BucketOwnerEnforced</code> de <code>ObjectOwnership</code> .	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidBucketName	El bucket especificado no es válido.	400 Bad Request	Cliente
InvalidBucketOwner AWSAccountID	El valor del parámetro de propietario del bucket esperado debe ser un ID de Cuenta de AWS.	400 Bad Request	Cliente
InvalidBucketState	La solicitud no es válida para el estado actual del bucket.	409 Conflict	Cliente
InvalidDigest	El valor de Content-MD5 o de la suma de comprobación que especificó no es válido.	400 Bad Request	Cliente
InvalidEncryptionAlgorithmError	La solicitud de cifrado que especificó no es válida. El valor válido es AES256.	400 Bad Request	Cliente
InvalidHostHeader	Los encabezados de host proporcionados en la solicitud utilizaban un estilo de direccionamiento incorrecto.	400 Bad Request	Cliente
InvalidHttpMethod	La solicitud se realiza mediante un método HTTP inesperado.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidLocationConstraint	La restricción de ubicación especificada (región) no es válida. Para obtener más información sobre cómo seleccionar una región para los buckets, consulte Información general de los buckets .	400 Bad Request	Cliente
InvalidObjectState	La operación no es válida para el estado actual del objeto.	403 Forbidden	Cliente
InvalidPart	No se han encontrado una o varias de las partes especificadas. Es posible que la parte no se haya cargado o que la etiqueta de entidad especificada no coincida con la etiqueta de entidad de la parte.	400 Bad Request	Cliente
InvalidPartOrder	La lista de partes no estaba en orden ascendente. La lista de partes se debe especificar en orden por número de parte.	400 Bad Request	Cliente
InvalidPayer	Se ha desactivado todo el acceso a este objeto. Para obtener más ayuda, consulte Contacte con nosotros .	403 Forbidden	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidPolicyDocument	El contenido del formulario no cumple las condiciones especificadas en el documento de la política.	400 Bad Request	Cliente
InvalidRange	No se puede satisfacer el rango solicitado.	El rango solicitado o no es satisfactorio, 416	Cliente
InvalidRegion	Ha intentado crear un punto de acceso multirregional en una región en la que no se ha registrado.	403: prohibido	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidRequest	Este error puede producirse por las siguientes razones: • Se realiza una solicitud ListBuckets sin paginar desde una cuenta que tiene una cuota de bucket de uso general aprobada superior a 10 000. Debe realizar solicitudes paginadas para mostrar los buckets de una cuenta con más de 10 000 buckets. • La solicitud utiliza la versión de firma incorrecta. Use AWS4-HMAC-SHA256 (Signature Version 4). • Solo se puede crear un punto de acceso para un bucket existente. • El punto de acceso no está en un estado en el que se pueda eliminar. • Solo se puede mostrar un punto de acceso para un bucket existente. • El token siguiente no es válido. •	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
	Se debe especificar al menos una acción en una regla del ciclo de vida. • Se debe especificar al menos una regla del ciclo de vida. • El número de reglas del ciclo de vida no debe superar el límite permitido de 1000 reglas. • El rango para el parámetro <code>MaxResults</code> no es válido. • Las solicitudes SOAP deben realizarse a través de una conexión HTTPS. • Los buckets no admiten la aceleración de transferencias de Amazon S3 con nombres no compatibles con DNS. • Los buckets no admiten la aceleración de transferencias de Amazon S3 con puntos (.) en los nombres. •		

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
	El punto de conexión de aceleración de transferencia de Amazon S3 solo admite solicitudes de estilo virtual. • La aceleración de transferencias de Amazon S3 no está configurada en este bucket. • La aceleración de transferencias de Amazon S3 está desactivada en este bucket. • La aceleración de transferencias de Amazon S3 no se admite en este bucket. Para obtener ayuda, póngase en contacto con Soporte. • La aceleración de transferencias de Amazon S3 no se puede habilitar en este bucket. Para obtener ayuda, póngase en contacto con Soporte. • Se proporcionan valores contradictorios en los encabezados HTTP y los parámetros de consulta.		

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
	- Se proporcionan valores contradictorios en los encabezados HTTP y en los campos del formulario POST. - Solicitud CopyObject realizada en objetos de más de 5 GB de tamaño.		
InvalidSessionException	Se devuelve si la sesión ya no existe porque se agotó el tiempo de espera o caducó.	400 Bad Request	Cliente
InvalidSignature	La firma de solicitud que calculó el servidor no coincide con la firma que proporcionó. Compruebe la clave de acceso secreta de AWS y el método de firma. Para obtener más información, consulte Firma y autenticación de solicitudes REST .	400 Bad Request	Cliente
InvalidSecurity	Las credenciales de seguridad proporcionadas no son válidas.	403 Forbidden	Cliente
InvalidSOAPRequest	El cuerpo de la solicitud SOAP no es válido.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
<code>InvalidStorageClass</code>	La clase de almacenamiento que especificó no es válida.	400 Bad Request	Cliente
<code>InvalidTargetBucketForLogging</code>	El bucket de destino para el registro no existe, no es de su propiedad o no tiene las concesiones adecuadas para el grupo de entrega de registros.	400 Bad Request	Cliente
<code>InvalidToken</code>	El token proporcionado es incorrecto o no es válido de otro modo.	400 Bad Request	Cliente
<code>InvalidURI</code>	El URI especificado no se pudo analizar.	400 Bad Request	Cliente
<code>KeyTooLongError</code>	La clave es demasiado larga.	400 Bad Request	Cliente
<code>KMS.DisabledException</code>	La solicitud se rechazó porque la clave KMS especificada no está habilitada.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
KMS.InvalidKeyUsageException	La solicitud se rechazó por una de las siguientes razones: • El valor KeyUsage de la clave KMS no es compatible con la operación de la API. • El algoritmo de cifrado o el algoritmo de firma especificado para la operación no es compatible con el tipo de material clave en la clave de KMS (KeySpec). Para cifrar, descifrar, volver a cifrar y generar claves de datos, KeyUsage debe ser ENCRYPT_DECRYPT. Para firmar y verificar los mensajes, KeyUsage debe ser SIGN_VERIFY. Para generar y verificar los códigos de autenticación de mensajes (MAC), KeyUsage debe ser GENERATE_VERIFY_MAC. Para obtener los secretos de los acuerdos de claves, KeyUsage debe ser KEY_AGREEMENT. Para encontrar KeyUsage de una clave de KMS, use la operación DescribeKey.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
	Para encontrar los algoritmos de cifrado y firma admitidos para una clave de KMS particular, use la operación DescribeKey.		

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
KMS.KMSInvalidStateException	La solicitud se rechazó porque el estado del recurso especificado no es válido para esta solicitud. Esta excepción es una de las siguientes opciones: - El estado de la clave de KMS no es compatible con la operación. Para encontrar el estado de la clave, utilice la operación <code>DescribeKey</code>. Para obtener más información sobre los estados de clave compatibles con cada operación de KMS, consulte Estados de clave de AWS KMS keys en la Guía para desarrolladores de AWS Key Management Service. - Para las operaciones criptográficas en claves de KMS en almacenes de claves personalizadas, esta excepción representa un error general con muchas posibles causas. Para identificar el problema, consulte el mensaje de error que acompaña a la excepción.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
<code>KMS.NotFoundException</code>	La solicitud se rechazó porque la entidad o el recurso especificados no se encontraron.	400 Bad Request	Cliente
<code>MalformedACLError</code>	La ACL proporcionada no tenía un formato válido o no pudo validarse con nuestro esquema publicado.	400 Bad Request	Cliente
<code>MalformedPOSTRequest</code>	El cuerpo de la solicitud POST no contiene datos multipartes/de formulario bien formados.	400 Bad Request	Cliente
<code>MalformedXML</code>	El XML proporcionado no tenía un formato válido o no podía validarse con nuestro esquema publicado.	400 Bad Request	Cliente
<code>MaxMessageLengthExceeded</code>	La solicitud era demasiado grande.	400 Bad Request	Cliente
<code>MaxPostPreDataLengthExceededError</code>	Los campos de solicitud POST que preceden al archivo de carga eran demasiado grandes.	400 Bad Request	Cliente
<code>MetadataTooLarge</code>	Los encabezados de los metadatos superan el tamaño máximo de metadatos permitido.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
MethodNotAllowed	El método especificado no está permitido en este recurso.	405 Method Not Allowed	Cliente
MissingAttachment	Se esperaba un archivo adjunto de SOAP, pero no se encontró ninguno.	400 Solicitud errónea	Cliente
MissingAuthenticationToken	La solicitud no estaba firmada.	403: prohibido	Cliente
MissingContentLength	Debe proporcionar el encabezado HTTP de longitud del contenido.	411 Length Required	Cliente
MissingRequestBodyError	Ha enviado un documento XML vacío como una solicitud.	400 Bad Request	Cliente
MissingSecurityElement	Falta un elemento de seguridad en la solicitud SOAP 1.1.	400 Bad Request	Cliente
MissingSecurityHeader	Falta un encabezado obligatorio en la solicitud.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
NoLoggingStatusForKey	No existe un subrecurso de estado de registro para una clave.	400 Bad Request	Cliente
NoSuchAsyncRequest	No se ha encontrado la solicitud especificada.	404 Not Found	Cliente
NoSuchBucket	El bucket especificado no existe.	404 Not Found	Cliente
NoSuchBucketPolicy	El bucket especificado no tiene una política de bucket.	404 Not Found	Cliente
NoSuchCORSConfiguration	El bucket especificado no tiene una configuración de CORS.	404 Not Found	Cliente
NoSuchKey	La clave especificada no existe.	404 Not Found	Cliente
NoSuchLifecycleConfiguration	La configuración del ciclo de vida especificada no existe.	404 Not Found	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
NoSuchMultiRegionAccessPoint	El punto de acceso multirregional especificado no existe.	404 Not Found	Cliente
NoSuchObjectLockConfiguration	El objeto especificado no tiene una configuración de ObjectLock.	404 Not Found	Cliente
NoSuchWebsiteConfiguration	El bucket especificado no tiene una configuración de sitio web.	404 Not Found	Cliente
NoSuchTagSet	La etiqueta especificada no existe.	404 Not Found	Cliente
NoSuchUpload	La carga multiparte especificada no existe. Es posible que el ID de carga no sea válido o que la carga multiparte se haya cancelado o completado.	404 Not Found	Cliente
NoSuchVersion	El ID de versión especificado en la solicitud no coincide con una versión existente.	404 Not Found	Cliente
NotDeviceOwnerError	El dispositivo que generó el token no es propiedad del usuario autenticado.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
NotImplemented	Un encabezado que ha proporcionado implica una funcionalidad que no está implementada.	501 Not Implemented	Server
NotModified	El recurso no se modificó.	304 Not Modified	Server
NoTransformationDefined	No se ha encontrado la transformación para este punto de acceso de Object Lambda.	404 Not Found	Cliente
NotSignedUp	La cuenta no está inscrita para el servicio de Amazon S3. Debe registrarse antes de poder utilizar Amazon S3. Puede registrarse en la siguiente URL: https://aws.amazon.com/s3	403 Forbidden	Cliente
ObjectLockConfigurationNotFoundError	La configuración del bloqueo de objetos no existe para este bucket.	404 Not Found	Cliente
OwnershipControlsNotFoundError	No se han encontrado los controles de propiedad del bucket.	404 Not Found	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
OperationAborted	Una operación condicional en conflicto está en curso actualmente en este recurso. Inténtelo de nuevo.	409 Conflict	Cliente
PermanentRedirect	El bucket al que intenta acceder se debe direccionar mediante el punto de conexión especificado. Envíe todas las solicitudes futuras a este punto de conexión.	301 Moved Permanently	Cliente
PermanentRedirectControlError	La operación de la API a la que intenta acceder se debe direccionar mediante el punto de conexión especificado. Envíe todas las solicitudes futuras a este punto de conexión.	301 Moved Permanently	Cliente
PreconditionFailed	Al menos una de las condiciones previas que especificó no se cumplía.	412 Precondition Failed	Cliente
Redirect	Redireccionamiento temporal. Se le dirige al bucket mientras el servidor del sistema de nombres de dominio (DNS) se está actualizando.	307 Temporary Redirect	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
RequestHeaderSectionTooLarge	El encabezado de la solicitud y los parámetros de consulta utilizados para realizar la solicitud superan el tamaño máximo permitido.	400 Bad Request	Cliente
RequestIsNotMultipartContent	Una solicitud POST de bucket debe ser del tipo de documento adjunto multiparte o datos de formulario.	412 Precondition Failed	Cliente
RequestTimeout	No se leyó ni se escribió en la conexión de socket al servidor durante el periodo de tiempo de espera.	400 Bad Request	Cliente
RequestTimeTooSkewed	La diferencia entre la hora de la solicitud y la hora del servidor es demasiado grande.	403 Forbidden	Cliente
RequestTorrentOfBucketError	No está permitido solicitar el archivo torrent de un bucket.	400 Bad Request	Cliente
ResponseInterrupted	Se devuelve al autor de la llamada original cuando se detecta un error al leer el cuerpo de WriteGetObjectResponse.	400 Bad Request	Cliente
RestoreAlreadyInProgress	La restauración del objeto ya está en curso.	409 Conflict	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
ServerSideEncryptionConfigurationNotFound	No se encontró la configuración de cifrado del servidor.	400 Bad Request	Cliente
ServiceUnavailable	El servicio no puede gestionar la solicitud.	503 Service Unavailable	Server
SignatureDoesNotMatch	La firma de solicitud que calculó el servidor no coincide con la firma que proporcionó. Compruebe la clave de acceso secreta de AWS y el método de firma. Para obtener más información, consulte Autenticación de REST y Autenticación de SOAP .	403 Forbidden	Cliente
SlowDown	Reduzca la tasa de solicitudes.	503 Slow Down	Server
503 SlowDown	Ralentizar	503 Slow Down	Server
TagPolicyException	La política de etiquetas no permite el valor especificado para la siguiente clave de etiqueta.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
TemporaryRedirect	Se le redirige al bucket mientras el servidor del sistema de nombres de dominio (DNS) se está actualizando.	307 Temporary Redirect	Cliente
TokenCodeInvalidError	El número de serie o el código del token que ha proporcionado no son válidos.	400 Bad Request	Cliente
TokenRefreshRequired	El token proporcionado se debe actualizar.	400 Bad Request	Cliente
TooManyAccessPoints	Ha intentado crear más puntos de acceso de los permitidos para una cuenta. Para obtener más información, consulte Puntos de conexión y cuotas de Amazon Simple Storage Service en Referencia general de AWS.	400 Bad Request	Cliente
TooManyBuckets	Ha intentado crear más buckets de los permitidos para una cuenta. Para obtener más información, consulte Puntos de conexión y cuotas de Amazon Simple Storage Service en Referencia general de AWS.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
TooManyMultiRegionAccessPointregionsError	Ha intentado crear un punto de acceso multirregional con más regiones de las permitidas para una cuenta. Para obtener más información, consulte Puntos de conexión y cuotas de Amazon Simple Storage Service en Referencia general de AWS.	400 Bad Request	Cliente
TooManyMultiRegionAccessPoints	Ha intentado crear más puntos de acceso multirregionales de los permitidos para una cuenta. Para obtener más información, consulte Puntos de conexión y cuotas de Amazon Simple Storage Service en Referencia general de AWS.	400 Bad Request	Cliente
UnauthorizedAccessError	Aplicable solo en las regiones de China. Se devuelve cuando se hace una solicitud a un bucket que no tiene una licencia ICP. Para obtener más información, consulte Registros de ICP .	403 Forbidden	Cliente
UnexpectedContent	Esta solicitud tiene contenido no compatible.	400 Bad Request	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
UnexpectedIPError	Aplicable solo en las regiones de China. Se ha rechazado esta solicitud porque la IP era inesperada.	403 Forbidden	Cliente
UnsupportedArgument	La solicitud contenía un argumento no admitido.	400 Bad Request	Cliente
UnsupportedSignature	La solicitud proporcionada está firmada con una versión de token de STS no compatible o la versión de firma no es compatible.	400 Bad Request	Cliente
UnresolvableGrantByEmailAddress	La dirección de correo electrónico que ha proporcionado no coincide con ninguna de las cuentas registradas.	400 Bad Request	Cliente
UserKeyMustBeSpecified	La solicitud POST del bucket debe contener el nombre de campo especificado. Si se especifica, compruebe el orden de los campos.	400 Bad Request	Cliente
NoSuchAccessPoint	El punto de acceso especificado no existe.	404 Not Found	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidTag	La solicitud contiene una entrada de etiqueta que no es válida. Por ejemplo, es posible que la solicitud contenga claves duplicadas, claves o valores demasiado largos o etiquetas de sistema.	400 Bad Request	Cliente
MalformedPolicy	La política contiene una entidad principal que no es válida.	400 Bad Request	Cliente

Lista de códigos de error de contenido de objeto SELECT

Important

Amazon S3 Select ya no está disponible para los nuevos clientes. Los clientes actuales de Amazon S3 Select pueden seguir utilizando la característica de la forma habitual. [Más información](#)

La tabla siguiente contiene errores especiales que SELECT Object Content podría devolver. Para obtener información general sobre los errores de Amazon S3 y una lista de códigos de error, consulte [Respuestas de error](#).

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
<code>AmbiguousFieldName</code>	El nombre del campo coincide con varios campos del archivo. Compruebe la expresión SQL y el archivo, e inténtelo de nuevo.	400	Cliente
<code>Busy</code>	El servicio no está disponible. Inténtelo de nuevo más tarde.	503	Cliente
<code>CastFailed</code>	Se produjo un error al intentar convertir un tipo de datos a otro mediante CAST en la expresión SQL.	400	Cliente
<code>ColumnTooLong</code>	La longitud de una columna del resultado es superior a <code>maxCharsPerColumn</code> de 1 MB.	400	Cliente
<code>CSVEscapingRecordDelimiter</code>	Se encontró un delimitador de registros entre comillas en el archivo. Para permitir los delimitadores de registros entre comillas, establezca <code>AllowQuotedRecordDelimiter</code> en 'TRUE'.	400	Cliente
<code>CSVParsingError</code>	Se produjo un error al analizar el archivo CSV. Compruebe el archivo e inténtelo de nuevo.	400	Cliente
<code>CSVUnescapedQuote</code>	Se encontró una comilla no escapada al analizar el archivo CSV. Para permitir los delimitadores de registros entre	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
	comillas, establezca <code>AllowQuotedRecordDelimiter</code> en <code>'TRUE'</code> .		
<code>EmptyRequestBody</code>	El cuerpo de la solicitud no puede estar vacío.	400	Cliente
<code>EvaluatorBindingDoesNotExist</code>	El nombre de columna o la ruta proporcionados no existen en la expresión SQL.	400	Cliente
<code>EvaluatorInvalidArguments</code>	Hay un número incorrecto de argumentos en la llamada a la función de la expresión SQL.	400	Cliente
<code>EvaluatorInvalidTimestampFormatPattern</code>	La cadena con formato de marca temporal de la expresión SQL no es válida.	400	Cliente
<code>EvaluatorInvalidTimestampFormatPatternSymbol</code>	El patrón de formato de marca temporal contiene un símbolo en la expresión SQL que no es válido.	400	Cliente
<code>EvaluatorInvalidTimestampFormatPatternSymbolForParsing</code>	El patrón de formato de marca temporal contiene un símbolo de formato válido que no se puede aplicar al análisis de marca temporal en la expresión SQL.	400	Cliente
<code>EvaluatorInvalidTimestampFormatPatternToken</code>	El patrón de formato de marca temporal contiene un token en la expresión SQL que no es válido.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
EvaluatorLikePatternInvalidEscapeSequence	Un argumento proporcionado a la expresión LIKE no era válido.	400	Cliente
EvaluatorNegativeLimit	LIMIT no debe ser negativo.	400	Cliente
EvaluatorTimestampFormatPatternDuplicateFields	El patrón de formato de marca temporal contiene varios especificadores de formato que representan el campo de marca temporal de la expresión SQL.	400	Cliente
EvaluatorTimestampFormatPatternHourClockAmPmMismatch	El patrón de formato de marca temporal contiene un símbolo de formato de hora en formato de 12 horas, pero no contiene un campo AM/PM o contiene un especificador de formato de hora en formato de 24 horas y contiene un campo AM/PM en la expresión SQL.	400	Cliente
EvaluatorUnterminatedTimestampFormatPatternToken	El patrón de formato de marca temporal contiene un token no finalizado en la expresión SQL.	400	Cliente
ExpressionTooLong	La expresión SQL es demasiado larga. La longitud máxima de bytes para una expresión SQL es de 256 KB.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
<code>ExternalEvalException</code>	No se puede evaluar la consulta. Compruebe el archivo e inténtelo de nuevo.	400	Cliente
<code>IllegalSqlFunctionArgument</code>	Se utilizó un argumento ilegal en la función SQL.	400	Cliente
<code>IncorrectSqlFunctionArgumentType</code>	Se especificó un tipo de argumento incorrecto en una llamada a una función de la expresión SQL.	400	Cliente
<code>IntegerOverflow</code>	Se ha producido un desbordamiento o un flujo insuficiente de enteros en la expresión SQL.	400	Cliente
<code>InternalError</code>	Se ha producido un error interno.	500	Cliente
<code>InvalidCast</code>	Se produjo un error al intentar convertir un tipo de datos a otro mediante CAST en la expresión SQL.	400	Cliente
<code>InvalidColumnIndex</code>	El índice de columnas de la expresión SQL no es válido.	400	Cliente
<code>InvalidCompressionFormat</code>	El archivo no tiene un formato de compresión compatible. Solo se admiten GZIP y BZIP2.	400	Cliente
<code>InvalidDataSource</code>	El tipo de origen de datos no es válido. Solo se admiten CSV, JSON y Parquet.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
<code>InvalidDataType</code>	La expresión SQL contiene un tipo de datos que no es válido.	400	Cliente
<code>InvalidExpressionType</code>	El valor <code>ExpressionType</code> no es válido. Solo se admiten las expresiones SQL.	400	Cliente
<code>InvalidFileHeaderInfo</code>	El valor <code>FileHeaderInfo</code> no es válido. Solo se admiten <code>NONE</code> , <code>USE</code> y <code>IGNORE</code> .	400	Cliente
<code>InvalidJsonType</code>	El valor <code>JsonType</code> no es válido. Solo se admiten <code>DOCUMENT</code> y <code>LINES</code> .	400	Cliente
<code>InvalidKeyPath</code>	La ruta clave en la expresión SQL no es válida.	400	Cliente
<code>InvalidQuoteFields</code>	El valor <code>QuoteFields</code> no es válido. Solo se admiten <code>ALWAYS</code> y <code>ASNEEDED</code> .	400	Cliente
<code>InvalidRequestParameter</code>	El valor de un parámetro del elemento <code>SelectRequest</code> no es válido. Consulte la documentación de la API del servicio e inténtelo de nuevo.	400	Cliente
<code>InvalidScanRange</code>	El rango de escaneo proporcionado no es válido.	400	Cliente
<code>InvalidTableAlias</code>	La expresión SQL contiene un alias de tabla que no es válido.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidTextEncoding	El tipo de codificación no es válido. Solo se admite la codificación UTF-8.	400	Cliente
JSONParsingError	Se produjo un error al analizar el archivo JSON. Compruebe el archivo e inténtelo de nuevo.	400	Cliente
LexerInvalidChar	La expresión SQL contiene un carácter que no es válido.	400	Cliente
LexerInvalidIONLiteral	La expresión SQL contiene un operador que no es válido.	400	Cliente
LexerInvalidLiteral	La expresión SQL contiene un operador que no es válido.	400	Cliente
LexerInvalidOperator	La expresión SQL contiene un literal que no es válido.	400	Cliente
LikeInvalidInputs	El argumento dado a la cláusula LIKE en la expresión SQL no es válido.	400	Cliente
MalformedXML	El XML proporcionado no tenía un formato válido o no pudo validarse con nuestro esquema publicado. Consulte la documentación del servicio e inténtelo de nuevo.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
MaxOperatorsExceeded	No se pudo analizar la expresión SQL. Intente reducir la complejidad. Por ejemplo, reduzca el número de operadores utilizados.	400	Cliente
MethodNotAllowed	El método especificado no está permitido en este recurso.	405 Method Not Allowed	Cliente
MissingRequiredParameter	Falta un parámetro obligatorio en la entidad <code>SelectRequest</code> . Consulte la documentación del servicio e inténtelo de nuevo.	400	Cliente
MultipleDataSourcesUnsupported	No se admiten múltiples orígenes de datos.	400	Cliente
NumberFormatError	Se ha producido un error al analizar un número. Este error puede deberse a un desbordamiento o flujo insuficiente de enteros.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
ObjectSerializationConflict	InputSerialization especifica más de un formato (CSV, JSON o Parquet) o OutputSerialization especifica más de un formato (CSV o JSON). Para InputSerialization y OutputSerialization, solo puede especificar un formato para cada uno.	400	Cliente
OverMaxColumn	El número de columnas del resultado es superior al número máximo de columnas permitido.	400	Cliente
OverMaxParquetBlockSize	El archivo Parquet supera el tamaño máximo del grupo de filas.	400	Cliente
OverMaxRecordSize	La longitud de un registro en la entrada o resultado es superior al límite de maxCharsPerRecord de 1 MB.	400	Cliente
ParquetParsingError	Se produjo un error al analizar el archivo Parquet. Compruebe el archivo e inténtelo de nuevo.	400	Cliente
ParquetUnsupportedCompressionCodec	El códec de compresión de Parquet especificado no es compatible.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
ParseAsteriskIsNotAloneInSelectList	No se permiten otras expresiones en la lista SELECT cuando * se utiliza sin notación de puntos en la expresión SQL.	400	Cliente
ParseCannotMixSqbandWildcardInSelectList	No se puede mezclar [] y * en la misma expresión en una lista SELECT de la expresión SQL.	400	Cliente
ParseCastArity	La expresión SQL CAST tiene una aridad incorrecta.	400	Cliente
ParseEmptySelect	La expresión SQL contiene una cláusula SELECT vacía.	400	Cliente
ParseExpected2TokenTypes	No se encontró el token esperado en la expresión SQL.	400	Cliente
ParseExpectedArgumentDelimiter	No se encontró el delimitador del argumento esperado en la expresión SQL.	400	Cliente
ParseExpectedDatePart	No se encontró la parte de la fecha esperada en la expresión SQL.	400	Cliente
ParseExpectedExpression	No se encontró la expresión SQL esperada.	400	Cliente
ParseExpectedIdentifierForAlias	No se encontró el identificador esperado para el alias de la expresión SQL.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
ParseExpectedIdentForAt	No se encontró el identificador esperado para el nombre de AT de la expresión SQL.	400	Cliente
ParseExpectedIdentForGroupName	GROUP no se admite en la expresión SQL.	400	Cliente
ParseExpectedKeyword	No se encontró la palabra clave esperada en la expresión SQL.	400	Cliente
ParseExpectedLeftParenAfterCast	No se encontró el paréntesis izquierdo después de CAST en la expresión SQL.	400	Cliente
ParseExpectedLeftParenBuiltinFunctionCall	No se encontró el paréntesis izquierdo en la expresión SQL.	400	Cliente
ParseExpectedLeftParenValueConstructor	No se encontró el paréntesis izquierdo en la expresión SQL.	400	Cliente
ParseExpectedMember	La expresión SQL contiene un uso no admitido de MEMBER.	400	Cliente
ParseExpectedNumber	No se encontró el número esperado en la expresión SQL.	400	Cliente
ParseExpectedRightParenBuiltinFunctionCall	No se encontró el carácter de paréntesis derecho en la expresión SQL.	400	Cliente
ParseExpectedTokenType	No se encontró el token esperado en la expresión SQL.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
ParseExpectedTypeName	No se encontró el nombre del tipo esperado en la expresión SQL.	400	Cliente
ParseExpectedWhenClause	No se encontró la cláusula WHEN esperada en la expresión SQL. CASE no se admite.	400	Cliente
ParseInvalidContextForWildcardInSelectList	El uso de * en la lista SELECT en la expresión SQL no es válido.	400	Cliente
ParseInvalidPathComponent	La expresión SQL contiene un componente de ruta que no es válido.	400	Cliente
ParseInvalidTypeParameter	La expresión SQL contiene un valor de parámetro que no es válido.	400	Cliente
ParseMalformedJoin	JOIN no se admite en la expresión SQL.	400	Cliente
ParseMissingIdentifierAfterAt	No se encontró el identificador esperado después del símbolo @ en la expresión SQL.	400	Cliente
ParseNonUnaryAggregateFunctionCall	Solo se admite un argumento para las funciones de agregación en la expresión SQL.	400	Cliente
ParseSelectMissingFrom	La expresión SQL contiene un FROM que falta después de la lista SELECT.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
ParseUnexpectedKeyword	La expresión SQL contiene una palabra clave inesperada.	400	Cliente
ParseUnexpectedOperator	La expresión SQL contiene un operador inesperado.	400	Cliente
ParseUnexpectedTerm	La expresión SQL contiene un término inesperado.	400	Cliente
ParseUnexpectedToken	La expresión SQL contiene un token inesperado.	400	Cliente
ParseUnknownOperator	La expresión SQL contiene un operador que no es válido.	400	Cliente
ParseUnsupportedAlias	La expresión SQL contiene un uso no admitido de ALIAS.	400	Cliente
ParseUnsupportedCallWithStar	En la expresión SQL, solo se admite COUNT con (*) como parámetro.	400	Cliente
ParseUnsupportedCase	La expresión SQL contiene un uso no admitido de CASE.	400	Cliente
ParseUnsupportedCaseClause	La expresión SQL contiene un uso no admitido de CASE.	400	Cliente
ParseUnsupportedLiteralsGroupBy	La expresión SQL contiene un uso no admitido de GROUP BY .	400	Cliente
ParseUnsupportedSelect	La expresión SQL contiene un uso no admitido de SELECT.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
ParseUnsupportedSyntax	La expresión SQL contiene sintaxis no admitida.	400	Cliente
ParseUnsupportedToken	La expresión SQL contiene un token no admitido.	400	Cliente
TruncatedInput	Error al descomprimir el objeto. Compruebe que el objeto está comprimido correctamente con el formato especificado en la solicitud.	400	Cliente
UnauthorizedAccess	No está autorizado para realizar esta operación.	401	Cliente
UnrecognizedFormatException	Hemos encontrado un tipo de registro que no es válido.	400	Cliente
UnsupportedFunction	Hemos encontrado una función SQL no compatible.	400	Cliente
UnsupportedParquetType	El tipo de Parquet especificado no se admite.	400	Cliente
UnsupportedRangeHeader	No se admite un encabezado de rango para esta operación.	400	Cliente
UnsupportedScanRangeInput	Las consultas de rangos de escaneo no se admiten en este tipo de objeto.	400	Cliente
UnsupportedSqlOperation	Hemos encontrado una operación SQL no compatible.	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
UnsupportedSqlStructure	Hemos encontrado una estructura SQL no compatible. Compruebe la referencia de SQL.	400	Cliente
UnsupportedStorageClass	Hemos encontrado una clase de almacenamiento que no es compatible. Solo se admiten las clases de almacenamiento STANDARD, STANDARD_IA y ONEZONE_IA.	400	Cliente
UnsupportedSyntax	Hemos encontrado una sintaxis no válida.	400	Cliente
UnsupportedTypeForQuerying	La consulta contiene un tipo no compatible para la comparación (por ejemplo, comprobar que un tipo de columna INT96 de Parquet es mayor que 0).	400	Cliente
ValueParseFailure	Se ha producido un error de análisis de la marca temporal en la expresión SQL.	400	Cliente

Lista de códigos de error relacionados con la replicación

La tabla siguiente contiene errores especiales que la operación `Replication` podría devolver. Para obtener información general sobre los errores de Amazon S3 y una lista de códigos de error, consulte [Respuestas de error](#).

