



Guia do desenvolvedor

Amazon Simple Storage Service



Versão da API 2006-03-01

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

Amazon Simple Storage Service: Guia do desenvolvedor

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

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestige a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

Table of Contents

O que é o Amazon S3?	1
Recomendações de chamada de API	2
Como fazer chamadas diretas de API REST	2
Permissões	3
Desenvolvimento com o Amazon S3	4
Fazer solicitações	4
Sobre as chaves de acesso	5
Endpoints de solicitações	7
Fazer solicitações por meio do IPv6	7
Fazer solicitações usando os AWS SDKs	18
Fazer solicitações usando a API REST	57
Usar a AWS CLI	70
Saiba mais sobre o AWS CLI	71
Desenvolvimento com SDKs da AWS	71
Interfaces de programação do SDK	72
Como especificar a versão da assinatura na autenticação de solicitações	73
Obter IDs de solicitação do Amazon S3 para Suporte	82
Usar a AWS CLI para obter IDs de solicitação	83
Usar o Windows PowerShell para obter IDs de solicitação	83
Usar eventos de dados do AWS CloudTrail para obter IDs de solicitação	83
Usar o registro de acesso ao servidor do S3 para obter IDs de solicitação	83
Usar HTTP para obter IDs de solicitação	84
Usar um navegador da web para obter IDs de solicitação	84
Usar os AWS SDKs para obter IDs de solicitação	84
Operações de API no nível do objeto do S3 com suporte para Tabelas do S3	90
Exemplos de código	95
Amazon S3	101
Conceitos básicos	106
Cenários	802
Exemplos sem servidor	1259
Amazon S3 Control	1271
Conceitos básicos	1271
Buckets de diretório do S3	1352
Conceitos básicos	1354

Cenários	1479
Autenticação de solicitações (AWS Signature Version 4)	1483
Métodos de autenticação	1484
Introdução à assinatura de solicitações	1485
Uso de um cabeçalho de autorização	1486
Visão geral	1487
Cálculo de assinatura: transferir carga útil em uma única parte	1492
Cálculo de assinatura: transferir carga útil em vários fragmentos	1511
Cálculo de assinatura: incluindo cabeçalhos finais	1524
Uso de parâmetros de consulta	1531
Cálculo de uma assinatura	1534
Um exemplo	1539
Exemplo 2	1541
Exemplos: cálculos de assinatura	1541
Exemplos de cálculo de assinatura usando Java	1542
Exemplos de cálculo de assinatura usando C#	1543
Autenticação de solicitações HTTP POST	1544
Cálculo de uma assinatura	1547
Chaves de política específicas de autenticação do Amazon S3 Signature Version 4.	1548
Exemplos de política de bucket que usam chaves de condição relacionadas ao Signature Version 4	1551
Uploads baseados no navegador usando POST	1555
Objeto POST	1556
Descrição	1556
Versionamento	1557
Solicitações	1557
Exemplos	1577
Recursos relacionados	1578
POST restauração de objeto	1579
Descrição	1579
Consulta em arquivos com solicitações de seleção	1579
Restauração de arquivos	1581
Solicitações	1583
Respostas	1598
Exemplos	1600
Mais informações	1602

Uploads baseados no navegador usando HTTP POST	1602
Cálculo de uma assinatura	1604
Criação de formulários HTML	1605
Declaração de formulário HTML	1606
Campos do formulário HTML	1607
Política de POST	1613
Expiração	1614
Correspondência de condição	1615
Condições	1617
Caracteres de escape	1621
Exemplo de upload POST	1623
Upload de um arquivo no Amazon S3 usando HTTP POST	1623
Uploads baseados no navegador usando o AWS Amplify	1626
Uso da biblioteca JavaScript do AWS Amplify para fazer upload de arquivos no Amazon S3	1626
Mais informações	1627
Cabeçalhos de solicitação comuns	1628
Cabeçalhos de resposta comuns	1633
Respostas de erro	1638
Respostas de erro de REST	1639
Lista de códigos de erro	1640
Lista de códigos de erro de conteúdo do objeto SELECT	1671
Lista de códigos de erro relacionados à replicação	1684
Lista de códigos de erro relacionados à marcação	1687
Lista de códigos de erro relacionados à anotação do Amazon S3	1688
Lista de códigos de erro do Amazon S3 em Outposts	1692
Lista de códigos de erro da Lente de Armazenamento do Amazon S3	1693
Lista de códigos de erro do Amazon S3 Object Lambda	1703
Lista de códigos de erro assíncronos do Amazon S3	1707
Lista de códigos de erro da Concessão de Acesso do Amazon S3	1709
Lista de códigos de erro relacionados ao Amazon FSx	1712
Lista de códigos de erro das Tabelas do Amazon S3	1713
Melhores práticas com relação a erros do Amazon S3	1714
Tentar InternalErrors novamente	1714
Ajustar o aplicativo para erros repetidos de SlowDown	1715
Erros isolados	1715

AWSGlossário da	1716
Recursos	1717
Apêndice	1719
Apêndice: resposta SelectObjectContent	1720
Descrição	1720
Respostas	1720
Recursos relacionados	1730
Apêndice: objeto OPTIONS	1732
Descrição	1732
Solicitações	1732
Respostas	1733
Exemplos	1735
Recursos relacionados	1735
Apêndice: API SOAP	1736
Operações no serviço (API SOAP)	1736
Operações em buckets (API SOAP)	1738
Operações em objetos (API SOAP)	1752
Autenticar solicitações SOAP	1777
Configurar políticas de acesso padrão com SOAP	1779
Elementos comuns	1780
Respostas de erro de SOAP	1781
Apêndice: autenticação de solicitações (AWS Signature Version 2)	1783
Autenticação de solicitações usando a API REST (AWS Signature Version 2)	1784
Assinatura e autenticação de solicitações REST (AWS Signature Version 2)	1787
Uploads baseados no navegador usando POST (AWS Signature Version 2)	1803
Apêndice: APIs de configuração do ciclo de vida (obsoletas)	1828
PUT Bucket lifecycle (obsoleto)	1829
GET Bucket lifecycle (obsoleto)	1846
Histórico do documento	1858

O que é o Amazon S3?

O Amazon Simple Storage Service (Amazon S3) é um serviço de armazenamento de objetos que oferece escalabilidade líder do setor, disponibilidade de dados, segurança e performance. Clientes de todos os tamanhos e setores podem usar o Amazon S3 para armazenar e proteger qualquer volume de dados para uma variedade de casos de uso, como data lakes, sites, aplicações móveis, backup e restauração, arquivamento, aplicações corporativas, dispositivos IoT e análises de big data. O Amazon S3 fornece recursos de gerenciamento para que você possa otimizar, organizar e configurar o acesso aos seus dados para atender aos seus requisitos específicos de negócios, organizacionais e de compatibilidade.

Este guia fornece informações focadas no desenvolvedor sobre o uso da API REST do Amazon S3 para criar aplicativos que armazenam e recuperam dados na nuvem.

Você pode usar qualquer toolkit compatível com HTTP para usar a API REST. Você pode até usar um navegador para buscar objetos, desde que eles possam ser lidos anonimamente.

A API REST usa os cabeçalhos padrão e os códigos de status HTTP, para que os navegadores e os toolkits padrão funcionem como esperado. Em algumas áreas, adicionamos funcionalidade ao HTTP (por exemplo, adicionamos cabeçalhos para oferecer suporte ao controle de acesso). Nesses casos, fizemos o melhor para adicionar nova funcionalidade de uma forma que correspondesse ao estilo de uso padrão do HTTP.

Note

O suporte de SOAP via HTTP está obsoleto, mas continua disponível via HTTPS. No entanto, novos recursos do Amazon S3 não serão compatíveis com o SOAP. Recomendamos usar esta API REST ou os SDKs da AWS no seguinte link:

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

Este guia do desenvolvedor inclui:

- [Cabeçalhos de solicitação comuns](#) e da [Cabeçalhos de resposta comuns](#)
- [Respostas de erro](#)
- [Uploads baseados no navegador usando POST \(AWS Signature Version 4\)](#)

 Important

Leia o seguinte sobre autenticação e controle de acesso antes de abordar tópicos específicos da API.

As solicitações para o Amazon S3 podem ser autenticadas ou anônimas. O acesso autenticado exige credenciais que a AWS pode usar para autenticar suas solicitações.

Recomendações de chamada de API

Fazer chamadas de API REST diretamente do seu código pode ser complicado. Exige que você grave o código necessário para calcular uma assinatura válida para autenticar suas solicitações. Recomendamos as seguintes alternativas:

- Use os [SDKs da AWS](#) para enviar suas solicitações.

Além disso, consulte os [Exemplos de código e bibliotecas](#)

Se você usar os SDKs, não precisará gravar código para calcular uma assinatura para autenticação de solicitações, pois os clientes do SDK autenticam suas solicitações usando as chaves de acesso fornecidas por você. Você sempre deve usar os SDKs da AWS, a menos que tenha um bom motivo para não fazer isso.

- Use a CLI da AWS para fazer chamadas de API do Amazon S3. Para obter informações sobre como configurar a CLI da AWS e exemplos de comandos do Amazon S3, consulte os seguintes tópicos:

[Configurar a CLI da AWS](#) no Guia de usuário do Amazon Simple Storage Service.

[Uso do Amazon S3 com a interface de linha de comando da AWS](#) no Guia do usuário da AWS Command Line Interface.

Como fazer chamadas diretas de API REST

 Note

O cabeçalho da solicitação PUT é limitado a 8 KB. No cabeçalho da solicitação PUT, os metadados definidos pelo sistema são limitados a 2 KB. O tamanho de metadados definidos

pelo sistema é medido pela soma do número de bytes na codificação US-ASCII de cada chave e valor.

Se você quiser fazer suas próprias chamadas de API REST em vez de usar uma das alternativas acima, há algumas considerações a fazer.

- Para fazer chamadas diretas de API REST a partir do seu código, crie uma assinatura usando credenciais válidas e inclua a assinatura na sua solicitação. Para obter informações sobre vários métodos de autenticação e cálculos de assinatura, consulte [Autenticação de solicitações \(AWS Signature Version 4\)](#).
- A API REST usa os cabeçalhos padrão e os códigos de status HTTP para que os navegadores e kits de ferramentas padrão funcionem como esperado. Em algumas áreas, adicionamos funcionalidade ao HTTP (por exemplo, adicionamos cabeçalhos para oferecer suporte ao controle de acesso). Nesses casos, fizemos o melhor para adicionar nova funcionalidade de uma forma que correspondesse ao estilo de uso padrão do HTTP. Para mais informações sobre fazer solicitações, consulte [Fazer solicitações](#).

Permissões

Você pode ter credenciais válidas para autenticar suas solicitações, mas, a menos que tenha permissões do S3 do proprietário da conta ou do bucket, não poderá criar ou acessar os recursos do Amazon S3. Essas permissões geralmente são concedidas por meio de uma [política](#) do AWS Identity and Access Management (IAM), como uma política de bucket. Por exemplo, você deve ter permissões para criar um bucket do S3 ou obter um objeto em um bucket. Para ver uma lista completa de permissões do S3, consulte [Ações, recursos e chaves de condição do Amazon S3](#).

Para obter mais informações sobre as permissões para operações de API do S3 por tipos de recursos do S3, consulte [Permissões obrigatórias para operações de API do Amazon S3](#) no Guia do usuário do Amazon Simple Storage Service.

Se usar as credenciais do usuário-raiz da sua Conta da AWS, você terá todas as permissões. No entanto, não se recomenda o uso de credenciais do usuário-raiz. Em vez disso, recomendamos que você crie perfis do AWS Identity and Access Management (IAM) em sua conta e gerencie as permissões do usuário. Para obter mais informações, consulte [Gerenciamento de acesso](#) no Guia do usuário do Amazon Simple Storage Service.

Desenvolvimento com o Amazon S3

Esta seção aborda tópicos relacionados ao desenvolvedor para usar o Amazon S3. Para obter mais informações, consulte os tópicos abaixo.

Tópicos

- [Fazer solicitações](#)
- [Desenvolvimento com o Amazon S3 usando a AWS CLI](#)
- [Desenvolvimento com o Amazon S3 usando AWS SDKs](#)
- [Obter IDs de solicitação do Amazon S3 para o AWS Support](#)
- [Operações de API no nível do objeto do Amazon S3 com suporte para Tabelas do S3](#)

Fazer solicitações

O Amazon S3 é um serviço REST. Você pode enviar solicitações para o Amazon S3 usando a API REST ou as bibliotecas wrapper do AWS SDK (consulte [Código de exemplo e bibliotecas](#)) que envolvem a API REST estrutural do Amazon S3, simplificando as tarefas de programação.

Cada interação com o Amazon S3 é autenticada ou anônima. A autenticação é um processo de verificação da identidade do solicitante que está tentando acessar um produto da Amazon Web Services (AWS). As solicitações autenticadas devem incluir um valor de assinatura que autentique o remetente da solicitação. O valor da assinatura é, em parte, gerado a partir das chaves de acesso da AWS do solicitante (ID de chave de acesso e chave de acesso secreta). Para obter mais informações sobre a obtenção de chaves de acesso, consulte [Como obter credenciais de segurança?](#) no Referência geral da AWS.

Se você estiver usando o AWS SDK, as bibliotecas calcularão a assinatura a partir das chaves fornecidas. No entanto, se fizer chamadas diretas da API REST no aplicativo, você deverá escrever o código para calcular a assinatura e adicioná-la à solicitação.

Tópicos

- [Sobre as chaves de acesso](#)
- [Endpoints de solicitações](#)
- [Fazer solicitações para o Amazon S3 por meio do IPv6](#)
- [Fazer solicitações usando os AWS SDKs](#)

- [Fazer solicitações usando a API REST](#)

Sobre as chaves de acesso

As seções a seguir avaliam os tipos de chaves de acesso que você pode usar para fazer solicitações autenticadas.

Conta da AWS

Chaves de acesso da

As chaves de acesso da conta fornecem acesso total aos recursos da AWS que pertencem à conta. Veja a seguir exemplos de chaves de acessos:

- ID de chave de acesso (uma string de 20 caracteres alfanuméricos). Por exemplo, .: AKIAIOSFODNN7EXAMPLE
- Chave de acesso secreta (uma string de 40 caracteres). Por exemplo, .: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

O ID de chave de acesso identifica uma de maneira única Conta da AWS. Use essas chaves de acesso para enviar solicitações autenticadas para o Amazon S3.

Chaves de acesso do usuário do IAM

Você pode criar uma Conta da AWS para sua empresa. No entanto, podem existir vários funcionários na organização que precisam de acesso aos recursos da AWS da organização. Compartilhar as chaves de acesso da Conta da AWS reduz a segurança e criar Contas da AWS individuais para cada funcionário pode não ser prático. Além disso, não é fácil compartilhar recursos como buckets e objetos, pois eles pertencem a contas diferentes. Para compartilhar recursos, você deve conceder permissões, o que gera trabalho adicional.

Em tais cenários, use o AWS Identity and Access Management (IAM) para criar usuários em sua Conta da AWS com suas próprias chaves de acesso e anexe políticas de usuário do IAM que concedam as permissões de acesso aos recursos apropriados para esses usuários. Para gerenciar melhor esses usuários, o IAM permite que você crie grupos de usuários e conceda permissões no nível de grupo que se aplicam a todos os usuário do grupo.

Esses usuários são conhecidos como usuários do IAM criados e gerenciados dentro da AWS. A conta pai controla a capacidade que um usuário tem de acessar a AWS. Quaisquer recursos que um usuário do IAM cria estão sob o controle da pai e são pagos por ela Conta da AWS. Esses usuários do IAM podem enviar solicitações autenticadas para o Amazon S3 usando as próprias credenciais

de segurança. Para obter mais informações sobre a criação e o gerenciamento de usuário em sua Conta da AWS, acesse a [página de detalhes do produto do AWS Identity and Access Management](#).

Credenciais de segurança temporárias

Além de criar usuários do IAM com suas próprias chaves de acesso, o IAM também permite que você conceda credenciais de segurança temporárias (chaves de acesso temporárias e um token de segurança) a qualquer usuário do IAM para que ele acesse serviços e recursos da AWS. Você também pode gerenciar usuários no sistema fora da AWS. Eles são conhecidos como usuários federados. Além disso, usuários podem ser aplicações criadas para acessar os recursos da AWS.

O IAM fornece a API do AWS Security Token Service para a solicitação de credenciais de segurança temporárias. Use a API do AWS STS ou o AWS SDK para solicitar essas credenciais. A API retorna as credenciais de segurança temporárias (ID de chave de acesso e chave de acesso secreta) e um token de segurança. Essas credenciais são válidas apenas pela duração especificada ao solicitá-las. Use o ID de chave de acesso e a chave secreta da mesma forma que os usa ao enviar solicitações usando a Conta da AWS ou as chaves de acesso do usuário do IAM. Além disso, é necessário incluir o token em cada solicitação enviada para o Amazon S3.

Um usuário do IAM pode solicitar essas credenciais de segurança temporárias para seu próprio uso ou enviá-las para usuários federados ou aplicações. Ao solicitar credenciais de segurança temporárias para usuários federados, você deve fornecer um nome de usuário e uma política do IAM definindo as permissões que deseja associar a essas credenciais. O usuário federado não pode obter mais permissões que o usuário pai do IAM que solicitou as credenciais temporárias.

Use as credenciais de segurança temporárias para fazer solicitações ao Amazon S3. As bibliotecas de API calculam o valor de assinatura necessário usando essas credenciais para autenticar sua solicitação. Se você enviar solicitações usando credenciais vencidas, o Amazon S3 negará a solicitação.

Para obter informações sobre a assinatura de solicitações usando credenciais de segurança temporárias nas solicitações da API REST, consulte [Assinatura e autenticação de solicitações REST \(AWS Signature Version 2\)](#). Para obter informações sobre o envio de solicitações usando AWS SDKs, consulte [Fazer solicitações usando os AWS SDKs](#).

Para obter mais informações sobre o suporte do IAM para credenciais de segurança temporárias, consulte [Credenciais de segurança temporárias](#) no Manual do usuário do IAM.

Para maior segurança, é possível exigir autenticação multifator (MFA) ao acessar os recursos do Amazon S3 configurando uma política do bucket. Para obter informações, consulte [Exemplo de](#)

[políticas de bucket: exigência de MFA](#). Depois de exigir a MFA para acesso aos recursos do Amazon S3, a única maneira de acessar esses recursos é fornecendo credenciais temporárias criadas com uma chave MFA. Para obter mais informações, consulte a página de detalhes do [AWS Multi-Factor Authentication](#) e [Configuração do acesso à API com proteção MFA](#) no Manual do usuário do IAM.

Endpoints de solicitações

Envie solicitações REST para o endpoint predefinido do serviço. Para obter uma lista de todos os serviços da AWS Endpoints os respectivos endpoints, acesse [Regiões e endpoints](#) na Referência geral da AWS.

Fazer solicitações para o Amazon S3 por meio do IPv6

O Amazon Simple Storage Service (Amazon S3) oferece suporte à capacidade de acessar buckets do S3 usando o protocolo IPv6 (Internet Protocol versão 6), além do protocolo IPv4. Os endpoints de pilha dupla do Amazon S3 oferecem suporte para buckets do S3 por IPv6 e IPv4. Não há custo adicional para acessar o Amazon S3 por meio do IPv6. Para obter mais informações sobre a definição de preços, consulte [Definição de preços do Amazon S3](#).

Note

As informações a seguir são para endpoints IPv6 públicos do Amazon S3. Para obter mais informações sobre endpoints da nuvem privada virtual (VPC) da AWS, consulte [AWS PrivateLink para o Amazon S3](#).

Tópicos

- [Conceitos básicos para fazer solicitações por meio do IPv6](#)
- [Como usar endereços do IPv6 em políticas do IAM](#)
- [Testar a compatibilidade com endereços IP](#)
- [Usar endpoints de pilha dupla do Amazon S3](#)

Conceitos básicos para fazer solicitações por meio do IPv6

Para fazer uma solicitação para um bucket do S3 por meio do IPv6, você precisa usar um endpoint de pilha dupla. A próxima seção descreve como fazer solicitações por meio do IPv6 usando endpoints de pilha dupla.

Estas são algumas coisas sobre as quais você deve estar ciente antes de tentar acessar um bucket por meio do IPv6:

- O cliente e a rede que estão acessando o bucket devem ter permissão para usar o IPv6.
- As solicitações de estilo hospedado virtual e de estilo de caminho são compatíveis para acessarem o IPv6. Para obter mais informações, consulte [Endpoints de pilha dupla do Amazon S3](#).
- Se você usar a filtragem de endereços IP de origem nas políticas de usuário ou de bucket do AWS Identity and Access Management (IAM), será necessário atualizar as políticas para incluir intervalos de endereços IPv6. Para obter mais informações, consulte [Como usar endereços do IPv6 em políticas do IAM](#).
- Ao usar o IPv6, os arquivos de log de acesso ao servidor fornecem endereços IP em um formato do IPv6. Você precisa atualizar as ferramentas, os scripts e o software existentes que usa para analisar os arquivos de log do Amazon S3 para que eles possam analisar os endereços Remote IP formatados para IPv6. Para obter mais informações, consulte [Registro em log das solicitações com registro em log de acesso ao servidor](#).

 Note

Se você tiver problemas relacionados à presença de endereços IPv6 nos arquivos de log, entre em contato com o [AWS Support](#).

Como fazer solicitações por meio do IPv6 usando endpoints de pilha dupla

Você faz solicitações com chamadas da API do Amazon S3 por meio do IPv6 usando endpoints de pilha dupla. As operações de API do Amazon S3 funcionam da mesma maneira se você estiver acessando o Amazon S3 por IPv6 ou por IPv4. A performance deve ser a mesma também.

Ao usar a API REST, você acessa um endpoint de pilha dupla diretamente. Para obter mais informações, consulte [Endpoints de pilha dupla](#).

Ao usar a AWS Command Line Interface (AWS CLI) e os AWS SDKs, você pode utilizar um parâmetro ou um sinalizador para mudar para um endpoint de pilha dupla. Você também pode especificar o endpoint de pilha dupla diretamente como uma substituição do endpoint do Amazon S3 no arquivo de configuração.

Você pode usar um endpoint de pilha dupla para acessar um bucket por meio do IPv6 de qualquer um dos seguintes:

- A AWS CLI, consulte [Usar endpoints de pilha dupla da AWS CLI](#).
- Os AWS SDKs, consulte [Usar endpoints de pilha dupla dos AWS SDKs](#).
- A API REST, consulte [Fazer solicitações para endpoints de pilha dupla usando a API REST](#).

Recursos não disponíveis por meio do IPv6

No momento, o recurso a seguir não é compatível ao acessar um bucket do S3 com IPv6: hospedagem de site estático proveniente de um bucket do S3.

Como usar endereços do IPv6 em políticas do IAM

Antes de tentar acessar um bucket usando o IPv6, você deve garantir que todas as políticas de usuário do IAM ou de bucket do S3 usadas para filtragem de endereços IP estejam atualizadas para incluir intervalos de endereços do IPv6. As políticas de filtragem de endereços IP que não estiverem atualizadas para lidar com endereços do IPv6 podem resultar na perda ou no ganho de acesso de clientes ao bucket quando começarem a usar o IPv6. Para ter mais informações sobre como gerenciar permissões de acesso com o IAM, consulte [Gerenciamento de identidade e acesso para o Amazon S3](#).

As políticas do IAM que filtram endereços IP usam [Operadores de condição de endereço IP](#). A política de bucket a seguir identifica o intervalo 54.240.143.* de endereços IPv4 permitidos usando operadores de condição de endereço IP. Todos os endereços IP fora deste intervalo terão o acesso ao bucket negado (examplebucket). Como todos os endereços do IPv6 estão fora do intervalo permitido, essa política impede que os endereços do IPv6 possam acessar o 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"}
      }
    }
  ]
}
```

```
}  
]  
}
```

Você pode modificar o elemento `Condition` da política do bucket para permitir os intervalos de endereços do IPv4 (54.240.143.0/24) e do IPv6 (2001:DB8:1234:5678::/64), conforme mostrado no exemplo a seguir. Você pode usar o mesmo tipo de bloqueio de `Condition` mostrado no exemplo para atualizar as políticas de usuário e de bucket do IAM.

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

Antes de usar o IPv6, é necessário atualizar todas as políticas de usuário e de bucket do IAM que usam a filtragem de endereços IP. Não recomendamos usar filtros de endereços IP em políticas de bucket.

Você pode revisar suas políticas de usuário do IAM usando o console do IAM em <https://console.aws.amazon.com/iam/>. Para obter mais informações sobre o IAM, consulte o [Manual do usuário do IAM](#). Para obter informações sobre como editar políticas de bucket do S3, consulte [Adição de uma política de bucket](#).

Testar a compatibilidade com endereços IP

Se estiver usando o Linux/Unix ou o Mac OS X, você poderá testar se é possível acessar um endpoint de pilha dupla por meio do IPv6 usando o comando `curl` conforme mostrado no exemplo a seguir:

Example

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

Você recebe de volta informações semelhantes ao exemplo a seguir. Se estiver conectado por meio do IPv6 o endereço IP conectado será um endereço do 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
```

Se estiver usando o Microsoft 7 ou 10, você poderá testar se é possível acessar um endpoint de pilha dupla por meio do IPv6 ou do IPv4 usando o comando ping conforme mostrado no exemplo a seguir:

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

Usar endpoints de pilha dupla do Amazon S3

Os endpoints de pilha dupla do Amazon S3 oferecem suporte para buckets do S3 por IPv6 e IPv4. Esta seção descreve como usar os endpoints de pilha dupla.

Tópicos

- [Endpoints de pilha dupla do Amazon S3](#)
- [Usar endpoints de pilha dupla da AWS CLI](#)
- [Usar endpoints de pilha dupla dos AWS SDKs](#)
- [Usar endpoints de pilha dupla da API REST](#)

Endpoints de pilha dupla do Amazon S3

Quando você faz uma solicitação para um endpoint de pilha dupla, o URL do bucket resolve para um endereço IPv6 ou IPv4. Para obter mais informações sobre como acessar um bucket por meio do IPv6, consulte [Fazer solicitações para o Amazon S3 por meio do IPv6](#).

Ao usar a API REST, você acessa diretamente um endpoint do Amazon S3 usando o nome do endpoint (URI). Você pode acessar um bucket do S3 por meio de um endpoint de pilha dupla usando um nome de endpoint de estilo de hospedagem virtual ou de estilo de caminho. O Amazon S3 oferece suporte apenas a nomes regionais de endpoint de pilha dupla, o que significa que você deve especificar a região como parte do nome.

Use as seguintes convenções de atribuição de nomes para os nomes de endpoint de estilo de hospedagem virtual e de estilo de caminho de pilha dupla:

- Endpoint de pilha dupla de estilo de hospedagem virtual

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

- Endpoint de pilha dupla de estilo de caminho:

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

Para obter mais informações sobre estilo de nome de endpoint, consulte [Acesso e listagem de um bucket do Amazon S3](#). Para obter uma lista dos endpoints do Amazon S3, consulte [Regiões e endpoints](#) na Referência geral da AWS.

Important

Você pode usar a aceleração de transferência com endpoints de pilha dupla. Para obter mais informações, consulte [Conceitos básicos do Amazon S3 Transfer Acceleration](#).

Note

Os dois tipos de endpoint da nuvem privada virtual (VPC) que acessam o Amazon S3 (endpoints de interface da VPC e endpoints da VPC do gateway) agora são compatíveis com pilha dupla. Para obter mais informações sobre os endpoints da VPC para o Amazon S3, consulte [AWS PrivateLink para Amazon S3](#).

Ao usar a AWS Command Line Interface (AWS CLI) e os AWS SDKs, você pode utilizar um parâmetro ou um sinalizador para mudar para um endpoint de pilha dupla. Você também pode especificar o endpoint de pilha dupla diretamente como uma substituição do endpoint do Amazon S3 no arquivo de configuração. As seções a seguir descrevem como usar endpoints de pilha dupla da AWS CLI e dos AWS SDKs.

Usar endpoints de pilha dupla da AWS CLI

Esta seção fornece exemplos de comandos da AWS CLI usados para fazer solicitações a um endpoint de pilha dupla. Para obter instruções de configuração da AWS CLI, consulte [Desenvolvimento com o Amazon S3 usando a AWS CLI](#).

Você define o valor de configuração `use_dualstack_endpoint` para `true` em um perfil no seu arquivo do AWS Config para direcionar todas as solicitações do Amazon S3 feitas pelos comandos `s3` e `s3api` da AWS CLI ao endpoint de pilha dupla para a região especificada. Especifique a região no arquivo de configuração ou em um comando usando a opção `--region`.

Quando se usa endpoints de pilha dupla com a AWS CLI, os estilos de endereçamento `path` e `virtual` são compatíveis. O estilo de endereçamento, definido no arquivo de configuração, controla se o nome do bucket está no hostname ou em parte do URL. Por padrão, a CLI tentará usar o estilo `virtual` sempre que possível, mas voltará ao estilo de caminho se necessário. Para obter mais informações, consulte [Configuração do Amazon S3 da AWS CLI](#).

Você também pode fazer alterações de configuração usando um comando, conforme mostrado no exemplo a seguir que define `use_dualstack_endpoint` para `true` e `addressing_style` para `virtual` no perfil padrão.

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

Se quiser usar um endpoint de pilha dupla apenas para comandos especificados da AWS CLI, (nem todos os comandos), você pode usar qualquer um dos métodos a seguir:

- Você pode usar o endpoint de pilha dupla por comando, definindo o parâmetro `--endpoint-url` como `https://s3.dualstack.aws-region.amazonaws.com` ou `http://s3.dualstack.aws-region.amazonaws.com` para qualquer comando `s3` ou `s3api`.

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

- Você pode configurar perfis separados em seu arquivo do AWS Config. Por exemplo, crie um perfil que defina `use_dualstack_endpoint` como `true` e um perfil que não defina `use_dualstack_endpoint`. Quando executar um comando, especifique qual perfil deseja usar, dependendo de querer ou não usar o endpoint de pilha dupla.

 Note

Atualmente, ao usar a AWS CLI, você não pode usar a aceleração de transferência com endpoints de pilha dupla. Contudo, o suporte para a AWS CLI estará disponível em breve. Consulte mais informações em [Como habilitar e usar o S3 Transfer Acceleration](#).

Usar endpoints de pilha dupla dos AWS SDKs

Esta seção fornece exemplos de como acessar um endpoint de pilha dupla usando os AWS SDKs.

AWS SDK para JavaExemplo do endpoint de pilha dupla do

O exemplo a seguir mostra como habilitar endpoints de pilha dupla ao criar um cliente do Amazon S3 usando o AWS SDK para Java.

Consulte instruções sobre como criar e testar uma amostra funcional de Java em [Getting Started](#) no Guia do desenvolvedor do 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();
    }
}
}
```

Se estiver usando o AWS SDK para Java no Windows, você talvez tenha de definir a seguinte propriedade da máquina virtual Java (JVM):

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

AWSExemplo do endpoint de pilha dupla do SDK para .NET

Ao usar o AWS SDK for .NET você, você usa a classe `AmazonS3Config` para permitir o uso de um endpoint de pilha dupla, como mostrado no exemplo a seguir.

```
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);
        }
    }
}
```

```
};
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 obter informações sobre como configurar e executar exemplos de código, consulte [Conceitos básicos do AWS SDK for .NET](#) no Guia do desenvolvedor do AWS SDK for .NET.

Usar endpoints de pilha dupla da API REST

Para obter informações sobre fazer solicitação para endpoints de pilha dupla usando a API REST, consulte [Fazer solicitações para endpoints de pilha dupla usando a API REST](#).

Fazer solicitações usando os AWS SDKs

Tópicos

- [Fazer solicitações usando credenciais de usuário do IAM ou da Conta da AWS](#)
- [Fazer solicitações usando credenciais temporárias de usuário do IAM](#)
- [Fazer solicitações usando credenciais temporárias de usuário federado](#)

Você pode enviar solicitações autenticadas para o Amazon S3 usando o AWS SDK ou fazendo chamadas de API REST diretamente em sua aplicação. A API do AWS SDK usa as credenciais fornecidas por você para computar a assinatura para autenticação. Se você usar a API REST diretamente em seus aplicativos, deverá gravar o código necessário para computar a assinatura para autenticar sua solicitação. Para obter uma lista de AWS SDKs disponíveis, acesse [Código de exemplo e bibliotecas](#).

Fazer solicitações usando credenciais de usuário do IAM ou da Conta da AWS

É possível usar suas credenciais de segurança de usuário do IAM ou da Conta da AWS para enviar solicitações autenticadas ao Amazon S3. Esta seção fornece exemplos de como você pode enviar solicitações autenticadas usando o AWS SDK para Java, o AWS SDK para .NET e o AWS SDK para PHP. Para obter uma lista de AWS SDKs disponíveis, acesse [Código de exemplo e bibliotecas](#).

Cada um desses AWS SDKs usa uma cadeia de provedor de credenciais específicas do SDK para encontrar e usar credenciais, além de realizar ações em nome do proprietário das credenciais. O que todas essas cadeias de provedor de credenciais têm em comum é que elas procuram por seu arquivo local de credenciais da AWS.

Para obter mais informações, consulte os tópicos abaixo.

Tópicos

- [Para criar um arquivo de credenciais da AWS local](#)
- [Enviar solicitações autenticadas usando os AWS SDKs](#)
- [Recursos relacionados](#)

Para criar um arquivo de credenciais da AWS local

A forma mais fácil de configurar credenciais para os AWS SDKs é usar um arquivo de credenciais da AWS. Se você utiliza a AWS Command Line Interface (AWS CLI), já deve ter um arquivo de

credenciais da AWS local configurado. Caso contrário, use o procedimento a seguir para configurar um arquivo de credenciais:

1. Faça login no Console de gerenciamento da AWS e abra o console do IAM em <https://console.aws.amazon.com/iam/>.
2. Crie um novo usuário com permissões limitadas aos serviços e ações aos quais você deseja que seu código tenha acesso. Para obter mais informações sobre como criar um usuário, consulte [Criar usuários do IAM \(console\)](#) e siga as instruções até a etapa 8.
3. Escolha Download .csv (Fazer download do arquivo .csv) para salvar uma cópia de suas credenciais da AWS.
4. Em seu computador, navegue para seu diretório inicial e crie um diretório `.aws`. Nos sistemas baseados em Unix, como Linux ou OS X, isso fica no seguinte local:

```
~/ .aws
```

No Windows, isso está no seguinte local:

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

5. No diretório `.aws`, crie um novo arquivo chamado `credentials`.
6. Abra o arquivo de credenciais `.csv` que você baixou do console do IAM e copie o conteúdo dele para o arquivo `credentials` usando o seguinte formato:

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

7. Salve o arquivo `credentials` e exclua o arquivo `.csv` que você baixou na etapa 3.

Seu arquivo de credenciais compartilhado agora está configurado em seu computador local, e ele está pronto para ser usado com os AWS SDKs.

Enviar solicitações autenticadas usando os AWS SDKs

Use os AWS SDKs para enviar solicitações autenticadas. Para obter mais informações sobre o envio de solicitações autenticadas, consulte [Credenciais de segurança da AWS](#) ou [Autenticação do IAM Identity Center](#).

Java

Para obter informações sobre como autenticar solicitações usando o SDK da AWS para Java, consulte [Uso de arquivos compartilhados de configuração e credenciais para configurar globalmente ferramentas e SDKs da AWS](#) e [Autenticação e acesso usando ferramentas e SDKs da AWS](#) no Guia de referência de ferramentas e SDKs da AWS.

.NET

Para enviar solicitações autenticadas usando as credenciais de Conta da AWS ou de usuário do IAM:

- Crie uma instância da classe `AmazonS3Client`.
- Execute um dos métodos do `AmazonS3Client` para enviar solicitações ao Amazon S3. O cliente gera a assinatura necessária das credenciais que você fornece e a inclui na solicitação enviada ao Amazon S3.

Para obter mais informações, consulte [Como fazer solicitações usando credenciais de usuário do IAM ou da conta da AWS](#).

Note

- Você pode criar o cliente `AmazonS3Client` sem fornecer suas credenciais de segurança. As solicitações enviadas usando esse cliente são solicitações anônimas, sem uma assinatura. O Amazon S3 retorna um erro se você enviar solicitações anônimas para um recurso que não esteja disponível publicamente.
- Você pode criar uma Conta da AWS e criar os usuários necessários. Você também pode gerenciar credenciais para esses usuários. Você precisa dessas credenciais para executar a tarefa no exemplo a seguir. Para obter mais informações, consulte [Configurar credenciais da AWS](#) no Guia do desenvolvedor do SDK para .NET.

Em seguida, você também pode configurar seu aplicativo para recuperar ativamente perfis e credenciais e, em seguida, usar explicitamente essas credenciais ao criar um cliente de serviço da AWS. Para obter mais informações, consulte [Acessando credenciais e perfis em uma aplicação](#) no Guia do desenvolvedor do SDK para .NET.

O exemplo de C# a seguir mostra como realizar as tarefas anteriores. Para obter informações sobre como configurar e executar exemplos de código, consulte [Conceitos básicos do AWS SDK for .NET](#) no Guia do desenvolvedor do AWS SDK for .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

Esta seção explica como usar uma classe da versão 3 do AWS SDK para PHP para enviar solicitações autenticadas usando suas credenciais da Conta da AWS ou de usuário do IAM. Para obter mais informações sobre a API do AWS SDK for Ruby, acesse [AWS SDK for Ruby Version 2](#).

O exemplo de PHP a seguir mostra como o cliente faz uma solicitação usando suas credenciais de segurança para listar todos os buckets para a sua conta.

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

Você pode criar o cliente `S3Client` sem fornecer suas credenciais de segurança. As solicitações enviadas usando esse cliente são solicitações anônimas, sem uma

assinatura. O Amazon S3 retorna um erro se você enviar solicitações anônimas para um recurso que não esteja disponível publicamente. Para obter mais informações, consulte [Criar clientes anônimos](#) na [Documentação do AWS SDK para PHP](#).

Ruby

Para poder usar a versão 3 do AWS SDK para Ruby para fazer chamadas para o Amazon S3, defina as credenciais de acesso da AWS que o SDK usa para verificar acesso aos seus buckets e objetos. Se você compartilhou credenciais configuradas no perfil de credenciais da AWS no sistema local, a versão 3 do SDK for Ruby poderá usar essas credenciais sem que você precise declará-las no código. Para obter mais informações sobre como configurar credenciais compartilhadas, consulte [Como fazer solicitações usando credenciais de usuário do IAM ou da conta da AWS](#).

O trecho de código seguinte do Ruby usa as credenciais de um arquivo de credenciais da AWS compartilhado em um computador local para autenticar uma solicitação a fim de obter todos os nomes de chaves de objeto em um bucket específico. Ela faz o seguinte:

1. Cria uma instância da classe `Aws::S3::Client`.
2. Faz uma solicitação para o Amazon S3, enumerando objetos em um bucket com o método `list_objects_v2` do `Aws::S3::Client`. O cliente gera o valor de assinatura necessário com base nas credenciais do arquivo de credenciais da AWS em seu computador e o inclui na solicitação que envia ao Amazon S3.
3. Imprime o array de nomes de chaves de objeto no 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__
```

Se você não tiver um arquivo de credenciais da AWS local, ainda poderá criar o recurso `Aws::S3::Client` e executar o código nos buckets e objetos do Amazon S3. As solicitações enviadas usando a versão 3 do SDK para Ruby são anônimas, sem assinatura por padrão. O Amazon S3 retornará um erro se você enviar solicitações anônimas para um recurso que não esteja disponível publicamente.

É possível usar e expandir o trecho de código anterior para aplicações do SDK para Ruby, como no seguinte exemplo mais 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

O exemplo a seguir usa credenciais da AWS carregadas automaticamente pelo SDK para Go pelo arquivo de credenciais compartilhado.

```
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

- [Desenvolvimento com o Amazon S3 usando AWS SDKs](#)
- [AWS SDK para PHP for Amazon S3, classe Aws\S3\S3Client](#)
- [AWS SDK para PHP Documentação](#)

Fazer solicitações usando credenciais temporárias de usuário do IAM

Uma Conta da AWS ou um usuário do IAM podem solicitar credenciais de segurança temporárias e usá-las para enviar solicitações autenticadas ao Amazon S3. Esta seção fornece exemplos de como usar AWS SDK para Java, .NET e PHP para obter credenciais de segurança temporárias e usá-las para autenticar suas solicitações para o Amazon S3.

Java

Um usuário do IAM ou uma Conta da AWS podem solicitar credenciais de segurança temporárias (consulte [Fazer solicitações](#)) usando o AWS SDK para Java e usá-las para acessar o Amazon S3. Essas credenciais expiram após a duração da sessão especificada.

Por padrão, a duração da sessão é de uma hora. Se usar credenciais de usuário do IAM, você poderá especificar a duração ao solicitar as credenciais de segurança temporárias de 15 minutos até a duração máxima de sessão para a função. Para obter mais informações sobre credenciais de segurança temporárias, consulte [Credenciais de segurança temporárias](#) no Manual do usuário do IAM. Para mais informações sobre fazer solicitações, consulte [Fazer solicitações](#).

Como obter credenciais de segurança temporárias e acesso ao Amazon S3

1. Crie uma instância da classe `AWSSecurityTokenService`.
2. Recupere as credenciais de segurança temporárias para a função desejada chamando o método `assumeRole()` do cliente STS (Security Token Service).
3. Empacote as credenciais de segurança temporárias em um objeto `BasicSessionCredentials`. Use esse objeto para fornecer as credenciais de segurança temporárias para o cliente do Amazon S3.
4. Crie uma instância da classe `AmazonS3Client` usando as credenciais de segurança temporárias. Envie solicitações ao Amazon S3 usando esse cliente. Se você enviar solicitações usando credenciais expiradas, o Amazon S3 retornará um erro.

Note

Se você obtiver as credenciais de segurança temporárias usando suas credenciais de segurança de Conta da AWS, as credenciais temporárias serão válidas somente por uma hora. Você pode especificar a duração de sessão somente se usar credenciais de usuário do IAM para solicitar uma sessão.

O exemplo a seguir lista um conjunto de chaves de objeto no bucket especificado. O exemplo obtém credenciais de segurança temporárias para uma sessão e usa essas credenciais para enviar uma solicitação autenticada ao Amazon S3.

Para testar o exemplo usando credenciais de usuário do IAM, crie um usuário do IAM em sua Conta da AWS. Para obter mais informações sobre como criar um usuário do IAM, consulte [Criar o primeiro usuário do IAM e grupo de administradores](#) no Manual do usuário do IAM.

Consulte instruções sobre como criar e testar uma amostra funcional em [Getting Started](#) no Guia do desenvolvedor do 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 =
            AWSSecurityTokenServiceClientBuilder.standard()
                .withCredentials(new ProfileCredentialsProvider())
                .withRegion(clientRegion)
                .build();
```



```
role of // Obtain credentials for the IAM role. Note that you cannot assume the
// 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

Um usuário do IAM ou uma Conta da AWS podem solicitar credenciais de segurança temporárias usando o AWS SDK para .NET e usá-las para acessar o Amazon S3. Essas credenciais expiram após a duração da sessão.

Por padrão, a duração da sessão é de uma hora. Se usar credenciais de usuário do IAM, você poderá especificar a duração ao solicitar as credenciais de segurança temporárias de 15 minutos até a duração máxima de sessão para a função. Para obter mais informações sobre credenciais de segurança temporárias, consulte [Credenciais de segurança temporárias](#) no Manual do usuário do IAM. Para mais informações sobre fazer solicitações, consulte [Fazer solicitações](#).

Como obter credenciais de segurança temporárias e acesso ao Amazon S3

1. Crie uma instância do cliente do AWS Security Token Service, `AmazonSecurityTokenServiceClient`.
2. Inicie uma sessão chamando o método `GetSessionToken` do cliente STS criado na etapa anterior. Forneça informações da sessão para esse método usando um objeto `GetSessionTokenRequest`.

O método retorna as credenciais de segurança temporárias.

3. Empacote as credenciais de segurança temporárias em uma instância do objeto `SessionAWSCredentials`. Use esse objeto para fornecer as credenciais de segurança temporárias para o cliente do Amazon S3.
4. Crie uma instância da classe `AmazonS3Client` passando as credenciais de segurança temporárias. Envie solicitações ao Amazon S3 usando esse cliente. Se você enviar solicitações usando credenciais expiradas, o Amazon S3 retornará um erro.

Note

Se você obtiver as credenciais de segurança temporárias usando suas credenciais de segurança de Conta da AWS, as credenciais serão válidas somente por uma hora. Você

poderá especificar a duração de sessão somente se usar as credenciais de usuário do IAM para solicitar uma sessão.

O exemplo do C# a seguir lista chaves de objeto no bucket especificado. Como ilustração, o exemplo obtém credenciais de segurança temporárias para uma sessão padrão de uma hora e usa essas credenciais para enviar uma solicitação autenticada ao Amazon S3.

Para testar o exemplo usando credenciais de usuário do IAM, crie um usuário do IAM em sua Conta da AWS. Para obter mais informações sobre como criar um usuário do IAM, consulte [Criar o primeiro usuário do IAM e grupo de administradores](#) no Manual do usuário do IAM. Para mais informações sobre fazer solicitações, consulte [Fazer solicitações](#).

Para obter informações sobre como configurar e executar exemplos de código, consulte [Conceitos básicos do AWS SDK for .NET](#) no Guia do desenvolvedor do AWS SDK for .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 obter mais informações sobre a API do AWS SDK for Ruby, acesse [AWS SDK for Ruby Version 2](#).

Um usuário do IAM ou uma Conta da AWS podem solicitar credenciais de segurança temporárias usando a versão 3 do AWS SDK para PHP. Posteriormente, as credenciais temporárias podem ser usadas para acessar o Amazon S3. As credenciais expiram quando a duração da sessão expira.

Por padrão, a duração da sessão é de uma hora. Se usar credenciais de usuário do IAM, você poderá especificar a duração ao solicitar as credenciais de segurança temporárias de 15 minutos até a duração máxima de sessão para a função. Para obter mais informações sobre credenciais de segurança temporárias, consulte [Credenciais de segurança temporárias](#) no Manual do usuário do IAM. Para mais informações sobre fazer solicitações, consulte [Fazer solicitações](#).

Note

Se você obtiver as credenciais de segurança temporárias usando suas credenciais de segurança de Conta da AWS, as credenciais de segurança temporárias serão válidas somente por uma hora. Você pode especificar a duração de sessão somente se usar credenciais de usuário do IAM para solicitar uma sessão.

Example

O exemplo de PHP a seguir lista as chaves de objeto no bucket especificado usando credenciais de segurança temporárias. O exemplo obtém credenciais de segurança temporárias para uma sessão padrão de uma hora e usa essas credenciais para enviar uma solicitação autenticada ao Amazon S3. Para obter mais informações sobre a API do AWS SDK for Ruby, acesse [AWS SDK for Ruby Version 2](#).

Para testar o exemplo usando credenciais de usuário do IAM, crie um usuário do IAM em sua Conta da AWS. Para obter informações sobre como criar um usuário do IAM, consulte [Criar o primeiro usuário do IAM e grupo de administradores](#) no Manual do usuário do IAM. Para obter exemplos de definição de duração de sessão usando credenciais de usuário do IAM para solicitar uma sessão, consulte [Fazer solicitações usando credenciais temporárias de usuário do 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

Um usuário do IAM ou uma Conta da AWS podem solicitar credenciais de segurança temporárias usando o AWS SDK para Ruby e usá-las para acessar o Amazon S3. Essas credenciais expiram após a duração da sessão.

Por padrão, a duração da sessão é de uma hora. Se usar credenciais de usuário do IAM, você poderá especificar a duração ao solicitar as credenciais de segurança temporárias de 15 minutos até a duração máxima de sessão para a função. Para obter mais informações sobre credenciais de segurança temporárias, consulte [Credenciais de segurança temporárias](#) no Manual do usuário do IAM. Para mais informações sobre fazer solicitações, consulte [Fazer solicitações](#).

Note

Se você obtiver as credenciais de segurança temporárias usando suas credenciais de segurança de Conta da AWS, as credenciais de segurança temporárias serão válidas somente por uma hora. É possível especificar a duração da sessão somente ao usar as credenciais de usuário do IAM para solicitar uma sessão.

O seguinte exemplo de Ruby cria um usuário temporário para listar os itens em um bucket especificado por uma hora. Para usar esse exemplo, você deve ter credenciais da AWS com as permissões necessárias para criar novos clientes do AWS Security Token Service (AWS STS) e listar buckets do 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
```

Fazer solicitações usando credenciais temporárias de usuário federado

Solicite credenciais de segurança temporárias e forneça-as às aplicações ou aos usuários federados que precisam de acesso aos seus recursos da AWS. Esta seção fornece exemplos de como usar o AWS SDK para obter credenciais de segurança temporárias para as aplicações ou usuários federados e enviar solicitações autenticadas ao Amazon S3 usando essas credenciais. Para obter uma lista de AWS SDKs disponíveis, consulte [Código de exemplo e bibliotecas](#).

Note

Tanto a Conta da AWS quanto um usuário do IAM podem solicitar credenciais de segurança temporárias para usuários federados. No entanto, para melhor segurança, somente um usuário do IAM com as permissões necessárias deve solicitar essas credenciais temporárias para garantir que o usuário federado obtenha, no máximo, as mesmas permissões do usuário do IAM solicitante. Em algumas aplicações, pode ser apropriado criar um usuário do IAM com permissões específicas com o único propósito de conceder credenciais de segurança temporárias às aplicações e aos usuários federados.

Java

Forneça credenciais de segurança temporárias para as aplicações e os usuários federados para que eles possam enviar solicitações autenticadas para acessar os seus recursos da AWS. Ao solicitar essas credenciais temporárias, forneça um nome de usuário e uma política do IAM que descreva as permissões de recursos que você deseja conceder. Por padrão, a duração da sessão é de uma hora. Defina explicitamente um valor de duração diferente ao solicitar as credenciais de segurança temporárias para aplicativos e usuários federados.

Note

Para garantir segurança adicional ao solicitar credenciais de segurança temporárias para usuários federados e aplicações, recomendamos usar um usuário dedicado do IAM apenas com as permissões de acesso necessárias. O usuário temporário criado nunca pode ter mais permissões que o usuário do IAM que solicitou as credenciais de segurança temporárias. Para obter mais informações, consulte [Perguntas frequentes sobre o AWS Identity and Access Management](#).

Para fornecer credenciais de segurança e enviar solicitações autenticadas para acessar recursos, faça o seguinte:

- Crie uma instância da classe `AWSSecurityTokenServiceClient`.
- Inicie uma sessão chamando o método `getFederationToken()` do cliente Security Token Service (STS). Forneça informações da sessão, incluindo o nome de usuário e uma política do IAM, que você deseja anexar às credenciais temporárias. Forneça uma duração de sessão opcional. Esse método retorna suas credenciais de segurança temporárias.
- Empacote as credenciais de segurança temporárias em uma instância do objeto `BasicSessionCredentials`. Use esse objeto para fornecer as credenciais de segurança temporárias para o cliente do Amazon S3.
- Crie uma instância da classe `AmazonS3Client` usando as credenciais de segurança temporárias. Envie solicitações ao Amazon S3 usando esse cliente. Se você enviar solicitações usando credenciais expiradas, o Amazon S3 retornará um erro.

Example

O exemplo lista chaves no bucket especificado do S3. No exemplo, você obtém credenciais de segurança temporárias para uma sessão de duas horas para o usuário federado e usa as credenciais para enviar solicitações autenticadas ao Amazon S3. Para executar o exemplo, crie um usuário do IAM com a política anexada que permita ao usuário solicitar as credenciais de segurança temporárias e listar os recursos da AWS. A política seguinte faz isso:

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

Para obter mais informações sobre como criar um usuário do IAM, consulte [Criar o primeiro usuário do IAM e grupo de administradores](#) no Manual do usuário do IAM.

Após criar um usuário do IAM e anexar a política anterior, você poderá executar o exemplo a seguir. Consulte instruções sobre como criar e testar uma amostra funcional em [Getting Started](#) no Guia do desenvolvedor do 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

Forneça credenciais de segurança temporárias para as aplicações e os usuários federados para que eles possam enviar solicitações autenticadas para acessar os seus recursos da AWS. Ao solicitar essas credenciais temporárias, forneça um nome de usuário e uma política do IAM que descreva as permissões de recursos que você deseja conceder. Por padrão, a duração de uma sessão é de uma hora. Defina explicitamente um valor de duração diferente ao solicitar as credenciais de segurança temporárias para aplicativos e usuários federados. Para obter informações sobre o envio de solicitações autenticadas, consulte [Fazer solicitações](#).

Note

Ao solicitar credenciais de segurança temporárias para usuários federados e aplicações, a fim de garantir segurança adicional, recomendamos que você use um usuário dedicado do IAM apenas com as permissões de acesso necessárias. O usuário temporário criado nunca pode ter mais permissões que o usuário do IAM que solicitou as credenciais de segurança temporárias. Para obter mais informações, consulte [Perguntas frequentes sobre o AWS Identity and Access Management](#).

Faça o seguinte:

- Crie uma instância de cliente AWS Security Token Service, classe `AmazonSecurityTokenServiceClient`.
- Inicie uma sessão chamando o método `GetFederationToken` do cliente STS. Forneça as informações da sessão, incluindo o nome de usuário e uma política do IAM que você deseja anexar às credenciais temporárias. Como opção, você pode fornecer uma duração de sessão. Esse método retorna suas credenciais de segurança temporárias.
- Empacote as credenciais de segurança temporárias em uma instância do objeto `SessionAWSCredentials`. Use esse objeto para fornecer as credenciais de segurança temporárias para o cliente do Amazon S3.
- Crie uma instância da classe `AmazonS3Client` enviando as credenciais de segurança temporárias. Use este cliente para enviar solicitações ao Amazon S3. Se você enviar solicitações usando credenciais expiradas, o Amazon S3 retornará um erro.

Example

O exemplo do C# a seguir lista as chaves no bucket especificado. No exemplo, você obtém credenciais de segurança temporárias para uma sessão de duas horas para o usuário federado (User1), e usa as credenciais para enviar solicitações autenticadas ao Amazon S3.

- Neste exercício, você criará um usuário do IAM com permissões mínimas. Usando as credenciais desse usuário do IAM, solicite credenciais temporárias para terceiros. Este exemplo lista somente os objetos em um bucket específico. Crie um usuário do IAM com a política a seguir anexada:

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

A política permite que o usuário do IAM solicite credenciais de segurança temporárias e permissão de acesso apenas para listar os recursos da AWS. Para obter mais informações sobre como criar um usuário do IAM, consulte [Criar o primeiro usuário do IAM e grupo de administradores](#) no Manual do usuário do IAM.

- Use as credenciais de segurança do usuário do IAM para testar o exemplo a seguir. O exemplo envia a solicitação autenticada ao Amazon S3 usando credenciais de segurança temporárias. O exemplo especifica a política a seguir ao solicitar credenciais de segurança temporárias para o usuário federado (User1), que restringe o acesso aos objetos de lista em um bucket específico (YourBucketName). É necessário atualizar a política e fornecer um nome de bucket existente.

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

```
}  
]  
}
```

- Example

Atualize o exemplo a seguir e forneça o nome de bucket especificado na política de acesso do usuário federado anterior. Para obter informações sobre como configurar e executar exemplos de código, consulte [Conceitos básicos do AWS SDK for .NET](#) no Guia do desenvolvedor do AWS SDK for .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

Este tópico explica como usar classes da versão 3 do AWS SDK para PHP para solicitar credenciais de segurança temporárias para aplicações e usuários federados e usá-las para acessar recursos armazenados no Amazon S3. Para obter mais informações sobre a API do AWS SDK for Ruby, acesse [AWS SDK for Ruby Version 2](#).

Forneça credenciais de segurança temporárias para as aplicações e os usuários federados para que eles possam enviar solicitações autenticadas para acessar os recursos da AWS. Ao solicitar essas credenciais temporárias, forneça um nome de usuário e uma política do IAM que descreva as permissões de recursos que você deseja conceder. Essas credenciais expiram quando a duração da sessão expira. Por padrão, a duração da sessão é de uma hora. Defina explicitamente um valor de duração diferente ao solicitar as credenciais de

segurança temporárias para aplicativos e usuários federados. Para obter mais informações sobre credenciais de segurança temporárias, consulte [Credenciais de segurança temporárias](#) no Manual do usuário do IAM. Para obter informações sobre como fornecer credenciais de segurança temporárias para aplicativos e usuários federados, consulte [Fazer solicitações](#).

Para garantir segurança adicional ao solicitar credenciais de segurança temporárias para usuários federados e aplicações, recomendamos usar um usuário dedicado do IAM apenas com as permissões de acesso necessárias. O usuário temporário criado nunca pode ter mais permissões que o usuário do IAM que solicitou as credenciais de segurança temporárias. Para obter mais informações sobre federação de identidades, consulte as [Perguntas frequentes do AWS Identity and Access Management](#).

Para obter mais informações sobre a API do AWS SDK for Ruby, acesse [AWS SDK for Ruby Version 2](#).

Example

O exemplo PHP a seguir lista chaves no bucket especificado. No exemplo, você obtém credenciais de segurança temporárias para uma sessão de uma hora para o usuário federado (User1). Depois, use as credenciais de segurança temporárias para enviar solicitações autenticadas ao Amazon S3.

Para maior segurança, ao solicitar credenciais temporárias para terceiros, use as credenciais de segurança de um usuário do IAM com permissões para solicitar credenciais de segurança temporárias. Para garantir que o usuário do IAM conceda apenas as permissões mínimas específicas da aplicação ao usuário federado, você também pode limitar as permissões de acesso desse usuário do IAM. Este exemplo lista somente objetos em um bucket específico. Crie um usuário do IAM com a política a seguir anexada:

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

A política permite que o usuário do IAM solicite credenciais de segurança temporárias e permissão de acesso apenas para listar os recursos da AWS. Para obter mais informações sobre como criar um usuário do IAM, consulte [Criar o primeiro usuário do IAM e grupo de administradores](#) no Manual do usuário do IAM.

Agora é possível usar as credenciais de segurança do usuário do IAM para testar o exemplo a seguir. O exemplo envia uma solicitação autenticada ao Amazon S3 usando credenciais de segurança temporárias. Ao solicitar credenciais de segurança temporárias para o usuário federado (User1), o exemplo especifica a política a seguir, que restringe o acesso aos objetos de lista em um bucket específico. Atualizar a política com o nome do seu bucket.

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

No exemplo a seguir, ao especificar o recurso de política, substitua YourBucketName pelo seu próprio nome de 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

Forneça credenciais de segurança temporárias para as aplicações e os usuários federados para que eles possam enviar solicitações autenticadas para acessar os seus recursos da AWS. Ao solicitar essas credenciais temporárias do serviço do IAM, forneça um nome de usuário e uma política do IAM que descreva as permissões de recurso que você deseja conceder. Por padrão, a duração da sessão é de uma hora. No entanto, se estiver solicitando credenciais temporárias usando credenciais de usuário do IAM, defina explicitamente um valor de duração diferente ao solicitar as credenciais de segurança temporárias para usuários federados e aplicações.

Para obter informações sobre credenciais de segurança temporárias para aplicativos e usuários federados, consulte [Fazer solicitações](#).

 Note

Para garantir segurança adicional ao solicitar credenciais de segurança temporárias para usuários federados e aplicações, use um usuário dedicado do IAM apenas com as permissões de acesso necessárias. O usuário temporário criado nunca pode ter mais permissões que o usuário do IAM que solicitou as credenciais de segurança temporárias. Para obter mais informações, consulte [Perguntas frequentes sobre o AWS Identity and Access Management](#).

Example

O exemplo de código Ruby a seguir permite que um usuário federado com um conjunto limitado de permissões listar as chaves no bucket específico.