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidArgument	Este error puede producirse por las siguientes razones: • El elemento <code><Account></code> está vacío. Debe contener un ID de cuenta válido. • La Cuenta de AWS especificada en el elemento <code><Account></code> debe coincidir con el propietario del bucket de destino. • <code>ReplicationTime-Status</code> debe contener un valor. • <code>ReplicationTime-ReplicationTimeValue</code> debe contener un valor. • El valor <code>Replication-ReplicationTimeValue-Minutes</code> debe ser 15. • <code>ReplicationMetrics</code> debe contener un <code>Status</code>. • <code>ReplicationMetrics</code> debe contener un <code>EventThreshold</code>. •	400	Cliente

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
	El valor <code>EventThreshold-ReplicationTimeValue-Minutes</code> debe ser 15. • <code>Rule ID</code> no debe contener caracteres que no sean ASCII.		

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidRequest	Este error puede producirse por las siguientes razones: • <code><Owner></code> en <code><AccessControlTranslation></code> tiene un valor, por lo que se debe especificar el elemento <code><Account></code> . • El elemento <code><Account></code> está vacío. Debe contener un ID de cuenta válido. • La replicación de <code>destination</code> debe contener <code>ReplicationTime</code> y <code>Metrics</code> o ninguno de los dos. • <code>ReplicationTime</code> y <code>ReplicationMetrics</code> deben tener el mismo estado. • Esta Región de AWS no admite el control del tiempo de replicación de S3 (S3 RTC).	400	Cliente
ReplicationConfigurationNotFoundError	No hay ninguna configuración de replicación para este bucket.	404 Not Found (No encontrado)	Cliente

Lista de códigos de error relacionados con el etiquetado

La tabla siguiente contiene errores especiales que las operaciones `TagResource`, `UntagResource` y `ListTagsForResource` podrían devolver para grupos de lentes de almacenamiento. Para obtener información general sobre los errores de Amazon S3 generales y una lista de códigos de error, consulte [Respuestas de error](#).

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
<code>InvalidRequest</code>	La Región de AWS del ARN del recurso no coincide con la región especificada en esta solicitud. La cuenta de AWS del ARN del recurso no coincide con el ID de la cuenta especificada en esta solicitud. La partición de AWS en <code>resourceArn</code> no es válida.	400 Bad Request	No admitido
<code>InvalidTag</code>	Esta solicitud contiene una clave o un valor de etiqueta que no es válido. Los caracteres válidos incluyen lo siguiente: <code>[a-zA-Z+-=._:/]</code> . Las claves de etiqueta pueden contener un máximo de 128 caracteres. Los valores de etiqueta pueden contener un máximo de 256 caracteres. Hay claves de etiquetas duplicadas en la solicitud. Las claves de etiquetas definidas por el usuario no pueden empezar por <code>aws:</code> .	400 Bad Request	No admitido
<code>NoSuchResource</code>	El recurso especificado no existe.	404 Not Found	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
TagPolicyException	La política de etiquetas no permite el valor especificado para la siguiente clave de etiqueta.	400 Bad Request	No admitido
TooManyTags	El número de etiquetas supera el límite de 50 etiquetas.	400 Bad Request	No admitido

Lista de códigos de error relacionados con las anotaciones de Amazon S3

La tabla siguiente contiene errores especiales que pueden devolver las operaciones de anotación (PutObjectAnnotation, GetObjectAnnotation, ListObjectAnnotations, DeleteObjectAnnotation, UpdateBucketMetadataAnnotationTableConfiguration) y el comportamiento de CopyObject relacionado con las anotaciones. Para obtener información general sobre los errores de Amazon S3 y una lista de códigos de error, consulte [Respuestas de error](#).

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
AccessDenied	Acceso denegado. La persona que llama no tiene permiso para realizar la operación de anotación solicitada o el objeto está protegido por Bloqueo de objetos en el modo COMPLIANCE.	403 Forbidden	No admitido
AnnotationLimitExceeded	El recuento de anotaciones de objetos supera el límite (1000).	400 Bad Request	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
AnnotationNameTooLong	El nombre de la anotación es demasiado largo. Los nombres de las anotaciones pueden tener hasta 512 bytes de longitud.	400 Bad Request	No admitido
AnnotationRequestConflict	Una operación en conflicto está en curso actualmente en esta anotación. Esto puede ocurrir cuando se elimina una anotación de origen durante una copia entre regiones.	409 Conflict	No admitido
EntityTooSmall	La carga propuesta es inferior al tamaño mínimo permitido. La carga útil de la anotación debe ser de al menos 1 byte.	400 Bad Request	No admitido
InvalidAnnotationName	El nombre de la anotación contiene caracteres no válidos o comienza con un prefijo reservado (aws o s3, sin distinción entre mayúsculas y minúsculas). Los nombres de las anotaciones solo pueden contener letras Unicode, números, guiones bajos (_), puntos (.) y guiones (-).	400 Bad Request	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidArgument	La directiva de anotación no es válida. Los valores válidos son COPY y EXCLUDE. Para ListObjectAnnotations , max-annotation-results debe estar comprendido entre 1 y 1000.	400 Bad Request	No admitido
InvalidPrefix	El prefijo de anotación proporcionado no es válido.	400 Bad Request	No admitido
InvalidRequest	Este error se puede producir por los siguientes motivos: el objeto está cifrado con SSE-C, que no admite anotaciones. El objeto de origen tiene anotaciones que no se pueden copiar en el destino porque el destino se encuentra en una región no compatible. La solicitud de configuración de la tabla de anotaciones no es válida (por ejemplo, desactivar la tabla mientras se especifica el cifrado, habilitar una tabla que ya está habilitada, actualizar una tabla en estado CREATING o enviar una solicitud vacía).	400 Bad Request	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
MethodNotAllowed	El método especificado no está permitido en este recurso. Por ejemplo, este error se produce cuando la solicitud se dirige a un marcador de eliminación.	405 Method Not Allowed	No admitido
NoSuchAnnotation	La anotación especificada no existe.	404 Not Found	No admitido
NoSuchKey	La clave especificada no existe.	404 Not Found	No admitido
NoSuchVersion	El ID de versión especificado en la solicitud no coincide con una versión existente.	404 Not Found	No admitido
NotImplemented	No se admiten anotaciones para los buckets de directorio de Amazon S3.	501 Not Implemented	No admitido
OperationAborted	Una operación condicional en conflicto está en curso actualmente en este recurso. Esto puede ocurrir durante modificaciones simultáneas o cuando el recuento de anotaciones de origen o la ETag cambian durante una copia entre regiones. Inténtelo de nuevo.	409 Conflict	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
PreconditionFailed	Al menos una de las condiciones previas que especificó no se cumplía. La ETag <code>x-amz-object-if-match</code> no coincide con la ETag del objeto actual.	412 Precondition Failed	No admitido
UnsupportedMediaType	La carga útil de la solicitud contiene un tipo de medio no compatible. La carga útil de anotación debe ser texto codificado en UTF-8 válido.	415 Unsupported Media Type	No admitido
UserAnnotationNameMustBeSpecified	Debe especificar un nombre de anotación de usuario.	400 Bad Request	No admitido

Lista de códigos de error de Amazon S3 en Outposts

La tabla siguiente contiene errores especiales que una operación de Amazon S3 en Outposts podría devolver. Para obtener información general sobre los errores de Amazon S3 y una lista de códigos de error, consulte [Respuestas de error](#).

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
BadRequest	El bucket se encuentra en un estado de transición debido a un intento de eliminación anterior. Inténtelo de nuevo más tarde.	400 Solicitud errónea	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidRequest	Este error puede producirse por las siguientes razones: - La configuración de Amazon VPC es obligatoria. - No se permite el acceso público en los puntos de acceso de S3 en Outposts.	400 Solicitud errónea	Cliente
InvalidOutpostState	La solicitud no es válida para el estado actual del Outpost.	Conflicto, 409	No admitido
InvalidRequest	El punto de acceso no está en un estado en el que se pueda eliminar.	400 Solicitud errónea	No admitido
NoSuchOutpost	El Outpost especificado no existe.	404 Not Found (No encontrado)	No admitido
UnsupportedOperation	La acción especificada no era compatible.	404 Not Found (No encontrado)	No admitido
InsufficientCapacity	Capacidad insuficiente.	507 Almacenamiento insuficiente	No admitido

Lista de códigos de error de Lente de almacenamiento de Amazon S3

La tabla siguiente contiene errores especiales que las operaciones de Lente de almacenamiento de Amazon S3 podrían devolver. Para obtener información general sobre los errores de Amazon S3 generales y una lista de códigos de error, consulte [Respuestas de error](#).

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
AccessDenied	Esta región no es compatible como región de origen para Lente de almacenamiento de S3.	403 Forbidden	No admitido
AccountNotAuthorized	Esta cuenta no está autorizada para usar AWS Organizations. Use la cuenta de administración o la cuenta de administrador delegado.	403 Forbidden	No admitido
ActivityMetricsMustEnabled	Las métricas de actividad deben estar habilitadas.	400 Solicitud errónea	No admitido
AWSOrganizationsNotInUseException	Esta cuenta ahora no es parte de la organización.	403 Forbidden	No admitido
DefaultConfigurationDeleteForbidden	La configuración predeterminada no se puede eliminar.	403 Forbidden	No admitido
DuplicateStorageLensGroupARN	Hay dos o más entradas del mismo ARN de grupo de lentes de	400 Bad Request	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
	almacenamiento en esta configuración.		
EmptyExcludeContainer	Este error se produce por las razones siguientes: • El contenedor de exclusión no puede estar vacío. • El contenedor de exclusión no puede tener ningún bucket. • El contenedor de exclusión no puede tener ninguna región.	400 Solicitud errónea	No admitido
EmptyExcludeElement	Debe especificar un grupo de lentes de almacenamiento con el elemento de exclusión.	400 Bad Request	No admitido
EmptyIncludeContainer	Este error se produce por las razones siguientes: • El contenedor de inclusión no puede estar vacío. • El contenedor de inclusión no puede tener ningún bucket. • El contenedor de inclusión no puede tener ninguna región.	400 Solicitud errónea	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidAWSOrgArn	Hay un ARN de AWS Organizations con formato incorrecto en la configuración.	400 Solicitud errónea	No admitido
EmptyIncludeElement	Debe especificar un grupo de lentes de almacenamiento con el elemento de inclusión.	400 Bad Request	No admitido
InvalidBucketFilter	Las configuraciones por organización no admiten filtros de bucket.	400 Solicitud errónea	No admitido
InvalidConfigId	El ID de la configuración no es válido.	400 Solicitud errónea	No admitido
InvalidDestination	El ARN del bucket de S3 tiene un formato incorrecto.	400 Solicitud errónea	No admitido
InvalidEncryptionMethod	Solo puede especificarse un método de cifrado.	400 Solicitud errónea	No admitido
InvalidFilterForDefaultConfiguration	La configuración predeterminada no debe incluir ningún filtro.	400 Solicitud errónea	No admitido
InvalidIncludeExcludeContainers	Puede especificar un contenedor de inclusión o un contenedor de exclusión en una configuración. No puede especificar ambos en una configuración.	400 Solicitud errónea	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
<code>InvalidIncludeExcludeElements</code>	Solo se permite un elemento de inclusión o exclusión. Debe haber al menos un elemento de inclusión o exclusión.	400 Bad Request	No admitido
<code>InvalidKMSEncryptionKeyId</code>	El ARN del ID de clave de KMS no es válido.	400 Solicitud errónea	No admitido
<code>InvalidMaximumPrefixDepth</code>	<code>MaxDepth</code> debe estar dentro del rango [1,10].	400 Solicitud errónea	No admitido
<code>InvalidMinimumStorageBytesPercentage</code>	<code>MinStorageBytesPercentage</code> debe estar dentro del rango [1.00,100.00].	400 Solicitud errónea	No admitido
<code>InvalidOrganizationARN</code>	El ARN de AWS Organizations en la configuración no es válido.	400 Solicitud errónea	No admitido
<code>InvalidOrganizationForDefaultConfiguration</code>	La configuración predeterminada no admite métricas por organización.	400 Solicitud errónea	No admitido
<code>InvalidRegionForDefaultConfiguration</code>	La región especificada no es compatible con la configuración predeterminada.	400 Solicitud errónea	No admitido
<code>InvalidRegionName</code>	El nombre de la región no es válido.	400 Solicitud errónea	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidStorageLensArn	El ARN de Lente de almacenamiento de S3 no es necesario en la entrada.	400 Solicitud errónea	No admitido
InvalidStorageLensGroupARN	El ARN de este grupo de lentes de almacenamiento no es válido o solo se permiten los grupos de lentes de almacenamiento en la cuenta. Además, debe seguir la estructura de ARN del grupo de lentes de almacenamiento: <code>arn::s3:::storage-lens-group/</code> y cumplir con el límite de 64 caracteres. Los nombres de grupos de lente de almacenamiento también pueden contener solo los siguientes caracteres: a-z, A-Z, 0-9, guiones (-) y guiones bajos (_).	400 Bad Request	No admitido
MissingAccountLevelActivityMetrics	Las métricas de actividad deben estar habilitadas por cuenta cuando las métricas de actividad están habilitadas por bucket.	400 Solicitud errónea	No admitido
MissingBucketLevelActivityMetrics	Las métricas de actividad deben estar habilitadas por bucket cuando las métricas de actividad están habilitadas por cuenta.	400 Solicitud errónea	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
MissingEncryptionMethod	El método de cifrado no puede estar en blanco. Especifique SSE-KMS o SSE-S3.	400 Solicitud errónea	No admitido
MissingPrefixLevelStorageMetrics	Las métricas de almacenamiento por prefijo son obligatorias cuando el nivel de prefijo está activado.	400 Solicitud errónea	No admitido
OrganizationAccessDenied	Esta cuenta no está autorizada para agregar AWS Organizations.	403: prohibido	No admitido
OrgConfigurationNotSupported	La región especificada no es compatible con AWS Organizations en la configuración.	403: prohibido	No admitido
ServiceNotEnabledForOrg	El rol vinculado al servicio de Lente de almacenamiento de S3 no está habilitado para la organización.	403: prohibido	No admitido
StorageMetricsMustEnabled	Las métricas de almacenamiento por prefijo deben estar habilitadas.	400 Solicitud errónea	No admitido
TooManyBuckets	El contenedor de buckets no puede tener más de 50 buckets.	400 Solicitud errónea	No admitido
TooManyRegions	El contenedor de regiones no puede tener más de 50 regiones.	400 Solicitud errónea	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
TooManyStorageLensGroups	No puede asociar más de 50 grupos de lentes de almacenamiento al panel de lente de almacenamiento.	400 Bad Request	No admitido

La tabla siguiente contiene errores especiales que las operaciones de grupos de lentes de almacenamiento de S3 podrían devolver. Para obtener información general sobre los errores de Amazon S3 generales y una lista de códigos de error, consulte [Respuestas de error](#).

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
AccessDenied	No tiene permiso para llevar a cabo acciones de grupo de lentes de almacenamiento. Esta región no es compatible como región de origen para grupos de lentes de almacenamiento de S3.	403 Forbidden	No admitido
ConfigurationAlreadyExists	La configuración especificada ya existe.	409 Conflict	No admitido
DuplicateElement	Las etiquetas deben ser únicas. El operador lógico And incluye claves de etiquetas duplicadas. El operador lógico Or incluye etiquetas duplicadas. El operador lógico incluye prefijos o sufijos duplicados.	400 Bad Request	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidAge	DaysLessThan y DaysGreaterThan deben ser números positivos.	400 Bad Request	No admitido
InvalidFilter	Un filtro debe incluir uno de los siguientes elementos: And, Or, MatchAnyTag , MatchAnyPrefix , MatchAnySuffix , MatchObjectAge , MatchObjectSize .	400 Bad Request	No admitido
InvalidLogicalOperator	Deben estar presentes al menos dos subelementos en los operadores lógicos And o Or.	400 Bad Request	No admitido
InvalidMatchAnyPrefix	El parámetro MatchAnyPrefix no puede estar vacío.	400 Bad Request	No admitido
InvalidMatchAnySuffix	El parámetro MatchAnySuffix no puede estar vacío.	400 Bad Request	No admitido
InvalidMatchAnyTag	El parámetro MatchAnyTag no puede estar vacío.	400 Bad Request	No admitido
InvalidMatchObjectAge	El parámetro MatchObjectAge no puede estar vacío.	400 Bad Request	No admitido
InvalidMatchObjectSize	El parámetro MatchObjectSize no puede estar vacío.	400 Bad Request	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
<code>InvalidName</code>	El parámetro de nombre del grupo de lentes de almacenamiento debe tener entre 1 y 64 caracteres. El parámetro Name del grupo de lentes de almacenamiento debe usar el patrón <code>^[a-zA-Z0-9\-_]+\$</code> .	400 Bad Request	No admitido
<code>InvalidNumericCombination</code>	Esta combinación de antigüedad o tamaño de objeto no es válida.	400 Bad Request	No admitido
<code>InvalidPrefix</code>	La longitud máxima de un prefijo es 1024 caracteres. La cadena de prefijo no puede estar vacía.	400 Bad Request	No admitido
<code>InvalidSize</code>	<code>BytesLessThan</code> y <code>BytesGreaterThan</code> deben ser números positivos. El tamaño máximo del objeto no puede superar los 50 TB. El tamaño mínimo del objeto no puede ser igual o superior a 50 TB.	400 Bad Request	No admitido
<code>InvalidSuffix</code>	La longitud máxima de un sufijo es 1024 caracteres. La cadena de sufijo no puede estar vacía.	400 Bad Request	No admitido

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
InvalidTag	La clave de la etiqueta del objeto no puede superar los 128 caracteres. La clave de la etiqueta del objeto no puede ser nula ni estar vacía. La longitud máxima de un valor de etiqueta es de 256 caracteres. La clave de la etiqueta del objeto contiene caracteres que no son válidos. La clave de la etiqueta del objeto debe contener solo a-z, A-Z, 0-9, espacios y los siguientes caracteres: ^(_ . : / = + \ - @] *) \$.	400 Bad Request	No admitido
MismatchedName	El nombre especificado en la solicitud no coincide con el nombre del grupo de lentes de almacenamiento.	400 Bad Request	No admitido
TooManyConfigurations	Ha intentado crear más configuraciones de grupos de lentes de almacenamiento de las 50 permitidas.	400 Bad Request	No admitido
TooManyElements	El Element supera el número máximo de elementos permitidos en un operador lógico. Solo se permiten 10 prefijos, sufijos o etiquetas.	400 Bad Request	No admitido

Lista de códigos de error de Amazon S3 Object Lambda

La tabla siguiente contiene errores especiales que S3 Object Lambda podría devolver. Para obtener información sobre los errores de Amazon S3 generales y una lista de códigos de error, consulte [Respuestas de error](#).

Las respuestas de error que se reciben de los puntos de acceso compatibles cuando no se ha realizado ninguna solicitud `GetObject` se envían a la persona que llama sin cambios.

Código de error	Descripción	Código de estado HTTP
<code>LambdaInvalidResponse</code>	Se devuelve a la persona que llama original cuando <code>WriteGetObjectResponse</code> responde con <code>ValidationError</code> a AWS Lambda. Consulte el mensaje <code>ValidationError</code> para obtener más información. No todos los casos de <code>ValidationError</code> dan lugar a un error <code>LambdaInvalidResponse</code>.	400 Bad Request
<code>LambdaInvocationFailed</code>	Se ha producido un error en la invocación de la función de Lambda. Las personas que llaman pueden recibir el siguiente error cuando S3 Object Lambda no puede invocar correctamente la función de Lambda configurada. El mensaje de error puede contener detalles sobre un posible error devuelto por el servicio de AWS Lambda al invocar la función (por	400 Bad Request

Código de error	Descripción	Código de estado HTTP	
	ejemplo, código de estado, código de error, mensaje de error e ID de solicitud).		
LambdaNotFound	No se ha encontrado la función de AWS Lambda. La función de Lambda configurada, la versión o el alias no se encontraron al intentar invocarla. Asegúrese de que la configuración del punto de acceso de S3 Object Lambda apunte al ARN correcto de la función de Lambda. El mensaje de error puede contener detalles sobre un posible error devuelto por el servicio de AWS Lambda al invocar la función (por ejemplo, código de estado, código de error, mensaje de error e ID de solicitud).	404 Not Found	

Código de error	Descripción	Código de estado HTTP	
LambdaPermissionError	La persona que llama no está autorizada a invocar la función de Lambda. La persona que llama debe tener permiso para invocar la función de Lambda. Compruebe las políticas adjuntas a la persona que llama y asegúrese de que se le haya permitido utilizar <code>Lambda:Invoke</code> para la función configurada. El mensaje de error puede contener detalles sobre un posible error devuelto por el servicio de AWS Lambda al invocar la función (por ejemplo, código de estado, código de error, mensaje de error e ID de solicitud).	403 Forbidden	

Código de error	Descripción	Código de estado HTTP	
LambdaResponseNotReceived	La función de Lambda se cerró sin llamar correctamente a <code>WriteGetObjectResponse</code> . Los datos de respuesta de <code>GetObject</code> los proporciona la función de Lambda al llamar a la operación de la API <code>WriteGetObjectResponse</code> . Los registros de Amazon CloudWatch de la función podrían proporcionar más información sobre por qué la función no llamó correctamente a esta operación de API a pesar de salir con normalidad.	500 Internal Service Error	
LambdaRuntimeError	La función de Lambda produjo un error durante la ejecución. Se recibió un error explícito de la función de Lambda. Para obtener detalles sobre el error, consulte los registros de AWS CloudFormation.	500 Internal Service Error	
LambdaTimeout	La función de Lambda no respondió en el tiempo permitido. La función de Lambda no pudo completar su llamada a <code>WriteGetObjectResponse</code> en 60 segundos.	500 Internal Service Error	

Código de error	Descripción	Código de estado HTTP	
SlowDown	Reduzca la tasa de solicitudes para las operaciones que impliquen AWS Lambda. La invocación de la función se ha visto limitada por AWS Lambda, tal vez porque ha alcanzado el límite de simultaneidad configurado. Para obtener más información, consulte Administración de simultaneidad para una función de Lambda en la Guía para desarrolladores de AWS Lambda. El mensaje de error puede contener detalles sobre un posible error devuelto por el servicio de AWS Lambda al invocar la función (por ejemplo, código de estado, código de error, mensaje de error e ID de solicitud).	503 Slow Down	
ValidationError	Es posible que la operación de la API WriteGetObjectResponse devuelva errores de validación y se produzcan por varios motivos. Consulte el mensaje de error para obtener más información.	400 Bad Request	

Lista de códigos de error asíncronos de Amazon S3

La tabla siguiente contiene errores especiales que las solicitudes asíncronas podrían devolver. Para obtener información general sobre los errores de Amazon S3 y una lista de códigos de error, consulte [Respuestas de error](#).

Estos errores se devuelven al consultar el estado de una solicitud asíncrona, por ejemplo, al utilizar `DescribeMultiRegionAccessPointOperation`. Como estas solicitudes son asíncronas, todos estos errores tienen un código de estado de 200 OK.

Código de error	Descripción	Código de estado HTTP	
<code>AccessDenied</code>	Acceso denegado.	200 OK	
<code>InternalErrors</code>	Se ha producido un error interno del servidor.	200 OK	
<code>MalformedPolicy</code>	La sintaxis de la política especificada no es válida.	200 OK	
<code>MultiRegionAccessPointAlreadyOwnedByYou</code>	Ya tiene un punto de acceso multirregional con el mismo nombre.	200 OK	
<code>MultiRegionAccessPointModifiedByAnotherRequest</code>	La acción produjo un error porque otra solicitud está modificando el recurso especificado. Intente volver a enviar la solicitud después de que se haya completado la solicitud anterior.	200 OK	
<code>MultiRegionAccessPointNotReady</code>	El punto de acceso multirregional especificado no está preparado para actualizarse.	200 OK	
<code>MultiRegionAccessPointSameBucketRegion</code>	Los buckets utilizados para crear un punto de acceso multirregional no pueden estar en la misma región.	200 OK	
<code>MultiRegionAccessPointUnsupportedRegion</code>	Uno de los buckets suministrados para crear el punto de acceso multirregional se encuentra en una región que no es compatible.	200 OK	

Código de error	Descripción	Código de estado HTTP	
NoSuchBucket	El bucket especificado no existe.	200 OK	
NoSuchMultiRegionAccessPoint	El punto de acceso multirregional especificado no existe.	200 OK	

Lista de códigos de error de concesiones de acceso a Amazon S3

La tabla siguiente contiene errores especiales que las solicitudes de concesiones de acceso a S3 podrían devolver. Para obtener información general sobre los errores de Amazon S3 y una lista de códigos de error, consulte [Respuestas de error](#).