```
# 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__

```

Fazer solicitações usando a API REST

Esta seção contém informações sobre como fazer solicitações para endpoints do Amazon S3 usando a API REST. Para obter uma lista dos endpoints do Amazon S3, consulte [Regiões e endpoints](#) na Referência geral da AWS.

Criar nomes de host do S3 para solicitações de API REST

Endpoints do Amazon S3 seguem a estrutura mostrada abaixo:

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

Os endpoints de pontos de acesso do Amazon S3 e endpoints de pilha dupla também seguem a estrutura padrão:

- Pontos de acesso do Amazon S - s3-accesspoint.*Region*.amazonaws.com
- Pilha dupla - s3.dualstack.*Region*.amazonaws.com

Para obter uma lista completa de regiões e endpoints do Amazon S3, consulte [Endpoints e cotas do Amazon S3](#) na Referência geral da Amazon Web Services.

Solicitações de estilo de hospedagem virtual e caminho

Ao fazer solicitações usando a API REST, use URIs no estilo de hospedagem virtual ou de caminho para os endpoints do Amazon S3. Para obter mais informações, consulte [Solicitações em estilo de caminho](#).

Example Solicitação no estilo de hospedagem virtual

Veja a seguir um exemplo de uma solicitação de estilo de hospedagem virtual para excluir o arquivo `puppy.jpg` do bucket `examplebucket` na região Oeste dos EUA (Oregon). Para obter mais informações sobre solicitações de estilo hospedadas virtualmente, consulte [Solicitações em estilo de caminho](#).

```
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 Solicitação no estilo de caminho

Veja a seguir um exemplo com a versão no estilo de caminho da mesma solicitação.

```
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
```

Atualmente, o Amazon S3 é compatível com o acesso a URLs no estilo de hospedagem virtual e no estilo de caminho em todas as Regiões da AWS. No entanto, os URLs no estilo de caminho serão descontinuados no futuro. Para obter mais informações, consulte a observação Importante a seguir.

Para obter mais informações sobre solicitações de estilo de caminho, consulte [Solicitações em estilo de caminho](#).

Important

Atualização (23 de setembro de 2020): para garantir que os clientes tenham o tempo necessário para fazer a transição para URLs no estilo de hospedagem virtual, decidimos

postergar a desativação de URLs no estilo de caminho. Para obter mais informações, consulte [Amazon S3 Path Deprecation Plan – The Rest of the Story](#) no Blog de notícias da AWS.

Fazer solicitações para endpoints de pilha dupla usando a API REST

Ao usar a API REST, acesse um endpoint de pilha dupla diretamente usando um nome de endpoint (URI) de hospedagem virtual ou de caminho. Todos os nomes de endpoint de pilha dupla do Amazon S3 incluem a região. Diferente dos endpoints somente-IPv4 padrão, os endpoints de hospedagem virtual e de caminho usam nomes de endpoint específicos para a região.

Example Solicitação de endpoint de pilha dupla do estilo de hospedagem virtual

Conforme mostrado no exemplo a seguir, use um endpoint no estilo de hospedagem virtual na solicitação REST que recupera o objeto do bucket chamado `puppy.jpg` do bucket `examplebucket` na região Oeste dos EUA (Oregon).

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

Example Solicitação de endpoint de pilha dupla no estilo de caminho

Ou use um endpoint no de estilo caminho na solicitação, conforme mostrado no exemplo a seguir.

```
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 obter mais informações sobre endpoints de pilha dupla, consulte [Usar endpoints de pilha dupla do Amazon S3](#).

Para obter mais informações sobre como fazer solicitações usando a API REST, consulte os tópicos a seguir.

Tópicos

- [Redirecionamento de solicitação e a API REST](#)
- [Roteamento de solicitação](#)

Redirecionamento de solicitação e a API REST

Tópicos

- [Redirecionamentos e agentes de usuário de HTTP](#)
- [Redirecionamentos e 100-Continue](#)
- [Exemplo de redirecionamento](#)

Esta seção descreve como processar redirecionamentos HTTP usando a API REST do Amazon S3. Para obter informações gerais sobre redirecionamentos do Amazon S3, consulte [Fazer solicitações](#) na Referência da API do Amazon Simple Storage Service.

Redirecionamentos e agentes de usuário de HTTP

Os programas que usam a API REST do Amazon S3 devem processar os redirecionamentos na camada de aplicação ou na camada HTTP. Muitas bibliotecas de clientes HTTP e agentes de usuário podem ser configurados para processar redirecionamentos de modo correto e automático; contudo, muitas outras têm implementações de redirecionamento incorretas ou incompletas.

Antes de confiar em uma biblioteca para atender aos requisitos de redirecionamento, teste os seguintes casos:

- Verifique se todos os cabeçalhos de solicitações HTTP estão incluídos corretamente na solicitação redirecionada (a segunda solicitação depois de receber um redirecionamento), incluindo os padrões HTTP como Autorização e Data.
- Verifique se redirecionamentos não GET, como PUT e DELETE, funcionam corretamente.
- Verifique se grandes solicitações PUT seguem o redirecionamento corretamente.
- Verifique se as solicitações PUT seguem redirecionamentos corretamente se a resposta 100-continue demorar muito tempo para chegar.

Os agentes de usuário HTTP que se conformam estritamente a RFC 2616 podem exigir confirmação explícita antes de seguir um redirecionamento quando o método de solicitação HTTP não

for GET nem HEAD. Em geral, é seguro seguir redirecionamentos gerados pelo Amazon S3 automaticamente, pois o sistema emitirá redirecionamentos somente para hosts no domínio amazonaws.com e o efeito da solicitação redirecionada será igual ao da solicitação original.

Redirecionamentos e 100-Continue

Para simplificar o processamento de redirecionamentos, aumentar a eficiência e evitar custos associados com o envio de um corpo de solicitação redirecionado duas vezes, configure seu aplicativo para usar 100-continues para operações PUT. Quando seu aplicativo usa 100-continue, ele não envia o corpo da solicitação até receber uma confirmação. Se a mensagem for rejeitada com base nos cabeçalhos, o corpo da mensagem não será enviado. Para obter mais informações sobre 100-continue, acesse [RFC 2616 Section 8.2.3](#).

Note

De acordo com o RFC 2616, ao usar Expect: Continue com um servidor HTTP desconhecido, você não deve esperar um período indefinido antes de enviar o corpo da solicitação. Isso porque alguns servidores HTTP não reconhecem 100-continue. Porém, o Amazon S3 reconhecerá se sua solicitação contiver um Expect: Continue e responderá com um status 100-continue temporário ou um código de status final. Além disso, um erro de não redirecionamento ocorrerá depois de receber o 100-continue temporário. Isto ajudará a evitar que você receba uma resposta de redirecionamento enquanto ainda estiver escrevendo o corpo da solicitação.

Exemplo de redirecionamento

Esta seção fornece um exemplo de interação de servidor cliente usando redirecionamento HTTP e 100-continue.

A seguir está um exemplo PUT para o 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
```

O Amazon S3 retorna o seguinte:

```
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>
```

O cliente segue a resposta de redirecionamento e emite uma nova solicitação ao endpoint temporário `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
```

O Amazon S3 retorna um 100-continue indicando que o cliente deve continuar com o envio de corpo da solicitação.

```
HTTP/1.1 100 Continue
```

O cliente envia o corpo da solicitação.

```
ha ha\n
```

O Amazon S3 retorna a resposta final.


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

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

Roteamento de solicitação

Programas que fazem solicitações em buckets criados usando a API [CreateBucket](#) e incluem uma [CreateBucketConfiguration](#) devem ser compatíveis com redirecionamentos. Além disso, alguns clientes que não respeitam TTLs DNS podem encontrar problemas.

Esta seção descreve problemas de roteamento e DNS a serem considerados ao projetar seu serviço ou aplicação para uso com o Amazon S3.

Redirecionamento de solicitação e a API REST

O Amazon S3 usa o Domain Name System (DNS) para rotear solicitações para instalações capazes de processá-las. Esse sistema funciona com eficiência, mas podem ocorrer erros de roteamento temporários. Se uma solicitação chega na localização errada do Amazon S3, o Amazon S3 responde com um redirecionamento temporário pedindo que o solicitante reenvie a solicitação para um novo endpoint. Se uma solicitação é formada de maneira incorreta, o Amazon S3 usa redirecionamentos constantes para fornecer direções sobre como executar a solicitação corretamente.

Important

Para usar esse recurso, você deve ter uma aplicação que possa lidar com respostas de redirecionamento do Amazon S3. A única exceção é para aplicativos que funcionam exclusivamente com buckets criados sem `<CreateBucketConfiguration>`. Para obter mais informações sobre restrições de local, consulte [Acesso e listagem de um bucket do Amazon S3](#).

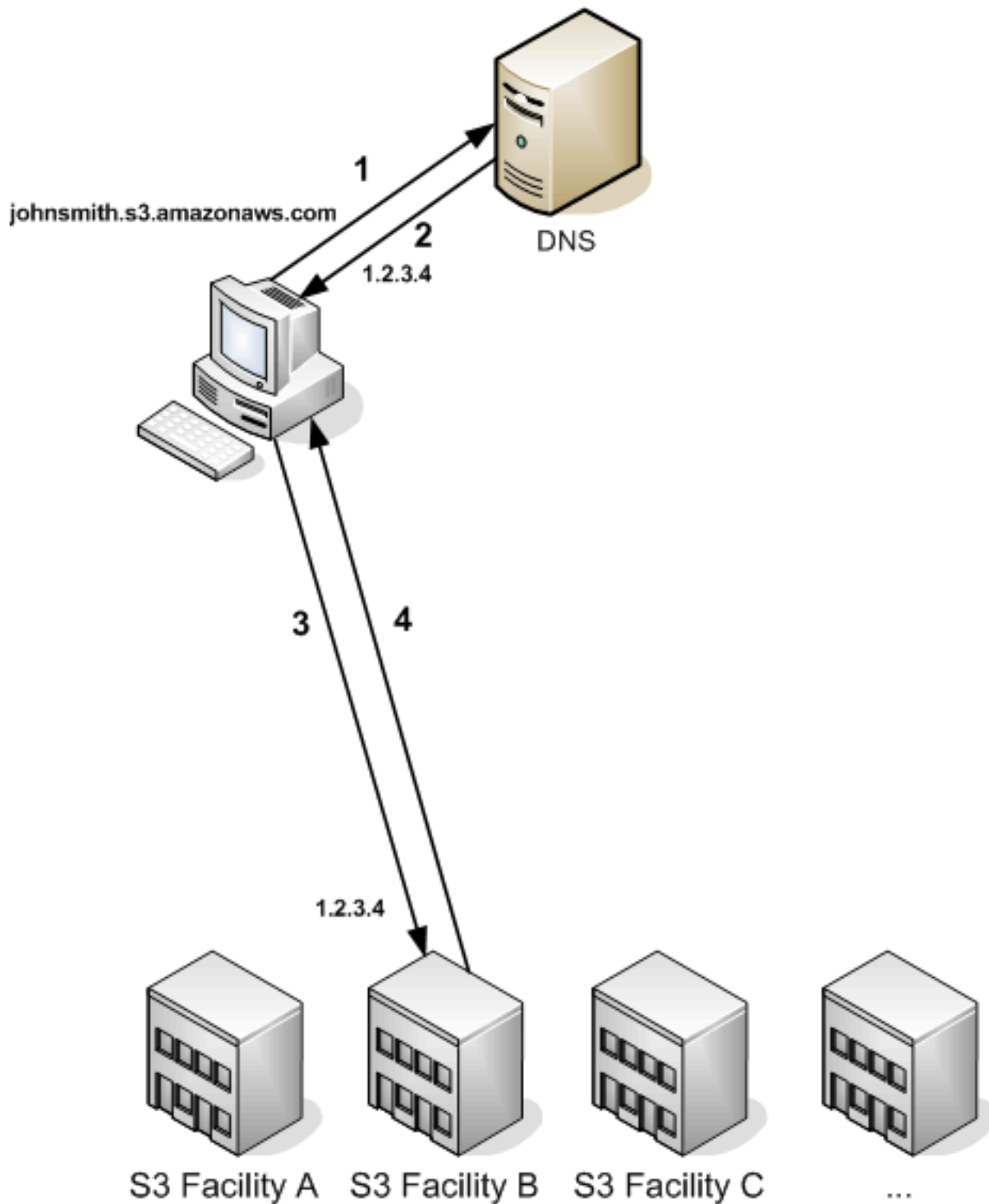
Para todas as regiões lançadas após 20 de março de 2019, se uma solicitação chega no local errado do Amazon S3, o Amazon S3 retorna um erro de solicitação incorreta HTTP 400. Para obter mais informações sobre como habilitar ou desabilitar uma Região da AWS, consulte [Regiões da AWS e endpoints](#) na Referência geral da AWS.

Tópicos

- [Roteamento de DNS](#)
- [Redirecionamento de solicitação temporário](#)
- [Redirecionamento permanente de solicitação](#)
- [Exemplos de redirecionamento de solicitação](#)

Roteamento de DNS

O roteamento de DNS encaminha solicitações para instalações apropriadas do Amazon S3. A figura e o procedimento a seguir mostram um exemplo de roteamento de DNS.



Etapas de solicitação de roteamento DNS

1. O cliente faz uma solicitação de DNS para obter um objeto armazenado no Amazon S3.

2. O cliente recebe um ou mais endereços IP para instalações capazes de processar a solicitação. Neste exemplo, o endereço IP é para a instalação B.
3. O cliente faz uma solicitação à instalação B do Amazon S3.
4. A instalação B retorna uma cópia do objeto ao cliente.

Redirecionamento de solicitação temporário

Um redirecionamento temporário é um tipo de resposta de erro que indica que o solicitante deve reenviar a solicitação para um endpoint diferente. Devido à natureza distribuída do Amazon S3, as solicitações podem ser roteadas temporariamente para a instalação errada. É mais provável que isso aconteça imediatamente após a criação ou exclusão de buckets.

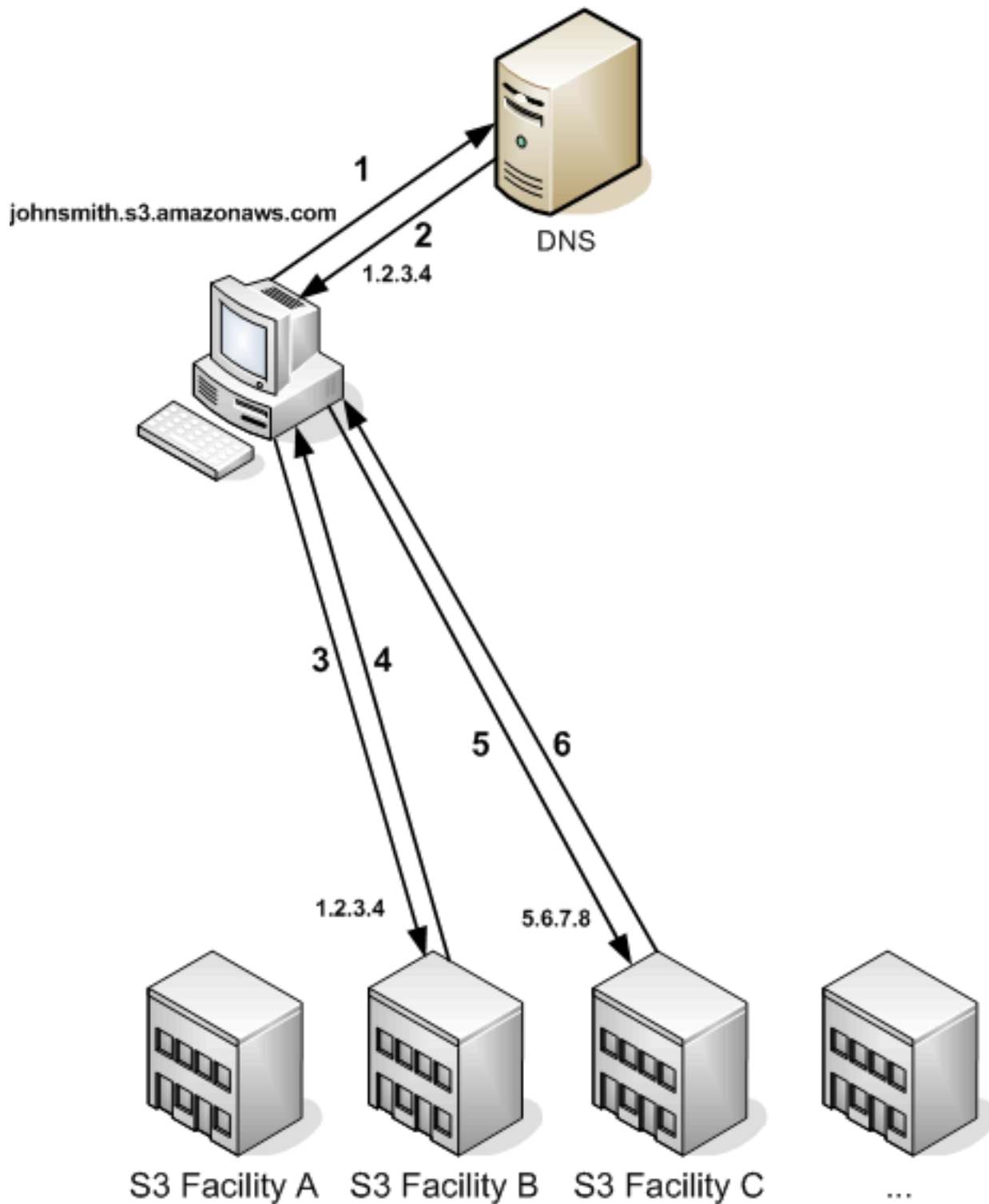
Por exemplo, se você criar um bucket novo e, em seguida, fizer uma solicitação a esse bucket, é possível que receba um redirecionamento temporário dependendo da restrição de localização do bucket. Se você criou o bucket na Região da AWS Leste dos EUA (Norte da Virgínia), não verá o redirecionamento, já que esse também é o endpoint padrão do Amazon S3.

No entanto, se o bucket for criado em qualquer outra região, todas as solicitações feitas ao bucket vão para o endpoint padrão enquanto a entrada do DNS do bucket é propagada. O endpoint padrão redireciona a solicitação para o endpoint correto com uma resposta HTTP 302. Os redirecionamentos temporários contêm um URI para a instalação correta, que pode ser usado para reenviar a solicitação imediatamente.

Important

Não reutilize um endpoint fornecido por uma resposta de redirecionamento anterior. Ele pode parecer funcionar (até mesmo durante longos períodos), mas pode fornecer resultados imprevisíveis e eventualmente falhará sem aviso.

A figura e o procedimento a seguir mostram um exemplo de um redirecionamento temporário.



Etapas de redirecionamento de solicitação temporário

1. O cliente faz uma solicitação de DNS para obter um objeto armazenado no Amazon S3.
2. O cliente recebe um ou mais endereços IP para instalações capazes de processar a solicitação.

3. O cliente faz uma solicitação à instalação B do Amazon S3.
4. A instalação B retorna um redirecionamento indicando que o objeto está disponível na localização C.
5. O cliente reenvia a solicitação para a instalação C.
6. A instalação C retorna uma cópia do objeto.

Redirecionamento permanente de solicitação

Um redirecionamento permanente indica que a solicitação abordou um recurso de maneira inapropriada. Por exemplo, redirecionamentos permanentes ocorrem se você usar uma solicitação no estilo de caminho para acessar um bucket criado com `<CreateBucketConfiguration>`. Para obter mais informações, consulte [Acesso e listagem de um bucket do Amazon S3](#).

Para ajudar a encontrar esses erros durante o desenvolvimento, esse tipo de redirecionamento não contém um cabeçalho HTTP de localização que permite o acompanhamento automático da solicitação para a localização correta. Consulte o documento de erros XML resultante para obter ajuda no uso do endpoint correto do Amazon S3.

Exemplos de redirecionamento de solicitação

Veja a seguir exemplos de respostas de redirecionamento de solicitação temporário.

API REST de resposta de redirecionamento temporário

```
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>
```

API SOAP de resposta de redirecionamento temporário

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

```
<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>
```

Considerações de DNS

Um dos requisitos de design do Amazon S3 é uma disponibilidade extremamente alta. Uma das maneiras de cumprirmos esse requisito é atualizando os endereços IP associados com o endpoint do Amazon S3 no DNS, conforme necessário. Essas alterações são refletidas automaticamente em clientes com tempo de vida curto, mas não em alguns clientes com tempo de vida longo. Os clientes com tempo de vida longo precisarão realizar uma ação especial para resolver de novo o endpoint do Amazon S3 periodicamente para se beneficiar com essas alterações. Para obter mais informações sobre Virtual Machines (VMs – Máquinas virtuais), consulte o seguinte:

- Para Java, por padrão, o JVM do Sun armazena em cache as pesquisas de DNS para sempre; acesse a seção "InetAddress Caching" da [documentação do InetAddress](#) para obter informações sobre como alterar esse comportamento.
- Para PHP, a VM PHP persistente que é executada nas configurações de implantação mais populares armazena em cache as pesquisas do DNS até a VM ser reiniciada. Acesse [os docs getHostByName PHP](#).

Desenvolvimento com o Amazon S3 usando a AWS CLI

Os comandos da AWS CLI do Amazon S3 são organizados em conjuntos diferentes na Referência de Comandos da AWS CLI e cada um tem seus próprios comandos disponíveis. Se você não encontrar o comando que está procurando em um conjunto, verifique um dos outros conjuntos. Os diferentes conjuntos são os seguintes:

- `s3`: descreve comandos de alto nível para trabalhar com objetos e buckets. Isso inclui ações de copiar, listar e mover. Para obter uma lista completa dos comandos desse conjunto, consulte [s3](#) na Referência de Comandos da AWS CLI.
- `s3api`: descreve comandos de baixo nível que gerenciam recursos do S3, como buckets, objetos, sessões e uploads fragmentados. Esses comandos correspondem às operações da [API do Amazon S3](#). Para obter uma lista completa dos comandos da CLI desse conjunto, consulte [s3api](#) na Referência de Comandos da AWS CLI.
- `s3control`: descreve comandos de baixo nível que gerenciam todos os outros recursos do S3, como Concessão de Acesso, grupos da Lente de Armazenamento e buckets do Amazon S3 em Outposts. Esses comandos correspondem às operações da API de [Controle do Amazon S3](#). Para obter uma lista completa dos comandos da CLI desse conjunto, consulte [s3control](#) na Referência de Comandos da AWS CLI.

Se você estiver com problemas para configurar a AWS CLI, consulte [Como resolvo o erro “Não é possível localizar credenciais” no Amazon S3](#) na Central de Conhecimento AWS re:Post.

Note

Os serviços na AWS, como o Amazon S3, exigem o fornecimento de credenciais quando acessados. Esse serviço pode determinar se você tem permissões para acessar os respectivos recursos próprios. O console requer sua senha. Você pode criar chaves de acesso para a Conta da AWS para acessar a AWS CLI ou a API. No entanto, não recomendamos que você acesse a AWS usando as credenciais de sua Conta da AWS. Em vez disso, recomendamos usar o AWS Identity and Access Management (IAM). Crie um usuário do IAM, adicione o usuário a um grupo do IAM com permissões administrativas e, em seguida, conceda permissões administrativas ao usuário do IAM criado. Depois, você pode acessar a AWS usando um URL especial e as credenciais desse usuário do IAM. Para obter instruções, acesse [Criar o primeiro usuário do IAM e o grupo de administradores](#) no Manual do usuário do IAM.

Saiba mais sobre o AWS CLI

Para saber mais sobre a AWS, consulte os seguintes recursos:

- [AWS Command Line Interface](#)
- [AWS Command Line Interface Guia do usuário da versão 2](#)

Desenvolvimento com o Amazon S3 usando AWS SDKs

AWSOs kits de desenvolvimento de software (SDKs) da estão disponíveis para muitas linguagens de programação populares. Cada SDK fornece uma API, exemplos de código e documentação que permitem que os desenvolvedores criem facilmente aplicações em seu idioma de preferência.

Note

É possível usar o AWS Amplify para o desenvolvimento full-stack de aplicações web e aplicativos móveis de ponta a ponta. O Amplify Storage integra perfeitamente os recursos de armazenamento e gerenciamento de arquivos em aplicações frontend web e aplicativos móveis, desenvolvidos com base no Amazon S3. Consulte mais informações em [Storage](#) no guia do usuário do Amplify.

Documentação do SDK	Exemplos de código
AWS SDK para C++	AWS SDK para C++ Exemplos de código
AWS CLI	AWS CLI Exemplos de código da
AWS SDK para Go	AWS SDK para Go Exemplos de código da
AWS SDK para Java	AWS SDK para Java Exemplos de código da
AWS SDK para JavaScript	AWS SDK para JavaScript Exemplos de código da
AWS SDK para Kotlin	AWS SDK para Kotlin Exemplos de código da
AWS SDK para .NET	AWS SDK para .NET Exemplos de código da

Documentação do SDK	Exemplos de código
AWS SDK para PHP	AWS SDK para PHP Exemplos de código da
Ferramentas da AWS para PowerShell	Tools for PowerShell code examples
AWS SDK para Python (Boto3)	AWS SDK para Python (Boto3) Exemplos de código
AWS SDK para Ruby	AWS SDK para Ruby Exemplos de código da
AWS SDK para Rust	AWS SDK para Rust Exemplos de código da
AWS SDK para SAP ABAP	AWS SDK para SAP ABAP Exemplos de código da
AWS SDK para Swift	AWS SDK para Swift Exemplos de código do

Para obter exemplos específicos, consulte [Exemplos de código do Amazon S3 usando SDKs da AWS](#).

Interfaces de programação do SDK

Cada AWS SDK fornece uma ou mais interfaces programáticas para trabalhar com o Amazon S3. Cada SDK fornece uma interface de baixo nível para o Amazon S3, com métodos que se assemelham a operações de API. Alguns SDKs fornecem interfaces de alto nível para o Amazon S3, que são abstrações com o objetivo de simplificar casos de uso comuns.

Por exemplo, ao realizar um upload fragmentado usando as operações de API de baixo nível, você usa uma operação para iniciar o upload, outra operação para fazer o upload das partes e uma operação final para concluir o upload. Uma operação de API de upload fragmentado de alto nível permite que você faça todas as operações necessárias para o upload em uma única chamada de API. Para ver exemplos, consulte [Carregamento de um objeto usando o upload fragmentado](#) no Guia do usuário do Amazon S3.

As operações de API de baixo nível permitem maior controle sobre o upload. Recomendamos usar as operações de API de baixo nível se quiser pausar e retomar uploads, variar os tamanhos das partes durante o upload ou começar uploads quando você não sabe o tamanho dos dados com antecedência.

Como especificar a versão da assinatura na autenticação de solicitações

O Amazon S3 oferece suporte apenas ao AWS Signature Version 4 na maioria das Regiões da AWS. No entanto, em algumas das Regiões da AWS mais antigas, o Amazon S3 é compatível com o Signature Version 4 e o Signature Version 2. No entanto, o Signature versão 2 está sendo desativado (defasado). Para obter mais informações sobre o fim do suporte ao Signature versão 2, consulte [AWS Signature Version 2 desativado \(defasado\) para o Amazon S3](#).

Para obter uma lista de todas as regiões do Amazon S3 e das versões do Signature com as quais elas são compatíveis, consulte [Regiões e endpoints](#) na Referência geral da AWS.

Para todas as Regiões da AWS, por padrão, os AWS SDKs usam o Signature Version 4 para autenticar solicitações. Ao usar os AWS SDKs liberados antes de maio de 2016, talvez seja necessário solicitar o Signature Versão 4 conforme mostrado na tabela a seguir.

SDK	Solicitar o Signature versão 4 para autenticação de solicitações
AWS CLI	Para o perfil padrão, execute o comando a seguir: <pre>\$ aws configure set default.s3.signature_version s3v4</pre> Para um perfil personalizado, execute o comando a seguir: <pre>\$ aws configure set profile.your_profile_name.s3.signature_version s3v4</pre>
SDK do Java	Adicione o seguinte ao código: <pre>System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");</pre> Ou, na linha de comando, especifique o seguinte: <pre>-Dcom.amazonaws.services.s3.enableV4</pre>
SDK do JavaScript	Defina o parâmetro <code>signatureVersion</code> como <code>v4</code> ao criar o cliente:

SDK	Solicitar o Signature versão 4 para autenticação de solicitações <pre>var s3 = new AWS.S3({signatureVersion: 'v4'});</pre>
SDK do PHP	Defina o parâmetro <code>signature</code> como <code>v4</code> ao criar o cliente de serviço do Amazon S3 para o SDK v2 do PHP: <pre><?php \$client = S3Client::factory(['region' => 'YOUR-REGION', 'version' => 'latest', 'signature' => 'v4']);</pre> Ao usar o SDK v3 do PHP, defina o parâmetro <code>signature_version</code> como <code>v4</code> durante a criação do cliente de serviço do Amazon S3: <pre><?php \$s3 = new Aws\S3\S3Client(['version' => '2006-03-01', 'region' => 'YOUR-REGION', 'signature_version' => 'v4']);</pre>
SDK do Python-Boto	Especifique o seguinte no arquivo de configuração boto padrão: <pre>[s3] use-sigv4 = True</pre>

SDK	Solicitar o Signature versão 4 para autenticação de solicitações
SDK do Ruby	SDK do Ruby - versão 1: defina o parâmetro <code>s3_signature_version</code> como <code>v4</code> ao criar o cliente: <pre>s3 = AWS::S3::Client.new(:s3_signature_version => :v4)</pre> SDK do Ruby - versão 3: defina o parâmetro <code>signature_version</code> como <code>v4</code> ao criar o cliente: <pre>s3 = Aws::S3::Client.new(signature_version: 'v4')</pre>
SDK do .NET	Adicione o código a seguir antes de criar o cliente do Amazon S3: <pre>AWSConfigsS3.UseSignatureVersion4 = true;</pre> Ou adicione o seguinte ao arquivo config: <pre><appSettings> <add key="AWS.S3.UseSignatureVersion4" value="true" /> </appSettings></pre>

AWS Signature Version 2 desativado (defasado) para o Amazon S3

O Signature versão 2 está sendo desativado (defasado) no Amazon S3. O Amazon S3 só aceitará solicitações de API assinadas com o Signature versão 4.

Esta seção apresenta respostas às perguntas mais comuns relacionadas ao fim do suporte ao Signature versão 2.

O que é o Signature versão 2/4 e o que “assinar solicitações” quer dizer?

O processo de assinatura com as versões 2 ou 4 do Signature é utilizado para autenticar solicitações de API do Amazon S3. A assinatura de solicitações permite que o Amazon S3 identifique quem está enviando uma solicitação e a protege contra agentes mal intencionados.

Para obter mais informações sobre como assinar solicitações da AWS, consulte [Assinar solicitações de API da AWS](#) na Referência geral da AWS.

Qual atualização está sendo feita?

No momento, oferecemos suporte a solicitações da API do Amazon S3 assinadas usando os processos das versões 2 e 4 do Signature. Depois disso, o Amazon S3 só aceitará solicitações assinadas com o Signature versão 4.

Para obter mais informações sobre a assinatura de solicitações da AWS, consulte [Alterações no Signature versão 4](#) na Referência geral da AWS.

Por que esta atualização está sendo feita?

O Signature versão 4 oferece maior segurança porque usa uma chave de assinatura em vez de sua chave de acesso secreta. No momento, o Signature Version 4 é compatível com todas as Regiões da AWS. O Signature Version 2 só é compatível com as regiões lançadas antes de janeiro de 2014. Essa atualização nos permite oferecer uma experiência mais consistente em todas as regiões.

Como posso ter certeza de que estou usando o Signature versão 4 e quais atualizações são necessárias?

Normalmente, a versão utilizada para assinar suas solicitações é definida pela ferramenta ou pelo SDK no lado do cliente. Por padrão, as versões mais recentes dos AWS SDKs usam o Signature Version 4. Para software de terceiro, entre em contato com a equipe de suporte dele para confirmar a versão necessária. Para enviar chamadas REST diretamente para o Amazon S3, modifique a aplicação para usar o processo de assinatura do Signature versão 4.

Para obter informações sobre qual versão dos AWS SDKs deve ser usada depois da transição para o Signature Version 4, consulte [Migração do Signature versão 2 para o Signature versão 4](#).

Para obter informações sobre como usar o Signature Version 4 com a API REST do Amazon S3, consulte [Autenticação de solicitações \(AWS Signature Version 4\)](#) na Referência da API do Amazon Simple Storage Service.

O que acontecerá se eu não atualizar?

As solicitações assinadas com o Signature versão 2 feitas após essa data não autenticarão com o Amazon S3. Os solicitantes verão erros com a mensagem de que a solicitação deve ser assinada com o Signature versão 4.

Devo fazer alterações mesmo se estiver usando um pre-signed URL que exige a assinatura por mais de sete dias?

Se você estiver usando um pre-signed URL que exige a assinatura por mais de sete dias, não é necessário fazer nada. Você pode continuar usando o AWS Signature Version 2 para assinar e autenticar URLs pré-assinados. Vamos fazer um acompanhamento e fornecer mais detalhes sobre como migrar para o Signature versão 4 para pre-signed URL.

Mais informações

- Para obter mais informações sobre como usar o Signature versão 4, consulte [Assinatura de solicitações de API da AWS](#).
- Visualize a lista de alterações entre o Signature versão 2 e o Signature versão 4 em [Alterações no Signature versão 4](#).
- Consulte a publicação [AWS Signature Version 4 como substituição ao AWS Signature Version 2 para assinar solicitações da API do Amazon S3](#) nos fóruns da AWS.
- Em caso de dúvidas ou preocupações, entre em contato conosco em [Suporte](#).

Migração do Signature versão 2 para o Signature versão 4

Se estiver usando o Signature versão 2 para autenticar solicitações de API do Amazon S3, migre para o Signature versão 4. Não haverá mais suporte para o Signature versão 2, conforme descrito em [AWS Signature Version 2 desativado \(defasado\) para o Amazon S3](#).

Para obter informações sobre como usar o Signature Version 4 com a API REST do Amazon S3, consulte [Autenticação de solicitações \(AWS Signature Version 4\)](#) na Referência da API do Amazon Simple Storage Service.

A tabela a seguir contém os SDKs que exigem a utilização do Signature versão 4 (SigV4). Se você utiliza URLs pré-assinados com os SDKs AWS Java, JavaScript (Node.js) ou Python (Boto/CLI), deverá definir a Região da AWS correta e o Signature versão 4 na configuração do cliente. Para obter mais informações sobre como definir o SigV4 na configuração do cliente, consulte [Como especificar a versão da assinatura na autenticação de solicitações](#).

Se você usa este SDK/ produto	Atualize para esta versão do SDK	Alteração de código necessária para o cliente usar o Sigv4?	Link para documentação do SDK
AWS SDK para Java v1	Atualize para Java 1.11.201+ ou v2.	Sim	Como especificar a versão da assinatura na autenticação de solicitações
AWS SDK para Java v2 da	Não é necessário atualizar os SDKs.	Não	AWS SDK para Java
AWS SDK para .NET v1	Atualize para o 3.1.10 ou versão superior.	Sim	AWS SDK para .NET
AWS SDK para .NET v2	Atualize para o 3.1.10 ou versão superior.	Não	AWS SDK para .NET v2
AWS SDK para .NET v3	Atualize para a versão 3.3.0.0 ou superior.	Sim	AWS SDK para .NET v3
AWS SDK para JavaScript v1	Atualize para o 2.68.0 ou versão superior.	Sim	AWS SDK para JavaScript

Se você usa este SDK/ produto	Atualize para esta versão do SDK	Alteração de código necessária para o cliente usar o Sigv4?	Link para documentação do SDK
AWS SDK para JavaScript v2	Atualize para o 2.68.0 ou versão superior.	Sim	AWS SDK para JavaScript
AWS SDK para JavaScript v3	Nenhuma outra ação é necessária no momento. Atualize para a versão principal V3 no terceiro trimestre de 2019.	Não	AWS SDK para JavaScript
AWS SDK para PHP v1	Recomenda-se atualizar para a versão mais recente do PHP ou, ao menos, para a versão 2.7.4 com o parâmetro de assinatura definido como v4 na configuração do cliente do S3.	Sim	AWS SDK para PHP

Se você usa este SDK/ produto	Atualize para esta versão do SDK	Alteração de código necessária para o cliente usar o Sigv4?	Link para documentação do SDK
AWS SDK para PHP v2	Recomenda-se atualizar para a versão mais recente do PHP ou, ao menos, para a versão 2.7.4 com o parâmetro de assinatura definido como v4 na configuração do cliente do S3.	Não	AWS SDK para PHP
AWS SDK para PHP v3	Não é necessário atualizar os SDKs.	Não	AWS SDK para PHP
Boto2	Atualize para Boto2 v2.49.0.	Sim	Atualização do Boto
Boto3	Atualize para 1.5.71 (Botocore), 1.4.6 (Boto3).	Sim	Boto 3 - AWS SDK for Python

Se você usa este SDK/ produto	Atualize para esta versão do SDK	Alteração de código necessária para o cliente usar o Sigv4?	Link para documentação do SDK
AWS CLI	Atualize para 1.11.108.	Sim	AWS Command Line Interface
AWS CLI v2 (pré-visualização)	Não é necessário atualizar os SDKs.	Não	AWS Command Line Interface versão 2
AWS SDK para Ruby v1	Atualize para Ruby V3.	Sim	Ruby V3 for AWS
AWS SDK para Ruby v2	Atualize para Ruby V3.	Sim	Ruby V3 for AWS
AWS SDK para Ruby v3	Não é necessário atualizar os SDKs.	Não	Ruby V3 for AWS
Go	Não é necessário atualizar os SDKs.	Não	AWS SDK para Go
C++	Não é necessário atualizar os SDKs.	Não	AWS SDK para C++

AWS Tools for Windows PowerShell ou AWS Tools for PowerShell Core

Se você estiver usando versões de módulo anteriores à 3.3.0.0, deverá atualizar para a 3.3.0.0.

Para obter informações sobre a versão, use o cmdlet do Get-Module:

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

Para atualizar para a versão 3.3.0.0, use o cmdlet do Update-Module:

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

É possível enviar o tráfego do Signature versão 2 para pre-signed URLs válidas por mais de sete dias.

Obter IDs de solicitação do Amazon S3 para o AWS Support

Sempre que você entrar em contato com o AWS Support devido a erros ou comportamento inesperado no Amazon S3, é necessário fornecer os IDs das solicitações associadas à ação com falha. O AWS Support usa esses IDs das solicitações para ajudar a resolver os problemas que está enfrentando.

Os IDs da solicitação são fornecidos em pares, são retornados em cada resposta que o Amazon S3 processa (mesmo errôneas) e podem ser acessados por meio de logs detalhados. Há vários métodos comuns para obter os IDs das solicitações, incluindo logs de acesso do servidor do S3 e eventos ou eventos de dados do AWS CloudTrail.

Depois de recuperar esses logs, copie e mantenha esses dois valores, pois você precisará deles ao entrar em contato com o AWS Support. Para obter informações sobre como entrar em contato com o AWS Support, consulte [Entrar em contato com a AWS](#) ou a [Documentação do AWS Support](#).

Tópicos

- [Usar a AWS CLI para obter IDs de solicitação](#)
- [Usar o Windows PowerShell para obter IDs de solicitação](#)
- [Usar eventos de dados do AWS CloudTrail para obter IDs de solicitação](#)
- [Usar o registro de acesso ao servidor do S3 para obter IDs de solicitação](#)

- [Usar HTTP para obter IDs de solicitação](#)
- [Usar um navegador da web para obter IDs de solicitação](#)
- [Usar os AWS SDKs para obter IDs de solicitação](#)

Usar a AWS CLI para obter IDs de solicitação

Para obter os IDs de solicitação ao usar a AWS Command Line Interface (AWS CLI), adicione `--debug` ao comando. Por exemplo, você deve ver `x-amz-request-id` e `x-amz-id-2` no log de depuração conforme mostrado abaixo:

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

Usar o Windows PowerShell para obter IDs de solicitação

Para obter informações sobre como recuperar logs com o Windows PowerShell, consulte [Log de respostas no AWS Tools for Windows PowerShell](#) no blog de ferramentas de desenvolvimento da AWS.

Usar eventos de dados do AWS CloudTrail para obter IDs de solicitação

Um bucket do Amazon S3 que é configurado com eventos de dados do CloudTrail para registrar operações de API no nível de objeto do S3 fornece informações detalhadas sobre as ações realizadas por um usuário, um perfil ou um serviço da AWS no Amazon S3. É possível identificar os IDs de solicitação do S3 consultando os eventos do CloudTrail com o Athena.

Para obter mais informações, consulte [Identificação de solicitações do Amazon S3 usando o CloudTrail](#) no Guia do usuário do Amazon Simple Storage Service.

Usar o registro de acesso ao servidor do S3 para obter IDs de solicitação

Um bucket do Amazon S3 configurado para o registro de acesso ao servidor do S3 fornece registros detalhados para cada solicitação feita ao bucket. É possível identificar os IDs de solicitação do S3 consultando os logs de acesso ao servidor usando o Athena.

Para obter mais informações, consulte [Consulta de logs de acesso para solicitações usando o Amazon Athena](#) no Guia do usuário do Amazon Simple Storage Service.

Usar HTTP para obter IDs de solicitação

Você pode obter os IDs da solicitação, `x-amz-request-id` e `x-amz-id-2`, registrando os bits de uma solicitação HTTP antes que ela chegue ao aplicativo de destino. Há várias ferramentas de terceiros que podem ser usadas para recuperar logs detalhados de solicitações HTTP. Escolha uma de sua confiança e execute a ferramenta para escutar na porta em que o tráfego do Amazon S3 passa ao enviar outra solicitação HTTP do Amazon S3.

Para solicitações HTTP, o par de IDs de solicitação será parecido com isto:

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

Note

As solicitações HTTP estão criptografadas e ocultas na maioria das capturas de pacotes.

Usar um navegador da web para obter IDs de solicitação

A maioria dos navegadores da web tem ferramentas de desenvolvedor que podem ser usadas para visualizar os cabeçalhos das solicitações.

Para solicitações baseadas em navegador da web que retornam um erro, o par de IDs de solicitações será parecido com os exemplos a seguir.

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

Para obter o par de IDs de solicitações bem-sucedidas, use as ferramentas de desenvolvedor do navegador para ver os cabeçalhos das respostas HTTP.

Usar os AWS SDKs para obter IDs de solicitação

As seções a seguir incluem informações para configuração do log usando diferentes SDKs da AWS.

Note

Embora seja possível habilitar o log detalhado em cada solicitação e resposta, não recomendamos habilitar o log em sistemas de produção, já que solicitações ou respostas grandes podem provocar uma lentidão significativa em uma aplicação.

Para solicitações do SDK da AWS, o par de IDs da solicitação será parecido com o seguinte:

```
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++

O tipo de registrador e a verbosidade são especificados durante a inicialização do SDK no argumento `SDKOptions`. O exemplo a seguir especifica o nível de verbosidade como `LogLevel::Debug`.

O registrador padrão gravará no sistema de arquivos e o arquivo será nomeado usando a seguinte convenção `aws_sdk_YYYY-MM-DD-HH.log`. O registrador cria um novo arquivo na hora.

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

Para obter mais informações, consulte [Como ativo o registro em log?](#) no wiki do SDK da AWS para C++ no GitHub.

Go

É possível configurar o registro em log usando o SDK para Go. Para obter mais informações, consulte [Registro em log](#) no Guia do desenvolvedor do AWS SDK para Go v2.

Java

Você pode habilitar o log para solicitações ou respostas específicas, para obter e retornar apenas cabeçalhos relevantes. Para fazer isso, importe a

classe com `amazonaws.services.s3.S3ResponseMetadata`. Mais tarde, você pode armazenar a solicitação em uma variável antes de executar a solicitação real. Para obter a solicitação ou a resposta registrada em log, chame `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());
```

Como alternativa, você pode usar log detalhado de cada solicitação e resposta de Java. Para obter mais informações, consulte [Registro em log detalhado](#) no Guia do desenvolvedor do AWS SDK para Java.

JavaScript

O AWS SDK para JavaScript tem um registrador integrado para que você possa registrar chamadas de API feitas com ele. Para ativar o registrador e imprimir entradas de log no console, adicione a instrução a seguir ao seu código:

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

Para obter mais informações, consulte [Registro de chamadas do AWS SDK para JavaScript](#) no Guia do desenvolvedor do AWS SDK para JavaScript.

Kotlin

Com o AWS SDK para Kotlin, você pode especificar o modo de log para mensagens no nível da rede usando configurações de código ou ambiente. Você pode definir o modo de log para solicitações e respostas HTTP.

Para optar pelo registro em log adicional, defina a propriedade `LogMode` ao criar um cliente de serviço:

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

// ...

val client = S3Client {
    // ...
```



```
logMode = LogMode.LogRequestWithBody + LogMode.LogResponse
}
```

Como alternativa, você pode definir o modo de log usando uma variável de ambiente:

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

Para obter mais informações, consulte [Registro em log](#) no Guia do desenvolvedor do AWS SDK para Kotlin.

.NET

É possível configurar o log com o SDK para .NET usando a ferramenta integrada de log System.Diagnostics. Para obter mais informações, consulte a postagem [Registro em log com o SDK para .NET](#) no Blog do desenvolvedor da AWS.

Note

Por padrão, o log retornado contém somente informações de erros. Para obter os IDs de solicitação, o arquivo de configuração deve ter `AWSLogMetrics` (e, opcionalmente, `AWSResponseLogging`) adicionado.

PHP

Você pode obter informações de depuração, incluindo os dados enviados pela rede, definindo a opção de depuração como `true` em um construtor de cliente.

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

Para obter mais informações, consulte [Como posso ver quais dados são enviados pela rede?](#) no Guia do desenvolvedor do AWS SDK para PHP.

Python (Boto3)

Com o AWS SDK para Python (Boto3), você pode registrar respostas específicas. É possível usar esse recurso para capturar somente os cabeçalhos relevantes. O código a seguir mostra como registrar em log partes da resposta a um arquivo:

```
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'])
```

Também é possível capturar exceções e registrar em log informações relevantes quando uma exceção é gerada. Para obter mais informações, consulte [Discernir informações úteis de respostas de erro](#) na Referência de API do AWS SDK para Python (Boto).

Além disso, é possível configurar o Boto3 para gerar logs de depuração detalhados usando o seguinte código:

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

Para obter mais informações, consulte [set_stream_logger](#) na Referência de API do AWS SDK para Python (Boto).

Ruby

É possível obter os IDs de solicitação usando o SDK para Ruby versão 3. Você pode ativar o registro em log de conexões HTTP em seu cliente usando o seguinte código:

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

Note

O registro em log eletrônico pode incluir informações confidenciais, como o ID da chave de acesso. As informações confidenciais devem ser higienizadas antes de serem compartilhadas com o Suporte da AWS.