Código de error	Descripción	Código de estado HTTP	
AccessGrantAlreadyExists	La concesión de acceso especificada ya existe	409	
AccessGrantsInstanceAlreadyExists	La instancia de concesiones de acceso ya existe	409	
AccessGrantsInstanceNotEmptyError	Limpie las ubicaciones antes de eliminar la instancia de concesiones de acceso	400	
AccessGrantsInstanceNotExistsError	La instancia de concesiones de acceso no existe	404	
AccessGrantsInstanceResourcePolicyNotExists	La política de recursos de la instancia de concesiones de acceso no existe	404	
AccessGrantsLocationAlreadyExistsError	La ubicación de las concesiones de acceso especificada ya existe	409	

Código de error	Descripción	Código de estado HTTP	
AccessGrantsLocationNotEmptyError	Limpie las concesiones de acceso antes de eliminar la ubicación de concesiones de acceso	400	
AccessGrantsLocationsQuotaExceededError	Se ha superado la cuota de ubicaciones de concesiones de acceso. Cuota de ubicaciones de concesiones de acceso: <i><value></i> . Comuníquese con S3 si se necesita un aumento.	409	
AccessGrantsQuotaExceededError	Se ha superado la cuota de concesiones de acceso. Cuota de concesiones de acceso: <i><value></i> . Comuníquese con S3 si se necesita un aumento.	409	
InvalidTag	Hay claves de etiquetas duplicadas en la solicitud. Elimine las claves de etiquetas duplicadas e inténtelo de nuevo.	400	
InvalidAccessGrant	La concesión de acceso especificada no es válida	400	
InvalidAccessGrantLocation	La ubicación de las concesiones de acceso especificada no es válida	400	
InvalidIamRole	El rol de IAM especificado no es válido	400	
InvalidIdentityCenterInstance	La instancia del centro de identidad especificada no es válida	400	

Código de error	Descripción	Código de estado HTTP	
InvalidResourcePolicy	La política de recursos especificada no es válida	400	
InvalidResourcePolicy	La política de recursos especificada no es válida	400	
InvalidTag	Esta solicitud contiene una clave o un valor de etiqueta que no es válido. Los caracteres válidos incluyen lo siguiente: [a-zA-Z+-. _:/]. Las claves de etiqueta pueden contener un máximo de 128 caracteres. Los valores de etiqueta pueden contener un máximo de 256 caracteres.	400	
NoSuchAccessGrantError	La concesión de acceso especificada no existe	404	
NoSuchAccessGrantsLocationError	La ubicación de las concesiones de acceso especificada no existe	404	
AccessDenied	No tiene permisos <i><requested permission></i> para el prefijo de S3 solicitado: <i><requested target></i>	403 Forbidden	
StsNotAuthorizedError	Se ha producido un error (StsNotAuthorizedError) al llamar a la operación GetDataAccess: usuario: access-grants.s3.amazonaws.com no tiene autorización para llevar a cabo: sts:AssumeRole en el recurso: <i><IAM Role ARN></i>	403	

Código de error	Descripción	Código de estado HTTP	
StsPackedPolicyTooLargeError	Se ha producido un error (StsPackedPolicyTooLargeError) al llamar a la operación GetDataAccess: el token serializado es demasiado grande para la sesión	400	
StsValidationError	El mensaje de error varía en función del error de validación.	400	
InvalidTags	Las claves de etiquetas no pueden empezar con el prefijo AWS reservado para las etiquetas del sistema.	400	
TooManyTags	El número de etiquetas supera el límite de 50 etiquetas. Elimine algunas etiquetas e inténtelo de nuevo.	400	

Lista de códigos de error relacionados con Amazon FSx

La tabla siguiente contiene errores especiales que pueden devolver las solicitudes realizadas a Amazon FSx a través de un punto de acceso de Amazon S3. Para obtener información general sobre los errores de Amazon S3 y una lista de códigos de error, consulte [Respuestas de error](#).

Código de error	Descripción	Código de estado HTTP	
AccessDenied	Este error puede producirse por las siguientes razones:	403 Forbidden	

Código de error	Descripción	Código de estado HTTP
	La persona que llama no está autorizada para realizar la operación de S3 especificada. La persona que llama no tiene autorización para acceder al recurso de Amazon FSx para la clave especificada.	
InsufficientCapacity	Se ha alcanzado la capacidad máxima de almacenamiento del sistema de archivos.	507 Insufficient Capacity
InvalidKey	La clave especificada no es válida.	400 Bad Request
SlowDown	Reduzca la tasa de solicitudes.	503 Slow Down

Lista de códigos de error de tablas de Amazon S3

La tabla siguiente contiene errores especiales que las operaciones de tablas de Amazon S3 podrían devolver. Para obtener información general sobre los errores de Amazon S3 y una lista de códigos de error, consulte [Respuestas de error](#).

Código de error	Descripción	Código de estado HTTP
AccessDeniedException	La acción no se puede realizar porque no tiene el permiso necesario.	403 Forbidden

Código de error	Descripción	Código de estado HTTP
<code>BadRequestException</code>	La solicitud no es válida o tiene un formato incorrecto.	400 Bad Request
<code>ConflictException</code>	Se produjo un error en la solicitud porque hay un conflicto con una escritura anterior. Puede reintentar la solicitud.	409 Conflict
<code>ForbiddenException</code>	La persona que llama no está autorizada a realizar la solicitud.	403 Forbidden
<code>InternalServerErrorException</code>	El error de la solicitud se produjo debido a un error de servidor interno.	500 Internal Server Error
<code>NotFoundException</code>	La solicitud se rechazó porque el recurso especificado no se encontró.	404 Not Found
<code>TooManyRequestsException</code>	Se superó el límite en el número de solicitudes por segundo.	429 Too Many Requests

Prácticas recomendadas para los errores de Amazon S3

Muchas respuestas de error contienen datos estructurados adicionales diseñados para ser leídos y comprendidos por un desarrollador que diagnostica errores de programación. Por ejemplo, si envía un encabezado `Content-MD5` con una solicitud `PUT` de REST que no coincide con el resumen calculado en el servidor, recibe un error `BadDigest`. La respuesta de error también incluye como elementos de detalle el resumen que calculamos y el resumen que usted nos anticipó. Durante el desarrollo, puede utilizar esta información para diagnosticar el error. En producción, un programa con buen comportamiento puede incluir esta información en el registro de error.

Cuando diseña una aplicación para utilizar con Amazon S3 es importante controlar correctamente los errores de Amazon S3. En esta sección se describen problemas que debe tener en cuenta al diseñar su aplicación.

Reintente en caso de recibir una respuesta de InternalErrors

Los errores internos son errores que se producen dentro del entorno de Amazon S3.

Es posible que las solicitudes que reciben una respuesta InternalError no se hayan procesado. Por ejemplo, si una solicitud PUT devuelve un error InternalError, una operación GET posterior puede recuperar el valor anterior o el valor actualizado.

Si Amazon S3 devuelve una respuesta de InternalError, repita la solicitud.

Ajustar la aplicación para errores SlowDown repetidos

Como en cualquier sistema distribuido, S3 tiene mecanismos de protección que detectan el consumo excesivo de recursos intencional o no intencional y reaccionan en consecuencia. Los errores de SlowDown se pueden producir cuando una velocidad de solicitud alta activa uno de estos mecanismos. La reducción de la velocidad de su solicitud disminuirá o eliminará errores de este tipo. En términos generales, la mayoría de los usuarios no experimentará estos errores de manera habitual. Sin embargo, si desea obtener más información o experimenta errores SlowDown repetidos o imprevistos, publique en nuestro [foro para desarrolladores de Amazon S3](#) o regístrese en Soporte en <https://aws.amazon.com/premiumsupport/>.

Aislar los errores

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

Amazon S3 proporciona un conjunto de códigos de error que se utilizan en la API de SOAP y en la de REST. La API de SOAP devuelve códigos de error estándares de Amazon S3. La API de REST está diseñada para tener el aspecto de un servidor HTTP estándar e interactuar con clientes HTTP existentes (p. ej., navegadores, bibliotecas de clientes HTTP, servidores proxy, cachés, etc.). Para asegurarnos de que los clientes HTTP controlen los errores correctamente, a cada error de Amazon S3 le asignamos un código de estado HTTP.

Los códigos de estado HTTP son menos costosos que los códigos de error de Amazon S3 e incluyen menos información sobre el error. Por ejemplo, los errores `NoSuchKey` y `NoSuchBucket` de Amazon S3 corresponden al código de estado HTTP `404 Not Found`.

Si bien los códigos de estado HTTP contienen menos información sobre el error, los clientes que comprenden el HTTP pero no la API de Amazon S3, por lo general, pueden controlar los errores correctamente.

Por lo tanto, al momento de controlar errores o informar errores de Amazon S3 a los usuarios finales, utilice el código de error de Amazon S3 en lugar del código de estado HTTP, ya que contiene más información sobre el error. Además, al depurar su aplicación, también debe consultar el elemento legible `<Details>` de la respuesta de error de XML.

AWSGlosario de

Para ver la terminología más reciente de AWS, consulte el [Glosario de AWS](#) en la Referencia de Glosario de AWS.

Recursos de Amazon S3

En la tabla siguiente, se enumeran los recursos relacionados que le resultarán útiles cuando trabaje con este servicio.

Recurso	Descripción
Guía del usuario de Amazon Simple Storage Service	La guía de introducción proporciona un tutorial rápido del servicio en función de un caso de uso sencillo.
Guía del usuario de Amazon Simple Storage Service	La guía para desarrolladores describe cómo realizar tareas mediante las operaciones de Amazon S3.
Preguntas frecuentes técnicas sobre Amazon S3	Las preguntas frecuentes tratan las 20 principales preguntas planteadas por los desarrolladores sobre este producto.
Notas de la versión de Amazon S3	Las Notas de la versión ofrecen información general de alto nivel de la versión actual. Destacan de forma específica las características nuevas, las correcciones y los problemas que se conocen.
Herramientas para Amazon Web Services	Punto de comienzo central para encontrar documentación, ejemplos de código, notas de la versión y otra información que ayudará a crear aplicaciones innovadoras con AWS SDK y herramientas.
Consola de administración de AWS	La consola permite llevar a cabo la mayoría de las funciones de Amazon S3 sin necesidad de programación.
Foros de debate de	Foro de la comunidad donde los desarrolladores analizan aspectos técnicos relacionados con Amazon Web Services.
Soporte Centro de	La página de inicio para la Asistencia técnica de AWS, que incluye el acceso a nuestros foros para desarrolladores, preguntas técnicas frecuentes, la página de estado del servicio y Premium Support.

Recurso	Descripción
Soporte	Página web principal para obtener información acerca de Soporte, un canal de soporte individualizado y de respuesta rápida que ayudará a crear y ejecutar aplicaciones en servicios de infraestructura de AWS.
Información de producto de Amazon S	Página web principal con información acerca de Amazon S3.
Contacto	Un punto de contacto centralizado para las consultas relacionadas con la facturación, cuentas, eventos, abuso, etc. de AWS.
Condiciones de uso	Información detallada acerca del uso de los derechos de autor y las marcas comerciales en Amazon.com y otros temas.

Apéndice

Temas

- [Apéndice: respuesta de SelectObjectContent](#)
- [Apéndice: objeto OPTIONS](#)
- [Apéndice: API de SOAP](#)
- [Apéndice: autenticación de solicitudes \(AWS Signature Version 2\)](#)
- [Apéndice: API de configuración del ciclo de vida \(obsoleto\)](#)

Apéndice: respuesta de SelectObjectContent

Descripción

La operación de Amazon S3 Select filtra el contenido de un objeto de Amazon S3 en función de una instrucción de lenguaje de consulta estructurado (SQL) simple. Dado que se desconoce el tamaño de la respuesta de esta operación, Amazon S3 Select transmite la respuesta como una serie de mensajes e incluye un encabezado `Transfer-Encoding` con **chunked** como su valor en la respuesta.

Para obtener más información sobre Amazon S3 Select, consulte [Selección de contenido de objetos](#) en la Guía del usuario de Amazon Simple Storage Service.

Para obtener más información acerca del uso de SQL con Amazon S3 Select, consulte [Referencia de SQL para Amazon S3 Select y Amazon Glacier Select](#) en la Guía del usuario de Amazon Simple Storage Service.

Respuestas

Si la operación de Amazon S3 Select se realiza correctamente, se devuelve el código de estado 200 OK.

Encabezados de respuesta

Esta implementación de la operación solo usa encabezados de respuesta que sean comunes a la mayoría de las respuestas. Para obtener más información, consulte [Encabezados de respuesta comunes](#).

Cuerpo de respuesta

Dado que se desconoce el tamaño de la respuesta de Amazon S3 Select, Amazon S3 transmite la respuesta como una serie de mensajes e incluye un encabezado `Transfer-Encoding` con **chunked** como su valor en la respuesta. En el ejemplo siguiente, se muestra el formato de respuesta en el nivel superior:

```
<Message 1>
<Message 2>
<Message 3>
.....
<Message n>
```

Cada mensaje se compone de dos secciones: el preludio y los datos. La sección de preludio contiene 1) la longitud total en bytes del mensaje y 2) la longitud en bytes combinada de todos los encabezados. La sección de datos contiene 1) los encabezados y 2) una carga útil.

Cada sección termina con una suma de comprobación de enteros big-endian de 4 bytes (CRC). Amazon S3 Select utiliza CRC32 (a menudo denominado CRC32 de GZIP) para calcular ambos CRC. Para obtener más información sobre CRC32, consulte [GZIP file format specification version 4.3](#).

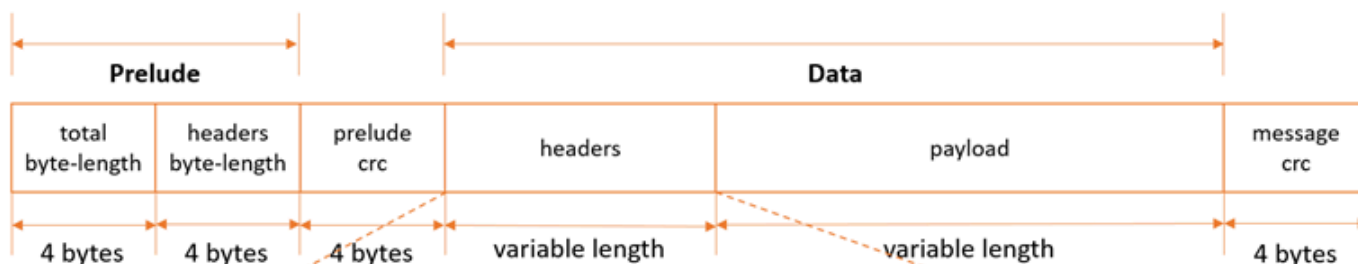
La carga total del mensaje incluido el preludio y las dos sumas de comprobación es de 16 bytes.

Note

Todos los valores de `integer` de los mensajes están en orden de bytes de red o en orden big-endian.

En el siguiente diagrama, se muestran los componentes que conforman un mensaje y un encabezado. Tenga en cuenta que hay varios encabezados por mensaje.

Message:



Headers (multiple headers per message):

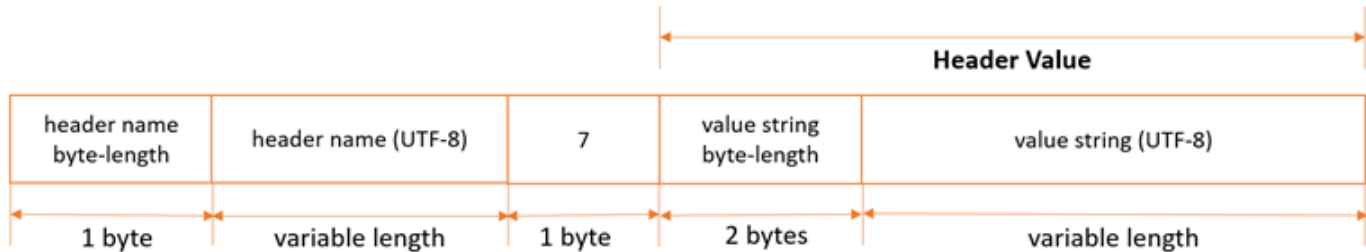


Note

En Amazon S3 Select, el tipo de valor del encabezado es siempre 7 (`type=String`). Para este tipo, el valor del encabezado consta de dos componentes: una longitud de enteros big-endian

de 2 bytes y una cadena UTF-8 que tiene esa longitud de bytes. En el siguiente diagrama, se muestran los componentes que conforman encabezados de Amazon S3 Select.

Amazon S3 Select Headers (type=String):



Cálculos de longitud de bytes de carga útil (estos dos cálculos son equivalentes):

- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - \text{sizeof}(\text{total\_length}) - \text{sizeof}(\text{header\_length}) - \text{sizeof}(\text{prelude\_crc}) - \text{sizeof}(\text{message\_crc})$
- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - 16$

Cada mensaje contiene los siguientes componentes:

- Preludio: siempre tiene un tamaño fijo de 8 bytes (dos campos de 4 bytes cada uno):
 - Primeros cuatro bytes: longitud total de bytes: se trata de la longitud en bytes indicada en números enteros big-endian de todo el mensaje (incluido el propio campo de longitud total de 4 bytes).
 - Segundos cuatro bytes: longitud en bytes de los encabezados: longitud en bytes de enteros big-endian de la parte de los encabezados del mensaje (sin incluir el propio campo de longitud de los encabezados).
- CRC del preludio: suma de comprobación de enteros big-endian de 4 bytes (CRC) de la parte del preludio del mensaje (sin incluir la propia CRC). El preludio tiene una CRC independiente de la CRC del mensaje (ver abajo) y que garantiza que la información dañada sobre la longitud de bytes se puede detectar de forma inmediata, sin generar comportamiento del almacenamiento en búfer compulsivo.
- Encabezados: un conjunto de metadatos que actúan como comentarios del mensaje; por ejemplo, el tipo de mensaje, el formato de carga útil, etc. Los mensajes pueden tener varios encabezados,

por lo que esta parte del mensaje puede tener longitudes de bytes diferentes según el tipo de mensaje. Los encabezados son pares clave-valor en los que la clave y el valor son cadenas UTF-8. Los encabezados pueden aparecer en cualquier orden en la parte del mensaje de los encabezados y cada tipo de encabezado solamente puede aparecer una vez.

Para Amazon S3 Select, a continuación, se muestra una lista de nombres de encabezados y el conjunto de valores válidos en función del tipo de mensaje.

- Encabezado `MessageType`:
 - `HeaderName` => `":message-type"`
 - `HeaderValues` válidos => `"error"`, `"event"`
- Encabezado `EventType`:
 - `HeaderName` => `":event-type"`
 - `HeaderValues` válidos => `"Records"`, `"Cont"`, `"Progress"`, `"Stats"`, `"End"`
- Encabezado `ErrorCode`:
 - `HeaderName` => `":error-code"`
 - `HeaderValues` válidos => Código de error de la tabla de la sección [Lista de códigos de error de contenido de objeto SELECT](#).
- Encabezado `ErrorMessage`:
 - `HeaderName` => `":error-message"`
 - `HeaderValues` válidos => Mensaje de error devuelto por el servicio para ayudar a diagnosticar los errores por solicitud.
- Carga útil: puede ser cualquier cosa.
- CRC del mensaje: suma de comprobación de enteros big-endian de 4 bytes (CRC) desde el principio del mensaje hasta el inicio de la suma de comprobación (es decir, todo lo que contiene el mensaje, excepto la propia CRC del mensaje).

Cada encabezado contiene los siguientes componentes. Puede haber varios encabezados por mensaje.

- Longitud en bytes del nombre de encabezado: longitud de bytes del nombre del encabezado.
- Nombre de encabezado: nombre del encabezado que indica el tipo de encabezado. Valores válidos: `":message-type"` `":event-type"` `":error-code"` `":error-message"`
- Tipo de valor del encabezado: enumeración que indica el tipo de valor del encabezado. Para Amazon S3 Select, siempre es 7.

- Longitud en bytes de cadena de valores: (para Amazon S3 Select) longitud en bytes de la cadena de valores del encabezado.
- Cadena de valores del encabezado: (para Amazon S3 Select) valor de la cadena del encabezado. Los valores válidos de este campo varían en función del tipo de encabezado. Consulte las secciones siguientes para ver los valores válidos para cada tipo de encabezado y tipo de mensaje.

Para Amazon S3 Select, las respuestas pueden ser mensajes de los siguientes tipos:

- Mensaje de registros: puede contener un registro único, registros parciales o varios registros. En función del tamaño del resultado, una respuesta puede contener uno o más de estos mensajes.
- Mensaje de continuación: Amazon S3 envía este mensaje periódicamente para mantener abierta la conexión TCP. Estos mensajes aparecen en las respuestas de forma aleatoria. El cliente debe detectar el tipo de mensaje y procesarlo en consecuencia.
- Mensaje de progreso: Amazon S3 envía este mensaje periódicamente, si se solicita. Contiene información acerca del progreso de una consulta que se ha iniciado, pero aún no se ha completado.
- Mensaje de estadísticas: Amazon S3 envía este mensaje al final de la solicitud. Contiene estadísticas sobre la consulta.
- Mensaje final: indica que la solicitud está completa y que no se enviarán más mensajes. No debe suponer que la solicitud está completa hasta que el cliente reciba un mensaje End.
- Mensaje RequestLevelError: Amazon S3 envía este mensaje si la solicitud ha producido un error por alguna razón. Contiene el código de error y el mensaje de error. Si Amazon S3 envía un mensaje RequestLevelError, no envía ningún mensaje End.

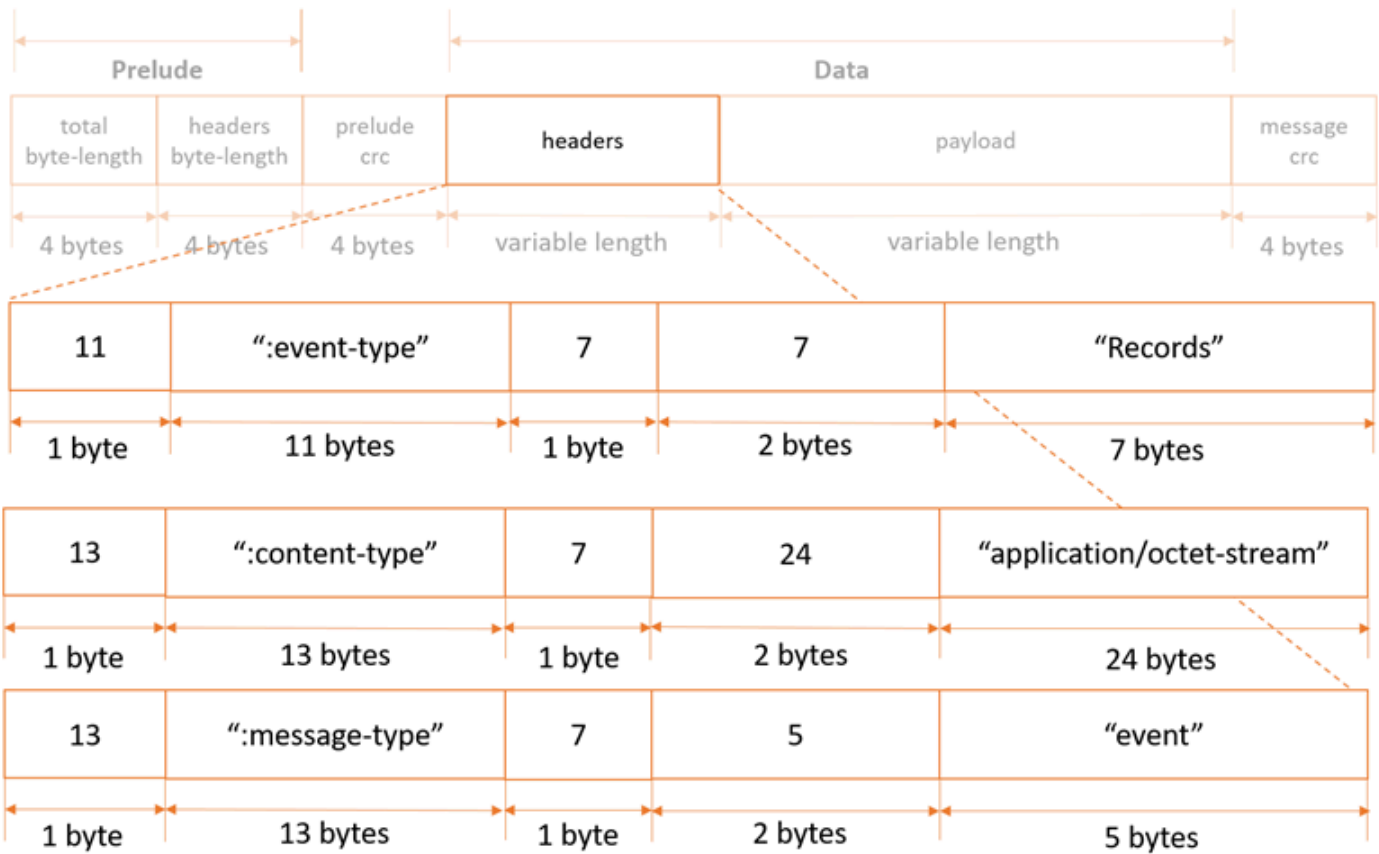
En las siguientes secciones, se explica con más detalle la estructura de cada tipo de mensaje.

Para ver ejemplos de código y pruebas unitarias que utilizan este protocolo, consulte [AWS C Event Stream](#) en el sitio web de GitHub.

Mensaje de registros

Especificación del encabezado

Los mensajes de registros contienen tres encabezados, de la siguiente manera:



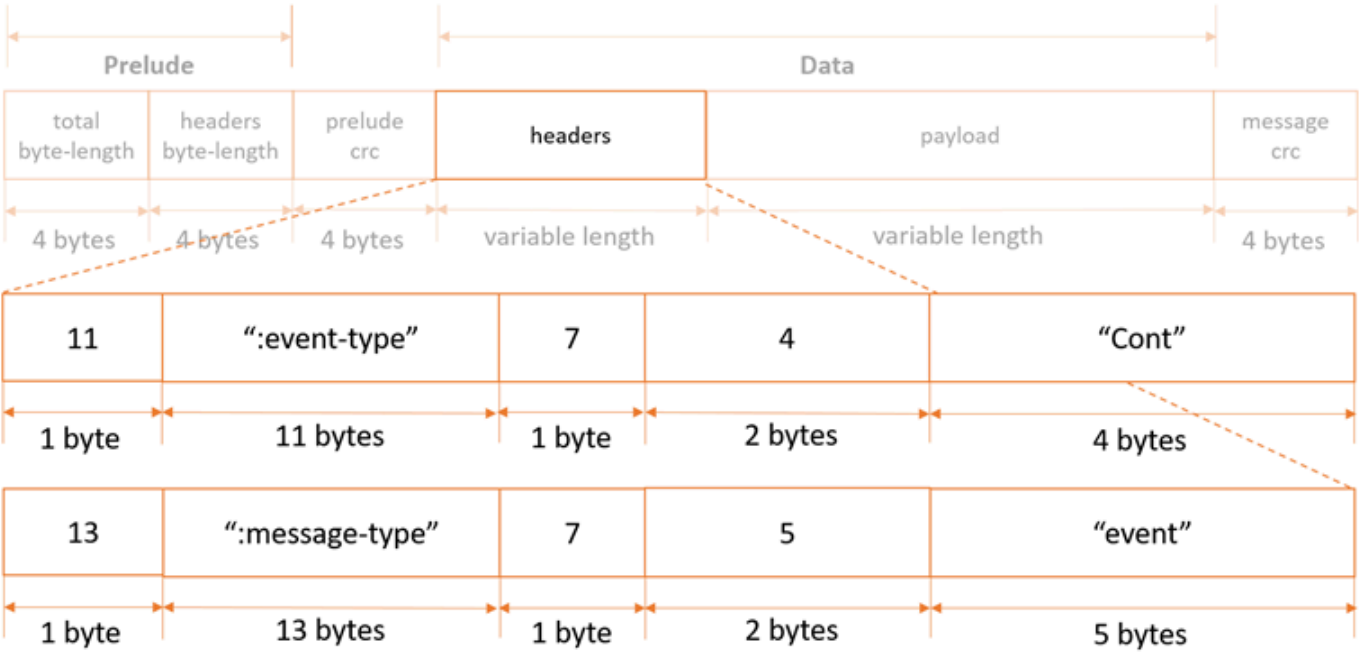
Especificación de carga útil

Las cargas útiles del mensaje de registros puede contener un registro único, registros parciales o varios registros.