Você também pode encontrar o ID da solicitação do objeto de contexto da solicitação na resposta ou no erro da solicitação:

```
# 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 obter mais informações, consulte [Depuração usando informações de rastreamento de rede de um cliente do AWS SDK para Ruby](#) no Guia do desenvolvedor do AWS SDK para Ruby.

Rust

Para habilitar o registro em log, adicione a caixa `tracing-subscriber` e inicialize-a na aplicação do Rust.

Adicione a biblioteca de rastreamento ao arquivo `Cargo.toml`:

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

Em seguida, no código Rust, inicialize o registrador na função principal antes de chamar qualquer operação do SDK:

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

Para obter mais informações, consulte [Configuração e uso de logs no AWS SDK para Rust](#) no Guia do desenvolvedor do AWS SDK para Rust.

Operações de API no nível do objeto do Amazon S3 com suporte para Tabelas do S3

A tabela a seguir inclui as operações de API no nível do objeto do S3 com suporte e os cabeçalhos correspondentes para as Tabelas do S3. Para obter uma lista de APIs de Tabelas do Amazon S3, consulte [Tabelas do Amazon S3](#). Para obter mais informações sobre Tabelas do Amazon S3, consulte [Trabalho com Tabelas do Amazon S3 e buckets de tabela](#) no Guia do usuário do Amazon S3.

API no nível do objeto	Cabeçalhos compatíveis	Observações	
AbortMultipartUpload	x-amz-expected-bucket-owner	Nenhum	
CompleteMultipartUpload	x-amz-checksum-crc32	Nenhum	
	x-amz-checksum-crc32c	Nenhum	
	x-amz-checksum-sha1	Nenhum	
	x-amz-checksum-sha256	Nenhum	
	x-amz-expected-bucket-owner	Nenhum	
	If-Match	Nenhum	
	If-None-Match	Nenhum	

API no nível do objeto	Cabeçalhos compatíveis	Observações	
CreateMultipartUpload	x-amz-acl: ACL	Para Tabelas do S3, o valor padrão é <code>bucket-owner-full-control</code> e não pode ser alterado.	
	Cache-Control	Nenhum	
	Content-Disposition	Nenhum	
	Content-Encoding	Nenhum	
	Content-Language	Nenhum	
	Content-Type	Nenhum	
	Expires	Nenhum	
	x-amz-checksum-algorithm	Nenhum	
	x-amz-storage-class	Para Tabelas do S3, o valor padrão é <code>STANDARD</code> e não pode ser alterado.	
GetObject	x-amz-server-side-encryption	Para Tabelas do S3, o valor padrão é <code>(SSE-S3) (AES256)</code> e não pode ser alterado.	
	If-Match	Nenhum	
	If-Modified-Since	Nenhum	

API no nível do objeto	Cabeçalhos compatíveis	Observações	
	If-None-Match	Nenhum	
	If-Unmodified-Since	Nenhum	
	Range	Nenhum	
	x-amz-expected-bucket-owner	Nenhum	
	x-amz-checksum-mode	Nenhum	
HeadObject	If-Match	Nenhum	
	If-Modified-Since	Nenhum	
	If-None-Match	Nenhum	
	If-Unmodified-Since	Nenhum	
	Range	Nenhum	
	x-amz-checksum-mode	Nenhum	
	x-amz-expected-bucket-owner	Nenhum	
ListParts	x-amz-expected-bucket-owner	Nenhum	

API no nível do objeto	Cabeçalhos compatíveis	Observações	
PutObject	x-amz-acl: ACL	Para Tabelas do S3, o valor padrão é <code>bucket-owner-full-control</code> e não pode ser alterado.	
	Cache-Control	Nenhum	
	Content-Disposition	Nenhum	
	Content-Encoding	Nenhum	
	Content-Language	Nenhum	
	Content-Length	Nenhum	
	Content-MD5	Nenhum	
	Content-Type	Nenhum	
	x-amz-sdk-checksum-algorithm	Nenhum	
	x-amz-checksum-crc32	Nenhum	
	x-amz-checksum-crc32c	Nenhum	
	x-amz-checksum-sha1	Nenhum	
	x-amz-checksum-sha256	Nenhum	

API no nível do objeto	Cabeçalhos compatíveis	Observações	
	Expires	Nenhum	
	If-None-Match	Nenhum	
	x-amz-expected-bucket-owner	Nenhum	
	x-amz-storage-class	Para Tabelas do S3, o valor padrão é STANDARD e não pode ser alterado.	
	x-amz-server-side-encryption	Para Tabelas do S3, o valor padrão é (SSE-S3) (AES256) e não pode ser alterado.	
UploadPart	Content-Length	Nenhum	
	Content-MD5	Nenhum	
	x-amz-checksum-crc32	Nenhum	
	x-amz-checksum-crc32c	Nenhum	
	x-amz-checksum-sha1	Nenhum	
	x-amz-checksum-sha256	Nenhum	
	x-amz-expected-bucket-owner	Nenhum	

Exemplos de código do Amazon S3 usando SDKs da AWS

Os exemplos de código a seguir mostram como usar o Amazon S3 com um Kit de Desenvolvimento de Software (SDK) da AWS.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos de código

- [Exemplos de código do Amazon S3 usando SDKs da AWS](#)
- [Exemplos básicos do Amazon S3 usando SDKs da AWS](#)
 - [Olá, Amazon S3](#)
 - [Conheça os conceitos básicos do Amazon S3 com um SDK da AWS](#)
 - [Ações do Amazon S3 usando SDKs da AWS](#)
 - [Usar AbortMultipartUpload com o AWS SDK ou a CLI](#)
 - [Usar CompleteMultipartUpload com o AWS SDK ou a CLI](#)
 - [Usar CopyObject com o AWS SDK ou a CLI](#)
 - [Usar CreateBucket com o AWS SDK ou a CLI](#)
 - [Usar CreateMultiRegionAccessPoint com um SDK da AWS](#)
 - [Usar CreateMultipartUpload com o AWS SDK ou a CLI](#)
 - [Usar CreatePresignedPost com um SDK da AWS](#)
 - [Usar DeleteBucket com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketAnalyticsConfiguration com uma CLI](#)
 - [Usar DeleteBucketCors com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketEncryption com uma CLI](#)
 - [Usar DeleteBucketInventoryConfiguration com uma CLI](#)
 - [Usar DeleteBucketLifecycle com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketMetricsConfiguration com uma CLI](#)
 - [Usar DeleteBucketPolicy com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketReplication com uma CLI](#)
 - [Usar DeleteBucketTagging com uma CLI](#)

- [Usar DeleteBucketWebsite com o AWS SDK ou a CLI](#)
- [Usar DeleteObject com o AWS SDK ou a CLI](#)
- [Usar DeleteObjectTagging com uma CLI](#)
- [Usar DeleteObjects com o AWS SDK ou a CLI](#)
- [Usar DeletePublicAccessBlock com uma CLI](#)
- [Usar GetBucketAccelerateConfiguration com uma CLI](#)
- [Usar GetBucketAcl com o AWS SDK ou a CLI](#)
- [Usar GetBucketAnalyticsConfiguration com uma CLI](#)
- [Usar GetBucketCors com o AWS SDK ou a CLI](#)
- [Usar GetBucketEncryption com o AWS SDK ou a CLI](#)
- [Usar GetBucketInventoryConfiguration com uma CLI](#)
- [Usar GetBucketLifecycleConfiguration com o AWS SDK ou a CLI](#)
- [Usar GetBucketLocation com o AWS SDK ou a CLI](#)
- [Usar GetBucketLogging com uma CLI](#)
- [Usar GetBucketMetricsConfiguration com uma CLI](#)
- [Usar GetBucketNotification com uma CLI](#)
- [Usar GetBucketPolicy com o AWS SDK ou a CLI](#)
- [Usar GetBucketPolicyStatus com uma CLI](#)
- [Usar GetBucketReplication com o AWS SDK ou a CLI](#)
- [Usar GetBucketRequestPayment com uma CLI](#)
- [Usar GetBucketTagging com uma CLI](#)
- [Usar GetBucketVersioning com uma CLI](#)
- [Usar GetBucketWebsite com o AWS SDK ou a CLI](#)
- [Usar GetObject com o AWS SDK ou a CLI](#)
- [Usar GetObjectAcl com o AWS SDK ou a CLI](#)
- [Usar GetObjectAttributes com o AWS SDK ou a CLI](#)
- [Usar GetObjectLegalHold com o AWS SDK ou a CLI](#)
- [Usar GetObjectLockConfiguration com o AWS SDK ou a CLI](#)
- [Usar GetObjectRetention com o AWS SDK ou a CLI](#)
- [Usar GetObjectTagging com uma CLI](#)

- [Usar GetPublicAccessBlock com uma CLI](#)
- [Usar HeadBucket com o AWS SDK ou a CLI](#)
- [Usar HeadObject com o AWS SDK ou a CLI](#)
- [Usar ListBucketAnalyticsConfigurations com uma CLI](#)
- [Usar ListBucketInventoryConfigurations com uma CLI](#)
- [Usar ListBuckets com o AWS SDK ou a CLI](#)
- [Usar ListMultipartUploads com o AWS SDK ou a CLI](#)
- [Usar ListObjectVersions com o AWS SDK ou a CLI](#)
- [Usar ListObjects com uma CLI](#)
- [Usar ListObjectsV2 com o AWS SDK ou a CLI](#)
- [Usar PutBucketAccelerateConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutBucketAcl com o AWS SDK ou a CLI](#)
- [Usar PutBucketCors com o AWS SDK ou a CLI](#)
- [Usar PutBucketEncryption com o AWS SDK ou a CLI](#)
- [Usar PutBucketLifecycleConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutBucketLogging com o AWS SDK ou a CLI](#)
- [Usar PutBucketNotification com uma CLI](#)
- [Usar PutBucketNotificationConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutBucketPolicy com o AWS SDK ou a CLI](#)
- [Usar PutBucketReplication com o AWS SDK ou a CLI](#)
- [Usar PutBucketRequestPayment com uma CLI](#)
- [Usar PutBucketTagging com uma CLI](#)
- [Usar PutBucketVersioning com o AWS SDK ou a CLI](#)
- [Usar PutBucketWebsite com o AWS SDK ou a CLI](#)
- [Usar PutObject com o AWS SDK ou a CLI](#)
- [Usar PutObjectAcl com o AWS SDK ou a CLI](#)
- [Usar PutObjectLegalHold com o AWS SDK ou a CLI](#)
- [Usar PutObjectLockConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutObjectRetention com o AWS SDK ou a CLI](#)
- [Usar RestoreObject com o AWS SDK ou a CLI](#)

- [Usar SelectObjectContent com o AWS SDK ou a CLI](#)
- [Usar UploadPart com o AWS SDK ou a CLI](#)
- [Usar UploadPartCopy com o AWS SDK ou a CLI](#)
- [Cenários do Amazon S3 usando AWS SDKs](#)
 - [Verificar se um bucket existe](#)
 - [Verificar se um objeto existe](#)
 - [Converter texto em fala e de volta em texto usando um AWS SDK](#)
 - [Criar um URL pré-assinado para o Amazon S3 usando um AWS SDK](#)
 - [Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos](#)
 - [Uma página da web que oferece uma lista de objetos do Amazon S3 usando um AWS SDK](#)
 - [Criar uma aplicação de exploração do Amazon Textract](#)
 - [Excluir todos os objetos de um bucket específico do Amazon S3 utilizando um SDK da AWS](#)
 - [Excluir uploads multiparte incompletos para o Amazon S3 usando um AWS SDK](#)
 - [Detectar EPI em imagens com o Amazon Rekognition usando um AWS SDK](#)
 - [Detectar entidades em texto extraído de uma imagem usando um AWS SDK](#)
 - [Detectar faces em uma imagem usando um AWS SDK](#)
 - [Detectar objetos em imagens com o Amazon Rekognition usando um AWS SDK](#)
 - [Detectar pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK](#)
 - [Baixar “diretórios” do S3 de um bucket do Amazon Simple Storage Service \(Amazon S3\)](#)
 - [Fazer download de todos os objetos de um bucket do Amazon Simple Storage Service \(Amazon S3\) em um diretório local](#)
 - [Baixar um fluxo de tamanho desconhecido em um objeto do Amazon S3 usando um AWS SDK](#)
 - [Obter um objeto do Amazon S3 de um ponto de acesso multirregional usando um AWS SDK](#)
 - [Obter um objeto de um bucket do Amazon S3 usando um AWS SDK e especificando um cabeçalho If-Modified-Since](#)
 - [Conceitos básicos de criptografia para objetos do Amazon S3 usando um AWS SDK](#)
 - [Conceitos básicos de etiquetas para objetos do Amazon S3 usando um AWS SDK](#)
 - [Introdução aos clusters de processamento de big data](#)

- [Conceitos básicos do gerenciamento de configurações](#)
 - [Introdução à extração de texto de documentos](#)
 - [Conceitos básicos de arquivos de atributos de machine learning](#)
 - [Introdução ao armazenamento de objetos](#)
 - [Introdução à análise de consulta](#)
 - [Trabalhar com os recursos de bloqueio de objetos do Amazon S3 usando um SDK da AWS](#)
 - [Fazer solicitações condicionais do Amazon S3 usando um AWS SDK](#)
 - [Gerenciar listas de controle de acesso \(ACLs\) para buckets do Amazon S3 usando um AWS SDK](#)
 - [Gerenciar mensagens grandes do Amazon SQS usando o Amazon S3 com um AWS SDK](#)
 - [Gerenciar objetos do Amazon S3 versionados em lotes com uma função do Lambda usando um AWS SDK](#)
 - [Analisar URIs do Amazon S3 usando um AWS SDK](#)
 - [Executar uma cópia multipart de um objeto do Amazon S3 usando um AWS SDK](#)
 - [Receber e processar notificações de eventos do Amazon S3 usando um AWS SDK](#)
 - [Salvar o EXIF e outras informações de imagem usando um AWS SDK](#)
 - [Enviar notificações de eventos do S3 ao Amazon EventBridge usando um AWS SDK](#)
 - [Monitorar o upload ou download de um objeto do Amazon S3 usando um SDK da AWS](#)
 - [Como transformar dados para sua aplicação com o S3 Object Lambda](#)
 - [Exemplos de abordagens para testes de unidade e integração com SDK da AWS](#)
 - [Fazer upload recursivo de um diretório local em um bucket do Amazon Simple Storage Service \(Amazon S3\)](#)
 - [Fazer upload ou download de arquivos grandes de e para o Amazon S3 usando um AWS SDK](#)
 - [Faça upload de um fluxo de tamanho desconhecido em um objeto do Amazon S3 usando um AWS SDK](#)
 - [Usar somas de verificação para trabalhar com um objeto do Amazon S3 usando um AWS SDK](#)
 - [Trabalhar com os recursos de integridade de objetos do Amazon S3 usando um AWS SDK](#)
 - [Trabalhar com objetos versionados do Amazon S3 usando um AWS SDK](#)
- [Exemplos de tecnologia sem servidor para o Amazon S3](#)

- [Invocar uma função do Lambda em um acionador do Amazon S3](#)
- [Exemplos de código do Amazon S3 Control usando AWS SDKs](#)
- [Exemplos básicos do Amazon S3 Control usando AWS SDKs](#)
 - [Olá, Amazon S3 Control](#)
 - [Conheça os conceitos básicos do Amazon S3 Control com um SDK da AWS](#)
 - [Ações do Amazon S3 Control usando AWS SDKs](#)
 - [Usar CreateJob com o AWS SDK ou a CLI](#)
 - [Usar DeleteJobTagging com um SDK da AWS](#)
 - [Usar DescribeJob com o AWS SDK ou a CLI](#)
 - [Usar GetJobTagging com um SDK da AWS](#)
 - [Usar PutJobTagging com um SDK da AWS](#)
 - [Usar UpdateJobPriority com o AWS SDK ou a CLI](#)
 - [Usar UpdateJobStatus com o AWS SDK ou a CLI](#)
- [Exemplos de código para buckets de diretório do S3 usando AWS SDKs](#)
- [Exemplos básicos de buckets de diretório do S3 usando AWS SDKs](#)
 - [Olá, Amazon S3 directory buckets](#)
 - [Aprender as noções básicas dos buckets de diretório do S3 com um SDK AWS](#)
 - [Ações para buckets de diretório do S3 usando AWS SDKs](#)
 - [Usar AbortMultipartUpload com um SDK da AWS](#)
 - [Usar CompleteMultipartUpload com um SDK da AWS](#)
 - [Usar CopyObject com um SDK da AWS](#)
 - [Usar CreateBucket com um SDK da AWS](#)
 - [Usar CreateMultipartUpload com um SDK da AWS](#)
 - [Usar CreateSession com um SDK da AWS](#)
 - [Usar DeleteBucket com um SDK da AWS](#)
 - [Usar DeleteBucketEncryption com um SDK da AWS](#)
 - [Usar DeleteBucketPolicy com um SDK da AWS](#)
 - [Usar DeleteObject com um SDK da AWS](#)
 - [Usar DeleteObjects com um SDK da AWS](#)
 - [Usar GetBucketEncryption com um SDK da AWS](#)

- [Usar GetBucketPolicy com um SDK da AWS](#)
- [Usar GetObject com um SDK da AWS](#)
- [Usar GetObjectAttributes com um SDK da AWS](#)
- [Usar HeadBucket com um SDK da AWS](#)
- [Usar HeadObject com um SDK da AWS](#)
- [Usar ListDirectoryBuckets com um SDK da AWS](#)
- [Usar ListMultipartUploads com um SDK da AWS](#)
- [Usar ListObjectsV2 com um SDK da AWS](#)
- [Usar ListParts com um SDK da AWS](#)
- [Usar PutBucketEncryption com um SDK da AWS](#)
- [Usar PutBucketPolicy com um SDK da AWS](#)
- [Usar PutObject com um SDK da AWS](#)
- [Usar UploadPart com um SDK da AWS](#)
- [Usar UploadPartCopy com um SDK da AWS](#)
- [Cenários para buckets de diretório do S3 usando AWS SDKs](#)
 - [Criar um URL pré-assinado para os buckets de diretório do Amazon S3 para obter um objeto usando um AWS SDK](#)

Exemplos de código do Amazon S3 usando SDKs da AWS

Os exemplos de código a seguir mostram como usar o Amazon S3 com um Kit de Desenvolvimento de Software (SDK) da AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Para ver uma lista completa dos Guias do desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Amazon S3

- [Exemplos básicos do Amazon S3 usando SDKs da AWS](#)
 - [Olá, Amazon S3](#)
 - [Conheça os conceitos básicos do Amazon S3 com um SDK da AWS](#)
 - [Ações do Amazon S3 usando SDKs da AWS](#)
 - [Usar AbortMultipartUpload com o AWS SDK ou a CLI](#)
 - [Usar CompleteMultipartUpload com o AWS SDK ou a CLI](#)
 - [Usar CopyObject com o AWS SDK ou a CLI](#)
 - [Usar CreateBucket com o AWS SDK ou a CLI](#)
 - [Usar CreateMultiRegionAccessPoint com um SDK da AWS](#)
 - [Usar CreateMultipartUpload com o AWS SDK ou a CLI](#)
 - [Usar CreatePresignedPost com um SDK da AWS](#)
 - [Usar DeleteBucket com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketAnalyticsConfiguration com uma CLI](#)
 - [Usar DeleteBucketCors com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketEncryption com uma CLI](#)
 - [Usar DeleteBucketInventoryConfiguration com uma CLI](#)
 - [Usar DeleteBucketLifecycle com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketMetricsConfiguration com uma CLI](#)
 - [Usar DeleteBucketPolicy com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketReplication com uma CLI](#)
 - [Usar DeleteBucketTagging com uma CLI](#)
 - [Usar DeleteBucketWebsite com o AWS SDK ou a CLI](#)
 - [Usar DeleteObject com o AWS SDK ou a CLI](#)
 - [Usar DeleteObjectTagging com uma CLI](#)
 - [Usar DeleteObjects com o AWS SDK ou a CLI](#)
 - [Usar DeletePublicAccessBlock com uma CLI](#)

- [Usar GetBucketAccelerateConfiguration com uma CLI](#)
- [Usar GetBucketAcl com o AWS SDK ou a CLI](#)
- [Usar GetBucketAnalyticsConfiguration com uma CLI](#)
- [Usar GetBucketCors com o AWS SDK ou a CLI](#)
- [Usar GetBucketEncryption com o AWS SDK ou a CLI](#)
- [Usar GetBucketInventoryConfiguration com uma CLI](#)
- [Usar GetBucketLifecycleConfiguration com o AWS SDK ou a CLI](#)
- [Usar GetBucketLocation com o AWS SDK ou a CLI](#)
- [Usar GetBucketLogging com uma CLI](#)
- [Usar GetBucketMetricsConfiguration com uma CLI](#)
- [Usar GetBucketNotification com uma CLI](#)
- [Usar GetBucketPolicy com o AWS SDK ou a CLI](#)
- [Usar GetBucketPolicyStatus com uma CLI](#)
- [Usar GetBucketReplication com o AWS SDK ou a CLI](#)
- [Usar GetBucketRequestPayment com uma CLI](#)
- [Usar GetBucketTagging com uma CLI](#)
- [Usar GetBucketVersioning com uma CLI](#)
- [Usar GetBucketWebsite com o AWS SDK ou a CLI](#)
- [Usar GetObject com o AWS SDK ou a CLI](#)
- [Usar GetObjectAcl com o AWS SDK ou a CLI](#)
- [Usar GetObjectAttributes com o AWS SDK ou a CLI](#)
- [Usar GetObjectLegalHold com o AWS SDK ou a CLI](#)
- [Usar GetObjectLockConfiguration com o AWS SDK ou a CLI](#)
- [Usar GetObjectRetention com o AWS SDK ou a CLI](#)
- [Usar GetObjectTagging com uma CLI](#)
- [Usar GetPublicAccessBlock com uma CLI](#)
- [Usar HeadBucket com o AWS SDK ou a CLI](#)
- [Usar HeadObject com o AWS SDK ou a CLI](#)
- [Usar ListBucketAnalyticsConfigurations com uma CLI](#)
- [Usar ListBucketInventoryConfigurations com uma CLI](#)

- [Usar ListBuckets com o AWS SDK ou a CLI](#)
- [Usar ListMultipartUploads com o AWS SDK ou a CLI](#)
- [Usar ListObjectVersions com o AWS SDK ou a CLI](#)
- [Usar ListObjects com uma CLI](#)
- [Usar ListObjectsV2 com o AWS SDK ou a CLI](#)
- [Usar PutBucketAccelerateConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutBucketAcl com o AWS SDK ou a CLI](#)
- [Usar PutBucketCors com o AWS SDK ou a CLI](#)
- [Usar PutBucketEncryption com o AWS SDK ou a CLI](#)
- [Usar PutBucketLifecycleConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutBucketLogging com o AWS SDK ou a CLI](#)
- [Usar PutBucketNotification com uma CLI](#)
- [Usar PutBucketNotificationConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutBucketPolicy com o AWS SDK ou a CLI](#)
- [Usar PutBucketReplication com o AWS SDK ou a CLI](#)
- [Usar PutBucketRequestPayment com uma CLI](#)
- [Usar PutBucketTagging com uma CLI](#)
- [Usar PutBucketVersioning com o AWS SDK ou a CLI](#)
- [Usar PutBucketWebsite com o AWS SDK ou a CLI](#)
- [Usar PutObject com o AWS SDK ou a CLI](#)
- [Usar PutObjectAcl com o AWS SDK ou a CLI](#)
- [Usar PutObjectLegalHold com o AWS SDK ou a CLI](#)
- [Usar PutObjectLockConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutObjectRetention com o AWS SDK ou a CLI](#)
- [Usar RestoreObject com o AWS SDK ou a CLI](#)
- [Usar SelectObjectContent com o AWS SDK ou a CLI](#)
- [Usar UploadPart com o AWS SDK ou a CLI](#)
- [Usar UploadPartCopy com o AWS SDK ou a CLI](#)

- [Cenários do Amazon S3 usando AWS SDKs](#)

- [Verificar se um bucket existe](#)

- [Verificar se um objeto existe](#)
- [Converter texto em fala e de volta em texto usando um AWS SDK](#)
- [Criar um URL pré-assinado para o Amazon S3 usando um AWS SDK](#)
- [Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos](#)
- [Uma página da web que oferece uma lista de objetos do Amazon S3 usando um AWS SDK](#)
- [Criar uma aplicação de exploração do Amazon Textract](#)
- [Excluir todos os objetos de um bucket específico do Amazon S3 utilizando um SDK da AWS](#)
- [Excluir uploads multiparte incompletos para o Amazon S3 usando um AWS SDK](#)
- [Detectar EPI em imagens com o Amazon Rekognition usando um AWS SDK](#)
- [Detectar entidades em texto extraído de uma imagem usando um AWS SDK](#)
- [Detectar faces em uma imagem usando um AWS SDK](#)
- [Detectar objetos em imagens com o Amazon Rekognition usando um AWS SDK](#)
- [Detectar pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK](#)
- [Baixar “diretórios” do S3 de um bucket do Amazon Simple Storage Service \(Amazon S3\)](#)
- [Fazer download de todos os objetos de um bucket do Amazon Simple Storage Service \(Amazon S3\) em um diretório local](#)
- [Baixar um fluxo de tamanho desconhecido em um objeto do Amazon S3 usando um AWS SDK](#)
- [Obter um objeto do Amazon S3 de um ponto de acesso multirregional usando um AWS SDK](#)
- [Obter um objeto de um bucket do Amazon S3 usando um AWS SDK e especificando um cabeçalho If-Modified-Since](#)
- [Conceitos básicos de criptografia para objetos do Amazon S3 usando um AWS SDK](#)
- [Conceitos básicos de etiquetas para objetos do Amazon S3 usando um AWS SDK](#)
- [Introdução aos clusters de processamento de big data](#)
- [Conceitos básicos do gerenciamento de configurações](#)
- [Introdução à extração de texto de documentos](#)
- [Conceitos básicos de arquivos de atributos de machine learning](#)
- [Introdução ao armazenamento de objetos](#)
- [Introdução à análise de consulta](#)
- [Trabalhar com os recursos de bloqueio de objetos do Amazon S3 usando um SDK da AWS](#)
- [Fazer solicitações condicionais do Amazon S3 usando um AWS SDK](#)

- [Gerenciar listas de controle de acesso \(ACLs\) para buckets do Amazon S3 usando um AWS SDK](#)
- [Gerenciar mensagens grandes do Amazon SQS usando o Amazon S3 com um AWS SDK](#)
- [Gerenciar objetos do Amazon S3 versionados em lotes com uma função do Lambda usando um AWS SDK](#)
- [Analisar URIs do Amazon S3 usando um AWS SDK](#)
- [Executar uma cópia multipart de um objeto do Amazon S3 usando um AWS SDK](#)
- [Receber e processar notificações de eventos do Amazon S3 usando um AWS SDK](#)
- [Salvar o EXIF e outras informações de imagem usando um AWS SDK](#)
- [Enviar notificações de eventos do S3 ao Amazon EventBridge usando um AWS SDK](#)
- [Monitorar o upload ou download de um objeto do Amazon S3 usando um SDK da AWS](#)
- [Como transformar dados para sua aplicação com o S3 Object Lambda](#)
- [Exemplos de abordagens para testes de unidade e integração com SDK da AWS](#)
- [Fazer upload recursivo de um diretório local em um bucket do Amazon Simple Storage Service \(Amazon S3\)](#)
- [Fazer upload ou download de arquivos grandes de e para o Amazon S3 usando um AWS SDK](#)
- [Faça upload de um fluxo de tamanho desconhecido em um objeto do Amazon S3 usando um AWS SDK](#)
- [Usar somas de verificação para trabalhar com um objeto do Amazon S3 usando um AWS SDK](#)
- [Trabalhar com os recursos de integridade de objetos do Amazon S3 usando um AWS SDK](#)
- [Trabalhar com objetos versionados do Amazon S3 usando um AWS SDK](#)
- [Exemplos de tecnologia sem servidor para o Amazon S3](#)
 - [Invocar uma função do Lambda em um acionador do Amazon S3](#)

Exemplos básicos do Amazon S3 usando SDKs da AWS

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon Simple Storage Service com SDKs da AWS.

Exemplos

- [Olá, Amazon S3](#)

- [Ações do Amazon S3 usando SDKs da AWS](#)
 - [Usar AbortMultipartUpload com o AWS SDK ou a CLI](#)
 - [Usar CompleteMultipartUpload com o AWS SDK ou a CLI](#)
 - [Usar CopyObject com o AWS SDK ou a CLI](#)
 - [Usar CreateBucket com o AWS SDK ou a CLI](#)
 - [Usar CreateMultiRegionAccessPoint com um SDK da AWS](#)
 - [Usar CreateMultipartUpload com o AWS SDK ou a CLI](#)
 - [Usar CreatePresignedPost com um SDK da AWS](#)
 - [Usar DeleteBucket com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketAnalyticsConfiguration com uma CLI](#)
 - [Usar DeleteBucketCors com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketEncryption com uma CLI](#)
 - [Usar DeleteBucketInventoryConfiguration com uma CLI](#)
 - [Usar DeleteBucketLifecycle com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketMetricsConfiguration com uma CLI](#)
 - [Usar DeleteBucketPolicy com o AWS SDK ou a CLI](#)
 - [Usar DeleteBucketReplication com uma CLI](#)
 - [Usar DeleteBucketTagging com uma CLI](#)
 - [Usar DeleteBucketWebsite com o AWS SDK ou a CLI](#)
 - [Usar DeleteObject com o AWS SDK ou a CLI](#)
 - [Usar DeleteObjectTagging com uma CLI](#)
 - [Usar DeleteObjects com o AWS SDK ou a CLI](#)
 - [Usar DeletePublicAccessBlock com uma CLI](#)
 - [Usar GetBucketAccelerateConfiguration com uma CLI](#)
 - [Usar GetBucketAcl com o AWS SDK ou a CLI](#)
 - [Usar GetBucketAnalyticsConfiguration com uma CLI](#)
 - [Usar GetBucketCors com o AWS SDK ou a CLI](#)
 - [Usar GetBucketEncryption com o AWS SDK ou a CLI](#)
 - [Usar GetBucketInventoryConfiguration com uma CLI](#)
 - [Usar GetBucketLifecycleConfiguration com o AWS SDK ou a CLI](#)

- [Usar GetBucketLocation com o AWS SDK ou a CLI](#)
- [Usar GetBucketLogging com uma CLI](#)
- [Usar GetBucketMetricsConfiguration com uma CLI](#)
- [Usar GetBucketNotification com uma CLI](#)
- [Usar GetBucketPolicy com o AWS SDK ou a CLI](#)
- [Usar GetBucketPolicyStatus com uma CLI](#)
- [Usar GetBucketReplication com o AWS SDK ou a CLI](#)
- [Usar GetBucketRequestPayment com uma CLI](#)
- [Usar GetBucketTagging com uma CLI](#)
- [Usar GetBucketVersioning com uma CLI](#)
- [Usar GetBucketWebsite com o AWS SDK ou a CLI](#)
- [Usar GetObject com o AWS SDK ou a CLI](#)
- [Usar GetObjectAcl com o AWS SDK ou a CLI](#)
- [Usar GetObjectAttributes com o AWS SDK ou a CLI](#)
- [Usar GetObjectLegalHold com o AWS SDK ou a CLI](#)
- [Usar GetObjectLockConfiguration com o AWS SDK ou a CLI](#)
- [Usar GetObjectRetention com o AWS SDK ou a CLI](#)
- [Usar GetObjectTagging com uma CLI](#)
- [Usar GetPublicAccessBlock com uma CLI](#)
- [Usar HeadBucket com o AWS SDK ou a CLI](#)
- [Usar HeadObject com o AWS SDK ou a CLI](#)
- [Usar ListBucketAnalyticsConfigurations com uma CLI](#)
- [Usar ListBucketInventoryConfigurations com uma CLI](#)
- [Usar ListBuckets com o AWS SDK ou a CLI](#)
- [Usar ListMultipartUploads com o AWS SDK ou a CLI](#)
- [Usar ListObjectVersions com o AWS SDK ou a CLI](#)
- [Usar ListObjects com uma CLI](#)
- [Usar ListObjectsV2 com o AWS SDK ou a CLI](#)
- [Usar PutBucketAccelerateConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutBucketAcl com o AWS SDK ou a CLI](#)

- [Usar PutBucketCors com o AWS SDK ou a CLI](#)
- [Usar PutBucketEncryption com o AWS SDK ou a CLI](#)
- [Usar PutBucketLifecycleConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutBucketLogging com o AWS SDK ou a CLI](#)
- [Usar PutBucketNotification com uma CLI](#)
- [Usar PutBucketNotificationConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutBucketPolicy com o AWS SDK ou a CLI](#)
- [Usar PutBucketReplication com o AWS SDK ou a CLI](#)
- [Usar PutBucketRequestPayment com uma CLI](#)
- [Usar PutBucketTagging com uma CLI](#)
- [Usar PutBucketVersioning com o AWS SDK ou a CLI](#)
- [Usar PutBucketWebsite com o AWS SDK ou a CLI](#)
- [Usar PutObject com o AWS SDK ou a CLI](#)
- [Usar PutObjectAcl com o AWS SDK ou a CLI](#)
- [Usar PutObjectLegalHold com o AWS SDK ou a CLI](#)
- [Usar PutObjectLockConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutObjectRetention com o AWS SDK ou a CLI](#)
- [Usar RestoreObject com o AWS SDK ou a CLI](#)
- [Usar SelectObjectContent com o AWS SDK ou a CLI](#)
- [Usar UploadPart com o AWS SDK ou a CLI](#)
- [Usar UploadPartCopy com o AWS SDK ou a CLI](#)

Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

.NET

SDK para .NET (v4)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [ListBuckets](#) na Referência da API AWS SDK para .NET.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Código para o arquivo CMakeLists.txt do CMake.

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

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

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

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

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

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

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

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

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

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

add_executable(${PROJECT_NAME}
    hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
```

```
${AWSSDK_LINK_LIBRARIES})
```

Código para o arquivo de origem `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 obter informações da API, consulte [ListBuckets](#) na Referência da API AWS SDK para C++.

Go

SDK para Go V2

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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)
        }
    }
}
```

```
}
```

- Consulte detalhes da API em [ListBuckets](#) na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter informações da API, consulte [ListBuckets](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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;
    }
  }
};
```

- Consulte detalhes da API em [ListBuckets](#) na Referência da API AWS SDK para JavaScript.

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
use Aws\S3\S3Client;

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

- Para obter informações da API, consulte [ListBuckets](#) na Referência da API AWS SDK para PHP.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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()
```

- Consulte detalhes da API em [ListBuckets](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
# 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

    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)
  end
end
```

```
manager.list_buckets
end
```

- Consulte detalhes da API em [ListBuckets](#) na Referência da API AWS SDK para Ruby.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

Utilitários 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)
    }
}
```

- Consulte detalhes da API em [ListBuckets](#) na Referência da API do AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Conheça os conceitos básicos do Amazon S3 com um SDK da AWS

Os exemplos de código a seguir mostram como:

- Crie um bucket e faça upload de um arquivo para ele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.
- Listar os objetos em um bucket.
- Exclua os objetos do bucket e o bucket.

.NET

SDK para .NET (v4)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Execute um cenário interativo que demonstre os recursos do 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($"\\n{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.WriteLine("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;
    }
}
```

Uma classe de wrapper para os métodos de SDK do Amazon S3.

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

namespace S3_Actions;

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

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

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

        var response = await _amazonS3.PutBucketAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error creating bucket: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Shows how to upload a file from the local computer to an Amazon S3
/// bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to which the object
```



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

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

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

```
        string bucketName,
        string objectName,
        string filePath)
    {
        var request = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
        };

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

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

    /// <summary>
    /// Copies an object in an Amazon S3 bucket to a folder within the
    /// same bucket.
    /// </summary>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
    /// object to copy is located.</param>
    /// <param name="objectName">The object to be copied.</param>
    /// <param name="folderName">The folder to which the object will
    /// be copied.</param>
    /// <returns>A boolean value that indicates the success or failure of
    /// the copy operation.</returns>
    public async Task<bool> CopyObjectInBucketAsync(
        string bucketName,
        string objectName,
        string folderName)
```

```

    {
        try
        {
            var request = new CopyObjectRequest
            {
                SourceBucket = bucketName,
                SourceKey = objectName,
                DestinationBucket = bucketName,
                DestinationKey = $"{folderName}\\{objectName}",
            };
            var response = await _amazonS3.CopyObjectAsync(request);
            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error copying object: '{ex.Message}'");
            return false;
        }
    }
}

/// <summary>
/// Shows how to list the objects in an Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket for which to list.
/// <param name="printList">True to print out the list.
/// <returns>The collection of objects.</returns>
public async Task<List<S3Object?>> ListBucketContentsAsync(string bucketName,
bool printList = true)
{
    try
    {
        var request = new ListObjectsV2Request
        {
            BucketName = bucketName,
            MaxKeys = 5,
        };

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

```

```

    }

    var listObjectsV2Paginator = _amazonS3.Paginators.ListObjectsV2(new
ListObjectsV2Request
    {
        BucketName = bucketName,
    });
    var s3Objects = new List<S3Object>();
    await foreach (var response in listObjectsV2Paginator.Responses)
    {
        if (response.S3Objects != null)
        {
            s3Objects.AddRange(response.S3Objects);
        }
    }

    if (printList)
    {
        Console.WriteLine($"Number of Objects: {s3Objects.Count}");
        foreach (var entry in s3Objects)
        {
            Console.WriteLine($"Key = {entry.Key} Size = {entry.Size}");
        }
    }

    return s3Objects;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' getting list of objects.");
    return null;
}
}

/// <summary>
/// Delete all of the objects stored in an existing Amazon S3 bucket.
/// </summary>
/// <param name="bucketName">The name of the bucket from which the
/// contents will be deleted.</param>
/// <returns>A boolean value that represents the success or failure of
/// deleting all of the objects in the bucket.</returns>

```

```

public async Task<bool> DeleteBucketContentsAsync(string bucketName)
{
    // Iterate over the contents of the bucket and delete all objects.
    try
    {
        // Delete all objects in the bucket.
        var deleteList = await ListBucketContentsAsync(bucketName, false);
        if (deleteList != null && deleteList.Any())
        {
            await _amazonS3.DeleteObjectsAsync(new DeleteObjectsRequest()
            {
                BucketName = bucketName,
                Objects = deleteList.Select(o => new KeyVersion { Key =
o.Key }).ToList(),
            });
        }

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

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

        await _amazonS3.DeleteBucketAsync(request);
        return true;
    }
}

```

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

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
#####
# 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
}

```

As funções do Amazon S3 usadas nesse cenário.

```

#####
# 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
}
```

```

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
}

```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
#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 {
            outStream << 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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina um struct que envolva as ações de bucket e objeto usadas pelo cenário.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)
```

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3)
// actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform
// bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket,
error) {
    var err error
    var output *s3.ListBucketsOutput
    var buckets []types.Bucket
    bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
&s3.ListBucketsInput{})
    for bucketPaginator.HasMorePages() {
        output, err = bucketPaginator.NextPage(ctx)
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
                fmt.Println("You don't have permission to list buckets for this account.")
                err = apiErr
            } else {
                log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
            }
            break
        } else {
            buckets = append(buckets, output.Buckets...)
        }
    }
    return buckets, err
}

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
(bool, error) {
```



```

_, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
    Bucket: aws.String(bucketName),
})
exists := true
if err != nil {
    var apiError smithy.APIError
    if errors.As(err, &apiError) {
        switch apiError.(type) {
        case *types.NotFound:
            log.Printf("Bucket %v is available.\n", bucketName)
            exists = false
            err = nil
        default:
            log.Printf("Either you don't have access to bucket %v or another error
occurred. "+
                "Here's what happened: %v\n", bucketName, err)
        }
    }
} else {
    log.Printf("Bucket %v exists and you already own it.", bucketName)
}

return exists, err
}

```

```

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
string) error {
_, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
    Bucket: aws.String(name),
    CreateBucketConfiguration: &types.CreateBucketConfiguration{
        LocationConstraint: types.BucketLocationConstraint(region),
    },
})
if err != nil {
    var owned *types.BucketAlreadyOwnedByYou
    var exists *types.BucketAlreadyExists
    if errors.As(err, &owned) {
        log.Printf("You already own bucket %s.\n", name)
        err = owned
    } else if errors.As(err, &exists) {
        log.Printf("Bucket %s already exists.\n", name)
    }
}
}

```

```
    err = exists
}
} else {
    err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
    }
}
return err
}

// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:     aws.String(objectKey),
            Body:    file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
                log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
                    fileName, bucketName, objectKey, err)
            }
        }
    }
} else {
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
}
```

```
    }
  }
  return err
}

// UploadLargeObject uses an upload manager to upload data to an object in a
// bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName
  string, objectKey string, largeObject []byte) error {
  largeBuffer := bytes.NewReader(largeObject)
  var partMiBs int64 = 10
  uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
    u.PartSize = partMiBs * 1024 * 1024
  })
  _, err := uploader.Upload(ctx, &s3.PutObjectInput{
    Bucket: aws.String(bucketName),
    Key:    aws.String(objectKey),
    Body:   largeBuffer,
  })
  if err != nil {
    var apiErr smithy.APIError
    if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
      log.Printf("Error while uploading object to %s. The object is too large.\n"+
        "The maximum size for a multipart upload is 5TB.", bucketName)
    } else {
      log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
        bucketName, objectKey, err)
    }
  } else {
    err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
      ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
        aws.String(objectKey)}, time.Minute)
    if err != nil {
      log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
    }
  }

  return err
}
```

```
// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
                objectKey, bucketName)
            err = noKey
        } else {
            log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName,
                objectKey, err)
        }
        return err
    }
    defer result.Body.Close()
    file, err := os.Create(fileName)
    if err != nil {
        log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
        return err
    }
    defer file.Close()
    body, err := io.ReadAll(result.Body)
    if err != nil {
        log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey,
            err)
    }
    _, err = file.Write(body)
    return err
}

// DownloadLargeObject uses a download manager to download an object from a
// bucket.
// The download manager gets the data in parts and writes them to a buffer until
// all of
// the data has been downloaded.
```

```
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10
    downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader)
    {
        d.PartSize = partMiBs * 1024 * 1024
    })
    buffer := manager.NewWriteAtBuffer([]byte{})
    _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    })
    if err != nil {
        log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return buffer.Bytes(), err
}
```

```
// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(ctx context.Context, bucketName string,
objectKey string, folderName string) error {
    objectDest := fmt.Sprintf("%v/%v", folderName, objectKey)
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(bucketName),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
        Key:         aws.String(objectDest),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
active tier.\n",
                objectKey, bucketName)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectDest)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectDest)
        }
    }
}
```

```
    }
    }
    return err
}

// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(destinationBucket),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
        Key:         aws.String(objectKey),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
active tier.\n",
                objectKey, sourceBucket)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }
    return err
}

// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
([]types.Object, error) {
    var err error
    var output *s3.ListObjectsV2Output
    input := &s3.ListObjectsV2Input{
        Bucket: aws.String(bucketName),
    }
}
```

```
var objects []types.Object
objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
for objectPaginator.HasMorePages() {
    output, err = objectPaginator.NextPage(ctx)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        }
        break
    } else {
        objects = append(objects, output.Contents...)
    }
}
return objects, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(ctx context.Context, bucketName string,
    objectKeys []string) error {
    var objectIds []types.ObjectIdentifier
    for _, key := range objectKeys {
        objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
    }
    output, err := basics.S3Client.DeleteObjects(ctx, &s3.DeleteObjectsInput{
        Bucket: aws.String(bucketName),
        Delete: &types.Delete{Objects: objectIds, Quiet: aws.Bool(true)},
    })
    if err != nil || len(output.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucketName)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
        } else if len(output.Errors) > 0 {
            for _, outErr := range output.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *output.Errors[0].Message)
        }
    }
}
```

```
}
} else {
    for _, delObjs := range output.Deleted {
        err = s3.NewObjectNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key: delObjs.Key},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                *delObjs.Key)
        } else {
            log.Printf("Deleted %s.\n", *delObjs.Key)
        }
    }
}
return err
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is
// returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
                bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
    return err
}
```



```
}
```

Execute um cenário interativo que mostre como trabalhar com buckets e objetos do 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))
```


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

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Uma situação de exemplo.

```
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.");
            }
        }
    }
}

```

Uma classe de wrapper que contém as operações.

```

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)
        .contentLength((long) (5 * mB)) // Specify the content length
        .build();

        CompletableFuture<CompletedPart> part1Future =
getAsyncClient().uploadPart(uploadPartRequest1,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(5 *
mB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(1)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Upload part 2.
        UploadPartRequest uploadPartRequest2 =
UploadPartRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .partNumber(2)
        .contentLength((long) (3 * mB))
        .build();

        CompletableFuture<CompletedPart> part2Future =
getAsyncClient().uploadPart(uploadPartRequest2,
        AsyncRequestBody.fromByteBuffer(getRandomByteBuffer(3 *
mB)))

        .thenApply(uploadPartResponse -> CompletedPart.builder()
        .partNumber(2)
        .eTag(uploadPartResponse.eTag())
        .build());

        // Combine the results of both parts.
        return CompletableFuture.allOf(part1Future, part2Future)
        .thenCompose(v -> {
            CompletedPart part1 = part1Future.join();
            CompletedPart part2 = part2Future.join();

            CompletedMultipartUpload completedMultipartUpload =
CompletedMultipartUpload.builder()
                .parts(part1, part2)
```

```

        .build();

        CompleteMultipartUploadRequest
completeMultipartUploadRequest = CompleteMultipartUploadRequest.builder()
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)
        .multipartUpload(completedMultipartUpload)
        .build();

        // Complete the multipart upload
        return
getAsyncClient().completeMultipartUpload(completeMultipartUploadRequest);
    });
})
    .thenAccept(response -> System.out.println("Multipart upload
completed successfully"))
    .exceptionally(ex -> {
        System.err.println("Failed to complete multipart upload: " +
ex.getMessage());
        throw new RuntimeException(ex);
    });
}

/**
 * Deletes an object from an S3 bucket asynchronously.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the key (file name) of the object to be deleted
 * @return a {@link CompletableFuture} that completes when the object has
been deleted
 */
public CompletableFuture<Void> deleteObjectFromBucketAsync(String bucketName,
String key) {
    DeleteObjectRequest deleteObjectRequest = DeleteObjectRequest.builder()
        .bucket(bucketName)
        .key(key)
        .build();

    CompletableFuture<DeleteObjectResponse> response =
getAsyncClient().deleteObject(deleteObjectRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {

```

```
        logger.info(key + " was deleted");
    } else {
        throw new RuntimeException("An S3 exception occurred during
delete", ex);
    }
});

return response.thenApply(r -> null);
}

/**
 * Deletes an S3 bucket asynchronously.
 *
 * @param bucket the name of the bucket to be deleted
 * @return a {@link CompletableFuture} that completes when the bucket
deletion is successful, or throws a {@link RuntimeException}
 * if an error occurs during the deletion process
 */
public CompletableFuture<Void> deleteBucketAsync(String bucket) {
    DeleteBucketRequest deleteBucketRequest = DeleteBucketRequest.builder()
        .bucket(bucket)
        .build();

    CompletableFuture<DeleteBucketResponse> response =
getAsyncClient().deleteBucket(deleteBucketRequest);
    response.whenComplete((deleteRes, ex) -> {
        if (deleteRes != null) {
            logger.info(bucket + " was deleted.");
        } else {
            throw new RuntimeException("An S3 exception occurred during
bucket deletion", ex);
        }
    });
    return response.thenApply(r -> null);
}

public CompletableFuture<String> performMultiCopy(String toBucket, String
bucketName, String key) {
    CreateMultipartUploadRequest createMultipartUploadRequest =
CreateMultipartUploadRequest.builder()
        .bucket(toBucket)
        .key(key)
        .build();
```

```

        getAsyncClient().createMultipartUpload(createMultipartUploadRequest)
            .thenApply(createMultipartUploadResponse -> {
                String uploadId = createMultipartUploadResponse.uploadId();
                System.out.println("Upload ID: " + uploadId);

                UploadPartCopyRequest uploadPartCopyRequest =
UploadPartCopyRequest.builder()
                    .sourceBucket(bucketName)
                    .destinationBucket(toBucket)
                    .sourceKey(key)
                    .destinationKey(key)
                    .uploadId(uploadId) // Use the valid uploadId.
                    .partNumber(1) // Ensure the part number is correct.
                    .copySourceRange("bytes=0-1023") // Adjust range as needed
                    .build();

                return getAsyncClient().uploadPartCopy(uploadPartCopyRequest);
            })
            .thenCompose(uploadPartCopyFuture -> uploadPartCopyFuture)
            .whenComplete((uploadPartCopyResponse, exception) -> {
                if (exception != null) {
                    // Handle any exceptions.
                    logger.error("Error during upload part copy: " +
exception.getMessage());
                } else {
                    // Successfully completed the upload part copy.
                    System.out.println("Upload Part Copy completed successfully.
ETag: " + uploadPartCopyResponse.copyPartResult().eTag());
                }
            });
        return null;
    }

    private static ByteBuffer getRandomByteBuffer(int size) {
        ByteBuffer buffer = ByteBuffer.allocate(size);
        for (int i = 0; i < size; i++) {
            buffer.put((byte) (Math.random() * 256));
        }
        buffer.flip();
        return buffer;
    }
}

```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Primeiro, importe todos os módulos necessários.

```
// 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";
```

As importações anteriores fazem referência a alguns utilitários auxiliares. Esses utilitários são locais no repositório do GitHub vinculado no início desta seção. Para sua referência, consulte as implementações a seguir desses utilitários.

```
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`;
};

```

Os objetos são armazenados em “buckets”. Vamos definir uma função para criar um 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;
};

```

Os buckets contêm “objetos”. Essa função faz upload do conteúdo de um diretório para seu bucket como 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.`);
  }
};
```

Depois de fazer upload dos objetos, confira se eles foram carregados corretamente. Você pode usar `ListObjects` para isso. Você usará a propriedade “Key”, mas também há outras propriedades úteis na resposta.

```
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`);
};
```

Às vezes é necessário copiar um objeto de um bucket em outro. Use o comando `CopyObject` para isso.

```
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();
};
```

Não há um método de SDK para obter vários objetos de um bucket. Em vez disso, você criará uma lista de objetos para baixar e iterar sobre eles.

```
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");
};
```

É hora de limpar seus recursos. Um bucket deve estar vazio para poder ser excluído. Essas duas funções esvaziam e excluem o 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`);
};
```

A função “principal” reúne tudo. Se você executar esse arquivo diretamente, a função principal será chamada.

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
suspend fun main(args: Array<String>) {  
    val usage = ""  
    Usage:  
        <bucketName> <key> <objectPath> <savePath> <toBucket>  
  
    Where:  
        bucketName - The Amazon S3 bucket to create.  
        key - The key to use.  
        objectPath - The path where the file is located (for example, C:/AWS/  
book2.pdf).  
        savePath - The path where the file is saved after it's downloaded (for  
example, C:/AWS/book2.pdf).  
        toBucket - An Amazon S3 bucket to where an object is copied to (for  
example, C:/AWS/book2.pdf).  
    ""  
  
    if (args.size != 4) {  
        println(usage)  
    }  
}
```

```
        exitProcess(1)
    }

    val bucketName = args[0]
    val key = args[1]
    val objectPath = args[2]
    val savePath = args[3]
    val toBucket = args[4]

    // Create an Amazon S3 bucket.
    createBucket(bucketName)

    // Update a local file to the Amazon S3 bucket.
    putObject(bucketName, key, objectPath)

    // Download the object to another local file.
    getObjectFromMrap(bucketName, key, savePath)

    // List all objects located in the Amazon S3 bucket.
    listBucketObs(bucketName)

    // Copy the object to another Amazon S3 bucket
    copyBucketOb(bucketName, key, toBucket)

    // Delete the object from the Amazon S3 bucket.
    deleteBucketObs(bucketName, key)

    // Delete the Amazon S3 bucket.
    deleteBucket(bucketName)
    println("All Amazon S3 operations were successfully performed")
}

suspend fun createBucket(bucketName: String) {
    val request =
        CreateBucketRequest {
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.createBucket(request)
        println("$bucketName is ready")
    }
}
```

```
suspend fun putObject(
    bucketName: String,
    objectKey: String,
    objectPath: String,
) {
    val metadataVal = mutableMapOf<String, String>()
    metadataVal["myVal"] = "test"

    val request =
        PutObjectRequest {
            bucket = bucketName
            key = objectKey
            metadata = metadataVal
            this.body = Paths.get(objectPath).asByteStream()
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        val response = s3.putObject(request)
        println("Tag information is ${response.eTag}")
    }
}

suspend fun getObjectFromMrap(
    bucketName: String,
    keyName: String,
    path: String,
) {
    val request =
        GetObjectRequest {
            key = keyName
            bucket = bucketName
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.getObject(request) { resp ->
            val myFile = File(path)
            resp.body?.writeToFile(myFile)
            println("Successfully read $keyName from $bucketName")
        }
    }
}

suspend fun listBucketObs(bucketName: String) {
    val request =
```



```
ListObjectsRequest {
    bucket = bucketName
}

S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->

    val response = s3.listObjects(request)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedEncodingException) {
        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 delObj =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delObj
        }

    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteObjects(request)
        println("$objectName was deleted from $bucketName")
    }
}

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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Kotlin.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)

- [PutObject](#)

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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'))
end
```

```
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
end
```

```
rescue Aws::Errors::ServiceError => e
  puts("Couldn't copy #{source_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Ruby.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Código para a caixa binária que executa o cenário.

```
#![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(())
}

```

Ações comuns usadas na situação.

```

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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Rust.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)

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

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
DATA(lo_session) = /aws1/cl_rt_session_aws=>create( cv_pfl ).
DATA(lo_s3) = /aws1/cl_s3_factory=>create( lo_session ).

" Create an Amazon Simple Storage Service (Amazon S3) bucket. "
TRY.
    " determine our region from our session
    DATA(lv_region) = CONV /aws1/s3_bucketlocationcnstrnt( lo_session-
>get_region( ) ).
    DATA lo_constraint TYPE REF TO /aws1/cl_s3_createbucketconf.
    " When in the us-east-1 region, you must not specify a constraint
    " In all other regions, specify the region as the constraint
    IF lv_region = 'us-east-1'.
        CLEAR lo_constraint.
    ELSE.
        lo_constraint = NEW /aws1/cl_s3_createbucketconf( lv_region ).
    ENDIF.

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
CATCH /aws1/cx_s3_bucketalrdyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
CATCH /aws1/cx_s3_bktalrdyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
```

```
ENDTRY.
```

```
"Upload an object to an S3 bucket."
```

```
TRY.
```

```
    "Get contents of file from application server."
```

```
    DATA lv_file_content TYPE xstring.
```

```
    OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
```

```
    READ DATASET iv_key INTO lv_file_content.
```

```
    CLOSE DATASET iv_key.
```

```
    lo_s3->putobject(
```

```
        iv_bucket = iv_bucket_name
```

```
        iv_key = iv_key
```

```
        iv_body = lv_file_content ).
```

```
    MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
```

```
    CATCH /aws1/cx_s3_nosuchbucket.
```

```
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
```

```
ENDTRY.
```

```
" Get an object from a bucket. "
```

```
TRY.
```

```
    DATA(lo_result) = lo_s3->getobject(
```

```
        iv_bucket = iv_bucket_name
```

```
        iv_key = iv_key ).
```

```
    DATA(lv_object_data) = lo_result->get_body( ).
```

```
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
```

```
    CATCH /aws1/cx_s3_nosuchbucket.
```

```
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
```

```
    CATCH /aws1/cx_s3_nosuchkey.
```

```
    MESSAGE 'Object key does not exist.' TYPE 'E'.
```

```
ENDTRY.
```

```
" Copy an object to a subfolder in a bucket. "
```

```
TRY.
```

```
    lo_s3->copyobject(
```

```
        iv_bucket = iv_bucket_name
```

```
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
```

```
        iv_copysource = |{ iv_bucket_name }/{ iv_key }| ).
```

```
    MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
```

```
    CATCH /aws1/cx_s3_nosuchbucket.
```

```
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
```

```
    CATCH /aws1/cx_s3_nosuchkey.
```

```
    MESSAGE 'Object key does not exist.' TYPE 'E'.
```

```
ENDTRY.

" List objects in the bucket. "
TRY.
    DATA(lo_list) = lo_s3->listobjects(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
DATA text TYPE string VALUE 'Object List - '.
DATA lv_object_key TYPE /aws1/s3_objectkey.
LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
    lv_object_key = lo_object->get_key( ).
    CONCATENATE lv_object_key ', ' INTO text.
ENDLOOP.
MESSAGE text TYPE 'I'.

" Delete the objects in a bucket. "
TRY.
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key ).
    lo_s3->deleteobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }| ).
    MESSAGE 'Objects deleted from S3 bucket.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Delete the bucket. "
TRY.
    lo_s3->deletebucket(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Deleted S3 bucket.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para SAP ABAP.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Swift

SDK para Swift

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Swift.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Ações do Amazon S3 usando SDKs da AWS

Os exemplos de código a seguir demonstram como realizar ações específicas do Amazon S3 com SDKs da AWS. Cada exemplo inclui um link para o GitHub, em que é possível encontrar instruções para configurar e executar o código.

Esses trechos chamam a API do Amazon S3 e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em [Cenários do Amazon S3 usando AWS SDKs](#).

Os exemplos a seguir incluem apenas as ações mais utilizadas. Para obter uma lista completa, consulte a [Referência de API do Amazon Simple Storage Service](#).

Exemplos

- [Usar AbortMultipartUpload com o AWS SDK ou a CLI](#)
- [Usar CompleteMultipartUpload com o AWS SDK ou a CLI](#)
- [Usar CopyObject com o AWS SDK ou a CLI](#)
- [Usar CreateBucket com o AWS SDK ou a CLI](#)
- [Usar CreateMultiRegionAccessPoint com um SDK da AWS](#)
- [Usar CreateMultipartUpload com o AWS SDK ou a CLI](#)

- [Usar CreatePresignedPost com um SDK da AWS](#)
- [Usar DeleteBucket com o AWS SDK ou a CLI](#)
- [Usar DeleteBucketAnalyticsConfiguration com uma CLI](#)
- [Usar DeleteBucketCors com o AWS SDK ou a CLI](#)
- [Usar DeleteBucketEncryption com uma CLI](#)
- [Usar DeleteBucketInventoryConfiguration com uma CLI](#)
- [Usar DeleteBucketLifecycle com o AWS SDK ou a CLI](#)
- [Usar DeleteBucketMetricsConfiguration com uma CLI](#)
- [Usar DeleteBucketPolicy com o AWS SDK ou a CLI](#)
- [Usar DeleteBucketReplication com uma CLI](#)
- [Usar DeleteBucketTagging com uma CLI](#)
- [Usar DeleteBucketWebsite com o AWS SDK ou a CLI](#)
- [Usar DeleteObject com o AWS SDK ou a CLI](#)
- [Usar DeleteObjectTagging com uma CLI](#)
- [Usar DeleteObjects com o AWS SDK ou a CLI](#)
- [Usar DeletePublicAccessBlock com uma CLI](#)
- [Usar GetBucketAccelerateConfiguration com uma CLI](#)
- [Usar GetBucketAcl com o AWS SDK ou a CLI](#)
- [Usar GetBucketAnalyticsConfiguration com uma CLI](#)
- [Usar GetBucketCors com o AWS SDK ou a CLI](#)
- [Usar GetBucketEncryption com o AWS SDK ou a CLI](#)
- [Usar GetBucketInventoryConfiguration com uma CLI](#)
- [Usar GetBucketLifecycleConfiguration com o AWS SDK ou a CLI](#)
- [Usar GetBucketLocation com o AWS SDK ou a CLI](#)
- [Usar GetBucketLogging com uma CLI](#)
- [Usar GetBucketMetricsConfiguration com uma CLI](#)
- [Usar GetBucketNotification com uma CLI](#)
- [Usar GetBucketPolicy com o AWS SDK ou a CLI](#)
- [Usar GetBucketPolicyStatus com uma CLI](#)

- [Usar GetBucketReplication com o AWS SDK ou a CLI](#)
- [Usar GetBucketRequestPayment com uma CLI](#)
- [Usar GetBucketTagging com uma CLI](#)
- [Usar GetBucketVersioning com uma CLI](#)
- [Usar GetBucketWebsite com o AWS SDK ou a CLI](#)
- [Usar GetObject com o AWS SDK ou a CLI](#)
- [Usar GetObjectAcl com o AWS SDK ou a CLI](#)
- [Usar GetObjectAttributes com o AWS SDK ou a CLI](#)
- [Usar GetObjectLegalHold com o AWS SDK ou a CLI](#)
- [Usar GetObjectLockConfiguration com o AWS SDK ou a CLI](#)
- [Usar GetObjectRetention com o AWS SDK ou a CLI](#)
- [Usar GetObjectTagging com uma CLI](#)
- [Usar GetPublicAccessBlock com uma CLI](#)
- [Usar HeadBucket com o AWS SDK ou a CLI](#)
- [Usar HeadObject com o AWS SDK ou a CLI](#)
- [Usar ListBucketAnalyticsConfigurations com uma CLI](#)
- [Usar ListBucketInventoryConfigurations com uma CLI](#)
- [Usar ListBuckets com o AWS SDK ou a CLI](#)
- [Usar ListMultipartUploads com o AWS SDK ou a CLI](#)
- [Usar ListObjectVersions com o AWS SDK ou a CLI](#)
- [Usar ListObjects com uma CLI](#)
- [Usar ListObjectsV2 com o AWS SDK ou a CLI](#)
- [Usar PutBucketAccelerateConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutBucketAcl com o AWS SDK ou a CLI](#)
- [Usar PutBucketCors com o AWS SDK ou a CLI](#)
- [Usar PutBucketEncryption com o AWS SDK ou a CLI](#)
- [Usar PutBucketLifecycleConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutBucketLogging com o AWS SDK ou a CLI](#)
- [Usar PutBucketNotification com uma CLI](#)
- [Usar PutBucketNotificationConfiguration com o AWS SDK ou a CLI](#)

- [Usar PutBucketPolicy com o AWS SDK ou a CLI](#)
- [Usar PutBucketReplication com o AWS SDK ou a CLI](#)
- [Usar PutBucketRequestPayment com uma CLI](#)
- [Usar PutBucketTagging com uma CLI](#)
- [Usar PutBucketVersioning com o AWS SDK ou a CLI](#)
- [Usar PutBucketWebsite com o AWS SDK ou a CLI](#)
- [Usar PutObject com o AWS SDK ou a CLI](#)
- [Usar PutObjectAcl com o AWS SDK ou a CLI](#)
- [Usar PutObjectLegalHold com o AWS SDK ou a CLI](#)
- [Usar PutObjectLockConfiguration com o AWS SDK ou a CLI](#)
- [Usar PutObjectRetention com o AWS SDK ou a CLI](#)
- [Usar RestoreObject com o AWS SDK ou a CLI](#)
- [Usar SelectObjectContent com o AWS SDK ou a CLI](#)
- [Usar UploadPart com o AWS SDK ou a CLI](#)
- [Usar UploadPartCopy com o AWS SDK ou a CLI](#)

Usar **AbortMultipartUpload** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar `AbortMultipartUpload`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Excluir uploads multiparte incompletos](#)
- [Trabalhar com a integridade de objetos do Amazon S3](#)

C++

SDK para C++

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

//! 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 receber detalhes da API, consulte [AbortMultipartUpload](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

Para interromper um carregamento fracionado especificado

O comando `abort-multipart-upload` a seguir interrompe um carregamento fracionado da chave `multipart/01` no bucket `amzn-s3-demo-bucket`.