Mensaje de continuación

Especificación del encabezado

Los mensajes de continuación contienen dos encabezados, de la siguiente manera:



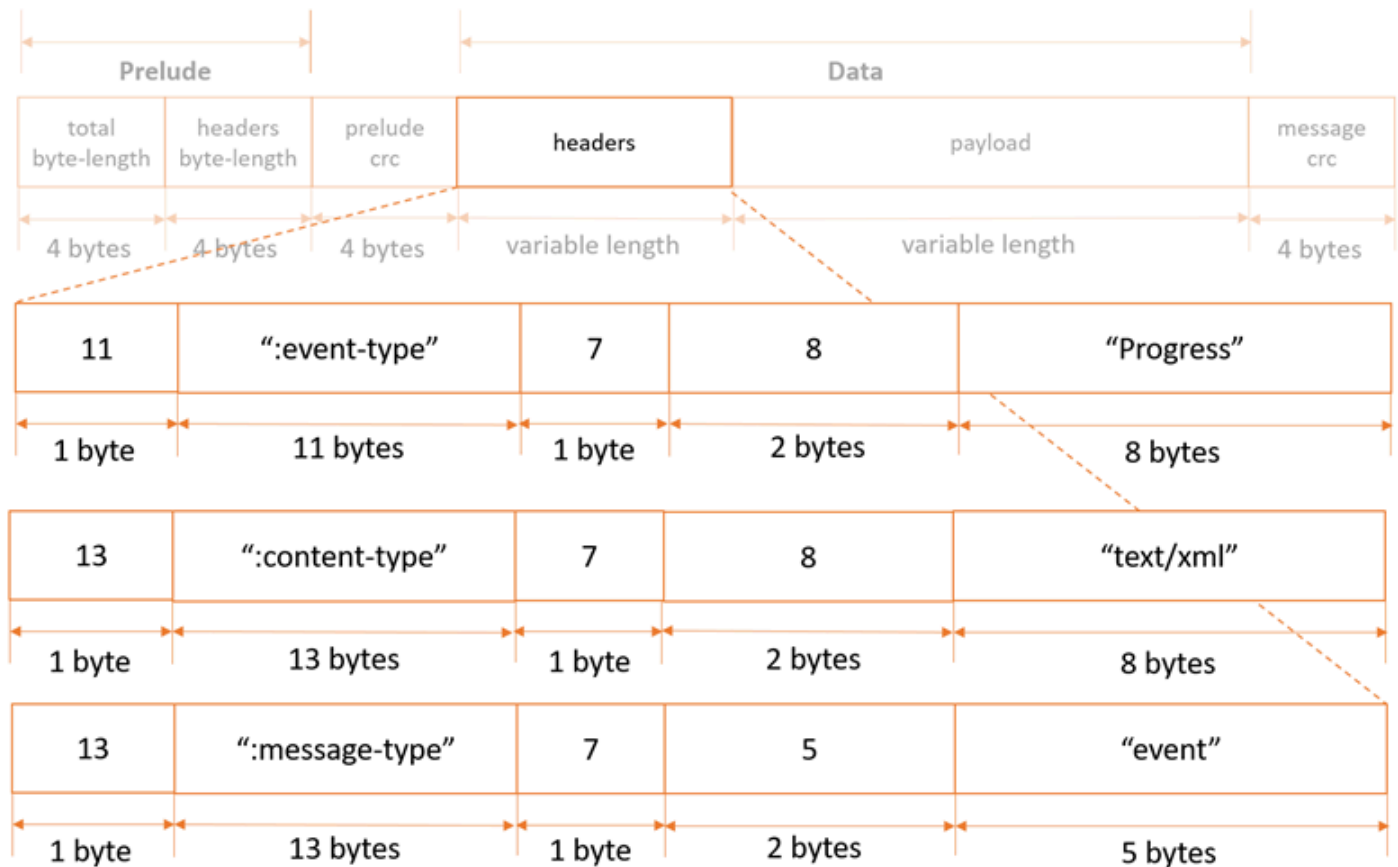
Especificación de carga útil

Los mensajes de continuación no tienen carga útil.

Mensaje de progreso

Especificación del encabezado

Los mensajes de progreso contienen tres encabezados, de la siguiente manera:



Especificación de carga útil

La carga útil de los mensajes de progreso es un documento XML que contiene información sobre el progreso de una solicitud.

- **BytesScanned** => Número de bytes que se procesaron antes de descomprimirse (si el archivo está comprimido).
- **BytesProcessed** => Número de bytes que se procesaron después de descomprimirse (si el archivo está comprimido).
- **BytesReturned** => Número actual de bytes de datos de carga útil de registros devueltos por Amazon S3.

Para archivos sin comprimir, **BytesScanned** y **BytesProcessed** son iguales.

Ejemplo:

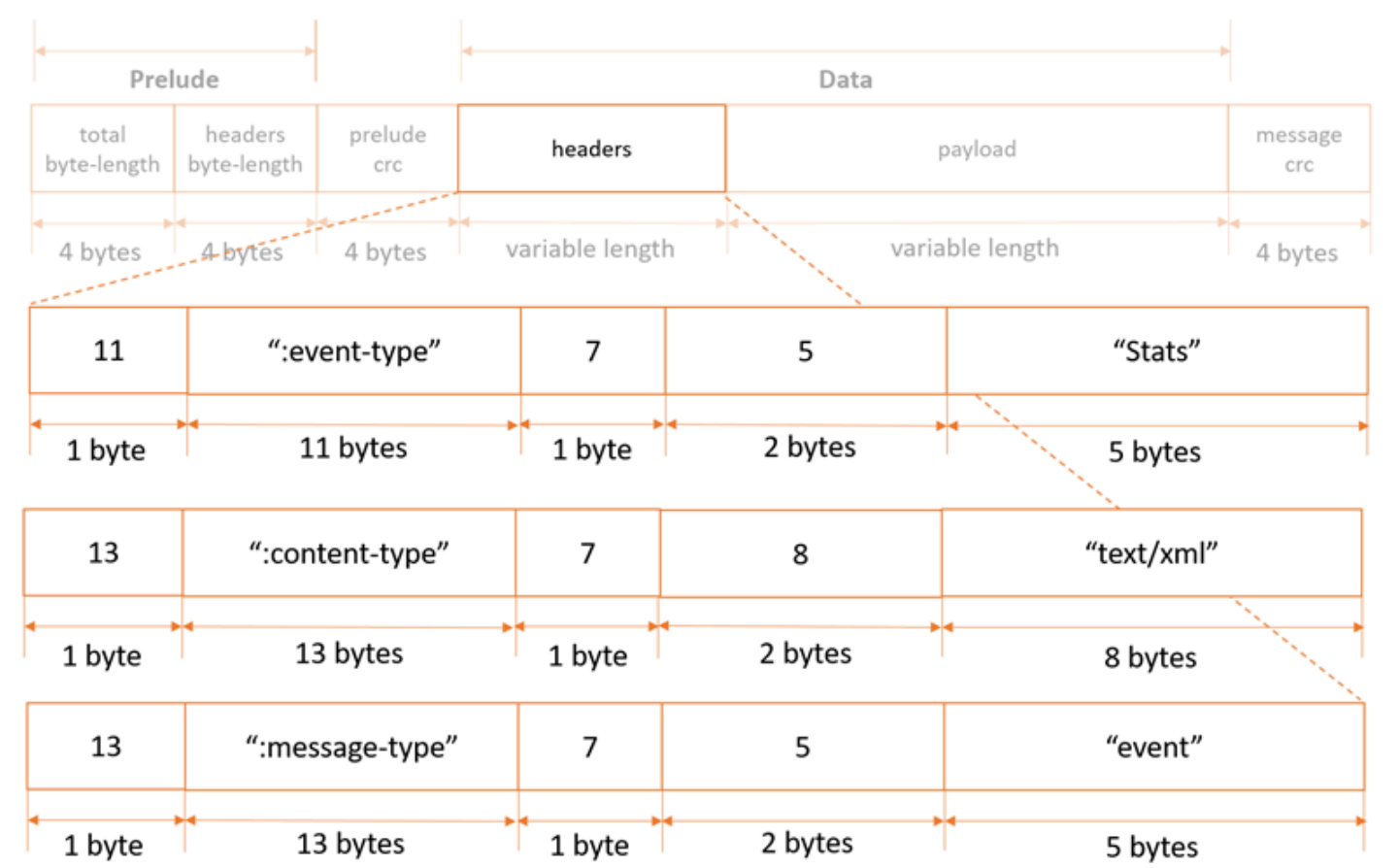
```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Progress>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Progress>
```

Mensaje de estadísticas

Especificación del encabezado

Los mensajes de estadísticas contienen tres encabezados, de la siguiente manera:



Especificación de carga útil

La carga útil del mensaje de estadísticas es un documento XML que contiene información sobre las estadísticas de una solicitud cuando se completa el procesamiento.

- BytesScanned => Número de bytes que se procesaron antes de descomprimirse (si el archivo está comprimido).

- BytesProcessed => Número de bytes que se procesaron después de descomprimirse (si el archivo está comprimido).
- BytesReturned => Número total de bytes de datos de carga útil de registros devueltos por Amazon S3.

Para archivos sin comprimir, BytesScanned y BytesProcessed son iguales.

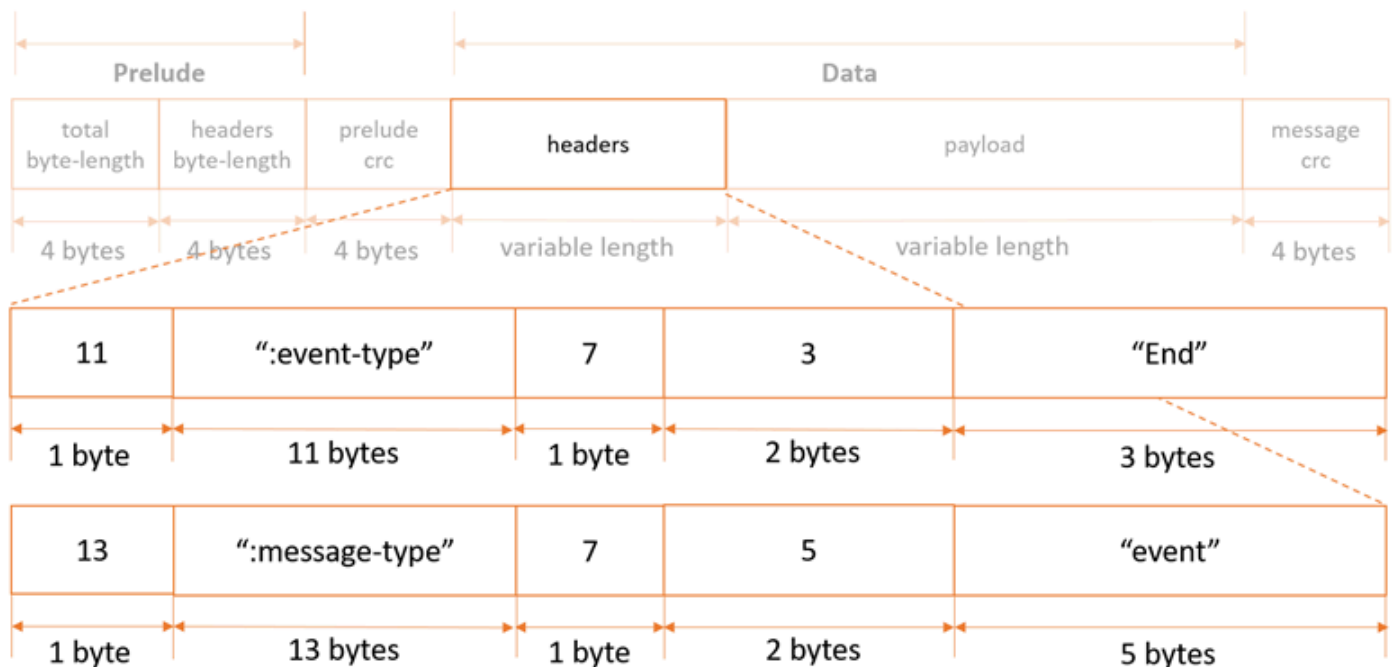
Ejemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<Stats>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Stats>
```

Mensaje final

Especificación del encabezado

Los mensajes finales contienen dos encabezados, de la siguiente manera:



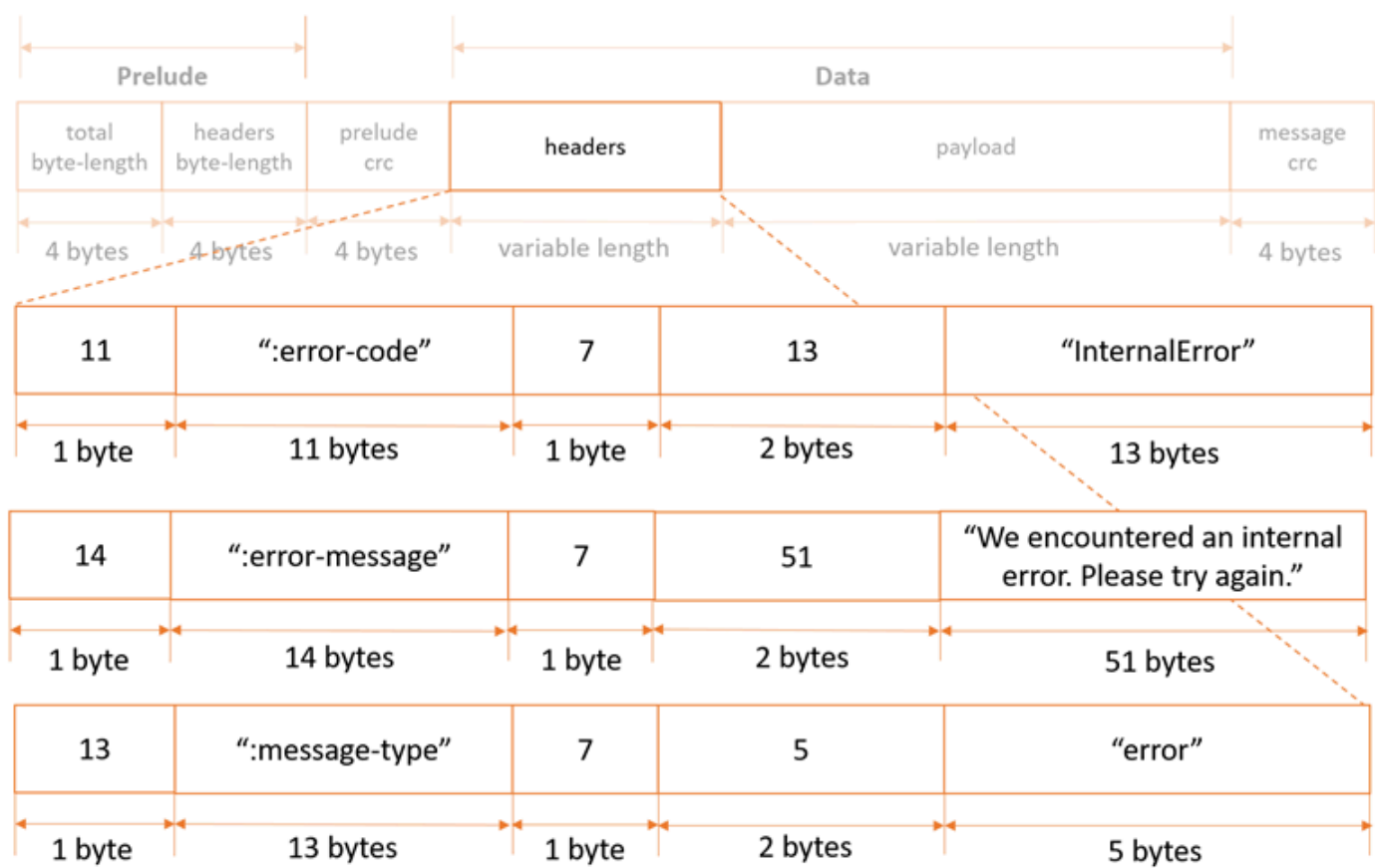
Especificación de carga útil

Los mensajes finales no tienen carga útil.

Mensaje de error por solicitud

Especificación del encabezado

Los mensajes de error por solicitud contienen tres encabezados, de la siguiente manera:



Para obtener una lista de posibles códigos de error y mensajes de error, consulte [Lista de códigos de error de contenido de objeto SELECT](#).

Especificación de carga útil

Los mensajes de error por solicitud no tienen carga útil.

Activos relacionados

- [SelectObjectContent](#)

- [GetObject](#)
- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)

Apéndice: objeto OPTIONS

Descripción

Un navegador puede enviar esta solicitud de verificación previa a Amazon S3 para determinar si puede enviar una solicitud real con el origen, el método HTTP y los encabezados específicos.

Amazon S3 admite el uso compartido de recursos entre orígenes (CORS), ya que permite agregar un subrecurso `cors` en un bucket. Cuando un navegador envía esta solicitud de verificación previa, Amazon S3 responde evaluando las reglas que se definen en la configuración de `cors`.

Si `cors` no está habilitado en el bucket, Amazon S3 devuelve una respuesta 403 Forbidden.

Para obtener más información acerca de CORS, vaya a [Habilitación del uso compartido de recursos entre orígenes](#) en la Guía del usuario de Amazon Simple Storage Service.

Solicitudes

Sintaxis

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

Parámetros de la solicitud

Esta operación no presenta ningún parámetro de solicitud específico, pero puede contener los parámetros de solicitud que requiera la solicitud real.

Encabezados de solicitud

Nombre	Descripción	Obligatorio
Origin	Identifica el origen de la solicitud entre orígenes a Amazon S3. Por ejemplo, <code>http://www.example.com</code> . Tipo: cadena	Sí

Nombre	Descripción	Obligatorio
	Valor predeterminado: None	
Access-Control-Request-Method	Identifica qué método HTTP se utilizará en la solicitud real. Tipo: cadena Valor predeterminado: None	Sí
Access-Control-Request-Headers	Una lista delimitada por comas de encabezados HTTP que se enviará en la solicitud real. Por ejemplo, para colocar un objeto con cifrado del servidor, esta solicitud de verificación previa determinará si puede incluir el encabezado <code>x-amz-server-side-encryption</code> en la solicitud. Tipo: cadena Valor predeterminado: None	No

Elementos de la solicitud

Esta implementación de la operación no utiliza elementos de solicitud.

Respuestas

Encabezados de respuesta

Encabezado	Descripción
Access-Control-Allow-Origin	El origen que envió en la solicitud. Si no se permite el origen de la solicitud, Amazon S3 no incluirá este encabezado en la respuesta.

Encabezado	Descripción
	Tipo: cadena
Access-Control-Max-Age	Cuánto tiempo, en segundos, se pueden almacenar en caché los resultados de una solicitud de verificación previa. Tipo: cadena
Access-Control-Allow-Methods	El método HTTP que se envió en la solicitud original. Si no se permite el método de la solicitud, Amazon S3 no incluirá este encabezado en la respuesta. Tipo: cadena
Access-Control-Allow-Headers	Una lista delimitada por comas de encabezados HTTP que el navegador puede enviar en la solicitud real. Si no se permite alguno de los encabezados solicitados, Amazon S3 no incluirá ese encabezado en la respuesta ni la respuesta contendrá ninguno de los encabezados con el prefijo Access-Control . Tipo: cadena
Access-Control-Expose-Headers	Una lista delimitada por comas de encabezados HTTP. Este encabezado proporciona al cliente de JavaScript acceso a estos encabezados en respuesta a la solicitud real. Tipo: cadena

Elementos de respuesta

Esta implementación de la operación no devuelve elementos de respuesta.

Ejemplos

Ejemplo: enviar una solicitud OPTIONS de verificación previa a un bucket habilitado para **cors**

Un navegador puede enviar esta solicitud de verificación previa a Amazon S3 para determinar si puede enviar la solicitud PUT real desde el origen de `http://www.example.com` al bucket de Amazon S3 denominado `examplebucket`.

Solicitud de muestra

```
OPTIONS /exampleobject HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

Respuesta de ejemplo

```
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
```

Activos relacionados

- [GetBucketCors](#)
- [DeleteBucketCors](#)
- [PutBucketCors](#)

Apéndice: API de SOAP

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

En esta sección, se describe la API de SOAP con respecto a las operaciones de servicio, bucket y objeto. Tenga en cuenta que las solicitudes SOAP, autenticadas y anónimas, se deben enviar a Amazon S3 con SSL. Amazon S3 devuelve un error si envía una solicitud SOAP por HTTP.

La versión más reciente del WSDL de Amazon S3 está disponible en docs.aws.amazon.com/2006-03-01/AmazonS3.wSDL.

Temas

- [Operaciones en el servicio \(API de SOAP\)](#)
- [Operaciones en buckets \(API de SOAP\)](#)
- [Operaciones en objetos \(API de SOAP\)](#)
- [Autenticar solicitudes SOAP](#)
- [Configurar políticas de acceso con SOAP](#)
- [Elementos comunes](#)
- [Respuestas de error de SOAP](#)

Operaciones en el servicio (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

En esta sección, se describen las operaciones de SOAP que puede realizar en el servicio de Amazon S3.

Temas

- [ListAllMyBuckets \(API de SOAP\)](#)

ListAllMyBuckets (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación `ListAllMyBuckets` devuelve una lista de todos los buckets pertenecientes al remitente de la solicitud.

Example

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>
```

```
</Bucket>
</Buckets>
</ListAllMyBucketsResult>
```

Cuerpo de respuesta

- **Owner :**

Esto proporciona información que Amazon S3 utiliza para representar su identidad con fines de autenticación y control de acceso. El ID es un identificador único y permanente del desarrollador que realizó la solicitud. DisplayName es un nombre legible para las personas que representa al desarrollador que realizó la solicitud. No es único y puede cambiar con el tiempo. Recomendamos hacer coincidir el DisplayName con el nombre del foro.

- **Name :**

Nombre del bucket de. Tenga en cuenta que si uno de los buckets se ha eliminado recientemente, es posible que el nombre del bucket eliminado siga presente en esta lista durante un tiempo.

- **CreationDate :**

La hora en que se creó el bucket.

Control de acceso

Debe autenticarse con un ID de clave de acceso de AWS válido. Las solicitudes anónimas nunca pueden mostrar buckets y solo se pueden mostrar buckets que se posean.

Operaciones en buckets (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

En esta sección, se describen las operaciones que puede realizar en buckets de Amazon S3.

Temas

- [CreateBucket \(API de SOAP\)](#)

- [DeleteBucket \(API de SOAP\)](#)
- [ListBucket \(API de SOAP\)](#)
- [GetBucketAccessControlPolicy \(API de SOAP\)](#)
- [SetBucketAccessControlPolicy \(API de SOAP\)](#)
- [GetBucketLoggingStatus \(API de SOAP\)](#)
- [SetBucketLoggingStatus \(API de SOAP\)](#)

CreateBucket (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación CreateBucket crea un bucket. No todas las cadenas tienen nombres de bucket aceptables. Para obtener información sobre las restricciones de nombres de los buckets, consulte [Cómo trabajar con buckets de Amazon S3](#).

Note

Para determinar si existe un nombre de bucket, utilice ListBucket y establezca MaxKeys en 0. Una respuesta de NoSuchBucket indica que el bucket está disponible, una respuesta de AccessDenied indica que otra persona es propietaria del bucket y una respuesta de éxito indica que usted posee el bucket o que tiene permiso para acceder a él.

Example Creación de un bucket llamado “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>
```

Elementos

- **Bucket**: el nombre del bucket que está intentando crear.
- **AccessControlList**: la lista de control de acceso para el bucket nuevo. Este elemento es opcional. Si no se proporciona, el bucket se crea con una política de acceso que otorga acceso de **FULL_CONTROL** al solicitante.

Control de acceso

Debe autenticarse con un ID de clave de acceso de AWS válido. Nunca se permite que las solicitudes anónimas creen buckets.

Activos relacionados

- [ListBucket \(API de SOAP\)](#)

DeleteBucket (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación `DeleteBucket` elimina un bucket. Todos los objetos del bucket se deben eliminar antes de eliminar el propio bucket.

Example

En este ejemplo, se elimina el bucket “quotes”.

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>
```

Elementos

- **Bucket** : el nombre del bucket que desea eliminar.

Control de acceso

Solo el propietario de un bucket puede eliminarlo, independientemente de la política de control de acceso del bucket.

ListBucket (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación **ListBucket** devuelve información sobre alguno de los elementos en el bucket.

Para obtener una introducción general a la operación de muestra, consulte [Listado de claves de objetos](#).

Solicitudes

En este ejemplo, se muestran hasta 1000 claves del bucket “quotes” que tienen el prefijo “notas”.

Sintaxis

```
<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>
```

Parámetros

Nombre	Descripción	Obligatorio
prefix	Limita la respuesta a las claves que comienzan con el prefijo indicado. Puede usar prefijos para separar un bucket en diferentes conjuntos de claves de forma similar a como un sistema de archivos usa las carpetas. Tipo: cadena Valor predeterminado: None	No
marker	Indica en qué parte del bucket se debe empezar a mostrar. La lista solo incluirá las claves que aparecen lexicográficamente después del marcador. Esto resulta práctico para la paginación: para pasar a la siguiente página de resultados, utilice la última clave de la página actual como marcador. Tipo: cadena	No

Nombre	Descripción	Obligatorio
	Valor predeterminado: None	
max-keys	El número máximo de claves que le gustaría ver en el cuerpo de la respuesta. Es posible que el servidor devuelva menos claves que esta cantidad, pero no devolverá más. Tipo: cadena Valor predeterminado: None	No
delimiter	Hace que las claves que contienen la misma cadena entre el prefijo y la primera aparición del delimitador se agrupen en un único elemento de resultado de la colección CommonPrefixes. Estas claves acumuladas no se devuelven en ninguna otra parte de la respuesta. Tipo: cadena Valor predeterminado: None	No

Respuesta de éxito

Esta respuesta asume que el bucket contiene las siguientes claves:

```
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
```

Sintaxis

```
<?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>"828ef3fdfa96f00ad9f27c383fc9ac7f"</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>
```

Como puede ver, muchos de los campos de la respuesta reflejan los parámetros de solicitud. IsTruncated, Contents y CommonPrefixes son los únicos elementos de respuesta que pueden contener información nueva.

Elementos de respuesta

Nombre	Descripción
Contents	Metadatos sobre cada objeto devuelto. Tipo: Metadatos XML Antecesor: ListBucketResult
CommonPrefixes	Una respuesta puede contener CommonPrefixes solo si se especifica un delimiter . Si lo hace, CommonPrefixes contiene todas las claves (si las hay) situadas entre Prefix y la siguiente aparición de la cadena especificada por delimiter . En efecto, CommonPrefixes muestra las claves que actúan como subdirectorios en el directorio especificado por Prefix. Por ejemplo, si prefix es notes/ y delimiter es una barra diagonal (/), en notes/summer/july , el prefijo común es notes/summer/ .

Nombre	Descripción
	Tipo: cadena Antecesor: ListBucketResult
Delimiter	Hace que las claves que contienen la misma cadena entre el prefijo y la primera aparición del delimitador se agrupen en un único elemento de resultado de la colección CommonPrefixes. Estas claves acumuladas no se devuelven en ninguna otra parte de la respuesta. Tipo: cadena Antecesor: ListBucketResult
IsTruncated	Especifica si se devolvieron todos los resultados (verdadero) o no (falso). Es posible que no se devuelvan todos los resultados si el número de resultados supera el especificado por MaxKeys. Tipo: cadena Antecesor: boolean
Marker	Indica en qué parte del bucket se debe empezar a mostrar. Tipo: cadena Antecesor: ListBucketResult
MaxKeys	El número máximo de claves devueltas en el cuerpo de respuesta. Tipo: cadena Antecesor: ListBucketResult
Name	Nombre del bucket. Tipo: cadena Antecesor: ListBucketResult

Nombre	Descripción
Prefix	Claves que comienzan con el prefijo indicado. Tipo: cadena Antecesor: ListBucketResult

Cuerpo de respuesta

Para obtener información sobre la respuesta de la lista, consulte [Lista de respuesta de claves](#).

Control de acceso

Para mostrar las claves de un bucket, es necesario que se haya concedido acceso de READ al bucket.

GetBucketAccessControlPolicy (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación GetBucketAccessControlPolicy obtiene la política de control de acceso de un bucket.

Example

En este ejemplo, se recupera la política de control de acceso para el bucket “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>
```

Cuerpo de respuesta

La respuesta contiene la política de control de acceso del bucket. Para obtener una explicación de esta respuesta, consulte [Política de acceso de SOAP](#).

Control de acceso

Debe tener derechos READ_ACP sobre el bucket para poder recuperar la política de control de acceso de un bucket.

SetBucketAccessControlPolicy (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación `SetBucketAccessControlPolicy` establece la política de control de acceso de un bucket existente. Si se ejecuta correctamente, la política de control de acceso anterior del bucket se sustituye por completo por la política de control de acceso especificada.

Example

Conceda al usuario especificado (normalmente el propietario) acceso de `FULL_CONTROL` al bucket "quotes".

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>
```

Control de acceso

Debe tener derechos `WRITE_ACP` sobre el bucket para poder establecer la política de control de acceso de un bucket.

GetBucketLoggingStatus (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

GetBucketLoggingStatus recupera el estado de registro de un bucket existente.

Para obtener una introducción general a esta característica, consulte [Registros del servidor](#).

Example

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>
```

Control de acceso

Solo el propietario de un bucket puede invocar esta operación.

SetBucketLoggingStatus (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación SetBucketLoggingStatus actualiza el estado de registro de un bucket existente.

Para obtener una introducción general a esta característica, consulte [Registros del servidor](#).

Example

Esta solicitud de ejemplo permite registrar el acceso al servidor para el bucket “mybucket” y configura los registros para que se envíen a “mylogs” con el prefijo “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>
```

Control de acceso

Solo el propietario de un bucket puede invocar esta operación.

Operaciones en objetos (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

En esta sección, se describen las operaciones que puede realizar en objetos de Amazon S3.

Temas

- [PutObjectInline \(API de SOAP\)](#)
- [PutObject \(API de SOAP\)](#)
- [CopyObject \(API de SOAP\)](#)
- [GetObject \(API de SOAP\)](#)
- [GetObjectExtended \(API de SOAP\)](#)
- [DeleteObject \(API de SOAP\)](#)
- [GetObjectAccessControlPolicy \(API de SOAP\)](#)
- [SetObjectAccessControlPolicy \(API de SOAP\)](#)

PutObjectInline (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación PutObjectInline agrega un objeto a un bucket. Los datos del objeto se proporcionan en el cuerpo del mensaje de SOAP.

Si un objeto ya existe en un bucket, el nuevo objeto lo sobrescribirá porque Amazon S3 almacena la última solicitud de escritura. Sin embargo, Amazon S3 es un sistema distribuido. Si Amazon S3

recibe varias solicitudes de escritura para el mismo objeto casi simultáneamente, es posible que todos los objetos se almacenen, aunque al final solo uno de ellos gane. Amazon S3 no proporciona bloqueo de objetos; si lo necesita, asegúrese de integrarlo en la capa de aplicación.

Para garantizar que un objeto no esté dañado en la red, puede calcular el MD5 de un objeto, colocarlo en Amazon S3 y comparar la Etag devuelta con el valor de MD5 calculado.

PutObjectInline no es adecuado para su uso con objetos grandes. El sistema limita esta operación al trabajo con objetos de 1 MB o menos. PutObjectInline producirá un error con el código de estado `InlineDataTooLargeError` si el parámetro Datos codifica un objeto de más de 1 MB. Para cargar objetos grandes, considere la posibilidad de utilizar la API PutObject no integrada o, en su lugar, la API de REST.

Example

En este ejemplo, se escriben algunos textos y metadatos en el objeto “Nelson” del bucket “quotes”, se proporciona a un usuario (normalmente el propietario) acceso de FULL_CONTROL al objeto y se permite que partes anónimas puedan leer el objeto.

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>

```

Elementos

- **Bucket**: el bucket en el que se va a agregar el objeto.
- **Key**: la clave que se va a asignar al objeto.
- **Metadata**: puede proporcionar pares de metadatos nombre-valor en el elemento de metadatos. Estos se almacenarán con el objeto.
- **Data**: el formulario codificado en base 64 de los datos.
- **ContentLength**: la longitud de los datos en bytes.
- **AccessControlList**: una lista de control de acceso para el recurso. Este elemento es opcional. Si se omite, el solicitante tiene acceso de FULL_CONTROL al objeto. Si el objeto ya existe, se reemplaza la política de control de acceso preexistente.

Respuestas

- **ETag**: la etiqueta de entidad es un hash MD5 del objeto que se puede utilizar para realizar recuperaciones condicionales del objeto mediante `GetObjectExtended`. La ETag solo refleja los cambios en el contenido de un objeto, no en los metadatos.
- **LastModified**: la marca temporal de Amazon S3 del objeto guardado.

Control de acceso

Debe tener acceso de `WRITE` al bucket para poder colocar objetos en él.

Activos relacionados

- [PutObject \(API de SOAP\)](#)
- [CopyObject \(API de SOAP\)](#)

PutObject (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación `PutObject` agrega un objeto a un bucket. Los datos del objeto se adjuntan como un archivo adjunto DIME.

Para garantizar que un objeto no esté dañado en la red, puede calcular el MD5 de un objeto, colocarlo en Amazon S3 y comparar la Etag devuelta con el valor de MD5 calculado.

Si un objeto ya existe en un bucket, el nuevo objeto lo sobrescribirá porque Amazon S3 almacena la última solicitud de escritura. Sin embargo, Amazon S3 es un sistema distribuido. Si Amazon S3 recibe varias solicitudes de escritura para el mismo objeto casi simultáneamente, es posible que todos los objetos se almacenen, aunque al final solo uno de ellos gane. Amazon S3 no proporciona bloqueo de objetos; si lo necesita, asegúrese de integrarlo en la capa de aplicación.

Example

En este ejemplo, se ponen algunos datos y metadatos en el objeto “Nelson” del bucket “quotes”, se proporciona a un usuario (normalmente el propietario) acceso de FULL_CONTROL al objeto y se permite que partes anónimas puedan leer el objeto. En este ejemplo, no se muestra el archivo adjunto real.

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>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2006-03-01T12:00:00.183Z</LastModified>
  </PutObjectResponse>
</PutObjectResponse>
```

Elementos

- **Bucket**: el bucket en el que se va a agregar el objeto.
- **Key**: la clave que se va a asignar al objeto.
- **Metadata**: puede proporcionar pares de metadatos nombre-valor en el elemento de metadatos. Estos se almacenarán con el objeto.
- **ContentLength**: la longitud de los datos en bytes.
- **AccessControlList**: una lista de control de acceso para el recurso. Este elemento es opcional. Si se omite, el solicitante tiene acceso de FULL_CONTROL al objeto. Si el objeto ya existe, se reemplaza la política de control de acceso preexistente.

Respuestas

- **ETag**: la etiqueta de entidad es un hash MD5 del objeto que se puede utilizar para realizar recuperaciones condicionales del objeto mediante `GetObjectExtended`. La ETag solo refleja los cambios en el contenido de un objeto, no en los metadatos.
- **LastModified**: la marca temporal de Amazon S3 del objeto guardado.

Control de acceso

Para colocar objetos en un bucket, debe tener acceso de `WRITE` al bucket.

Activos relacionados

- [CopyObject \(API de SOAP\)](#)

CopyObject (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

Descripción

La operación `CopyObject` crea una copia de un objeto al especificar la clave y el bucket de un objeto de origen y la clave y el bucket de un destino.

Al copiar un objeto, puede conservar todos los metadatos (opción predeterminada) o especificar metadatos nuevos. Sin embargo, la ACL no se conserva y se establece en `private` para el usuario que realiza la solicitud. Para invalidar la configuración de ACL predeterminada, especifique una nueva ACL al generar una solicitud de copia. Para obtener más información, consulte [Uso de ACL](#).

Todas las solicitudes de copia se deben autenticar. Además, debe tener acceso de lectura al objeto de origen y de escritura al bucket de destino. Para obtener más información, consulte [Uso de la autenticación de acceso](#).

Para copiar un objeto solo en determinadas condiciones, por ejemplo, si la ETag coincide o si el objeto se modificó antes o después de una fecha específica, use los parámetros de solicitud `CopySourceIfUnmodifiedSince`, `CopyIfUnmodifiedSince`, `CopySourceIfMatch` o `CopySourceIfNoneMatch`.

Note

Puede que necesite configurar el tiempo de espera del socket de la pila de SOAP para copiar objetos grandes.

Sintaxis de la solicitud

```
<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>
...
<AccessControllList>
  <Grant>
    <Grantee xsi:type="user_type">
      <ID>user_id</ID>
      <DisplayName>display_name</DisplayName>
    </Grantee>
    <Permission>permission</Permission>
  </Grant>
  ...
</AccessControllList>
<CopySourceIfMatch>etag</CopySourceIfMatch>
<CopySourceIfNoneMatch>etag</CopySourceIfNoneMatch>
<CopySourceIfModifiedSince>date_time</CopySourceIfModifiedSince>
<CopySourceIfUnmodifiedSince>date_time</CopySourceIfUnmodifiedSince>
<AWSAccessKeyId>AWSAccessKeyId</AWSAccessKeyId>
<Timestamp>TimeStamp</Timestamp>
<Signature>Signature</Signature>
</CopyObject>
```

Parámetros de la solicitud

Nombre	Descripción	Obligatorio
SourceBucket	El nombre del bucket de origen.	Sí
	Tipo: cadena	
	Valor predeterminado: None	
	Restricciones: un bucket de origen válido.	
SourceKey	El nombre de la clave del objeto de origen.	Sí
	Tipo: cadena	

Nombre	Descripción	Obligatorio
	Valor predeterminado: None Restricciones: la clave de un objeto de origen válido al que tiene acceso de READ.	
DestinationBucket	El nombre del bucket de de destino. Tipo: cadena Valor predeterminado: None Restricciones: debe tener acceso de WRITE al bucket de destino.	Sí
DestinationKey	La clave del objeto de destino. Tipo: cadena Valor predeterminado: None Restricciones: debe tener acceso de WRITE al bucket de destino.	Sí

Nombre	Descripción	Obligatorio
MetadataDirective	Especifica si los metadatos se copian del objeto de origen o se sustituyen por los metadatos proporcionados en la solicitud. Tipo: cadena Predeterminado: COPY Valores válidos: COPY REPLACE Restricciones: los valores distintos de COPY o REPLACE producirán un error inmediato. No se puede copiar un objeto a sí mismo a menos que se especifique el encabezado MetadataDirective y se establezca su valor en REPLACE.	No
Metadata	Especifica los pares nombre-valor de los metadatos que se van a establecer para el objeto. Si MetadataDirective se establece en COPY, se ignoran todos los metadatos. Tipo: cadena Valor predeterminado: None Restricciones: ninguna.	No
AccessControlList	Concede acceso a los usuarios mediante direcciones de correo electrónico o un ID de usuario canónico. Tipo: cadena Valor predeterminado: None Restricciones: ninguna	No

Nombre	Descripción	Obligatorio
CopySourceIfMatch	Copia el objeto si su etiqueta de entidad (ETag) coincide con la etiqueta especificada; de lo contrario, devuelve un PreconditionFailed. Tipo: cadena Valor predeterminado: None Restricciones: ninguna. Si la ETag no coincide, el objeto no se copia.	No
CopySourceIfNoneMatch	Copia el objeto si su etiqueta de entidad (ETag) es diferente de la ETag especificada; de lo contrario, devuelve un error. Tipo: cadena Valor predeterminado: None Restricciones: ninguna.	No
CopySourceIfUnmodifiedSince	Copia el objeto si no se ha modificado desde el momento especificado; de lo contrario, devuelve PreconditionFailed. Tipo: dateTime Valor predeterminado: none	No
CopySourceIfModifiedSince	Copia el objeto si se ha modificado desde el momento especificado; de lo contrario devuelve un error. Tipo: dateTime Valor predeterminado: none	No

Sintaxis de la respuesta

```
<CopyObjectResponse xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"etag"</ETag>
    <LastModified>timestamp</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Elementos de respuesta

A continuación, se muestra una lista de elementos de respuesta.

Note

La API de SOAP no devuelve espacios en blanco adicionales. Los espacios en blanco adicionales solo los devuelve la API de REST.

Nombre	Descripción
Etag	Devuelve la etag del nuevo objeto. La ETag solo refleja los cambios en el contenido de un objeto, no en los metadatos. Tipo: cadena Antecesor: CopyObjectResult
LastModified	Devuelve la fecha en la que se modificó el objeto por última vez. Tipo: cadena Antecesor: CopyObjectResult

Para obtener información sobre los elementos de respuesta generales, consulte [Uso de encabezados de respuesta a errores de REST](#).

Errores especiales

No hay errores especiales para esta operación. Para obtener información acerca de errores de Amazon S3 generales, consulte [Lista de códigos de error](#).

Ejemplos

En este ejemplo se copia el objeto `flotsam` desde el bucket `pacific` al objeto `jetsam` del bucket `atlantic`, conservando sus metadatos.

Solicitud de muestra

```
<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>
```

Respuesta de ejemplo

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

En este ejemplo, se copia el objeto “`tweedledee`” del bucket de `wonderland` al objeto “`tweedledum`” del bucket de `wonderland`, lo que sustituye sus metadatos.

Solicitud de muestra

```
<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>
```

Respuesta de ejemplo

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Activos relacionados

- [PutObject \(API de SOAP\)](#)
- [PutObjectInline \(API de SOAP\)](#)

GetObject (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación `GetObject` devuelve la versión actual de un objeto. Si intenta `GetObject` un objeto que tiene un marcador de eliminación como versión actual, S3 devuelve un error 404. No puede usar la API de SOAP para recuperar una versión específica de un objeto. Para ello, utilice la API de

REST. Para obtener más información, consulte [Control de versiones](#). Para obtener más opciones, utilice la operación [GetObjectExtended \(API de SOAP\)](#).

Note

Los nombres de clave de objeto con el valor "soap" no son compatibles con las [solicitudes de tipo host virtual](#). Para los valores de los nombres de las claves de los objetos en los que se utiliza "soap", se debe utilizar en su lugar una [URL de tipo ruta](#).

Example

En este ejemplo, se obtiene el objeto "Nelson" del bucket "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>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
</GetObjectResponse>
</GetObjectResponse>
```

Elementos

- **Bucket**: el bucket del que se va a recuperar el objeto.
- **Key**: la clave que identifica el objeto.
- **GetMetadata**: los metadatos se devuelven con el objeto si es verdadero.
- **GetData**: los datos del objeto se devuelven si es verdadero.
- **InlineData**: si es verdadero, los datos se devuelven, codificados en base 64, como parte del cuerpo de SOAP de la respuesta. Si es falso, los datos se devuelven como un archivo adjunto de SOAP. La opción **InlineData** no es adecuada para su uso con objetos grandes. El sistema limita esta operación al trabajo con 1 MB de datos o menos. Una solicitud de **GetObject** con el indicador **InlineData** establecido producirá un error con el código de estado **InlineDataTooLargeError** si el parámetro **Datos** resultante hubiera codificado más de 1 MB. Para descargar objetos grandes, considere llamar a **GetObject** sin establecer el indicador **InlineData** o, en su lugar, use la API de REST.

Elementos devueltos

- **Metadata**: los metadatos de pares nombre-valor almacenados con el objeto.
- **Data**: si **InlineData** era verdadero en la solicitud, contiene los datos del objeto codificados en base 64.
- **LastModified**: la hora en la que se almacenó el objeto en Amazon S3.
- **ETag**: la etiqueta de entidad del objeto. Este es un hash del objeto que se puede usar para realizar obtenciones condicionales. La **ETag** solo refleja los cambios en el contenido de un objeto, no en los metadatos.

Control de acceso

Puede leer un objeto solo si se le ha concedido acceso de **READ** al objeto.

Descargas fragmentadas y reanudables de SOAP

Para ofrecer flexibilidad GET, Amazon S3 admite descargas fragmentadas y reanudables.

Seleccione entre lo siguiente:

- Para descargas de objetos grandes, es posible que desee dividirlos en fragmentos más pequeños. Para obtener más información, consulte [Range GET](#)
- Para operaciones GET que produzcan un error, puede diseñar la aplicación para que descargue el resto del archivo en lugar de todo el archivo. Para obtener más información, consulte [Recuperación del error GET de REST](#)

Range GET

En el caso de algunos clientes, es posible que desee dividir las descargas grandes en descargas más pequeñas. Para dividir GET en unidades más pequeñas, use Range.

Antes de poder dividir GET en unidades más pequeñas, debe determinar su tamaño. Por ejemplo, la siguiente solicitud obtiene el tamaño del objeto 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 devuelve la respuesta siguiente.

```
<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>

```

A continuación, se muestra una solicitud que descarga el primer megabyte del objeto 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 devuelve el primer megabyte del archivo y la Etag del archivo.

```

<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>
```

Para asegurarse de que el archivo no ha cambiado desde que se descargó la parte anterior, especifique el elemento `IfMatch`. Aunque el elemento `IfMatch` no es obligatorio, se recomienda para el contenido que pueda cambiar.

A continuación, se muestra una solicitud que obtiene el resto del archivo mediante el encabezado de solicitud `IfMatch`.

```
<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>"828ef3fd96f00ad9f27c383fc9ac7f"</IfMatch>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>
```

Amazon S3 devuelve la siguiente respuesta y el resto del archivo.

```
<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>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
```

```
</GetObjectResponse>
</GetObjectResponse>
```

GetObject versionado

La siguiente solicitud devuelve la versión especificada del objeto del bucket.

```
<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>
```

Respuesta de ejemplo

```
<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>
```


Recuperación del error GET de REST

Si un objeto GET produce un error, puede obtener el resto del archivo especificando el rango que desea descargar. Para ello, debe obtener el tamaño del objeto mediante `ListBucket` y realizar un rango GET en el resto del archivo. Para obtener más información, consulte [GetObjectExtended \(API de SOAP\)](#).

Activos relacionados

[Operaciones en objetos \(API de SOAP\)](#)

GetObjectExtended (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

`GetObjectExtended` es exactamente igual que [GetObject \(API de SOAP\)](#), excepto que admite los siguientes elementos adicionales que se pueden utilizar para realizar gran parte de la misma funcionalidad que proporcionan los encabezados HTTP GET (vaya a <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>).

`GetObjectExtended` admite los siguientes elementos además de los compatibles con `GetObject`:

- `ByteRangeStart`, `ByteRangeEnd`: estos elementos especifican que solo se debe recuperar una parte de los datos del objeto. Siguen el comportamiento de los rangos de bytes HTTP (vaya a <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35>).
- `IfModifiedSince`: devuelve el objeto solo si la marca temporal del objeto es posterior a la marca temporal especificada. (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.25>)
- `IfUnmodifiedSince`: devuelve el objeto solo si la marca temporal del objeto es anterior o igual a la marca temporal especificada. (Vaya a <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.28>)
- `IfMatch`: devuelve el objeto solo si su ETag coincide con las etiquetas proporcionadas. (Vaya a <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24>)
- `IfNoneMatch`: devuelve el objeto solo si su ETag no coincide con las etiquetas proporcionadas. (Vaya a <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26>)

- `ReturnCompleteObjectOnConditionFailure:ReturnCompleteObjectOnConditionFailure`: si es verdadero, si la solicitud incluye un elemento de rango y uno o ambos elementos `IfUnmodifiedSince`/`IfMatch`, y la condición produce un error, devuelva el objeto completo en lugar de un error. Esto habilita la funcionalidad `If-Range` (vaya a <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27>).

DeleteObject (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación `DeleteObject` quita el objeto especificado de Amazon S3. Una vez eliminado, no hay ningún método para restaurar o recuperar un objeto.

Note

Si elimina un objeto que no existe, Amazon S3 devolverá un mensaje de éxito (no un mensaje de error).

Example

En este ejemplo, se elimina el objeto “Nelson” del bucket “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>
```

Elementos

- **Bucket** : el bucket que contiene el objeto.
- **Key** : la clave que identifica el objeto.

Control de acceso

Puede eliminar un objeto solo si tiene acceso de **WRITE** al bucket, independientemente de quién sea el propietario del objeto o de los derechos que se le concedan.

GetObjectAccessControlPolicy (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación `GetObjectAccessControlPolicy` obtiene la política de control de acceso de un objeto.

Example

En este ejemplo, se recupera la política de control de acceso para el objeto “Nelson” del bucket “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>
```

Cuerpo de respuesta

La respuesta contiene la política de control de acceso del bucket. Para obtener una explicación de esta respuesta, [Política de acceso de SOAP](#).

Control de acceso

Debe tener derechos READ_ACP sobre el objeto para poder recuperar la política de control de acceso de un objeto.

SetObjectAccessControlPolicy (API de SOAP)

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

La operación `SetObjectAccessControlPolicy` establece la política de control de acceso de un objeto existente. Si se ejecuta correctamente, la política de control de acceso anterior del objeto se sustituye por completo por la política de control de acceso especificada.

Example

En este ejemplo, se proporciona al usuario especificado (normalmente el propietario) acceso de `FULL_CONTROL` al objeto “Nelson” desde el bucket “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>
```

Key

Control de acceso

Debe tener derechos de WRITE_ACP sobre el objeto para poder establecer la política de control de acceso de un bucket.

Autenticar solicitudes SOAP

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API REST o los SDK de AWS.

Cada solicitud no anónima debe contener información de autenticación para establecer la identidad de la solicitud principal que hace la solicitud. En SOAP, la información de autenticación se coloca en los siguientes elementos de la solicitud SOAP:

- Su ID de clave de acceso de AWS

Note

Cuando se hacen solicitudes autenticadas SOAP, no se admiten credenciales de seguridad temporales. Para obtener más información acerca de estos tipos de credenciales, consulte [Realizar solicitudes](#).

- **Timestamp:** Debe ser una fecha y hora (consulte <http://www.w3.org/TR/xmlschema-2/#dateTime>) en la zona horaria del Tiempo universal coordinado (Tiempo medio de Greenwich), como 2009-01-01T12:00:00.000Z. La autorización fallará si esta marca temporal está adelantada más de 15 minutos que el reloj de los servicios de Amazon S3.
- **Signature:** el resumen RFC 2104 HMAC-SHA1 (consulte <http://www.ietf.org/rfc/rfc2104.txt>) de la concatenación de "AmazonS3" + OPERATION + Timestamp, con su clave de acceso secreta de AWS como clave. Por ejemplo, en la siguiente solicitud de ejemplo

CreateBucket, el elemento de firma debe contener el resumen HMAC-SHA1 del valor "AmazonS3CreateBucket2009-01-01T12:00:00.000Z":

Por ejemplo, en la siguiente solicitud de ejemplo CreateBucket, el elemento de firma debe contener el resumen HMAC-SHA1 del valor "AmazonS3CreateBucket2009-01-01T12:00:00.000Z":

Example

```
<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

Las solicitudes SOAP, tanto autenticadas como anónimas, deben enviarse a Amazon S3 con SSL. Amazon S3 devuelve un error si envía una solicitud SOAP por HTTP.

Important

Debido a diferentes interpretaciones con respecto a cómo se debe descartar la precisión de tiempo adicional, los usuarios de .NET deben tener cuidado y no enviar a Amazon S3 demasiadas marcas temporales específicas. Esto se puede lograr creando objetos DateTime manualmente con solo una precisión de milisegundos.

Configurar políticas de acceso con SOAP

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

El control de acceso se puede establecer en el momento en el que se escribe un bucket o un objeto incluyendo el elemento "AccessControlList" en la solicitud para CreateBucket, PutObjectInline o PutObject. El elemento AccessControlList se describe en [Administración de identidades y accesos para Amazon S3](#). Si no se ha especificado ninguna lista de control de acceso con estas operaciones, el recurso se creará con una política de acceso predeterminada que otorga al solicitante un acceso FULL_CONTROL (este será el caso incluso aunque la solicitud sea del tipo PutObjectInline o PutObject para un objeto que ya exista).

A continuación presentamos una solicitud que escribe datos en un objeto, hace que el objeto sea legible desde principales anónimos y proporciona al usuario especificado derechos FULL_CONTROL sobre el bucket (la mayoría de desarrolladores querrán concederse a sí mismos un acceso FULL_CONTROL a su propio bucket).

Example

A continuación aparece una solicitud que escribe datos en un objeto y hace que el objeto sea legible desde principales anónimos.

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>
</PutObjectInline>
```



```
</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>"828ef3fd9a96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

Esta política de control de acceso se puede leer o configurar para un bucket o un objeto existentes mediante los métodos `GetBucketAccessControlPolicy`, `GetObjectAccessControlPolicy`, `SetBucketAccessControlPolicy` y `SetObjectAccessControlPolicy`. Para obtener más información, consulte la explicación detallada de estos métodos.

Elementos comunes

Puede incluir los siguientes elementos relacionados a autorización con cualquier solicitud de SOAP:

- **AWSAccessKeyId**: el ID de clave de acceso de AWS del solicitante
- **Timestamp**: La hora actual en el sistema
- **Signature**: La firma de la solicitud

Respuestas de error de SOAP

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API de REST o los SDK de AWS.

En SOAP, el cliente recibe un resultado de error como una falla de SOAP, con el código de respuesta HTTP 500. Si no recibe una falla de SOAP, su solicitud se completó correctamente. El código de falla de SOAP de Amazon S3 consta de un código de falla SOAP 1.1 estándar (“Servidor” o “Cliente”) concatenado con el código de error específico de Amazon S3. Por ejemplo: “Server.InternalError” o “Client.NoSuchBucket”. El elemento de cadena de falla de SOAP incluye un mensaje de error genérico legible en inglés. Por último, el elemento de detalle de falla de SOAP incluye información variada relacionada con el error.

Por ejemplo, si intenta eliminar el objeto “Fred”, que no existe, el cuerpo de la respuesta de SOAP incluye una falla “NoSuchKey” de SOAP.

El ejemplo siguiente muestra una respuesta de error de SOAP de ejemplo.