```
aws s3api abort-multipart-upload \  
  --bucket amzn-s3-demo-bucket \  
  --key multipart/01 \  
  --upload-  
id dfRtDYU0WMCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P
```

O ID de upload exigido por esse comando é a saída de `create-multipart-upload` e também pode ser recuperado com `list-multipart-uploads`.

- Consulte detalhes da API em [AbortMultipartUpload](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import software.amazon.awssdk.core.exception.SdkException;  
import software.amazon.awssdk.core.sync.RequestBody;  
import software.amazon.awssdk.services.s3.S3Client;  
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;  
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;  
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;  
import software.amazon.awssdk.services.s3.model.CompletedPart;  
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;  
import software.amazon.awssdk.services.s3.model.LifecycleRule;  
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;  
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;  
import software.amazon.awssdk.services.s3.model.MultipartUpload;
```

```
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import java.util.Objects;
import java.util.UUID;

import static software.amazon.awssdk.transfer.s3.SizeConstant.KB;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 * <p>
 * For more information, see the following documentation topic:
 * <p>
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-started.html
 */

public class AbortMultipartUploadExamples {
    static final S3Client s3Client = S3Client.create();
    static final String classPathFilePath = "/multipartUploadFiles/s3-
userguide.pdf";
    static final String filePath = getFullFilePath(classPathFilePath);
    private static final Logger logger =
        LoggerFactory.getLogger(AbortMultipartUploadExamples.class);
    private static final String accountId = getAccountId();

    public static void main(String[] args) {
```

```
String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
name.

String key = UUID.randomUUID().toString();

createBucket(bucketName);
try {
    initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread");
    abortIncompleteMultipartUploadsFromList(bucketName);
    abortMultipartUploadUsingUploadId(bucketName, key);
    abortMultipartUploadsUsingLifecycleConfig(bucketName);
} catch (S3Exception e) {
    logger.error(e.getMessage());
} finally {
    deleteResources(bucketName, key);
}

}

/**
 * Aborts all incomplete multipart uploads from the specified S3 bucket.
 *
 * @param bucketName the name of the S3 bucket
 */
public static void abortIncompleteMultipartUploadsFromList(String bucketName)
{
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        abortMultipartUploadRequest = AbortMultipartUploadRequest.builder()
            .bucket(bucketName)
            .key(upload.key())
            .expectedBucketOwner(accountId)
            .uploadId(upload.uploadId())
            .build();

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
    }
}
```



```

        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}

/**
 * Aborts incomplete multipart uploads older than the specified point in
time.
 *
 * @param bucketName the name of the S3 bucket
 * @param pointInTime the cutoff time; uploads initiated before this are
aborted
 */
static void abortIncompleteMultipartUploadsOlderThan(String bucketName,
Instant pointInTime) {
    ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
        .bucket(bucketName)
        .build();

    ListMultipartUploadsResponse response =
s3Client.listMultipartUploads(listMultipartUploadsRequest);
    List<MultipartUpload> uploads = response.uploads();

    AbortMultipartUploadRequest abortMultipartUploadRequest;
    for (MultipartUpload upload : uploads) {
        logger.info("Found multipartUpload with upload ID [{}], initiated
[{}]", upload.uploadId(), upload.initiated());
        if (upload.initiated().isBefore(pointInTime)) {
            abortMultipartUploadRequest =
AbortMultipartUploadRequest.builder()
                .bucket(bucketName)
                .key(upload.key())
                .expectedBucketOwner(accountId)
                .uploadId(upload.uploadId())
                .build();

            AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
            if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {

```

```

        logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
    }
}

/**
 * Aborts a multipart upload using the upload ID.
 *
 * @param bucketName the name of the S3 bucket
 * @param key         the object key
 */
static void abortMultipartUploadUsingUploadId(String bucketName, String key)
{
    String uploadId = startUploadReturningUploadId(bucketName, key);
    AbortMultipartUploadResponse response = s3Client.abortMultipartUpload(b -
> b
        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

    if (response.sdkHttpResponse().isSuccessful()) {
        logger.info("Upload ID [{}] to bucket [{}] successfully aborted.",
uploadId, bucketName);
    }
}

/**
 * Configures a lifecycle rule to abort incomplete multipart uploads after 7
days.
 *
 * @param bucketName the name of the S3 bucket
 */
static void abortMultipartUploadsUsingLifecycleConfig(String bucketName) {
    Collection<LifecycleRule> lifeCycleRules =
List.of(LifecycleRule.builder()
        .abortIncompleteMultipartUpload(b -> b.
            daysAfterInitiation(7))
        .status("Enabled")
        .filter(SdkBuilder::build) // Filter element is required.
        .build());
}

```

```

        PutBucketLifecycleConfigurationResponse response =
s3Client.putBucketLifecycleConfiguration(b -> b
        .bucket(bucketName)
        .lifecycleConfiguration(b1 -> b1.rules(lifeCycleRules)));

        if (response.sdkHttpResponse().isSuccessful()) {
            logger.info("Rule to abort incomplete multipart uploads added to
bucket.");
        } else {
            logger.error("Unsuccessfully applied rule. HTTP status code is [{}]",
response.sdkHttpResponse().statusCode());
        }
    }

    /*****
    Multipart upload methods
    *****/

    static void initiateAndInterruptMultiPartUpload(String bucketName, String
key, String threadName) {
        Runnable upload = () -> {
            try {
                doMultipartUpload(bucketName, key);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        };
        Thread uploadThread = new Thread(upload, threadName);
        uploadThread.start();
        try {
            Thread.sleep(Duration.ofSeconds(1).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        uploadThread.interrupt();
    }

    static Instant initiateAndInterruptTwoUploads(String bucketName, String key)
{
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread1");
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
    }

```

```
    }
    Instant secondUploadInstant = Instant.now();
    initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread2");
    return secondUploadInstant;
}

static void doMultipartUpload(String bucketName, String key) {
    String uploadId = step1CreateMultipartUpload(bucketName, key);
    List<CompletedPart> completedParts = step2UploadParts(bucketName, key,
uploadId);
    step3CompleteMultipartUpload(bucketName, key, uploadId, completedParts);
}

static String step1CreateMultipartUpload(String bucketName, String key) {
    CreateMultipartUploadResponse createMultipartUploadResponse =
s3Client.createMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key));
    return createMultipartUploadResponse.uploadId();
}

static List<CompletedPart> step2UploadParts(String bucketName, String key,
String uploadId) {
    int partNumber = 1;
    List<CompletedPart> completedParts = new ArrayList<>();
    ByteBuffer bb = ByteBuffer.allocate(Long.valueOf(1024 * KB).intValue());

    try (RandomAccessFile file = new RandomAccessFile(filePath, "r")) {
        long fileSize = file.length();
        long position = 0;
        while (position < fileSize) {
            file.seek(position);
            long read = file.getChannel().read(bb);

            bb.flip();
            UploadPartRequest uploadPartRequest = UploadPartRequest.builder()
                .bucket(bucketName)
                .key(key)
                .uploadId(uploadId)
                .partNumber(partNumber)
                .build();

            UploadPartResponse partResponse = s3Client.uploadPart(
                uploadPartRequest,
```

```
RequestBody.fromByteBuffer(bb));

CompletedPart part = CompletedPart.builder()
    .partNumber(partNumber)
    .eTag(partResponse.eTag())
    .build();
completedParts.add(part);
logger.info("Part {} upload", partNumber);

bb.clear();
position += read;
partNumber++;
}
} catch (IOException | S3Exception e) {
    logger.error(e.getMessage());
    return null;
}
return completedParts;
}

static void step3CompleteMultipartUpload(String bucketName, String key,
String uploadId, List<CompletedPart> completedParts) {
    s3Client.completeMultipartUpload(b -> b
        .bucket(bucketName)
        .key(key)
        .uploadId(uploadId)

    .multipartUpload(CompletedMultipartUpload.builder().parts(completedParts).build()));
}

static String startUploadReturningUploadId(String bucketName, String key) {
    String uploadId = step1CreateMultipartUpload(bucketName, key);
    doMultipartUploadWithUploadId(bucketName, key, uploadId);
    return uploadId;
}

static void doMultipartUploadWithUploadId(String bucketName, String key,
String uploadId) {
    new Thread(() -> {
        try {
            List<CompletedPart> completedParts = step2UploadParts(bucketName,
key, uploadId);
            step3CompleteMultipartUpload(bucketName, key, uploadId,
completedParts);
        }
    })
    .start();
}
```

```

        } catch (SdkException e) {
            logger.error(e.getMessage());
        }
    }, "upload thread").start();
    try {
        Thread.sleep(Duration.ofSeconds(2L).toMillis());
    } catch (InterruptedException e) {
        logger.error(e.getMessage());
        System.exit(1);
    }
}

/*****
Resource handling methods
*****/

static void createBucket(String bucketName) {
    logger.info("Creating bucket: [{}]", bucketName);
    s3Client.createBucket(b -> b.bucket(bucketName));
    try (S3Waiter s3Waiter = s3Client.waiter()) {
        s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
    }
    logger.info("Bucket created.");
}

static void deleteResources(String bucketName, String key) {
    logger.info("Deleting resources ...");
    s3Client.deleteObject(b -> b.bucket(bucketName).key(key));
    s3Client.deleteBucket(b -> b.bucket(bucketName));
    try (S3Waiter s3Waiter = s3Client.waiter()) {
        s3Waiter.waitUntilBucketNotExists(b -> b.bucket(bucketName));
    }
    logger.info("Resources deleted.");
}

private static String getAccountId() {
    try (StsClient stsClient = StsClient.create()) {
        return stsClient.getCallerIdentity().account();
    }
}

static String getFullFilePath(String filePath) {
    URL uploadDirectoryURL =
PerformMultiPartUpload.class.getResource(filePath);

```

```
String fullFilePath;
try {
    fullFilePath =
Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
} catch (URISyntaxException e) {
    throw new RuntimeException(e);
}
return fullFilePath;
}
}
```

- Para receber detalhes da API, consulte [AbortMultipartUpload](#) na Referência da API AWS SDK for Java 2.x.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando interrompe os carregamentos fracionados criados há mais de cinco dias.

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

Exemplo 2: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de, 2014.

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

Exemplo 3: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de 2014, às, 2014, 10:45:37.

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

- Consulte detalhes da API em [AbortMultipartUpload](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando interrompe os carregamentos fracionados criados há mais de cinco dias.

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

Exemplo 2: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de, 2014.

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

Exemplo 3: este comando interrompe os carregamentos fracionados criados antes de 2 de janeiro de 2014, às, 2014, 10:45:37.

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

- Consulte detalhes da API em [AbortMultipartUpload](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **CompleteMultipartUpload** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar CompleteMultipartUpload.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Executar uma cópia multipart](#)
- [Usar somas de verificação](#)
- [Trabalhar com a integridade de objetos do Amazon S3](#)

C++

SDK para C++

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
///  
//! 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 receber detalhes da API, consulte [CompleteMultipartUpload](#) na Referência da API do AWS SDK para C++.

CLI

AWS CLI

O seguinte comando conclui um upload fracionado da chave `multipart/01` no bucket `amzn-s3-demo-bucket`:

```
aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZlJF.Yxwh6XG7WfS2vC4to6HiV6YjLx.cph0gtNBtJ8P
```

O ID de upload exigido por esse comando é a saída de `create-multipart-upload` e também pode ser recuperado com `list-multipart-uploads`.

A opção de upload fracionado no comando acima usa uma estrutura JSON que descreve quais partes do upload fracionado devem ser reagrupadas no arquivo completo. Neste exemplo, o prefixo `file://` é usado para carregar a estrutura JSON de um arquivo na pasta local chamada `mpustruct`.

`mpustruct`:

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

```
    },  
    {  
      "ETag": "d0a0112e841abec9c9ec83406f0159c8",  
      "PartNumber": 3  
    }  
  ]  
}
```

O valor de ETag de cada parte carregada é gerado sempre que você fizer o upload de uma parte usando o comando `upload-part`. Ele também pode ser recuperado ao chamar `list-parts` ou calculado fazendo a soma de verificação MD5 de cada parte.

Resultado:

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

- Consulte detalhes da API em [CompleteMultipartUpload](#) na Referência de comandos da AWS CLI.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
// 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 obter detalhes da API, consulte [CompleteMultipartUpload](#) na Referência da API do AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **CopyObject** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar CopyObject.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Conheça os conceitos básicos](#)
- [Conceitos básicos de criptografia](#)
- [Introdução ao armazenamento de objetos](#)
- [Fazer solicitações condicionais](#)

.NET

SDK para .NET (v4)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [CopyObject](#) em Referência da API AWS SDK para .NET.

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Copie um objeto usando uma solicitação 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;
    }
}

```

- Consulte detalhes da API em [CopyObject](#) em Referência da API AWS SDK para .NET.

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

#####
# 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
}
```

- Consulte detalhes da API em [CopyObject](#) na Referência de comandos da AWS CLI.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();
}
```

- Consulte detalhes da API em [CopyObject](#) em Referência da API AWS SDK para C++.

CLI

AWS CLI

O comando a seguir copia um objeto de bucket-1 para bucket-2:

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

Resultado:

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

- Consulte detalhes da API em [CopyObject](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

- Consulte detalhes da API em [CopyObject](#) em Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Copie um objeto usando um [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);
}

```

Use um [S3TransferManager](#) para [copiar um objeto](#) de um bucket para outro. Veja o [arquivo completo](#) e [teste](#).

```

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();
    }
```

- Consulte detalhes da API em [CopyObject](#) em Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Copie o 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;
    }
}
};

```

Copie o objeto caso seu ETag não corresponda ao fornecido.

```

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"));
  }
}

```

Copie o objeto caso seu ETag não corresponda ao fornecido.

```

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"));
    }
}
```

Copie o objeto usando a condição de que ele tenha sido criado ou modificado em determinado período.

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

Copie o objeto usando a condição de que ele não tenha sido criado ou modificado em determinado período.

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

- Consulte detalhes da API em [CopyObject](#) em Referência da API AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
suspend fun copyBucketObject(  
    fromBucket: String,  
    objectKey: String,  
    toBucket: String,  
) {  
    var encodedUrl = ""  
    try {  
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",  
StandardCharsets.UTF_8.toString())  
    } catch (e: UnsupportedEncodingException) {  
        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)
}
}
```

- Consulte detalhes da API em [CopyObject](#) na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Cópia simples de um 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.");
}
```

- Consulte detalhes da API em [CopyObject](#) em Referência da API AWS SDK para PHP.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando copia o objeto “sample.txt” do bucket “test-files” para o mesmo bucket, mas com uma nova chave de “sample-copy.txt”.

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

Exemplo 2: este comando copia o objeto “sample.txt” do bucket “test-files” para o bucket “backup-files” com uma nova chave 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
```

Exemplo 3: este comando baixa o objeto “sample.txt” do bucket “test-files” em um arquivo local com o nome “local-sample.txt”.

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

Exemplo 4: um único objeto é baixado no arquivo especificado. O arquivo baixado encontra-se em c:\downloads\data\archive.zip

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

Exemplo 5: todos os objetos que correspondem ao prefixo de chave especificado são baixados na pasta local. A hierarquia relativa de chaves será preservada como subpastas no local geral do download.

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

- Consulte detalhes da API em [CopyObject](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando copia o objeto “sample.txt” do bucket “test-files” para o mesmo bucket, mas com uma nova chave de “sample-copy.txt”.

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

Exemplo 2: este comando copia o objeto “sample.txt” do bucket “test-files” para o bucket “backup-files” com uma nova chave 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
```

Exemplo 3: este comando baixa o objeto “sample.txt” do bucket “test-files” em um arquivo local com o nome “local-sample.txt”.

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

Exemplo 4: um único objeto é baixado no arquivo especificado. O arquivo baixado encontra-se em c:\downloads\data\archive.zip

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

Exemplo 5: todos os objetos que correspondem ao prefixo de chave especificado são baixados na pasta local. A hierarquia relativa de chaves será preservada como subpastas no local geral do download.

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

Exemplo 6: baixa um único objeto em um arquivo local usando o download paralelo de várias partes para melhorar o desempenho em arquivos grandes.

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

Exemplo 7: baixa um objeto grande usando o modo RANGE com um tamanho de parte personalizado de 64 MB. O modo RANGE usa solicitações HTTP de intervalo de bytes e funciona para todos os objetos, independentemente de como eles foram carregados.

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

Exemplo 8: baixa todos os objetos que correspondem ao prefixo de chave em uma pasta local usando o download de várias partes com downloads simultâneos de arquivos.

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

Exemplo 9: baixa um único objeto usando o download de várias partes com 20 conexões simultâneas para obter o throughput máximo em redes de alta largura de banda.

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

Exemplo 10: baixa todos os objetos que correspondem ao prefixo de chave para uma pasta local usando o download de várias partes, continuando com as falhas de arquivos individuais em vez de interrompê-las.

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

- Consulte detalhes da API em [CopyObject](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

Copie um objeto usando uma solicitação 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

```

- Consulte detalhes da API em [CopyObject](#) na Referência da API do AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Copie um 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 um objeto e adicione criptografia do lado do servidor ao 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

  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__

```

- Consulte detalhes da API em [CopyObject](#) em Referência da API AWS SDK para Ruby.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [CopyObject](#) em Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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.
```

- Consulte detalhes da API em [CopyObject](#) na Referência da API AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
import AWSS3  
  
public func copyFile(from sourceBucket: String, name: String, to destBucket:  
String) async throws {
```

```
let srcUrl = ("\"(sourceBucket)/\n(name)").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)\n\nlet input = CopyObjectInput(\n    bucket: destBucket,\n    copySource: srcUrl,\n    key: name\n)\ndo {\n    _ = try await client.copyObject(input: input)\n}\ncatch {\n    print("ERROR: ", dump(error, name: "Copying an object."))\n    throw error\n}\n}
```

- Consulte detalhes da API em [CopyObject](#) na Referência da API AWS SDK para Swift.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **CreateBucket** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar CreateBucket.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Conheça os conceitos básicos](#)
- [Conceitos básicos do gerenciamento de configurações](#)
- [Conceitos básicos de arquivos de atributos de machine learning](#)
- [Introdução ao armazenamento de objetos](#)
- [Gerenciar mensagens grandes usando o S3](#)
- [Trabalhar com objetos versionados](#)

.NET

SDK para .NET (v4)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK para .NET.

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie um bucket com o bloqueio 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;
    }
}
```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK para .NET.

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
#####
# 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
}

```

```
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
}
```

- Consulte detalhes da API em [CreateBucket](#) na Referência de comandos da AWS CLI.

C++

SDK para C++

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();
}
```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

Exemplo 1: como criar um bucket

O seguinte exemplo de `create-bucket` cria um bucket chamado `amzn-s3-demo-bucket`:

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

Resultado:

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

Para obter mais informações, consulte [Como criar um bucket](#) no Guia do usuário do Amazon S3.

Exemplo 2: como criar um bucket com aplicação de políticas de proprietário

O exemplo de `create-bucket` a seguir cria um bucket chamado `amzn-s3-demo-bucket` que usa a configuração aplicada de proprietário de bucket para a propriedade de objetos do S3.

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

Resultado:

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

Para obter mais informações, consulte [Controlar a propriedade de objetos e desabilitar ACLs](#), no Guia do usuário do Amazon S3.

Exemplo 3: como criar um bucket fora da região ``us-east

O exemplo de `create-bucket` a seguir cria um bucket chamado `amzn-s3-demo-bucket` na região `eu-west-1`. Regiões fora da `us-east-1` exigem que o `LocationConstraint` adequado seja especificado para poder criar o bucket na região desejada.

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

Resultado:

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

Para obter mais informações, consulte [Como criar um bucket](#) no Guia do usuário do Amazon S3.

- Consulte detalhes da API em [CreateBucket](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Criar um bucket com a configuração padrão.

```
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
}
```

Crie um bucket com bloqueio de objetos e espere que ele exista.

```
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  
}
```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API do AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie um 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);
        }
    });
}
```

Crie um bucket com o bloqueio 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");
}
```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie o 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [CreateBucket](#) na Referência da API do AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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")  
    }  
}
```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie um 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.");
}
```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK para PHP.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie um bucket com as configurações padrão.

```
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

```

Crie um bucket versionado com uma configuração 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
```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
```

```
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
  # This is a client-side object until
  # create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
    else
      @bucket.client.get_bucket_location(bucket:
@bucket.name).location_constraint
    end
  rescue Aws::Errors::ServiceError => e
    "Couldn't get the location of #{@bucket.name}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  region = "us-west-2"
  wrapper = BucketCreateWrapper.new(Aws::S3::Bucket.new("amzn-s3-demo-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)
```

```
puts "Created bucket #{wrapper.bucket.name}."
puts "Your bucket's region is: #{wrapper.location}"
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK para Ruby.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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))
    }
}))
}

```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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_bktalrdyownedbyyou.
MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.
```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
    }
}
```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK para Swift.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **CreateMultiRegionAccessPoint** com um SDK da AWS

O código de exemplo a seguir mostra como usar `CreateMultiRegionAccessPoint`.

Kotlin

SDK para Kotlin

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Configure o cliente de controle do S3 para enviar a solicitação à região 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
    }
}
```

Crie o ponto de acesso multirregional.

```
suspend fun createMrap(
    s3Control: S3ControlClient,
    accountIdParam: String,
    bucketName1: String,
    bucketName2: String,
    mrmapName: String,
): String {
    println("Creating MRAP ...")
    val createMrapResponse: CreateMultiRegionAccessPointResponse =
        s3Control.createMultiRegionAccessPoint {
            accountId = accountIdParam
            clientToken = UUID.randomUUID().toString()
            details {
                name = mrmapName
                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"
}

```

Aguarde até que o ponto de acesso multirregional fique disponível.

```

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 ter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para Kotlin](#).

- Para obter detalhes da API, consulte [CreateMultiRegionAccessPoint](#) na Referência da API do AWS SDK para Kotlin.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **CreateMultipartUpload** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar CreateMultipartUpload.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Executar uma cópia multipart](#)
- [Usar somas de verificação](#)
- [Trabalhar com a integridade de objetos do Amazon S3](#)

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
//! 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 receber detalhes da API, consulte [CreateMultipartUpload](#) na Referência da API do AWS SDK para C++.

CLI

AWS CLI

O seguinte comando cria um upload fracionado no bucket `amzn-s3-demo-bucket` com a chave `multipart/01`:

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

Resultado:

```
{
  "Bucket": "amzn-s3-demo-bucket",
```

```

    "UploadId":
    "dfRtDYU0WWCCcH43C3WfbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P3
    "Key": "multipart/01"
}

```

O arquivo concluído terá o nome 01 e estará em uma pasta chamada multipart no bucket amzn-s3-demo-bucket. Salve o ID de upload, a chave e o nome do bucket para usar com o comando upload-part.

- Consulte detalhes da API em [CreateMultipartUpload](#) na Referência de comandos da AWS CLI.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

// 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 {
        size_of_chunk
    };
    let part_number = chunk_index + 1;
    let upload_part_res = client
        .upload_part()
        .bucket(&bucket_name)
        .key(&key)
        .upload_id(&upload_id)
        .part_number(part_number)
        .data(chunk)
        .send()
        .await?;
    upload_parts.push(upload_part_res);
}
let complete_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .upload_id(&upload_id)
    .multipart_upload(upload_parts)
    .send()
    .await?;

```

```

    } 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 obter detalhes da API, consulte [CreateMultipartUpload](#) na Referência da API do AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **CreatePresignedPost** com um SDK da AWS

O código de exemplo a seguir mostra como usar `CreatePresignedPost`.

.NET

SDK para .NET (v4)

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie um URL POST pré-assinado.

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

- Consulte detalhes da API em [CreatePresignedPost](#) na Referência de API do AWS SDK para .NET.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucket** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar DeleteBucket.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Conheça os conceitos básicos](#)
- [Conceitos básicos do gerenciamento de configurações](#)
- [Conceitos básicos de arquivos de atributos de machine learning](#)
- [Introdução ao armazenamento de objetos](#)

.NET

SDK para .NET (v4)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
/// <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;
    }
}

```

- Consulte detalhes da API em [DeleteBucket](#) na Referência da API AWS SDK para .NET.

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

#####
# 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
}
```

- Consulte detalhes da API em [DeleteBucket](#) na Referência de comandos da AWS CLI.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();
}
```

- Consulte detalhes da API em [DeleteBucket](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte comando exclui o bucket `amzn-s3-demo-bucket`:

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

- Consulte detalhes da API em [DeleteBucket](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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  
}
```

- Consulte detalhes da API em [DeleteBucket](#) na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [DeleteBucket](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua o 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [DeleteBucket](#) na Referência da API do AWS SDK para JavaScript.

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua um bucket vazio.

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

- Consulte detalhes da API em [DeleteBucket](#) na Referência da API AWS SDK para PHP.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando remove todos os objetos e versões de objetos do bucket “test-files” e, em seguida, exclui o bucket. O comando solicitará a confirmação antes de continuar. Adicione a opção -Force para ignorar a confirmação. Observe que os buckets que não estão vazios não podem ser excluídos.

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

- Consulte detalhes da API em [DeleteBucket](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando remove todos os objetos e versões de objetos do bucket “test-files” e, em seguida, exclui o bucket. O comando solicitará a confirmação antes de continuar. Adicione a opção -Force para ignorar a confirmação. Observe que os buckets que não estão vazios não podem ser excluídos.

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

- Consulte detalhes da API em [DeleteBucket](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

- Consulte detalhes da API em [DeleteBucket](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
# 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
```

- Consulte detalhes da API em [DeleteBucket](#) na Referência da API AWS SDK para Ruby.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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))
            }
        }
    }
}
```

- Consulte detalhes da API em [DeleteBucket](#) na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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.
```

- Consulte detalhes da API em [DeleteBucket](#) na Referência da API AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
    }
}
```

- Consulte detalhes da API em [DeleteBucket](#) na Referência da API AWS SDK para Swift.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketAnalyticsConfiguration** com uma CLI

Os exemplos de código a seguir mostram como usar DeleteBucketAnalyticsConfiguration.

CLI

AWS CLI

Para excluir uma configuração de análise de um bucket

O exemplo delete-bucket-analytics-configuration a seguir recupera a configuração referente ao bucket e ID especificados.

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

Este comando não produz saída.

- Consulte detalhes da API em [DeleteBucketAnalyticsConfiguration](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando remove o filtro de análise com o nome “testfilter” no bucket do S3 em questão.

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

- Consulte detalhes da API em [DeleteBucketAnalyticsConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando remove o filtro de análise com o nome “testfilter” no bucket do S3 em questão.

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

- Consulte detalhes da API em [DeleteBucketAnalyticsConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketCors** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar DeleteBucketCors.

.NET

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [DeleteBucketCors](#) na Referência da API AWS SDK para .NET.

CLI

AWS CLI

O seguinte comando exclui a configuração de compartilhamento de recursos de origem cruzada do bucket `amzn-s3-demo-bucket`:

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

- Consulte detalhes da API em [DeleteBucketCors](#) na Referência de comandos da AWS CLI.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

- Consulte detalhes da API em [DeleteBucketCors](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #
  # @return [Boolean] True if the CORS rules were deleted; otherwise, false.
```



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

end
```

- Consulte detalhes da API em [DeleteBucketCors](#) na Referência da API AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [DeleteBucketCors](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketEncryption** com uma CLI

Os exemplos de código a seguir mostram como usar DeleteBucketEncryption.

CLI

AWS CLI

Para excluir a configuração de criptografia do lado do servidor de um bucket

O exemplo delete-bucket-encryption a seguir exclui a configuração de criptografia do lado do servidor do bucket especificado.

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

Este comando não produz saída.

- Consulte detalhes da API em [DeleteBucketEncryption](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: isso desabilita a criptografia habilitada para o bucket do S3 fornecido.

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

Saída:

```
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
```

- Consulte detalhes da API em [DeleteBucketEncryption](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: isso desabilita a criptografia habilitada para o bucket do S3 fornecido.

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

Saída:

```
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
```

- Consulte detalhes da API em [DeleteBucketEncryption](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketInventoryConfiguration** com uma CLI

Os exemplos de código a seguir mostram como usar DeleteBucketInventoryConfiguration.

CLI

AWS CLI

Para excluir a configuração de inventário de um bucket

O exemplo delete-bucket-inventory-configuration a seguir exclui a configuração do inventário com ID 1 do bucket especificado.

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

Este comando não produz saída.

- Consulte detalhes da API em [DeleteBucketInventoryConfiguration](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando remove o inventário denominado “testInventoryName” correspondente ao bucket do S3 em questão.

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

Saída:

```
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
```

- Consulte detalhes da API em [DeleteBucketInventoryConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando remove o inventário denominado “testInventoryName” correspondente ao bucket do S3 em questão.

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

Saída:

```
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
```

- Consulte detalhes da API em [DeleteBucketInventoryConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketLifecycle** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar DeleteBucketLifecycle.

.NET

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [DeleteBucketLifecycle](#) na Referência da API AWS SDK para .NET.

CLI

AWS CLI

O seguinte comando exclui a configuração de ciclo de vida do bucket `amzn-s3-demo-bucket`:

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

- Consulte detalhes da API em [DeleteBucketLifecycle](#) na Referência de comandos da AWS CLI.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

- Consulte detalhes da API em [DeleteBucketLifecycle](#) na Referência da API AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [DeleteBucketLifecycle](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketMetricsConfiguration** com uma CLI

Os exemplos de código a seguir mostram como usar DeleteBucketMetricsConfiguration.

CLI

AWS CLI

Para excluir uma configuração de métricas referente a um bucket

O exemplo delete-bucket-metrics-configuration a seguir remove a configuração referente ao bucket e ID especificados.

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

Este comando não produz saída.

- Consulte detalhes da API em [DeleteBucketMetricsConfiguration](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando remove o filtro de métricas com o nome “testmetrics” no bucket do S3 em questão.

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

- Consulte detalhes da API em [DeleteBucketMetricsConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando remove o filtro de métricas com o nome “testmetrics” no bucket do S3 em questão.


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

- Consulte detalhes da API em [DeleteBucketMetricsConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketPolicy** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar DeleteBucketPolicy.

C++

SDK para C++

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();
}
```

- Consulte detalhes da API em [DeleteBucketPolicy](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte comando exclui uma política de bucket do bucket `amzn-s3-demo-bucket`:

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

- Consulte detalhes da API em [DeleteBucketPolicy](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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);
    }
}
```

- Consulte detalhes da API em [DeleteBucketPolicy](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua a política de 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [DeleteBucketPolicy](#) na Referência da API do AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
suspend fun deleteS3BucketPolicy(bucketName: String?) {  
    val request =  
        DeleteBucketPolicyRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteBucketPolicy(request)  
        println("Done!")  
    }  
}
```

- Consulte detalhes da API em [DeleteBucketPolicy](#) na Referência da API AWS SDK para Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando remove a política de bucket associada ao bucket do S3 em questão.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [DeleteBucketPolicy](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando remove a política de bucket associada ao bucket do S3 em questão.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [DeleteBucketPolicy](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

- Consulte detalhes da API em [DeleteBucketPolicy](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
# 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
```

- Consulte detalhes da API em [DeleteBucketPolicy](#) na Referência da API AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [DeleteBucketPolicy](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketReplication** com uma CLI

Os exemplos de código a seguir mostram como usar DeleteBucketReplication.

CLI

AWS CLI

O seguinte comando exclui uma configuração de replicação de um bucket denominado amzn-s3-demo-bucket:

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

- Consulte detalhes da API em [DeleteBucketReplication](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: exclui a configuração de replicação associada ao bucket denominado "amzn-s3-demo-bucket". Observe que essa operação exige permissão para a ação `s3:DeleteReplicationConfiguration`. Será solicitada uma confirmação antes que a operação continue. Para ignorar a confirmação, use a opção `-Force`.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Consulte detalhes da API em [DeleteBucketReplication](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: exclui a configuração de replicação associada ao bucket denominado "amzn-s3-demo-bucket". Observe que essa operação exige permissão para a ação `s3:DeleteReplicationConfiguration`. Será solicitada uma confirmação antes que a operação continue. Para ignorar a confirmação, use a opção `-Force`.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Consulte detalhes da API em [DeleteBucketReplication](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketTagging** com uma CLI

Os exemplos de código a seguir mostram como usar `DeleteBucketTagging`.

CLI

AWS CLI

O seguinte comando exclui uma configuração de marcação de um bucket denominado `amzn-s3-demo-bucket`:

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

- Consulte detalhes da API em [DeleteBucketTagging](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando remove todas as tags associadas ao bucket do S3 em questão.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
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
```

- Consulte detalhes da API em [DeleteBucketTagging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando remove todas as tags associadas ao bucket do S3 em questão.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
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
```

- Consulte detalhes da API em [DeleteBucketTagging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketWebsite** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar DeleteBucketWebsite.

C++

SDK para C++

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();  
}
```

- Consulte detalhes da API em [DeleteBucketWebsite](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte comando exclui uma configuração de site do bucket `amzn-s3-demo-bucket`:

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

- Consulte detalhes da API em [DeleteBucketWebsite](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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);
        }
    }
}
```

- Consulte detalhes da API em [DeleteBucketWebsite](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua a configuração de site do 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).