```
<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>
```

En la siguiente tabla, se explican los elementos de respuesta de error de SOAP

Nombre	Descripción
Detail	Contenedor de la clave implicada en el error Tipo: Contenedor Antecesor: Body.Fault
Fault	Contenedor de información sobre errores. Tipo: Contenedor Antecesor: Body
Faultcode	El código de error es una cadena que identifica de forma exclusiva una condición de error. Está diseñado para que los programas que detectan y

Nombre	Descripción
	administran errores por tipo puedan leerlo y comprenderlo. Para obtener más información, consulte Lista de códigos de error. Tipo: cadena Antecesor: Body.Fault
Faultstring	La cadena de error contiene una descripción genérica de la condición de error en inglés. Está destinado a un público humano. Los programas simples muestran el mensaje directamente al usuario final si se detecta una condición de error que no sabe cómo controlar o no le interesa hacerlo. Los programas sofisticados con un control de errores más exhaustivo y una internacionalización adecuada tienen más probabilidades de ignorar la cadena de error. Tipo: cadena Antecesor: Body.Fault
Key	Identifica la clave implicada en el error Tipo: cadena Antecesor: Body.Fault

Apéndice: autenticación de solicitudes (AWS Signature Version 2)

Important

En esa sección, se explica cómo se autentican solicitudes con AWS Signature Version 2. Signature Version 2 se va a desactivar (esta característica quedará obsoleta). Amazon S3 solo aceptará solicitudes de API que estén firmadas con Signature Version 4. Para obtener más información, consulte [AWS Signature Version 2 se va a desactivar \(obsoleta\) para Amazon S3](#)

Signature Version 4 es compatible con todas las Regiones de AWS y es la única versión que puede utilizarse en las regiones nuevas. Para obtener más información, consulte [Autenticación de solicitudes \(AWS Signature Version 4\)](#) en la Referencia de la API de Amazon Simple Storage Service.

Amazon S3 le ofrece la posibilidad de identificar qué versión de la API de firma se utilizó para firmar una solicitud. Es importante que identifique si alguno de sus flujos de trabajo está utilizando la firma de Signature Version 2 y que los actualice para que utilicen Signature Version 4 con el fin de evitar que su negocio resulte afectado.

- Si utiliza los registros de eventos de CloudTrail (opción recomendada), consulte [Identificación de las solicitudes de Amazon S3 Signature Version 2 mediante CloudTrail](#) para saber cómo buscar e identificar dichas solicitudes.
- Si utiliza los registros de acceso del servidor de Amazon S3, consulte [Uso de los registros de acceso del servidor de Amazon S3 para identificar solicitudes](#)

Temas

- [Autenticación de solicitudes con la API de REST \(AWS Signature Version 2\)](#)
- [Firma y autenticación de solicitudes de REST \(AWS Signature Version 2\)](#)
- [Cargas basadas en el navegador con POST \(AWS Signature Version 2\)](#)

Autenticación de solicitudes con la API de REST (AWS Signature Version 2)

Cuando accede a Amazon S3 con REST, debe proporcionar los siguientes elementos en su solicitud para que esta se pueda autenticar:

Elementos de la solicitud

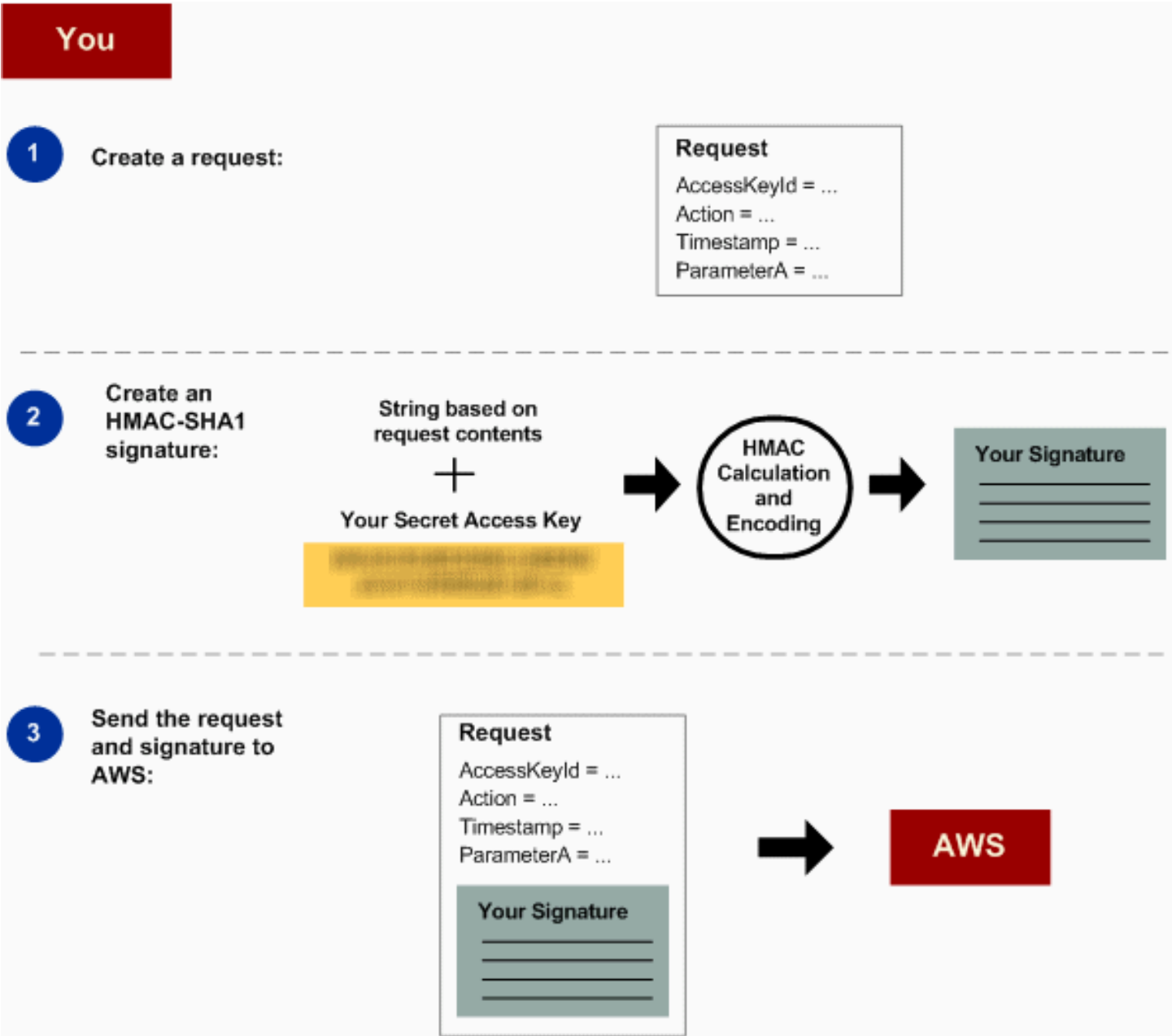
- ID de clave de acceso de AWS: cada solicitud debe contener el ID de clave de acceso de la identidad que utiliza para enviar la solicitud.
- Firma: cada solicitud debe incluir una firma de solicitud válida; de lo contrario, la solicitud se rechaza.

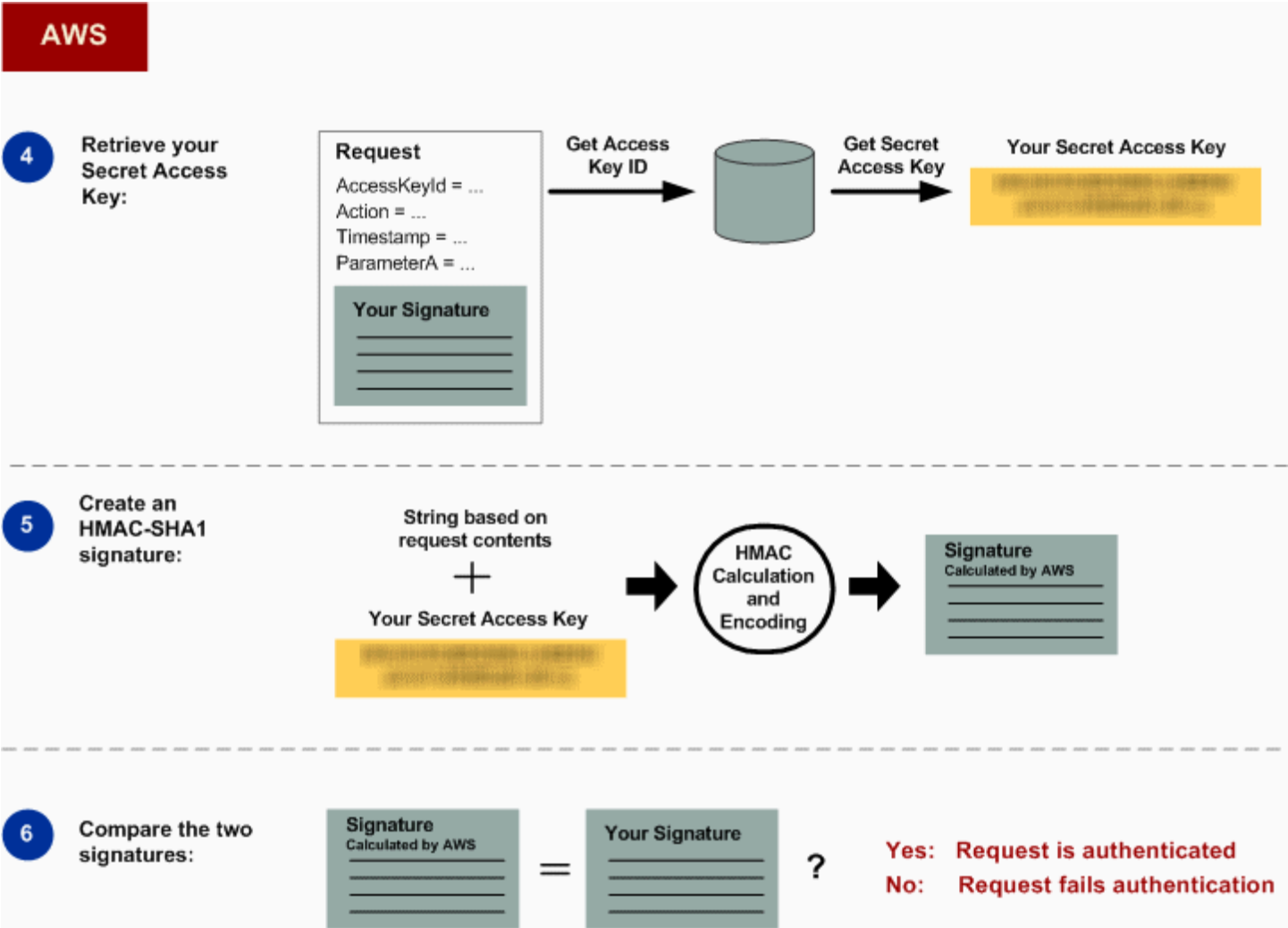
Una firma de solicitud se calcula mediante su clave de acceso secreta, que es un secreto compartido que solo conocen usted y AWS.

- Marca temporal: cada solicitud debe incluir la fecha y la hora en que se creó la solicitud, representadas como una cadena en Universal Time Coordinated (UTC, Hora universal coordinada).
- Fecha: cada solicitud debe incluir la marca temporal de la solicitud.

Según la acción de la API que utiliza, puede proporcionar una hora y una fecha de vencimiento para la solicitud, en lugar o además de la marca temporal. Consulte el tema de autenticación para la acción particular para determinar lo que requiere.

A continuación se indican los pasos generales para autenticar solicitudes en Amazon S3. Se asume que usted cuenta con las credenciales de seguridad, el ID de clave de acceso y la clave de acceso secreta necesarios.





Información de autenticación detallada

Para obtener información detallada acerca de la autenticación de REST, consulte [Firma y autenticación de solicitudes de REST \(AWS Signature Version 2\)](#).

Firma y autenticación de solicitudes de REST (AWS Signature Version 2)

Temas

- [Uso de credenciales de seguridad temporales](#)
- [El encabezado de autenticación](#)
- [Solicitar canonicalización para firmas](#)
- [Crear elemento CanonicalizedResource](#)
- [Crear el elemento CanonicalizedAmzHeaders](#)
- [Elementos StringToString de los encabezados HTTP: posicionales frente a denominados](#)
- [Requisitos de marca temporal](#)
- [Ejemplos de autenticación](#)
- [Problemas de firma con solicitudes REST](#)
- [Alternativa de autenticación por cadena de consulta de solicitudes](#)

Note

En este tema se explica la autenticación de solicitudes mediante el uso de Signature Versión 2. Amazon S3 ya es compatible con Signature Version 4. Esta última versión de Signature puede utilizarse en todas las regiones, y las nuevas regiones solo permitirán el uso de Signature versión 4 tras el 30 de enero de 2014. Para obtener más información, vaya a [Autenticación de solicitudes \(AWS Signature Version 4\)](#) en la Referencia de la API de Amazon Simple Storage Service.

La autenticación es el proceso que consiste en demostrar la identidad del usuario a un sistema. La identidad es un factor importante en las decisiones sobre el control de accesos en Amazon S3. Las solicitudes se admiten o se rechazan, en parte, en función de la identidad del solicitante. Por ejemplo, el derecho a crear buckets se reserva a los desarrolladores registrados, y el derecho a crear objetos en un bucket (de manera predeterminada) se reserva al propietario del bucket en cuestión. Como desarrollador, estará realizando solicitudes que invoquen estos privilegios, por lo que tendrá

que demostrar su identidad al sistema autenticando sus solicitudes. En esta sección le demostramos cómo.

 Note

El contenido de esta sección no se aplica al método HTTP POST. Para obtener más información, consulte [Cargas basadas en el navegador con POST \(AWS Signature Version 2\)](#).

La API de REST de Amazon S3 usa un esquema HTTP personalizado basado en un HMAC (Hash Message Authentication Code) con clave para la autenticación. Para autenticar una solicitud, primero ha de concatenar los elementos seleccionados en la solicitud para formar una cadena. A continuación, utilice su clave de acceso secreta de AWS para calcular el HMAC de esa cadena. Este proceso se denomina informalmente "firmar la solicitud", y al resultado del algoritmo HMAC se le llama la firma, ya que simula las propiedades de seguridad de una firma real. Por último, tendrá que agregar esta firma como parámetro de la solicitud empleando la sintaxis descrita en esta sección.

Cuando el sistema recibe una solicitud autenticada, toma la clave de acceso secreta de AWS que usted afirma tener y la utiliza del mismo modo para computar una firma para el mensaje recibido. A continuación, compara la firma calculada con la firma que presenta el solicitante. Si ambas firmas coinciden, el sistema concluye que el solicitante debe de tener acceso a la clave de acceso secreta de AWS y, por lo tanto, actúa con la autoridad de la solicitud principal para la que se emitió la clave. Si las dos firmas no coinciden, la solicitud se abandona y el sistema responde con un mensaje de error.

Example Solicitud REST autenticada de Amazon S3

```
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=
```

Uso de credenciales de seguridad temporales

Si firma su solicitud con credenciales de seguridad temporales (consulte [Realizar solicitudes](#)), deberá incluir el token de seguridad correspondiente en la solicitud agregando el encabezado `x-amz-security-token`.

Al obtener credenciales de seguridad temporales con la API de AWS Security Token Service, la respuesta incluye credenciales de seguridad temporales y un token de sesión. El valor del token de la sesión lo facilita en el encabezado `x-amz-security-token` al enviar solicitudes a Amazon S3. Para obtener más información acerca de la API de AWS Security Token Service proporcionada por IAM, consulte la sección [Acción](#) en la Guía de referencia de la API de AWS Security Token Service.

El encabezado de autenticación

La API de REST de Amazon S3 usa el encabezado estándar HTTP `Authorization` para transmitir la información de autenticación. (El nombre "encabezado estándar" no es demasiado preciso, ya que lo que transmite es información de autenticación, no autorización). Según el esquema de autenticación de Amazon S3, el encabezado de autorización tiene la siguiente forma:

```
Authorization: AWS AWSAccessKeyId:Signature
```

Los desarrolladores reciben una ID de clave de acceso de AWS y una clave de acceso secreta de AWS cuando se registran. Para la autenticación de solicitudes, el elemento `AWSAccessKeyId` identifica la ID de clave de acceso utilizada para computar la firma e, indirectamente, también al desarrollador que realiza la solicitud.

El elemento `Signature` es el RFC 2104 HMAC-SHA1 de los elementos seleccionados de la solicitud, y por tanto, la parte `Signature` del encabezado de la autorización variará entre solicitudes. Si la firma de la solicitud calculada por el sistema coincide con la `Signature` incluida en la solicitud, el solicitante habrá demostrado la posesión de la clave de acceso secreta de AWS. La solicitud será procesada bajo la identidad y con la autoridad del desarrollador al que se le emitió la clave.

A continuación presentamos pseudogramática que ilustra la construcción del encabezado de solicitudes `Authorization`. (en el ejemplo, `\n` representa el punto de código en Unicode U+000A, normalmente denominado nueva línea).

```

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 es un algoritmo definido por [RFC 2104, código de autenticación de mensajes en clave-hash](#). El algoritmo toma como entrada dos cadenas de bytes, una clave y un mensaje. Para la autenticación de solicitudes en Amazon S3, utilice su clave de acceso secreta de AWS (YourSecretAccessKey) como clave y la codificación UTF-8 de StringToSign como mensaje. El resultado del HMAC SHA1 también es una cadena de bytes, denominada resumen. El parámetro Signature de la solicitud se construye en Base64 mediante la codificación de este resumen.

Solicitar canonicalización para firmas

Recuerde que cuando el sistema recibe una solicitud autenticada, compara la firma de la solicitud computada con la firma proporcionada en la solicitud en StringToSign. Por este motivo, se debe computar la firma utilizando el mismo método que use Amazon S3. Al proceso de poner una solicitud en una forma acordada para la firma se le denomina canonicalización.

Crear elemento CanonicalizedResource

CanonicalizedResource representa el recurso de Amazon S3 que es el destino de la solicitud. Debe construirlo para adaptarlo a una solicitud REST de la siguiente forma:

1. Comience por una cadena vacía ("").
2. Si la solicitud especifica un bucket con el encabezado del host HTTP (estilo de alojamiento virtual), adjunte el nombre del bucket precedido por un "/" (por ejemplo, "/nombredelbucket"). Para las

solicitudes con estilo de ruta y las solicitudes que no se dirigen a un bucket, no haga nada. Para obtener más información acerca de las solicitudes de tipo alojamiento virtual, consulte [Alojamiento virtual de buckets](#).

Para una solicitud de tipo alojamiento virtual "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg", el CanonicalizedResource es "/"awsexamplebucket1".

Para la solicitud de tipo ruta, "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg", el CanonicalizedResource es "".

3. Adjunte la parte de la ruta del URI de la solicitud HTTP sin decodificar, hasta la cadena de la consulta, sin incluirla.

Para una solicitud de tipo alojamiento virtual "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg", el CanonicalizedResource es "/"awsexamplebucket1/photos/puppy.jpg".

Para una solicitud de tipo ruta, "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg", el CanonicalizedResource es "/awsexamplebucket1/photos/puppy.jpg". En este punto, el CanonicalizedResource es el mismo tanto para la solicitud de estilo de alojamiento virtual como para la de estilo de ruta.

Para una solicitud que no se dirija a un bucket, como [GET Service](#), adjunte "/".

4. Si la solicitud se dirige a un subrecurso, como ?versioning, ?location, ?acl, ?lifecycle o ?versionid, adjunte el subrecurso, su valor (si lo tiene) y el signo de interrogación. Tenga en cuenta que, en caso de que haya varios subrecursos, estos deben estar ordenados lexicográficamente por el nombre del subrecurso y separados por "&". Por ejemplo: ?acl&versionId=*valor*.

Los subrecursos que se deben incluir cuando se crea el elemento CanonicalizedResource son los siguientes: acl, lifecycle, location, logging, notification, partNumber, policy, requestPayment, uploadId, uploads, versionId, versioning, versions y website.

Si la solicitud especifica parámetros de la cadena de consulta que sobrescriban a los valores del encabezado de respuesta (véase [Get Object](#)), adjunte los parámetros de la cadena de consulta y sus valores. Al firmar no estará codificando estos valores. Sin embargo, al realizar la solicitud, debe codificar estos valores de parámetros. Los parámetros de la cadena de consulta en una solicitud GET incluyen response-content-type, response-content-language,

`response-expires`, `response-cache-control`, `response-content-disposition` y `response-content-encoding`.

El parámetro de la cadena de consulta `delete` debe incluirse al crear el `CanonicalizedResource` para una solicitud de eliminación de varios objetos.

Los elementos del `CanonicalizedResource` que provienen del URI de la solicitud HTTP deben firmarse literalmente según aparecen en la solicitud HTTP, incluidos los caracteres meta de codificación de la URL.

El `CanonicalizedResource` podría ser diferente del URI de la solicitud HTTP. En particular, si su solicitud usa el encabezado HTTP `Host` para especificar un bucket, el bucket no aparecerá en el URI de la solicitud HTTP. Sin embargo, el `CanonicalizedResource` seguirá incluyendo el bucket. Los parámetros de la cadena de consulta podrían aparecer también en el URI de la solicitud, pero no están incluidos en el `CanonicalizedResource`. Para obtener más información, consulte [Alojamiento virtual de buckets](#).

Crear el elemento `CanonicalizedAmzHeaders`

Para construir la parte de `CanonicalizedAmzHeaders` de `StringToSign`, seleccione todos los encabezados de las solicitudes HTTP que comiencen por "x-amz" (con una comparación que no distinga mayúsculas y minúsculas) y emplee el siguiente proceso.

1. Convierta cada nombre de encabezado HTTP a minúsculas. Por ejemplo, "X-Amz-Date" se ha de convertir en "x-amz-date".
2. Ordene la colección de encabezados lexicográficamente por nombre de encabezado.
3. Combine los campos de encabezado que tengan el mismo nombre en un par de encabezados "nombre-de-encabezado:lista-de-valores-separados-por-comas" según se indica en la RFC 2616, sección 4.2, sin espacios entre los valores. Por ejemplo, dos encabezados de metadatos "x-amz-meta-username: fred" y "x-amz-meta-username: barney" se combinarían en el encabezado único "x-amz-meta-username: fred,barney".
4. "Desdoble" los encabezados largos que ocupen varias líneas (según lo permite la RFC 2616, sección 4.2) sustituyendo el espacio de desdoble (incluida la línea nueva) por un espacio único.
5. Elimine los espacios que haya en torno a los dos puntos del encabezado. Por ejemplo, el encabezado "x-amz-meta-username: fred,barney" se convertiría en "x-amz-meta-username:fred,barney".

6. Por último, adjunte un nuevo carácter (U+000A) a cada encabezado canonicalizado de la lista resultante. Construye el elemento CanonicalizedResource concatenando todos los encabezados de esta lista en una cadena única.

Elementos StringToString de los encabezados HTTP: posicionales frente a denominados

Los primeros elementos del encabezado de `StringToSign` (Content-Type, Date y Content-MD5) tienen naturaleza posicional. `StringToSign` no incluye los nombres de estos encabezados, solo sus valores procedentes de la solicitud. Por contraste, los elementos "x-amz-" son denominados. Tanto los nombres como los valores de los encabezados aparecen en `StringToSign`.

Si un encabezado posicional al que se llama en la definición de `StringToSign` no está presente en su solicitud (por ejemplo, Content-Type o Content-MD5 son opcionales para las solicitudes PUT y no significativos para las solicitudes GET), sustituya la cadena vacía ("") para dicha posición.

Requisitos de marca temporal

Una marca temporal válida (en la que se use el encabezado HTTP Date o una alternativa x-amz-date) es obligatoria para las solicitudes autenticadas. Además, la marca temporal del cliente incluida en una solicitud autenticada debe estar en el intervalo de 15 minutos del momento, según la hora del sistema de Amazon S3 en el que se recibe la solicitud. En caso contrario, ocurrirá un error en la solicitud y recibirá el código de error `RequestTimeTooSkewed`. El objetivo de estas restricciones es limitar la posibilidad de que las solicitudes interceptadas pudieran ser reproducidas por un adversario. Para implementar una protección más sólida ante el acceso no autorizado, use el transporte HTTPS para solicitudes autenticadas.

Note

La limitación de validación con la fecha de solicitud solo se aplica a las solicitudes autenticadas en las que no se use la autenticación por cadena de consulta. Para obtener más información, consulte [Alternativa de autenticación por cadena de consulta de solicitudes](#).

Algunas bibliotecas de cliente HTTP no permiten configurar el encabezado Date para una solicitud. Si encuentra problemas al incluir el valor del encabezado "Date" en los encabezados canonicalizados, puede establecer la marca temporal de la solicitud mediante un encabezado "x-

amz-date". El valor del encabezado x-amz-date debe encontrarse en uno de los formatos que se indican en RFC 2616 (<http://www.ietf.org/rfc/rfc2616.txt>). Cuando hay un encabezado x-amz-date presente en una solicitud, el sistema ignorará cualquier encabezado Date al computarla firma de la misma. Por tanto, si incluye el encabezado x-amz-date, use la cadena vacía para Date al construir el StringToSign. Consulte la siguiente sección para ver un ejemplo.

Ejemplos de autenticación

Los ejemplos de esta sección emplean las credenciales (no funcionales) de la siguiente tabla.

Parámetro	Valor
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

En el ejemplo, el formato de StringToSign no es significativo, y el \n es el punto de código Unicode U+000A, normalmente denominado línea nueva. Además, en los ejemplos se usa "+0000" para designar la zona horaria. También puede usar "GMT" para designar la zona horaria, pero las firmas que se mostrarán en los ejemplos serán diferentes.

Object GET

En este ejemplo se obtiene un objeto del bucket awsexamplebucket1.

Solicitud	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 AKIAIOSFODNN7EXAMPLE: qgk2+6Sv9/oM7G3qLEjTH1a1l1g=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:36:42 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Tenga en cuenta que el CanonicalizedResource incluye el nombre del bucket, pero la URI de la solicitud HTTP no. (El encabezado del host especifica el bucket).

Note

La siguiente secuencia de comandos de Python calcula la firma anterior, utilizando los parámetros proporcionados. Puede utilizar este script para construir sus propias firmas, reemplazando las claves y StringToSign según corresponda.

```
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()}")
```

Object PUT

En este ejemplo se coloca un objeto en el bucket `awsexamplebucket1`.

Solicitud	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</pre>	<pre>PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Solicitud	StringToSign
<pre>Authorization: AWS AKIAIOSFODNN7EXAMPLE LE: iqRzw+ileNPu1fhspnRs8n0jjIA=</pre>	

Tenga en cuenta el encabezado Content-Type en la solicitud y en el StringToSign. Tenga en cuenta también que el Content-MD5 se deja vacío en el StringToSign, porque no aparece en la solicitud.

Enumeración

En este ejemplo se muestra el contenido del bucket `awsexamplebucket1`.

Solicitud	StringToSign
<pre>GET /?prefix=photos&max-keys=50&marker=puppy HTTP/1.1 User-Agent: Mozilla/5.0 Host: awsexamplebucket1.s3.us-west-1.amazonaws.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>

Tenga en cuenta la barra al final del CanonicalizedResource y la ausencia de parámetros en la cadena de consulta.

Fetch

En este ejemplo se obtiene el subrecurso de la política de control de acceso para el bucket `"awsexamplebucket1"`.

Solicitud	StringToSign
<pre>GET /?acl HTTP/1.1</pre>	<pre>GET\n \n</pre>

Solicitud	StringToSign
<pre>Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 19:44:46 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: 82ZHiFIjc+WbcwFKGUVeQspPn+0=</pre>	<pre>\n Tue, 27 Mar 2007 19:44:46 +0000\n /awsexamplebucket1/?acl</pre>

Ahora, tenga en cuenta cómo el parámetro de la cadena de consulta del subrecurso está incluido en el CanonicalizedResource.

Eliminar

En este ejemplo se elimina un objeto del bucket "awsexamplebucket1" de tipo ruta y la alternativa Date.

Solicitud	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: LE:XbyTlbQdu9Xw5o8P4iMwPktxQd8=</pre>	<pre>DELETE\n \n \n Tue, 27 Mar 2007 21:20:26 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Tenga en cuenta que hemos usado el método alternativo "x-amz-date" para especificar la fecha (porque nuestra biblioteca de clientes, por ejemplo, nos impide establecer la fecha). En este caso, el x-amz-date tiene prioridad sobre el encabezado Date. Por tanto, la fecha introducida en la firma ha de contener el valor del encabezado x-amz-date.

Cargar

En este ejemplo se carga un objeto en un bucket con alojamiento virtual de estilo CNAME y con metadatos.

Solicitud	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 X-Amz-Meta-ReviewedBy: jane@exam ple.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 AKIAIOSFODNN7EXAMP LE: jtBQa0Aq+DkULFI8qrpwIjGEx0E=</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:c rc32\n 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>

Tenga en cuenta cómo los encabezados "x-amz-" se ordenan, se les recortan los espacios extra o se convierten en minúscula. Tenga en cuenta también que se han agrupado varios encabezados con el mismo nombre utilizando comas para separar valores.

Tenga en cuenta que solo los encabezados de entidades HTTP Content-Type y Content-MD5 aparecen en el StringToSign. El resto de encabezados de entidades Content-* no aparecen.

Además, tenga en cuenta que el CanonicalizedResource incluye el nombre del bucket, pero el URI de la solicitud HTTP no lo incluye. (El encabezado del host especifica el bucket).

Mostrar todos mis buckets

Solicitud	StringToSign
<pre>GET / HTTP/1.1</pre>	<pre>GET\n</pre>

Solicitud	StringToSign
Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:29:59 +0000 <i>Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdE RIC03wnaRNKh60qZehG9s=</i>	\n \n Wed, 28 Mar 2007 01:29:59 +0000\n /

Claves de Unicode

Solicitud	StringToSign
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 <i>Authorization: AWS AKIAIOSFODNN7EXAMP LE:DNEZGsoieTZ92F3bUfSPQcbGmLM=</i>	GET\n \n \n Wed, 28 Mar 2007 01:49:49 +0000\n /dictionary/fran%C3%A7ais/pr %c3%a9f%c3%a8re

Note

Los elementos en el StringToSign que se derivaron del URI de la solicitud se toman literalmente, incluida la codificación y capitalización de la URL.

Problemas de firma con solicitudes REST

Cuando falla la autenticación de REST, el sistema responde a la solicitud con un documento de errores XML. La información que contiene este documento de errores tiene como objetivo ayudar a los desarrolladores a diagnosticar el problema. En particular, el elemento StringToSign del documento de errores SignatureDoesNotMatch le dice exactamente qué canonicalización de solicitudes está empleando el sistema.

Algunas herramientas insertan encabezados en modo silencioso cuya existencia usted no conocía, como la agregación del encabezado Content-Type durante un PUT. En la mayoría de estos casos, el valor del encabezado insertado permanece constante, con lo que podrá descubrir los encabezados faltantes con herramientas como *Ethereal* o *tcpmon*.

Alternativa de autenticación por cadena de consulta de solicitudes

Puede autenticar determinado tipo de solicitudes pasando la información requerida como parámetros de una cadena de consulta, en lugar de usar el encabezado HTTP `Authorization`. Esto resulta útil para habilitar el acceso directo por navegador de terceros a sus datos de Amazon S3 privados sin hacer pasar la solicitud por un proxy. La idea es construir una solicitud "prefirmada" y codificarla como una URL que pueda recuperar el navegador de un usuario final. Además, puede limitar una solicitud prefirmada especificando una fecha de vencimiento.

Para obtener más información acerca del uso de parámetros de consulta para autenticar solicitudes, consulte [Autenticación de solicitudes: Uso de los parámetros de consulta\(AWS Signature Version 4\)](#) en la Referencia de la API de Amazon Simple Storage Service. Con el fin de ver ejemplos sobre cómo usar los AWS SDK para generar URL prefirmadas, consulte [Compartir objetos con URL prefirmadas](#).

Crear una firma

A continuación figura un ejemplo de una solicitud REST de Amazon S3 autenticada por cadena de consulta.

```
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
```

El método de autenticación de solicitudes con cadena de consulta no requiere ningún encabezado HTTP especial. Los elementos de autenticación necesarios se especifican como parámetros de la cadena de consulta:

Nombre del parámetro de la cadena de consulta	Ejemplo de valor	Descripción
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE	El ID de la clave de acceso de AWS. Especifica la clave de acceso

Nombre del parámetro de la cadena de consulta	Ejemplo de valor	Descripción
		secreta de AWS utilizada para firmar la solicitud e, indirectamente, la identidad del desarrollador que realiza la solicitud.
Expires	1141889120	El momento de vencimiento de la firma, especificado como el número de segundos a partir de la fecha de inicio (00:00:00 UTC del 1 de enero de 1970). Cualquier solicitud recibida con posterioridad a este momento (según el servidor) será rechazada.
Signature	vjbyPxybdZaNmGa%2B yT272YEAiv4%3D	La codificación URL de la codificación Base64 del HMAC SHA1 de StringToSign.

El método de autenticación de solicitudes con cadena de consulta difiere levemente del método ordinario, pero solo en el formato del parámetro de la solicitud `Signature` y el elemento `StringToSign`. A continuación presentamos pseudogramática que ilustra el método de autenticación de solicitudes por cadena de consulta.

```
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 es el ID de clave de acceso secreta de AWS que le asigna Amazon cuando se registra para ser un desarrollador de Amazon Web Services. Tenga en cuenta que la Signature está codificada en formato de URL para que se pueda colocar en la cadena de consulta. Tenga en cuenta también que, en StringToSign, el elemento posicional HTTP Date ha sido sustituido por Expires. El CanonicalizedAmzHeaders y el CanonicalizedResource son iguales.

Note

En el método de autenticación por cadena de consulta, no utilice el encabezado Date ni x-amz-date request al calcular la cadena para firmar.

Autenticación por cadena de consulta de solicitudes

Solicitud	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>

Suponemos que, cuando un navegador realiza la solicitud GET, no facilitará un encabezado Content-MD5 ni Content-Type, ni tampoco creará encabezados x-amz-, por lo que esas partes de StringToSign se dejan vacías.

Usar codificación Base64

Las firmas de solicitudes HMAC deben tener codificación Base64. La codificación Base64 convierte la firma en una cadena ASCII sencilla que se puede adjuntar a la solicitud. Los caracteres que podrían aparecer en la cadena de firma, como más (+), barra inclinada (/) e igual (=) deben estar codificados si se usan en un URI. Por ejemplo, si el código de autenticación incluye un signo más (+), codifíquelo como %2B en la solicitud. Las barras inclinadas se codifican como %2F, y los signos de igual, como %3D.

Para ver más ejemplos de codificación en Base64, consulte los de Amazon 3 [Ejemplos de autenticación](#).

Cargas basadas en el navegador con POST (AWS Signature Version 2)

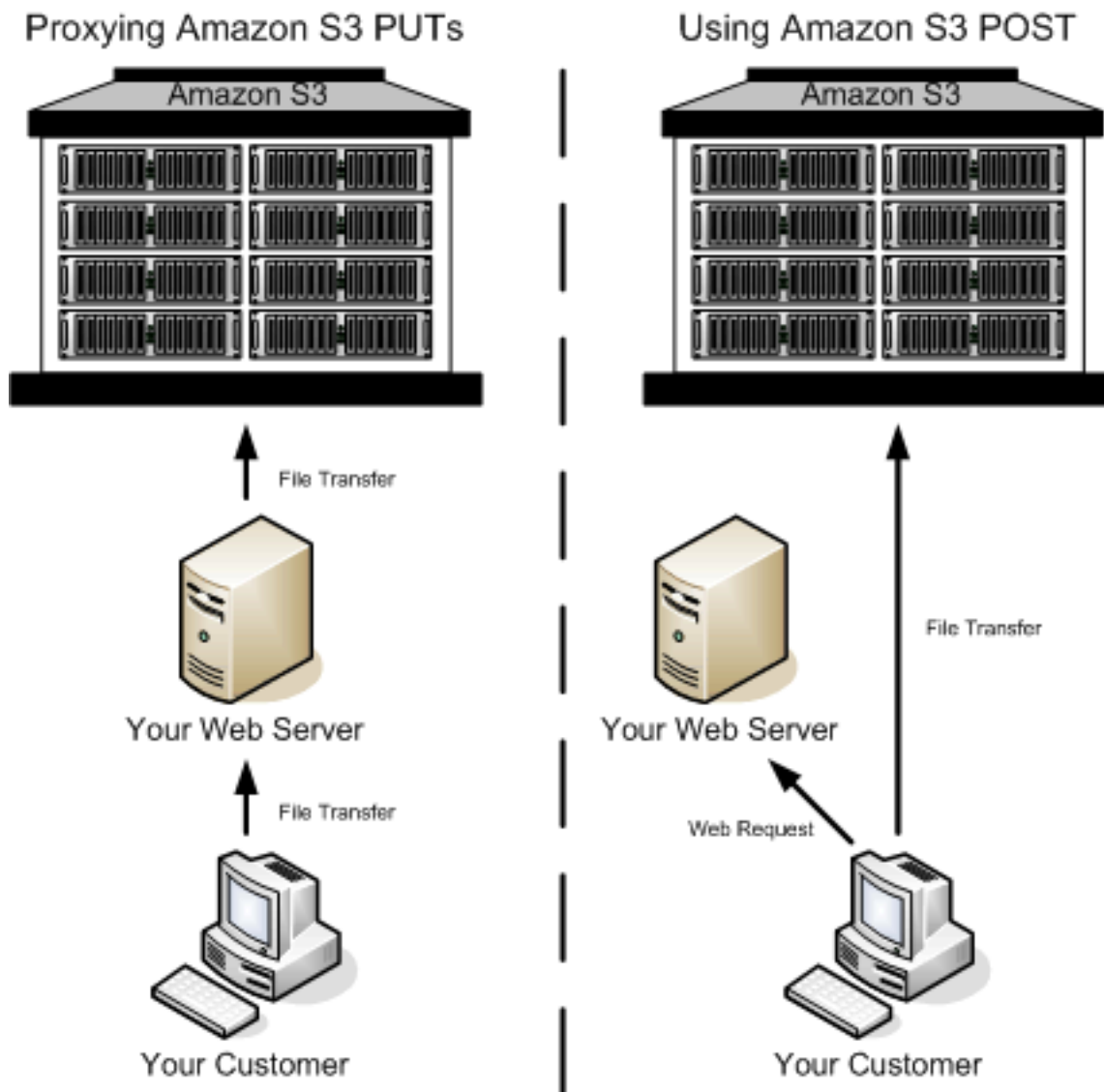
Amazon S3 admite POST, que le permite a los usuarios cargar contenido directamente en Amazon S3. POST está diseñado para simplificar las cargas, reducir la latencia de las cargas y ahorrar dinero en aplicaciones que los usuarios utilizan para cargar datos y guardarlos en Amazon S3.

Note

La autenticación de solicitudes que se analiza en esta sección se basa en AWS Signature Version 2, un protocolo para autenticar las solicitudes de la API entrantes para los servicios de AWS.

Amazon S3 ahora admite Signature Version 4, un protocolo para autenticar las solicitudes de la API entrantes para los servicios de AWS, en todas las Regiones de AWS. En este momento, las Regiones de AWS creadas antes del 30 de enero de 2014 seguirán admitiendo el protocolo anterior, Signature Version 2. Cualquier región nueva después del 30 de enero de 2014 solo admitirá Signature Version 4, y por lo tanto, todas las solicitudes de esas regiones se deben realizar con Signature Version 4. Para obtener más información, consulte [Autenticación de solicitudes de las cargas basadas en el navegador con POST \(AWS Signature Version 4\)](#) en la Referencia de la API de Amazon Simple Storage Service.

En la siguiente figura se muestra una carga con POST de Amazon S3.



Cargar con POST

- 1 El usuario abre un navegador web y obtiene acceso a su página web.
- 2 Su página web incluye un formulario HTTP que contiene toda la información necesaria para que el usuario pueda cargar contenido en Amazon S3.
- 3 El usuario carga contenido directamente en Amazon S3.

Note

POST no admite la autenticación por query string.

Formularios HTML (AWS Signature Version 2)

Temas

- [Codificar formulario HTML](#)
- [Declaración de formulario HTML](#)
- [Campos de formulario HTML](#)
- [Construcción de la política](#)
- [Crear una firma](#)
- [Redireccionamiento](#)

Cuando se comunica con Amazon S3, por lo general utiliza la API REST o SOAP para realizar operaciones como PUT, GET, DELETE, y otras. Con POST, los usuarios cargan datos directamente en Amazon S3 a través de sus navegadores, que no pueden procesar la API de SOAP ni crear una solicitud PUT de REST.

Note

Las API de SOAP para Amazon S3 no están disponibles para nuevos clientes y se acercan al fin de la vida útil (EOL) el 31 de agosto de 2025. Le recomendamos que utilice la API REST o los SDK de AWS.

Para que los usuarios puedan cargar contenido en Amazon S3 con sus navegadores, debe utilizar los formularios HTML. Los formularios HTML constan de una declaración de formulario y campos de formulario. Cada declaración de formulario incluye información de alto nivel acerca de la solicitud. Los campos de formulario incluyen información detallada acerca de la solicitud, así como la política que se utiliza para autenticarla y asegurar que cumpla con las condiciones que usted especifica.

Note

Los datos y límites del formulario (sin incluir los contenidos del archivo) no pueden exceder los 20 KB.

En esta sección se explica cómo utilizar los formularios HTML.

Codificar formulario HTML

El formulario y la política deben estar cifrados con UTF-8. Para aplicar la codificación UTF-8 en el formulario puede especificarlo en el encabezado HTML o como un encabezado de solicitud.

Note

La declaración de formulario HTML no acepta parámetros de autenticación por query string.

A continuación, se muestra un ejemplo de la codificación UTF-8 en el encabezado HTML:

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

A continuación, se muestra un ejemplo de la codificación UTF-8 en un encabezado de solicitud:

```
Content-Type: text/html; charset=UTF-8
```

Declaración de formulario HTML

La declaración de formulario tiene tres componentes: la acción, el método y el tipo de documento adjunto. Si cualquiera de estos valores se configura de manera inadecuada, la solicitud falla.

La acción especifica el URL que procesa la solicitud, que debe establecerse en el URL del bucket. Por ejemplo, si el nombre de su bucket es `awsexamplebucket1` y la región es EE. UU. Oeste (Norte de California), la URL es `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/`.

Note

El nombre de clave se especifica en un campo de formulario.

El método debe ser POST.

Se debe especificar el tipo de documento adjunto (enctype) y se debe establecer en datos de formulario/multiparte para cargas de archivos y cargas de área de texto. Para obtener más información, visite [RFC 1867](#).

Example

El siguiente ejemplo es una declaración de formulario para el bucket "awsexamplebucket1".

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

Campos de formulario HTML

En la siguiente tabla se describen los campos que se pueden utilizar en un formulario HTML.

 Note

La variable `${filename}` se reemplaza automáticamente con el nombre del archivo provisto por el usuario y es reconocida por todos los campos de formulario. Si el navegador o el cliente proporciona una ruta total o parcial al archivo, solo se utilizará el texto después de la última barra inclinada (/) o barra inversa (\). Por ejemplo, "C:\Program Files\directory1\file.txt" se interpretará como "file.txt". Si no se brinda ningún archivo o nombre de archivo, la variable se reemplaza con una cadena vacía.

Nombre del campo	Descripción	Obligatorio
AWSAccessKeyId	El ID de clave de acceso de AWS del propietario del bucket que otorga un acceso de usuario anónimo para una solicitud que cumple con el conjunto de restricciones en la política. Este campo es obligatorio si la solicitud incluye un documento de política.	Condicional
acl		No

Nombre del campo	Descripción	Obligatorio
	Una lista de control de acceso (ACL) de Amazon S3. Si se especifica una lista de control de acceso no válida, se genera un error. Tipo: cadena Valor predeterminado: privado Valores válidos: <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>	
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	Encabezados específicos de REST. Para obtener más información, consulte PUT Object .	No
key	El nombre de la clave cargada. Para utilizar el nombre de archivo provisto por el usuario, utilice la variable <code>\${filename}</code>. Por ejemplo si la usuaria Betty carga el archivo <code>lolcatz.jpg</code> y usted especifica <code>/user/betty/\${filename}</code>, el archivo se guarda como <code>/user/betty/lolcatz.jpg</code>. Para obtener más información, consulte Trabajar con metadatos de objeto.	Sí

Nombre del campo	Descripción	Obligatorio
policy	Política de seguridad que describe qué está permitido en la solicitud. Las solicitudes sin una política de seguridad se consideran anónimas y solo se aceptarán en buckets que se pueden escribir públicamente.	No
success_action_redirect, redirect	URL al que el cliente es redirigido después de la carga exitosa. Amazon S3 adjunta al URL el bucket, la clave y los valores de la etag como parámetros de cadena de consulta. Si no se especifica el campo success_action_redirect, Amazon S3 devuelve el tipo de documento vacío especificado en el campo success_action_status. Si Amazon S3 no puede interpretar la URL, ignora el campo . Si la carga falla, Amazon S3 muestra un error y no redirige al usuario a una URL. Para obtener más información, consulte Redireccionamiento.  Note El nombre de campo de redirect está obsoleto y el soporte para el nombre de campo de redirect se eliminará en el futuro.	No

Nombre del campo	Descripción	Obligatorio
success_action_status	El código de estado devuelto al cliente después de la carga exitosa si no se especifica a success_action_redirect. Los valores válidos son 200, 201 o 204 (predeterminado). Si el valor se establece en 200 o 204, Amazon S3 devuelve un documento vacío con un código de estado 200 o 204. Si el valor se establece en 201, Amazon S3 devuelve un documento XML con un código de estado 201. Para obtener información acerca del contenido del documento XML, consulte POST Object. Si el valor no se establece o si se establece en un valor no válido, Amazon S3 devuelve un documento vacío con un código de estado 204.	No
 Note Algunas versiones de Adobe Flash Player no controlan debidamente las respuestas HTTP con un cuerpo vacío. Para admitir cargas con Adobe Flash, recomendamos establecer success_action_status en 201.		

Nombre del campo	Descripción	Obligatorio
signature	La firma HMAC que se crea con la clave de acceso secreta que corresponde al <code>AWSSecretAccessKey</code> provisto. Este campo es obligatorio si se incluye un documento de política en la solicitud. Para obtener más información, consulte Administración de identidad y acceso para Amazon S3.	Condicional
x-amz-security-token	Un token de seguridad utilizado por las credenciales de sesión. Si la solicitud utiliza Amazon DevPay, se requieren dos campos de formulario <code>x-amz-security-token</code> : uno para el token de producto y otro para el token de usuario. Si la solicitud utiliza credenciales de sesión, se requiere un formulario <code>x-amz-security-token</code> . Para obtener más información, consulte Credenciales de seguridad temporales en la guía del usuario de IAM.	No
Otros nombres de archivo con prefijos x-amz-meta-	Metadatos especificados por el usuario. Amazon S3 no valida ni utiliza estos datos. Para obtener más información, consulte PUT Object.	No

Nombre del campo	Descripción	Obligatorio
file	Contenido de texto o archivo. El archivo o el contenido debe ser el último campo en el formulario. Cualquier campo debajo de estos se ignora. No puede cargar más de un archivo a la vez.	Sí

Construcción de la política

Temas

- [Expiration](#)
- [Condiciones](#)
- [Coincidencia de condiciones](#)
- [Secuencia de escape de caracteres](#)

La política es un documento JSON con codificación UTF-8 y Base64 que especifica las condiciones que debe cumplir la solicitud, y se utiliza para autenticar el contenido. Según cómo diseñe sus documentos de política, puede utilizarlos por carga, por usuario, para todas las cargas o de acuerdo con otros diseños que se ajusten a sus necesidades.

Note

Si bien el documento de política es opcional, lo recomendamos ampliamente en lugar de hacer un bucket que se pueda escribir públicamente.

A continuación se muestra el ejemplo de un documento de política:

```
{ "expiration": "2007-12-01T12:00:00.000Z",  
  
  "conditions": [  
  
    {"acl": "public-read" },
```

```
{ "bucket": "awsexamplebucket1" },  
  
  [ "starts-with", "$key", "user/eric/" ],  
  
]  
  
}
```

El documento de política incluye los vencimientos y las condiciones.

Expiration

El elemento de vencimiento especifica la fecha de vencimiento de la política en el formato de fecha según la norma ISO 8601 en Universal Time Coordinated (UTC, Hora universal coordinada). Por ejemplo, “2007-12-01T12:00:00.000Z” especifica que la política no es válida después de la medianoche, UTC, del 01/12/2007. El vencimiento es obligatorio en una política.

Condiciones

Las condiciones en el documento de política validan los contenidos del objeto cargado. Cada campo de formulario que especifica en el formulario (salvo `AWSAccessKeyId`, firma, archivo, política y nombres de archivos que tienen un prefijo `x-ignore-`) se debe incluir en la lista de condiciones.

Note

Si tiene varios campos con el mismo nombre, los valores deben estar separados por comas. Por ejemplo, si tiene dos campos denominados “x-amz-meta-tag” y el primero tiene el valor “Ninja” y el segundo tiene el valor “Stallman”, configurará el documento de política como `Ninja,Stallman`.

Todas las variables en el formulario se expanden antes de que la política se valide. Por lo tanto, se deben realizar todas las coincidencias de condiciones con respecto a los campos expandidos. Por ejemplo, si configuró el campo de clave en `user/betty/${filename}`, su política puede ser `[ "starts-with", "$key", "user/betty/" ]`. No escriba `[ "starts-with", "$key", "user/betty/${filename}" ]`. Para obtener más información, consulte [Coincidencia de condiciones](#).

En la siguiente tabla se describen las condiciones de los documentos de política.

Nombre del elemento	Descripción
acl	Especifica las condiciones que debe cumplir la ACL. Admite las coincidencias exactas y <code>starts-with</code> .
content-length-range	Especifica el tamaño mínimo y máximo permitido para el contenido cargado. Admite la coincidencia de rango.
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	Encabezados específicos de REST. Admite las coincidencias exactas y <code>starts-with</code> .
clave	El nombre de la clave cargada. Admite las coincidencias exactas y <code>starts-with</code> .
success_action_redirect, redirect	URL al que el cliente es redirigido después de la carga exitosa. Admite las coincidencias exactas y <code>starts-with</code> .
success_action_status	El código de estado devuelto al cliente después de la carga exitosa si no se especifica <code>success_action_redirect</code>. Admite las coincidencias exactas.
x-amz-security-token	Token de seguridad de Amazon DevPay. Cada solicitud que utiliza Amazon DevPay requiere dos campos de formulario <code>x-amz-security-token</code> : uno para el token de producto y otro para el token de usuario. Como consecuencia, los valores deben estar separados por comas. Por ejemplo, si el token de usuario es <code>ew91dHViZ</code>

Nombre del elemento	Descripción
	Q== y el token del producto es b0hnNVNKWVJIQTA= , establece la entrada de política en { "x-amz-security-token": "eW91dHVIZQ==,b0hnNVNKWVJIQT A=" } .
Otros nombres de archivo con prefijos x-amz-meta-	Metadatos especificados por el usuario. Admite las coincidencias exactas y starts-with .

 **Note**

Si su conjunto de herramientas añade campos adicionales (p. ej., Flash añade el nombre de archivo), debe añadirlos al documento de política. Si puede controlar esta funcionalidad, añada el prefijo x-ignore- al campo para que Amazon S3 omita la característica y no afecte futuras versiones de esta característica.

Coincidencia de condiciones

En la siguiente tabla se describen los tipos de coincidencias de condiciones. Si bien debe especificar una condición para cada campo de formulario que especifica en el formulario, puede crear criterios de coincidencia más complejos especificando varias condiciones para un campo de formulario.

Condición	Descripción
Coincidencias exactas	Las coincidencias exactas verifican que los campos coincidan con valores específicos. En este ejemplo se indica que la ACL se debe establecer en public-read: <pre>{"acl": "public-read" }</pre> Este ejemplo es una alternativa para indicar que la ACL se debe establecer en public-read:

Condición	Descripción
	<pre>["eq", "\$acl", "public-read"]</pre>
Empieza por	Si el valor debe empezar con un valor determinado, utilice starts-with. En este ejemplo se indica que la clave debe empezar con user/betty: <pre>["starts-with", "\$key", "user/betty/"]</pre>
Coincidencia con cualquier contenido	Para configurar la política para permitir cualquier contenido dentro de un campo, utilice starts-with con un valor vacío. Este ejemplo permite cualquier campo success_action_redirect: <pre>["starts-with", "\$success_action_redirect", ""]</pre>
Especificación de rangos	Para campos que aceptan rangos, separe los rangos superiores e inferiores con una coma. Este ejemplo permite un tamaño de archivo de 1 a 10 megabytes: <pre>["content-length-range", 1048579, 10485760]</pre>

Secuencia de escape de caracteres

En la siguiente tabla se describen los caracteres a los que se debe aplicar una secuencia de escape en un documento de política.

Secuencia de escape	Descripción
\\	Barra inversa
\\\$	Signo de dólar

Secuencia de escape	Descripción
<code>\b</code>	Retroceso
<code>\f</code>	Salto de página
<code>\n</code>	Nueva línea
<code>\r</code>	Salto de línea
<code>\t</code>	Tabulador horizontal
<code>\v</code>	Tabulador vertical
<code>\uxxxx</code>	Todos los caracteres Unicode

Crear una firma

Paso	Descripción
1	Codifique la política con UTF-8.
2	Codifique los bytes de UTF-8 con Base64.
3	Firme la política con su clave de acceso secreta mediante el uso del algoritmo HMAC SHA-1.
4	Codifique la firma de SHA-1 con Base64.

Para obtener información general sobre autenticación, consulte [Administración de identidad y acceso para Amazon S3](#).

Redireccionamiento

En esta sección se describe cómo administrar los direccionamientos.

Redireccionamiento general

Al finalizar la solicitud POST, el usuario es redirigido a la ubicación que especificó en el campo `success_action_redirect`. Si Amazon S3 no puede interpretar la URL, ignora el campo `success_action_redirect`.

Si no se especifica el campo `success_action_redirect`, Amazon S3 devuelve el tipo de documento vacío especificado en el campo `success_action_status`.

Si la solicitud POST falla, Amazon S3 muestra un error y no proporciona un redireccionamiento.

Redireccionamiento de carga previa

Si su bucket se creó con `<CreateBucketConfiguration>`, sus usuarios finales pueden necesitar una redirección. Si esto sucede, algunos navegadores pueden administrar la redirección de manera incorrecta. Esto es relativamente poco frecuente, pero es muy probable que suceda inmediatamente después de que se crea un bucket.

Ejemplos de carga (AWS Signature Version 2)

Temas

- [Cargar archivo](#)
- [Cargar área de texto](#)

Note

La autenticación de solicitudes que se analiza en esta sección se basa en AWS Signature Version 2, un protocolo para autenticar las solicitudes de la API entrantes para los servicios de AWS.

Amazon S3 ahora admite Signature Version 4, un protocolo para autenticar las solicitudes de la API entrantes para los servicios de AWS, en todas las Regiones de AWS. En este momento, las Regiones de AWS creadas antes del 30 de enero de 2014 seguirán admitiendo

el protocolo anterior, Signature Version 2. Cualquier región nueva después del 30 de enero de 2014 solo admitirá Signature Version 4, y por lo tanto, todas las solicitudes de esas regiones se deben realizar con Signature Version 4. Para obtener más información, consulte [Ejemplos de cargas basadas en el navegador con HTTP POST \(mediante AWS Signature Version 4\)](#) en la Referencia de la API de Amazon Simple Storage Service.

Cargar archivo

En este ejemplo se muestra el proceso completo para crear una política y un formulario que se puede utilizar para cargar un archivo adjunto.

Crear política y formulario

La siguiente política admite cargas en Amazon S3 para el bucket `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", ""]
  ]
}
```

Esta política requiere lo siguiente:

- La carga se debe realizar el 1 de diciembre de 2007 antes de las 12:00 UTC.
- El contenido se debe cargar en el bucket `awsexamplebucket1`.
- La clave debe empezar con `“user/eric/”`.
- La ACL se establece en `public-read`.
- El campo `success_action_redirect` se establece en `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html`.
- El objeto es un archivo de imagen.

- La etiqueta x-amz-meta-uuid se debe establecer en 14365123651274.
- x-amz-meta-tag puede incluir cualquier valor.

A continuación, se incluye una versión de esta política codificada en Base64.

```
eyJhZiZxhwaXJhdGlvbiI6IClIyMDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
```

Con su credenciales, cree una firma, por ejemplo 0RavWzkygo6QX9caELEqKi9kDbU= es la firma para el documento de política anterior.

El siguiente formulario admite una solicitud POST al bucket amzn-s3-demo-bucket que utiliza esta política.

```
<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 />
      <input type="hidden" name="AWSAccessKeyId" value="AKIAIOSFODNN7EXAMPLE" />
      <input type="hidden" name="Policy" value="POLICY" />
      <input type="hidden" name="Signature" value="SIGNATURE" />
      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>
```

Solicitar ejemplo

Esta solicitud asume que la imagen cargada tiene un tamaño de 117 108 bytes; los datos de la imagen no se incluyen.

```
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"

eyJAiZXhwaXJhdGlvbI6ICiYMDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
--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--
```

Respuesta de ejemplo

```
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
```

Cargar área de texto

Temas

- Crear política y formulario
- Solicitar ejemplo
- Respuesta de ejemplo

En el siguiente ejemplo se muestra el proceso completo para crear una política y un formulario para cargar un área de texto. Cargar un área de texto es útil para presentar contenido creado por el usuario, como publicaciones de blog.

Crear política y formulario

La siguiente política admite cargas de área de texto en Amazon S3 para el bucket `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", ""]
  ]
}
```

Esta política requiere lo siguiente:

- La carga se debe realizar antes de las 12:00 GMT del 01/12/2007.
- El contenido se debe cargar en el bucket `awsexamplebucket1`.
- La clave debe empezar con `“user/eric/”`.
- La ACL se establece en `public-read`.
- El campo `success_action_redirect` se establece en `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html`.
- El objeto es un texto HTML.
- La etiqueta `x-amz-meta-uuid` se debe establecer en `14365123651274`.
- `x-amz-meta-tag` puede incluir cualquier valor.

A continuación, se incluye una versión de esta política codificada en Base64.

```
eyJhZlZxhwaXJhdGlvbiI6IClYMDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXR
```

```
pb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXR0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCaidXNlci
LAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2p
C5zMy5hbWV6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSI6ICJ0ZXh0L2h0
CAgIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LWFtei1t
IsICIiXQogIF0KfQo=
```

Con sus credenciales, cree una firma. Por ejemplo, qA7FWXKq6VvU681I9KdveT1cWgF= es la firma para el documento de política anterior.

El siguiente formulario admite una solicitud POST al bucket amzn-s3-demo-bucket que utiliza esta política.

```
<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" />
      <input type="hidden" name="Policy" value="POLICY" />
      <input type="hidden" name="Signature" value="SIGNATURE" />
      Entry: <textarea name="file" cols="60" rows="10">
```

Your blog post goes here.

```
    </textarea><br />
    <!-- The elements after this will be ignored -->
    <input type="submit" name="submit" value="Upload to Amazon S3" />
  </form>
  ...
</html>
```

Solicitar ejemplo

Esta solicitud asume que la imagen cargada tiene un tamaño de 117 108 bytes; los datos de la imagen no se incluyen.

```
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"
eyJhZlZxhwaXJhdGlvbiI6IClYMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidW
--178521717625888
Content-Disposition: form-data; name="Signature"

qA7FWXKq6VvU68lI9KdveT1cWgF=
--178521717625888
Content-Disposition: form-data; name="file"

...content goes here...
--178521717625888
Content-Disposition: form-data; name="submit"

Upload to Amazon S3
--178521717625888--

```

Respuesta de ejemplo

```

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 con Adobe Flash (AWS Signature Version 2)

En esta sección se describe cómo utilizar POST con Adobe Flash.

Seguridad de Adobe Flash Player

De forma predeterminada, el modelo de seguridad de Adobe Flash Player les prohíbe a los usuarios de Adobe Flash Player conectarse a servidores fuera del dominio al que sirve el archivo SWF.

Para sobrescribir el valor predeterminado, debe cargar un archivo `crossdomain.xml` que se pueda leer públicamente al bucket que aceptará cargas POST. A continuación se muestra un archivo `crossdomain.xml` de ejemplo.

```
<?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

Para obtener más información acerca del modelo de seguridad de Adobe Flash, visite el sitio web de Adobe.

La incorporación del archivo `crossdomain.xml` a su bucket permite que cualquier usuario de Adobe Flash Player se conecte al archivo `crossdomain.xml` en su bucket; sin embargo, esto no brinda acceso al bucket de Amazon S3 real.

Consideraciones sobre Adobe Flash

La API `FileReference` en Adobe Flash añade el campo de formulario `Filename` a la solicitud POST. Al crear aplicaciones de Adobe Flash que se cargan en Amazon S3 con la acción de la API `FileReference`, incluya la siguiente condición en su política:

```
['starts-with', '$Filename', '']
```

Algunas versiones de Adobe Flash Player no controlan debidamente las respuestas HTTP que tienen un cuerpo vacío. Para configurar POST de manera que devuelva una respuesta que no tenga un cuerpo vacío, establezca el campo `success_action_status` en 201. Amazon S3 devolverá un documento XML con un código de estado 201. Para obtener información acerca del contenido del documento XML, consulte [POST Object](#). Para obtener información acerca de los campos del formulario, consulte [Campos de formulario HTML](#).

Apéndice: API de configuración del ciclo de vida (obsoleto)

La configuración del ciclo de vida del bucket se ha actualizado para admitir filtros basados en etiquetas de objetos. Es decir, ahora puede especificar una regla que especifique el prefijo de

nombre de clave, una o varias etiquetas de objeto o ambos factores para seleccionar un subconjunto de objetos al que aplicar la regla. Las API se han actualizado en consecuencia. En los siguientes temas, se describe la versión anterior de las operaciones del ciclo de vida de los buckets PUT y GET para garantizar la compatibilidad con versiones anteriores.

Temas

- [Ciclo de vida de bucket PUT \(obsoleto\)](#)
- [Ciclo de vida de bucket GET \(obsoleto\)](#)

Ciclo de vida de bucket PUT (obsoleto)

Descripción

Important

Para obtener una versión actualizada de esta API, consulte [PutBucketLifecycleConfiguration](#). Esta versión ya no está disponible. Las configuraciones del ciclo de vida existentes funcionarán. Para las nuevas configuraciones del ciclo de vida, use la API actualizada.

Crea una nueva una configuración de ciclo de vida para el bucket o sustituye una configuración de ciclo de vida existente. Para obtener información sobre la configuración del ciclo de vida, consulte [Administración del ciclo de vida de los objetos](#) en la Guía del usuario de Amazon Simple Storage Service.

Permisos

De forma predeterminada, todos los recursos de Amazon S3, incluidos buckets, objetos y subrecursos relacionados (por ejemplo, configuración del ciclo de vida y del sitio web) son privados. Solo el propietario del recurso, la Cuenta de AWS que creó el recurso, puede acceder a este. El propietario del recurso puede conceder permisos de acceso opcionalmente a terceros si escribe una política de acceso. Para esta operación, los usuarios deben obtener el permiso `s3:PutLifecycleConfiguration`.

También puede denegar explícitamente los permisos. La denegación explícita también reemplaza cualquier otro permiso. Si desea impedir que los usuarios o las cuentas eliminen objetos del bucket, debe negarles los permisos para realizar las siguientes acciones:

- `s3:DeleteObject`
- `s3:DeleteObjectVersion`
- `s3:PutLifecycleConfiguration`

Para obtener más información sobre los permisos, consulte [Administración de los permisos de acceso a los recursos de Amazon S3](#) en la Guía del usuario de Amazon Simple Storage Service.

Solicitudes

Sintaxis

```
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

Para obtener información sobre las cadenas de autorización, consulte [Autenticación de solicitudes \(AWS Signature Version 4\)](#).

Parámetros de la solicitud

Esta aplicación de la operación no utiliza parámetros de solicitud.

Encabezados de solicitud

Nombre	Descripción	Obligatorio
Content-MD5	Resumen MD5 de 128 bits de los datos cifrados en base64. Debe utilizar este encabezado como comprobación de la integridad del mensaje para comprobar que el cuerpo de la solicitud no se dañó durante el tránsito. Para obtener más información, consulte RFC 1864. Tipo: cadena Valor predeterminado: None	Sí

Cuerpo de la solicitud

En la solicitud, debe especificar la configuración del ciclo de vida en el cuerpo de la solicitud. La configuración del ciclo de vida se especifica como XML. A continuación, se muestra un ejemplo

de configuración de un ciclo de vida básico. Especifica una regla. `Prefix` en la regla identifica los objetos a los que se aplica la regla. La regla también especifica dos acciones (`Transition` y `Expiration`). Cada acción especifica un cronograma en el que Amazon S3 debe realizar la acción. `Status` indica si la regla está habilitada o desactivada.

```
<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 el estado del bucket es control de versiones habilitado o control de versiones suspendido, puede tener muchas versiones del mismo objeto: una versión actual y ninguna o varias versiones no actuales. La configuración siguiente del ciclo de vida especifica las acciones (`NoncurrentVersionTransition`, `NoncurrentVersionExpiration`) que son específicas de las versiones de objetos no actuales.

```
<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>
```

Puede usar la interfaz de programa de aplicación (API, Application Program Interface) de carga multiparte para cargar objetos grandes en partes. Para obtener más información acerca de las cargas multiparte, consulte [Información general de carga multiparte](#) en la Guía del usuario de Amazon Simple Storage Service. Con la configuración del ciclo de vida, puede indicar a Amazon S3 que cancele las cargas multiparte incompletas, identificadas por el prefijo de nombre de clave especificado en la regla, si no se completan en una cantidad de días especificada. Cuando Amazon S3 cancela una carga multiparte, elimina todas las partes asociadas con la carga. Esto garantiza que no tenga cargas multiparte incompletas que tengan partes incompletas almacenadas en Amazon S3 y, por lo tanto, que no tenga que pagar costos de almacenamiento por ellas. El siguiente es un ejemplo de configuración de ciclo de vida que especifica una regla con la acción `AbortIncompleteMultipartUpload`. Esta acción indica a Amazon S3 que cancele las cargas multiparte incompletas siete días después de iniciarse.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>SomeKeyPrefix</Prefix>
    <Status>rule-status</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

En las siguientes tablas, se describen los elementos XML de la configuración de ciclo de vida.

Nombre	Descripción	Obligatorio
<code>AbortIncompleteMultipartUpload</code>	Contenedor para especificar cuándo una carga multiparte incompleta pasa a ser apta para una operación de anulación. Elemento secundario: <code>DaysAfterInitiation</code> Tipo: Contenedor Antecesor: <code>Rule</code>	Sí, si no se especifica a ninguna otra acción para la regla

Nombre	Descripción	Obligatorio
Date	La fecha en la que desea que Amazon S3 realice la acción. Para obtener más información, consulte Reglas de ciclo de vida: en función de una fecha específica en la Guía del usuario de Amazon Simple Storage Service. El valor de la fecha debe estar en formato ISO 8601. La hora siempre es medianoche UTC. Tipo: cadena Antecesor: <code>Expiration</code> o <code>Transition</code>	Sí, si <code>Days</code> y <code>ExpiredObjectDeleteMarker</code> están ausentes
Days	Especifica el número de días después de la creación del objeto cuando entra en vigor la acción de regla específica. Tipo: entero no negativo cuando se usa con <code>Transition</code> , entero positivo cuando se usa con <code>Expiration</code> Antecesor: <code>Expiration</code> , <code>Transition</code>	Sí, si <code>Date</code> y <code>ExpiredObjectDeleteMarker</code> están ausentes

Nombre	Descripción	Obligatorio
<code>DaysAfterInitiation</code>	Especifica el número de días después de iniciar una carga multiparte en los que debe completarse la carga multiparte. Si no se completa en el número de días especificado, se podrá aplicar una operación de anulación y Amazon S3 cancela la carga multiparte incompleta. Tipo: entero positivo Antecesor: <code>AbortIncompleteMultipartUpload</code>	Sí, si se especifica a una etiqueta principal

Nombre	Descripción	Obligatorio
Expiration	Esta acción especifica un periodo de la vida útil de un objeto en el que Amazon S3 debe llevar a cabo la acción de caducidad adecuada. La acción que tome Amazon S3 depende de si el bucket tiene habilitado el control de versiones. • Si el control de versiones nunca se ha habilitado en el bucket, Amazon S3 elimina la única copia del objeto de forma permanente. • Si el bucket tiene el control de versiones habilitado (o el control de versiones está suspendido), la acción se aplica solo a la versión actual del objeto. Un bucket con el control de versiones habilitado puede tener muchas versiones del mismo objeto: una versión actual y ninguna o varias versiones no actuales. En lugar de eliminar la versión actual, Amazon S3 la convierte en una versión no actual agregándole un marcador de eliminación como la nueva versión actual.	Sí, si no hay ninguna otra acción presente en la Rule.

 **Important**

Si el estado de un bucket es con el control de versiones suspendido, Amazon S3 crea un marcador de eliminación con el ID de versión null. Si tiene una versión con un

Nombre	Descripción	Obligatorio
	ID de versión null, Amazon S3 sobrescribe esa versión. Note Para establecer la caducidad de los objetos no actuales, utilice la acción NoncurrentVersionExpiration . Tipo: Contenedor Secundario: Días o Fecha Antecesor: Rule	
ID	Identificador único de la regla. El valor no puede tener más de 255 caracteres. Tipo: cadena Antecesor: Rule	No

Nombre	Descripción	Obligatorio
LifecycleConfiguration	Contenedor para las reglas del ciclo de vida. Puede añadir hasta 1000 reglas. Tipo: Contenedor Secundario: Regla Ancesor: ninguno	Sí
ExpiredObjectDeleteMarker	En un bucket con control de versiones (un bucket con el control de versiones habilitado o suspendido), puede agregar este elemento en la configuración del ciclo de vida para indicar a Amazon S3 que elimine los marcadores de eliminación de objetos caducados. Para ver un ejemplo, consulte Ejemplo 8: Eliminación de marcadores de eliminación de objetos caducados en la Guía del usuario de Amazon Simple Storage Service. No lo agregue a un bucket sin control de versiones, ya que ese tipo de bucket no puede incluir marcadores de eliminación. Tipo: cadena Valores válidos: true false (el valor false está permitido, pero no es operativo, lo que significa que Amazon S3 no realizará ninguna acción) Antecesor: Expiration	Sí, si Date y Days están ausentes

Nombre	Descripción	Obligatorio
NoncurrentDays	Especifica el número de días que un objeto no es actual antes de Amazon S3 pueda realizar la acción asociada. Para obtener información acerca de los cálculos de los días no actuales, consulte Cómo calcula Amazon S3 el tiempo que un objeto lleva siendo no actual en la Guía del usuario de Amazon Simple Storage Service. Tipo: entero no negativo cuando se usa con NoncurrentVersionTransition , entero positivo cuando se usa con NoncurrentVersionExpiration Antecesor: NoncurrentVersionExpiration o NoncurrentVersionTransition	Sí
NoncurrentVersionExpiration	Especifica cuándo expiran las versiones de objetos no actuales. Cuando expiran, Amazon S3 elimina de forma permanente las versiones de objetos no actuales. Establezca esta acción de configuración del ciclo de vida en un bucket que tenga habilitado (o suspendido) el control de versiones para indicar a Amazon S3 que elimine las versiones de objetos no actuales en un periodo específico o de la vida útil del objeto. Tipo: Contenedor Secundario: NoncurrentDays Antecesor: Rule	Sí, si no hay ninguna otra acción presente en la Rule

Nombre	Descripción	Obligatorio
NoncurrentVersionTransition	Contenedor de la regla de transición que describe cuándo los objetos no actuales se transfieren a STANDARD_IA , ONEZONE_IA o clase de almacenamiento. Si el bucket tiene habilitado (o suspendido) el control de versiones, puede establecer esta acción para indicar a Amazon S3 que transfiera a las versiones de objetos no actuales en periodo específico de la vida útil del objeto. Tipo: Contenedor Secundario: NoncurrentDays y StorageClass Antecesor: Rule	Sí, si no hay ninguna otra acción presente en la Rule
Prefix	Prefijo de clave de objeto que identifica uno o varios objetos a los que se aplica la regla. Tipo: cadena Antecesor: Rule	Sí
Rule	Contenedor para una regla del ciclo de vida. Una configuración de ciclo de vida puede contener hasta 1000 reglas. Tipo: Contenedor Antecesor: LifecycleConfiguration	Sí

Nombre	Descripción	Obligatorio
Status	Si está habilitada, Amazon S3 ejecuta la regla según lo programado. Si está desactivada, Amazon S3 ignora la regla. Tipo: cadena Antecesor: Rule Valores válidos: habilitado, desactivado	Sí
StorageClass	Especifica la clase de almacenamiento de Amazon S3 a la que quiere que pase el objeto. Tipo: cadena Antecesor: Transition y NoncurrentVersionTransition Valores válidos: STANDARD_IA ONEZONE_IA GLACIER	Sí Este elemento solo es necesario si especifica a uno o ambos de sus antecesor es.

Nombre	Descripción	Obligatorio
Transition	Esta acción especifica un periodo de la vida útil de los objetos en el que Amazon S3 debe hacer la transición a STANDARD_IA , ONEZONE_IA o clase de almacenamiento. Cuando esta acción esté en vigor, lo que haga Amazon S3 dependerá de si el bucket tiene habilitado el control de versiones. • Si el control de versiones no se habilita o nunca en el bucket, Amazon S3 pasa la única copia del objeto a la clase de almacenamiento especificada. • Si el bucket tiene habilitado (o suspendido) el control de versiones, Amazon S3 pasa solo las versiones actuales de los objetos identificados en la regla.  Note El bucket con control de versiones habilitado puede tener muchas versiones de un objeto. Esta acción no afecta a las versiones de objetos no actuales. Para realizar la transición de objetos no actuales, debe utilizar la acción NoncurrentVersionTransition .	Sí, si no hay ninguna otra acción presente en la Rule
	Tipo: Contenedor	

Nombre	Descripción	Obligatorio
	Secundario: Días o Fecha y StorageClass	
	Antecesor: Rule	

Respuestas

Encabezados de respuesta

Esta implementación de la operación solo usa encabezados de respuesta que sean comunes a la mayoría de las respuestas. Para obtener más información, consulte [Encabezados de respuesta comunes](#).

Elementos de respuesta

Esta implementación de la operación no devuelve elementos de respuesta.

Errores especiales

Esta implementación de la operación no devuelve errores especiales. Para obtener información sobre los errores de Amazon S3 y una lista de códigos de error, consulte [Respuestas de error](#).

Ejemplos

Ejemplo 1: Agregar la configuración del ciclo de vida a un bucket que no tenga habilitado el control de versiones

La configuración de ciclo de vida siguiente especifica dos reglas, cada una con una acción.

- La acción de transición indica a Amazon S3 que realice la transición de los objetos con el prefijo “documents/” a la clase de almacenamiento 30 días después de su creación.
- La acción de caducidad indica a Amazon S3 que elimine los objetos con el prefijo “logs/” 365 días después de su creación.

```
<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>

```

A continuación, se muestra una solicitud PUT `/?lifecycle` de ejemplo que agrega la configuración del ciclo de vida anterior al bucket `examplebucket`.

```

PUT /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Wed, 14 May 2014 02:11:21 GMT
Content-MD5: q6yJD1IkBaGGfb3QLY69A==
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>

```



```
</Rule>
</LifecycleConfiguration>
```

A continuación, se muestra una respuesta de ejemplo.

```
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
```

Ejemplo 2: Agregar la configuración del ciclo de vida a un bucket con el control de versiones habilitado

La configuración de ciclo de vida siguiente especifica dos reglas, cada una con una acción para que realice Amazon S3. Debe especificar estas acciones cuando el bucket tenga el control de versiones habilitado o suspendido:

- La acción `NoncurrentVersionExpiration` indica a Amazon S3 que caduque las versiones no actuales de los objetos con el prefijo “logs/” 100 días después de que los objetos pasen a ser no actuales.
- La acción `NoncurrentVersionTransition` indica a Amazon S3 que pase las versiones no actuales de los objetos con el prefijo “documents/” a la clase de almacenamiento 30 días después de convertirse en no actuales.

```
<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>

```

A continuación, se muestra una solicitud PUT `/?lifecycle` de ejemplo que agrega la configuración del ciclo de vida anterior al bucket `examplebucket`.

```

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>

```

A continuación, se muestra una respuesta de ejemplo.

```

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
```

Ejemplos adicionales

Para ver más ejemplos de la transición de objetos a clases de almacenamiento, como STANDARD_IA o ONEZONE_IA, consulte [Ejemplos de configuración del ciclo de vida](#).

Activos relacionados

- [GetBucketLifecycleConfiguration](#)
- [POST Object restore](#)
- De forma predeterminada, el propietario de un recurso (en este caso, el propietario del bucket, que es la Cuenta de AWS que creó el bucket) puede realizar cualquiera de las operaciones. El propietario del recurso también puede conceder permisos a otras personas para realizar la operación. Para obtener más información, consulte los siguientes temas en la Guía del usuario de Amazon Simple Storage Service:
 - [Especificación de permisos en una política](#)
 - [Administración de permisos de acceso para los recursos de Amazon S](#)

Ciclo de vida de bucket GET (obsoleto)

Descripción

Important

Para obtener una versión actualizada de esta API, consulte [GetBucketLifecycleConfiguration](#). Si configuró el ciclo de vida de un bucket con el elemento <filter>, debería ver una versión actualizada de este tema. Este tema se proporciona por compatibilidad con versiones anteriores.

Devuelve la información de configuración de `lifecycle` establecida en el bucket. Para obtener información sobre la configuración del ciclo de vida, vaya a [Administración del ciclo de vida de los objetos](#) en la Guía del usuario de Amazon Simple Storage Service.

Para usar esta operación, debe tener el permiso para realizar la acción `s3:GetLifecycleConfiguration`. De forma predeterminada, el propietario del bucket tiene este permiso. El propietario del bucket puede conceder este permiso a otros. Para obtener más información sobre los permisos, consulte [Administración de los permisos de acceso a los recursos de Amazon S3](#) en la Guía del usuario de Amazon Simple Storage Service.

Solicitudes

Sintaxis

```
GET /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Autenticación de solicitudes \(AWS Signature Version 4\))
```

Parámetros de la solicitud

Esta aplicación de la operación no utiliza parámetros de solicitud.

Encabezados de solicitud

Esta implementación de la operación solo usa encabezados de solicitud que sean comunes para todas las operaciones. Para obtener más información, consulte [Encabezados de solicitud comunes](#).

Elementos de la solicitud

Esta implementación de la operación no utiliza elementos de solicitud.

Respuestas

Encabezados de respuesta

Esta implementación de la operación solo usa encabezados de respuesta que sean comunes a la mayoría de las respuestas. Para obtener más información, consulte [Encabezados de respuesta comunes](#).

Elementos de respuesta

Esta implementación de GET devuelve los siguientes elementos de respuesta.

Nombre	Descripción	Obligatorio
<code>AbortIncompleteMultipartUpload</code>	Contenedor para especificar cuándo una carga multiparte incompleta pasa a ser apta para una operación de anulación. Elemento secundario: <code>DaysAfterInitiation</code> Tipo: Contenedor Antecesor: <code>Rule</code>	Sí, si no se especifica a ninguna otra acción para la regla
<code>Date</code>	La fecha en la que desea que Amazon S3 realice la acción. Para obtener más información, consulte Reglas de ciclo de vida: en función de una fecha específica en la Guía del usuario de Amazon Simple Storage Service. El valor de la fecha debe estar en formato ISO 8601. La hora siempre es medianoche UTC. Tipo: cadena	Sí, si <code>Days</code> y <code>ExpiredObjectDeleteMarker</code> están ausentes

Nombre	Descripción	Obligatorio
	Antecesor: <code>Expiration</code> o <code>Transition</code>	
<code>Days</code>	Especifica el número de días después de la creación del objeto cuando entra en vigor la acción de regla específica. El tiempo de aptitud del objeto se calcula como el tiempo de creación más el número de días, redondeando el tiempo resultante a la medianoche UTC del día siguiente. Tipo: entero no negativo cuando se usa con <code>Transition</code> , entero positivo cuando se usa con <code>Expiration</code> . Antecesor: <code>Transition</code> o <code>Expiration</code>	Sí, si <code>Date</code> y <code>ExpiredObjectDeleteMarker</code> están ausentes
<code>DaysAfterInitiation</code>	Especifica el número de días después de iniciar una carga multiparte en los que debe completarse la carga multiparte. Si no se completa en el número de días especificado, se podrá aplicar una operación de anulación y Amazon S3 cancela la carga multiparte incompleta. Tipo: entero positivo Antecesor: <code>AbortIncompleteMultipartUpload</code>	Sí, si <code>Date</code> está ausente

Nombre	Descripción	Obligatorio
Expiration	Esta acción especifica un periodo de la vida útil del objeto en el que Amazon S3 debe llevar a cabo la acción de caducidad adecuada. La acción de caducidad solo se produce en los objetos que son aptos según el periodo especificado en el elemento secundario Date o Days. La acción que tome Amazon S3 depende de si el bucket tiene habilitado el control de versiones. • Si el control de versiones nunca se ha habilitado en el bucket, Amazon S3 elimina la única copia del objeto de forma permanente. • De lo contrario, si el bucket tiene el control de versiones habilitado (o el control de versiones está suspendido), la acción se aplica solo a la versión actual del objeto. Los buckets que tienen habilitado o suspendido el control de versiones, pueden tener muchas versiones del mismo objeto: una versión actual y ninguna o varias versiones no actuales. En lugar de eliminar la versión actual, Amazon S3 la convierte en una versión no actual agregándole un marcador de eliminación como la nueva versión actual.	Sí, si se especifica la etiqueta principal

Nombre	Descripción	Obligatorio
	 Important Si el estado de un bucket es con el control de versiones suspendido, Amazon S3 crea un marcador de eliminación con el ID de versión null. Si tiene una versión con un ID de versión null, Amazon S3 sobrescribe esa versión.  Note Para establecer la caducidad de los objetos no actuales, debe usar la acción NoncurrentVersionExpiration . Tipo: Contenedor Secundario: Días o Fecha Antecesor: Rule	
ID	Identificador único de la regla. El valor no puede tener más de 255 caracteres. Tipo: cadena Antecesor: Rule	No

Nombre	Descripción	Obligatorio
LifecycleConfiguration	Contenedor para las reglas del ciclo de vida. Puede añadir hasta 1000 reglas. Tipo: Contenedor Secundario: Regla Ancestor: ninguno	Sí
ExpiredObjectDeleteMarker	En un bucket con control de versiones (un bucket con el control de versiones habilitado o suspendido), este elemento indica si Amazon S3 eliminará los marcadores de eliminación de objetos caducados en el bucket. Para ver un ejemplo, vaya a Ejemplo 8: Especificar acción de vencimiento para eliminar marcadores de eliminación de objetos caducados en la Guía del usuario de Amazon Simple Storage Service. Tipo: cadena Valores válidos: true false (el valor false está permitido pero no es operativo, Amazon S3 no realiza ninguna acción si el valor es false) Antecesor: Expiration	Sí, si Date y Days están ausentes

Nombre	Descripción	Obligatorio
NoncurrentDays	Especifica el número de días que un objeto no es actual antes de que Amazon S3 pueda realizar la acción asociada. Para obtener información sobre el cálculo de los días no actuales, consulte Reglas del ciclo de vida basadas en el número de días en la Guía del usuario de Amazon Simple Storage Service. Tipo: entero no negativo cuando se usa con NoncurrentVersionTransition , entero positivo cuando se usa con NoncurrentVersionExpiration Antecesor: NoncurrentVersionExpiration o NoncurrentVersionTransition	Sí, solo si el antecesor está presente
NoncurrentVersionExpiration	Especifica cuándo expiran las versiones de objetos no actuales. Cuando expiran, Amazon S3 elimina de forma permanente las versiones de objetos no actuales. Establezca esta acción de configuración del ciclo de vida en un bucket que tenga habilitado (o suspendido) el control de versiones a fin de solicitar que Amazon S3 elimine las versiones de objetos no actuales en un periodo específico o de la vida útil del objeto. Tipo: Contenedor Secundario: NoncurrentDays Antecesor: Rule	Sí, si no hay ninguna otra acción presente en la Rule

Nombre	Descripción	Obligatorio
NoncurrentVersionTransition	Contenedor de la regla de transición que describe cuándo los objetos no actuales se transfieren a STANDARD_IA , ONEZONE_IA o la clase de almacenamiento. Si el bucket tiene habilitado (o desactivado) el control de versiones, puede establecer esta acción para solicitar que Amazon S3 transfiera las versiones de objetos no actuales a la clase de almacenamiento en un periodo específico de la vida útil del objeto. Tipo: Contenedor Secundario: NoncurrentDays y StorageClass Antecesor: Rule	Sí, si no hay ninguna otra acción presente en la Rule
Prefix	Prefijo de clave de objeto que identifica uno o varios objetos a los que se aplica la regla. Tipo: cadena Antecesor: Rule	Sí
Rule	Contenedor para una regla del ciclo de vida. Tipo: Contenedor Antecesor: LifecycleConfiguration	Sí

Nombre	Descripción	Obligatorio
Status	Si está habilitada, Amazon S3 ejecuta la regla según lo programado. Si está desactivada, Amazon S3 ignora la regla. Tipo: cadena Antecesor: Rule Valores válidos: Habilitado o Desactivado	Sí
StorageClass	Especifica la clase de almacenamiento de Amazon S3 a la que quiere que pase el objeto. Tipo: cadena Antecesor: Transition y NoncurrentVersionTransition Valores válidos: STANDARD_IA ONEZONE_IA	Sí

Nombre	Descripción	Obligatorio
Transition	Esta acción especifica un periodo de la vida útil de los objetos en el que Amazon S3 debe hacer la transición a STANDARD_IA , ONEZONE_IA o clase de almacenamiento. Cuando esta acción esté en vigor, lo que haga Amazon S3 dependerá de si el bucket tiene habilitado el control de versiones. • Si el control de versiones no se habilita o nunca en el bucket, Amazon S3 pasa la única copia del objeto a la clase de almacenamiento especificada. • Cuando el bucket tiene habilitado (o suspendido) el control de versiones, Amazon S3 pasa solo las versiones actuales de los objetos identificados en la regla.  Note Un bucket con control de versiones habilitado o suspendido puede contener varias versiones de un objeto. Esta acción no afecta a las versiones de objetos no actuales. Para realizar la transición de objetos no actuales, debe utilizar la acción NoncurrentVersionTransition .	Sí, si no hay ninguna otra acción presente en la Rule

Nombre	Descripción	Obligatorio
	Tipo: Contenedor	
	Secundario: Días o Fecha y StorageClass	
	Antecesor: Rule	

Errores especiales

Código de error	Descripción	Código de estado HTTP	Prefijo de código de error de SOAP
NoSuchLifecycleConfiguration	La configuración del ciclo de vida no existe.	404 Not Found (No encontrado)	Cliente

Para obtener información general sobre los errores de Amazon S3 y una lista de códigos de error, consulte [Respuestas de error](#).

Ejemplos

Ejemplo 1: Recuperar un subrecurso del ciclo de vida

Este ejemplo es una solicitud GET para recuperar el subrecurso `lifecycle` del bucket especificado y un ejemplo de respuesta con la configuración del ciclo de vida devuelta.

Solicitud de muestra

```
GET /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Thu, 15 Nov 2012 00:17:21 GMT
Authorization: signatureValue
```

Respuesta de ejemplo

```
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>
```

Activos relacionados

- [PutBucketLifecycleConfiguration](#)
- [DeleteBucketLifecycle](#)

Historial de revisión