- Consulte detalhes da API em [DeleteBucketWebsite](#) na Referência da API do AWS SDK para JavaScript.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando desabilita a propriedade de hospedagem de site estático do bucket do S3 em questão.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
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
```

- Consulte detalhes da API em [DeleteBucketWebsite](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando desabilita a propriedade de hospedagem de site estático do bucket do S3 em questão.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
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
```

- Consulte detalhes da API em [DeleteBucketWebsite](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteObject** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar `DeleteObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Trabalhar com a integridade de objetos do Amazon S3](#)
- [Trabalhar com objetos versionados](#)

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua um objeto de um bucket do S3 sem versionamento.

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

/// <summary>
/// This example shows how to delete an object from a non-versioned Amazon
/// Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class DeleteObject
{
```

```
/// <summary>
/// The Main method initializes the necessary variables and then calls
/// the DeleteObjectNonVersionedBucketAsync method to delete the object
/// named by the keyName parameter.
/// </summary>
public static async Task Main()
{
    const string bucketName = "amzn-s3-demo-bucket";
    const string keyName = "testfile.txt";

    // If the Amazon S3 bucket is located in an AWS Region other than the
    // Region of the default account, define the AWS Region for the
    // Amazon S3 bucket in your call to the AmazonS3Client constructor.
    // For example RegionEndpoint.USSouth1.
    IAmazonS3 client = new AmazonS3Client();
    await DeleteObjectNonVersionedBucketAsync(client, bucketName,
keyName);
}

/// <summary>
/// The DeleteObjectNonVersionedBucketAsync takes care of deleting the
/// desired object from the named bucket.
/// </summary>
/// <param name="client">An initialized Amazon S3 client used to delete
/// an object from an Amazon S3 bucket.</param>
/// <param name="bucketName">The name of the bucket from which the
/// object will be deleted.</param>
/// <param name="keyName">The name of the object to delete.</param>
public static async Task DeleteObjectNonVersionedBucketAsync(IAmazonS3
client, string bucketName, string keyName)
{
    try
    {
        var deleteObjectRequest = new DeleteObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
        };

        Console.WriteLine($"Deleting object: {keyName}");
        await client.DeleteObjectAsync(deleteObjectRequest);
        Console.WriteLine($"Object: {keyName} deleted from
{bucketName}.");
    }
}
```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error encountered on server.
Message: '{ex.Message}' when deleting an object.");
        }
    }
}
```

Exclua um objeto de um bucket do S3 com versionamento.

```
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;
    }
}

```

- Consulte detalhes da API em [DeleteObject](#) na Referência da API AWS SDK para .NET.

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

#####
# 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
}
```

- Consulte detalhes da API em [DeleteObject](#) na Referência de comandos da AWS CLI.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();
}

```

- Consulte detalhes da API em [DeleteObject](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte comando exclui um objeto `test.txt` do bucket `amzn-s3-demo-bucket`:

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

Se o versionamento de bucket estiver habilitado, a saída conterá o ID de versão do marcador excluído:

```

{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tND1w1Gq",
  "DeleteMarker": true
}

```


Para obter mais informações sobre como excluir objetos, consulte Excluir objetos no Guia do desenvolvedor do Amazon S3.

- Consulte detalhes da API em [DeleteObject](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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)
    } else {
        deleted = true
    }
}
return deleted, err
}
```

- Consulte detalhes da API em [DeleteObject](#) na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [DeleteObject](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua um 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;  
    }  
}  
};
```

- Consulte detalhes da API em [DeleteObject](#) na Referência da API AWS SDK para JavaScript.

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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;
    }
}
```

- Consulte detalhes da API em [DeleteObject](#) na Referência da API do AWS SDK para PHP.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua um 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

```

Reverta um objeto para uma versão anterior excluindo versões posteriores do 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}."
    )

```

Restaure um objeto excluído removendo o marcador de exclusão ativo do 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)
```

Crie um manipulador do Lambda que remova um marcador de exclusão de um objeto do S3. Esse manipulador pode ser usado para limpar com eficiência marcadores de exclusão estranhos em um bucket versionado.

```
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(
obj_version_id
            "Object %s, version %s is a delete marker.", obj_key,
        )
        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
retrieved.

            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,
    }

```

- Consulte detalhes da API em [DeleteObject](#) na Referência da API AWS SDK para Python (Boto3).

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [DeleteObject](#) na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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.
```

- Consulte detalhes da API em [DeleteObject](#) na Referência da API AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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  
    }  
}
```

- Consulte detalhes da API em [DeleteObject](#) na Referência da API AWS SDK para Swift.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteObjectTagging** com uma CLI

Os exemplos de código a seguir mostram como usar DeleteObjectTagging.

CLI

AWS CLI

Para excluir os conjuntos de tags de um objeto

O exemplo delete-object-tagging a seguir exclui a tag com a chave especificada do objeto doc1.rtf.

```
aws s3api delete-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc1.rtf
```

Este comando não produz saída.

- Consulte detalhes da API em [DeleteObjectTagging](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando remove todas as tags associadas ao objeto com a chave “testfile.txt” no bucket do S3 em questão.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -
Select '^Key'
```

Saída:

```
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
```

- Consulte detalhes da API em [DeleteObjectTagging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando remove todas as tags associadas ao objeto com a chave “testfile.txt” no bucket do S3 em questão.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

Saída:

```
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
```

- Consulte detalhes da API em [DeleteObjectTagging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteObjects** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar DeleteObjects.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Conheça os conceitos básicos](#)
- [Excluir todos os objetos em um bucket](#)
- [Introdução ao armazenamento de objetos](#)

.NET

SDK para .NET (v4)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK para .NET. SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua vários objetos de um bucket do S3 sem versionamento.

```
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;
    }
}

```

Exclua vários objetos de um bucket do S3 com versionamento.

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete objects in a version-enabled Amazon
/// Simple StorageService (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    public static async Task Main()
    {

```

```

        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region for your Amazon S3 bucket is different from
        // the AWS Region of the default user, define the AWS Region for
        // the Amazon S3 bucket and pass it to the client constructor
        // like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
        IAmazonS3 s3Client;

        s3Client = new AmazonS3Client();
        await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
    }

    /// <summary>
    /// This method removes multiple versions and objects from a
    /// version-enabled Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
    {
        // Delete objects (specifying object version in the request).
        await DeleteObjectVersionsAsync(client, bucketName);

        // Delete objects (without specifying object version in the request).
        var deletedObjects = await DeleteObjectsAsync(client, bucketName);

        // Additional exercise - remove the delete markers Amazon S3 returned
from
        // the preceding response. This results in the objects reappearing
        // in the bucket (you can verify the appearance/disappearance of
        // objects in the console).
        await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
    }

    /// <summary>

```

```

        /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
        /// them again. The method returns a list of the Amazon S3 objects
deleted.
        /// </summary>
        /// <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>
/// <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;
}
}

```

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK para .NET.

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_items_in_bucket

```

```

#
# This function deletes the specified list of keys from the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.
#     $2 - A list of keys in the bucket to delete.

# Returns:
#     0 - If successful.
#     1 - If it fails.
#####
function delete_items_in_bucket() {
    local bucket_name=$1
    local keys=$2
    local response

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

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

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

```

- Consulte detalhes da API em [DeleteObjects](#) na Referência de comandos da AWS CLI.

C++

SDK para C++

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();
}
```

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte comando exclui um objeto do bucket `amzn-s3-demo-bucket`:

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

`delete.json` é um documento JSON no diretório atual que especifica o objeto a ser excluído:

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

Resultado:

```
{
  "Deleted": [
    {
      "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUP01koHwzU",
      "Key": "test1.txt",
      "DeleteMarker": true
    }
  ]
}
```

- Consulte detalhes da API em [DeleteObjects](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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);
        }
    }
}
```

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua vários 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;  
  }  
}  
};
```

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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")
}
}
```

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua um conjunto de objetos de uma lista de chaves.

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

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK para PHP.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando remove o objeto “sample.txt” do bucket “test-files”. Será solicitada uma confirmação antes que o comando seja executado. Para ignorar a solicitação, use a opção -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemplo 2: este comando remove a versão especificada do objeto “sample.txt” do bucket “test-files”, supondo que o bucket tenha sido configurado para habilitar versões de objetos.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
HLbxxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Exemplo 3: este comando remove objetos “sample1.txt”, “sample2.txt” e “sample3.txt” do bucket “test-files” como uma operação em lote única. A resposta do serviço listará todas as chaves processadas, independentemente do status de êxito ou erro da exclusão. Para obter somente erros de chaves que não puderam ser processadas pelo serviço, adicione o parâmetro -ReportErrorsOnly (esse parâmetro também pode ser especificado com o alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
"sample2.txt", "sample3.txt" )
```

Exemplo 4: este exemplo usa uma expressão em linha com o parâmetro -KeyCollection para obter as chaves dos objetos a serem excluídos. Get-S3Object retorna um conjunto de

instâncias `Amazon.S3.Model.S3Object`, cada uma com um membro de chave do tipo `string` identificando o objeto.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Exemplo 5: este exemplo obtém todos os objetos que têm um prefixo de chave “prefix/subprefix” no bucket e os exclui. Observe que os objetos de entrada são processados um de cada vez. Para conjuntos grandes, considere a possibilidade de passar o conjunto para o parâmetro `-InputObject` (alias `-S3ObjectCollection`) do cmdlet para permitir que a exclusão ocorra em lote com uma única chamada ao serviço.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

Exemplo 6: este exemplo envia um conjunto de instâncias `Amazon.S3.Model.S3ObjectVersion` que representam marcadores de exclusão do cmdlet para exclusão. Observe que os objetos de entrada são processados um de cada vez. Para conjuntos grandes, considere a possibilidade de passar o conjunto para o parâmetro `-InputObject` (alias `-S3ObjectCollection`) do cmdlet para permitir que a exclusão ocorra em lote com uma única chamada ao serviço.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Exemplo 7: este script mostra como realizar uma exclusão em lote de um conjunto de objetos (nesse caso, marcadores de exclusão) construindo uma matriz de objetos a serem usados com o parâmetro `-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
```

- Consulte detalhes da API em [DeleteObjects](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando remove o objeto “sample.txt” do bucket “test-files”. Será solicitada uma confirmação antes que o comando seja executado. Para ignorar a solicitação, use a opção -Force.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemplo 2: este comando remove a versão especificada do objeto “sample.txt” do bucket “test-files”, supondo que o bucket tenha sido configurado para habilitar versões de objetos.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId  
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Exemplo 3: este comando remove objetos “sample1.txt”, “sample2.txt” e “sample3.txt” do bucket “test-files” como uma operação em lote única. A resposta do serviço listará todas as chaves processadas, independentemente do status de êxito ou erro da exclusão. Para obter somente erros de chaves que não puderam ser processadas pelo serviço, adicione o parâmetro -ReportErrorsOnly (esse parâmetro também pode ser especificado com o alias -Quiet).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

Exemplo 4: este exemplo usa uma expressão em linha com o parâmetro -KeyCollection para obter as chaves dos objetos a serem excluídos. Get-S3Object retorna um conjunto de instâncias Amazon.S3.Model.S3Object, cada uma com um membro de chave do tipo string identificando o objeto.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Exemplo 5: este exemplo obtém todos os objetos que têm um prefixo de chave “prefix/subprefix” no bucket e os exclui. Observe que os objetos de entrada são processados um de cada vez. Para conjuntos grandes, considere a possibilidade de passar o conjunto para o parâmetro -InputObject (alias -S3ObjectCollection) do cmdlet para permitir que a exclusão ocorra em lote com uma única chamada ao serviço.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

Exemplo 6: este exemplo envia um conjunto de instâncias `Amazon.S3.Model.S3ObjectVersion` que representam marcadores de exclusão do cmdlet para exclusão. Observe que os objetos de entrada são processados um de cada vez. Para conjuntos grandes, considere a possibilidade de passar o conjunto para o parâmetro `-InputObject` (alias `-S3ObjectCollection`) do cmdlet para permitir que a exclusão ocorra em lote com uma única chamada ao serviço.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Exemplo 7: este script mostra como realizar uma exclusão em lote de um conjunto de objetos (nesse caso, marcadores de exclusão) construindo uma matriz de objetos a serem usados com o parâmetro `-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
```

- Consulte detalhes da API em [DeleteObjects](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua um conjunto de objetos de uma lista de chaves.

```

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

```

Exclua todos os objetos em um 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

```

Exclua permanentemente um objeto versionado excluindo todas as suas versões.

```
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
```

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
# 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

```

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK para Ruby.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

/// 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(())
  }

```

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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 obter detalhes da API, consulte [DeleteObjects](#) na Referência da API do SDK da AWS para SAP ABAP.

Swift

SDK para Swift

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
    }
}
```

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API AWS SDK para Swift.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeletePublicAccessBlock** com uma CLI

Os exemplos de código a seguir mostram como usar DeletePublicAccessBlock.

CLI

AWS CLI

Para excluir a configuração do Bloqueio de Acesso Público para um bucket

O exemplo delete-public-access-block a seguir remove a configuração do Bloqueio de Acesso Público no bucket especificado.

```
aws s3api delete-public-access-block \
  --bucket amzn-s3-demo-bucket
```

Este comando não produz saída.

- Consulte detalhes da API em [DeletePublicAccessBlock](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando desativa a configuração do Bloqueio de Acesso Público do bucket em questão.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select
'^BucketName'
```

Saída:

```
amzn-s3-demo-bucket
```

- Consulte detalhes da API em [DeletePublicAccessBlock](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando desativa a configuração do Bloqueio de Acesso Público do bucket em questão.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

Saída:

```
amzn-s3-demo-bucket
```

- Consulte detalhes da API em [DeletePublicAccessBlock](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketAccelerateConfiguration** com uma CLI

Os exemplos de código a seguir mostram como usar GetBucketAccelerateConfiguration.

CLI

AWS CLI

Para recuperar a configuração de aceleração de um bucket

O exemplo get-bucket-accelerate-configuration a seguir recupera a configuração de aceleração do bucket especificado.

```
aws s3api get-bucket-accelerate-configuration \  
--bucket amzn-s3-demo-bucket
```

Resultado:

```
{  
  "Status": "Enabled"  
}
```

- Consulte detalhes da API em [GetBucketAccelerateConfiguration](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: se as configurações de aceleração de transferência estiverem habilitadas para o bucket especificado, este comando retornará o valor Habilitado.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Value
-----
Enabled
```

- Consulte detalhes da API em [GetBucketAccelerateConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: se as configurações de aceleração de transferência estiverem habilitadas para o bucket especificado, este comando retornará o valor Habilitado.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Value
-----
Enabled
```

- Consulte detalhes da API em [GetBucketAccelerateConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketAc1** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar GetBucketAc1.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Verificar se um bucket existe](#)
- [Gerenciar listas de controle de acesso \(ACLs\)](#)

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [GetBucketAcl](#) na Referência da API AWS SDK para .NET.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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";
    }
}

```

- Consulte detalhes da API em [GetBucketAcl](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte comando recupera a lista de controle de acesso do bucket `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "Owner": {
    "DisplayName": "my-username",
    "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
  },
  "Grants": [
    {
      "Grantee": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
      },
      "Permission": "FULL_CONTROL"
    }
  ]
}
```

- Consulte detalhes da API em [GetBucketAcl](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 "";
    }
}
```

- Consulte detalhes da API em [GetBucketAcl](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha as permissões 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [GetBucketAcl](#) na Referência da API do AWS SDK para JavaScript.

PowerShell

Ferramentas para PowerShell V5

Exemplo 1: o comando obtém os detalhes do proprietário do objeto do S.

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

Saída:

```
DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Consulte detalhes da API em [GetBucketAcl](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

- Consulte detalhes da API em [GetBucketAcl](#) na Referência da API AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [GetBucketAcl](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketAnalyticsConfiguration** com uma CLI

Os exemplos de código a seguir mostram como usar **GetBucketAnalyticsConfiguration**.

CLI

AWS CLI

Para recuperar a configuração de análise de um bucket com um ID específico

O exemplo `get-bucket-analytics-configuration` a seguir recupera a configuração de análise referente ao bucket e ID especificados.

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

Resultado:

```
{
  "AnalyticsConfiguration": {
    "StorageClassAnalysis": {},
    "Id": "1"
  }
}
```

- Consulte detalhes da API em [GetBucketAnalyticsConfiguration](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna os detalhes do filtro de análise com o nome “testfilter” no bucket do S3 em questão.

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

- Consulte detalhes da API em [GetBucketAnalyticsConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna os detalhes do filtro de análise com o nome “testfilter” no bucket do S3 em questão.

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


- Consulte detalhes da API em [GetBucketAnalyticsConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketCors** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar GetBucketCors.

.NET

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [GetBucketCors](#) na Referência da API AWS SDK para .NET.

CLI

AWS CLI

O seguinte comando recupera a configuração de compartilhamento de recursos de origem cruzada para o bucket `amzn-s3-demo-bucket`:

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

Resultado:

```
{
  "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": [
        "*"
    ]
  }
]
```

- Consulte detalhes da API em [GetBucketCors](#) na Referência de comandos da AWS CLI.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha a política de CORS para o 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [GetBucketCors](#) na Referência da API do AWS SDK para JavaScript.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

- Consulte detalhes da API em [GetBucketCors](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  # an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS
  # configuration for the bucket.
  def cors
    @bucket_cors.data
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
      why: #{e.message}"
      nil
    end
  end
end
```

- Consulte detalhes da API em [GetBucketCors](#) na Referência da API AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes sobre a API, consulte [GetBucketCors](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketEncryption** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar GetBucketEncryption.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
/// <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 ter detalhes da API, consulte [GetBucketEncryption](#) na Referência da API do AWS SDK para .NET.

CLI

AWS CLI

Para recuperar a configuração de criptografia do lado do servidor para um bucket

O exemplo `get-bucket-encryption` a seguir recupera a configuração de criptografia do lado do servidor do bucket `amzn-s3-demo-bucket`.

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

Resultado:

```
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
```

- Consulte detalhes da API em [GetBucketEncryption](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna todas as regras de criptografia do lado do servidor associadas ao bucket em questão.

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

- Consulte detalhes da API em [GetBucketEncryption](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna todas as regras de criptografia do lado do servidor associadas ao bucket em questão.

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

- Consulte detalhes da API em [GetBucketEncryption](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketInventoryConfiguration** com uma CLI

Os exemplos de código a seguir mostram como usar GetBucketInventoryConfiguration.

CLI

AWS CLI

Para recuperar a configuração de inventário de um bucket

O exemplo get-bucket-inventory-configuration a seguir recupera a configuração de inventário do bucket especificado com ID 1.

```
aws s3api get-bucket-inventory-configuration \  
  --bucket amzn-s3-demo-bucket \
```

--id 1**Resultado:**

```
{
  "InventoryConfiguration": {
    "IsEnabled": true,
    "Destination": {
      "S3BucketDestination": {
        "Format": "ORC",
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
        "AccountId": "123456789012"
      }
    },
    "IncludedObjectVersions": "Current",
    "Id": "1",
    "Schedule": {
      "Frequency": "Weekly"
    }
  }
}
```

- Consulte detalhes da API em [GetBucketInventoryConfiguration](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna os detalhes do inventário denominado “testinventory” para o bucket do S3 em questão.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId 'testinventory'
```

- Consulte detalhes da API em [GetBucketInventoryConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna os detalhes do inventário denominado “testinventory” para o bucket do S3 em questão.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId 'testinventory'
```

- Consulte detalhes da API em [GetBucketInventoryConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketLifecycleConfiguration** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar `GetBucketLifecycleConfiguration`.

.NET

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [GetBucketLifecycleConfiguration](#) na Referência da API AWS SDK para .NET.

CLI

AWS CLI

O seguinte comando recupera a configuração do ciclo de vida do bucket `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket
```

Resultado:

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

- Consulte detalhes da API em [GetBucketLifecycleConfiguration](#) na Referência de comandos da AWS CLI.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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,\n        self.bucket.name,\n    )\nexcept:\n    logger.exception(\n        "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name\n    )\n    raise\nelse:\n    return config.rules
```

- Consulte detalhes da API em [GetBucketLifecycleConfiguration](#) na Referência da API AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
TRY.\n    oo_result = lo_s3->getbucketlifecycleconf(          " oo_result is\nreturned for testing purposes. "\n    iv_bucket = iv_bucket_name ).\n    MESSAGE 'Retrieved bucket lifecycle configuration.' TYPE 'I'. \n    CATCH /aws1/cx_s3_nosuchbucket.\n    MESSAGE 'Bucket does not exist.' TYPE 'E'. \nENDTRY.
```

- Para obter detalhes da API, consulte [GetBucketLifecycleConfiguration](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketLocation** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar `GetBucketLocation`.

CLI

AWS CLI

O seguinte comando recupera a restrição de localização do bucket `amzn-s3-demo-bucket`, se houver uma restrição:

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "LocationConstraint": "us-west-2"
}
```

- Consulte detalhes da API em [GetBucketLocation](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: se houver uma restrição, este comando retornará a restrição de localização do bucket `"amzn-s3-demo-bucket"`.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Value
-----
ap-south-1
```


- Consulte detalhes da API em [GetBucketLocation](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: se houver uma restrição, este comando retornará a restrição de localização do bucket "amzn-s3-demo-bucket".

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Value
-----
ap-south-1
```

- Consulte detalhes da API em [GetBucketLocation](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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(())
}
```

- Consulte detalhes da API em [GetBucketLocation](#) na Referência da API AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketLogging** com uma CLI

Os exemplos de código a seguir mostram como usar `GetBucketLogging`.

CLI

AWS CLI

Para recuperar o status de registro em log de um bucket

O exemplo `get-bucket-logging` a seguir recupera o status de registro em log do bucket especificado.

```
aws s3api get-bucket-logging \  
  --bucket amzn-s3-demo-bucket
```

Resultado:

```
{  
  "LoggingEnabled": {  
    "TargetPrefix": "",  
    "TargetBucket": "amzn-s3-demo-bucket-logs"  
  }  
}
```

- Consulte detalhes da API em [GetBucketLogging](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna o status de registro em log do bucket especificado.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Saída:

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- Consulte detalhes da API em [GetBucketLogging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna o status de registro em log do bucket especificado.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Saída:

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- Consulte detalhes da API em [GetBucketLogging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketMetricsConfiguration** com uma CLI

Os exemplos de código a seguir mostram como usar GetBucketMetricsConfiguration.

CLI

AWS CLI

Para recuperar a configuração de métricas de um bucket com um ID específico

O exemplo get-bucket-metrics-configuration a seguir exibe a configuração de métricas referente ao bucket e ID especificados.

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

Resultado:

```
{
  "MetricsConfiguration": {
    "Filter": {
```

```
        "Prefix": "logs"
      },
      "Id": "123"
    }
  }
```

- Consulte detalhes da API em [GetBucketMetricsConfiguration](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna detalhes sobre o filtro de métricas denominado “testfilter” para o bucket do S3 em questão.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- Consulte detalhes da API em [GetBucketMetricsConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna detalhes sobre o filtro de métricas denominado “testfilter” para o bucket do S3 em questão.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- Consulte detalhes da API em [GetBucketMetricsConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketNotification** com uma CLI

Os exemplos de código a seguir mostram como usar `GetBucketNotification`.

CLI

AWS CLI

O seguinte comando recupera a configuração de notificação do bucket denominado amzn-s3-demo-bucket:

```
aws s3api get-bucket-notification --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "TopicConfiguration": {
    "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
    "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNL",
    "Event": "s3:ObjectCreated:*",
    "Events": [
      "s3:ObjectCreated:*"
    ]
  }
}
```

- Consulte detalhes da API em [GetBucketNotification](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo recupera a configuração de notificação do bucket em questão

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
TopicConfigurations
```

Saída:

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Consulte detalhes da API em [GetBucketNotification](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo recupera a configuração de notificação do bucket em questão

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty TopicConfigurations
```

Saída:

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Consulte detalhes da API em [GetBucketNotification](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketPolicy** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o `GetBucketPolicy`.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();
}

```

- Consulte detalhes da API em [GetBucketPolicy](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte comando recupera política de bucket do bucket `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
```



```
"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/*\" } ] }"
```

Obter e colocar uma política de bucket em vigor O exemplo a seguir mostra como fazer download de uma política do Amazon S3, fazer alterações no arquivo e usar `put-bucket-policy` para aplicar a política de bucket modificada. Para fazer o download de uma política de bucket em um arquivo, você pode executar:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output text > policy.json
```

Em seguida, modifique o arquivo `policy.json` conforme necessário. Por fim, aplique essa política modificada ao bucket do S3 executando:

o arquivo `policy.json` conforme necessário. Por fim, aplique essa política modificada ao bucket do S3 executando:

conforme necessário. Por fim, aplique essa política modificada ao bucket do S3 executando:

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://policy.json
```

- Consulte detalhes da API em [GetBucketPolicy](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 "";
}
```

- Consulte detalhes da API em [GetBucketPolicy](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha a política de 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [GetBucketPolicy](#) na Referência da API do AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
    }
}
```

- Consulte detalhes da API em [GetBucketPolicy](#) na Referência da API AWS SDK para Kotlin.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando gera a política de bucket associada ao bucket do S3 em questão.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetBucketPolicy](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando gera a política de bucket associada ao bucket do S3 em questão.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetBucketPolicy](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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)
```

- Consulte detalhes da API em [GetBucketPolicy](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object
  # configured with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def policy
    policy = @bucket_policy.data.policy
```

```

    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
    #{e.message}"
    nil
  end
end
end

```

- Consulte detalhes da API em [GetBucketPolicy](#) na Referência da API AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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 obter detalhes da API, consulte [GetBucketPolicy](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketPolicyStatus** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketPolicyStatus`.

CLI

AWS CLI

Para recuperar o status da política de um bucket que indica se o bucket é público

O exemplo `get-bucket-policy-status` a seguir recupera o status da política do bucket `amzn-s3-demo-bucket`.

```
aws s3api get-bucket-policy-status \
  --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "PolicyStatus": {
    "IsPublic": false
  }
}
```

- Consulte detalhes da API em [GetBucketPolicyStatus](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna o status da política do bucket do S3 em questão, indicando se o bucket é público.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetBucketPolicyStatus](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna o status da política do bucket do S3 em questão, indicando se o bucket é público.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetBucketPolicyStatus](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketReplication** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o GetBucketReplication.

CLI

AWS CLI

O seguinte comando recupera a configuração de replicação de um bucket denominado amzn-s3-demo-bucket:

```
aws s3api get-bucket-replication --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "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"
    }
}
```

- Consulte detalhes da API em [GetBucketReplication](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [GetBucketReplication](#) na Referência de API do AWS SDK for Java 2.x.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna as informações da configuração de replicação definida no bucket denominado "amzn-s3-demo-bucket".

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Consulte detalhes da API em [GetBucketReplication](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna as informações da configuração de replicação definida no bucket denominado "amzn-s3-demo-bucket".

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Consulte detalhes da API em [GetBucketReplication](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketRequestPayment** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetBucketRequestPayment`.

CLI

AWS CLI

Para recuperar a configuração de pagamento de solicitação de um bucket

O exemplo `get-bucket-request-payment` a seguir recupera a configuração de pagamento de solicitação do bucket especificado.

```
aws s3api get-bucket-request-payment \
  --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "Payer": "BucketOwner"
}
```

- Consulte detalhes da API em [GetBucketRequestPayment](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: retorna as informações da configuração de pagamento de solicitação do bucket denominado "amzn-s3-demo-bucket". Por padrão, o proprietário do bucket paga pelos downloads feitos no bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Consulte detalhes da API em [GetBucketRequestPayment](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: retorna as informações da configuração de pagamento de solicitação do bucket denominado "amzn-s3-demo-bucket". Por padrão, o proprietário do bucket paga pelos downloads feitos no bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Consulte detalhes da API em [GetBucketRequestPayment](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketTagging** com uma CLI

Os exemplos de código a seguir mostram como usar o GetBucketTagging.

CLI

AWS CLI

O seguinte comando recupera a configuração de marcação de um bucket denominado amzn-s3-demo-bucket:

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "TagSet": [
    {
      "Value": "marketing",
      "Key": "organization"
    }
  ]
}
```

- Consulte detalhes da API em [GetBucketTagging](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna todas as tags associadas ao bucket em questão.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetBucketTagging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna todas as tags associadas ao bucket em questão.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetBucketTagging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketVersioning** com uma CLI

Os exemplos de código a seguir mostram como usar o GetBucketVersioning.

CLI

AWS CLI

O seguinte comando recupera a configuração de versionamento de um bucket denominado amzn-s3-demo-bucket:

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
```

```
"Status": "Enabled"
}
```

- Consulte detalhes da API em [GetBucketVersioning](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna o status de versionamento referente ao bucket em questão.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetBucketVersioning](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna o status de versionamento referente ao bucket em questão.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetBucketVersioning](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketWebsite** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o GetBucketWebsite.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [GetBucketWebsite](#) na Referência da API AWS SDK para .NET.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();
}

```

- Consulte detalhes da API em [GetBucketWebsite](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte comando recupera a configuração estática do site do bucket `amzn-s3-demo-bucket`:

```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- Consulte detalhes da API em [GetBucketWebsite](#) na Referência de comandos da AWS CLI.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha a configuração do site.

```
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)}\n`,  
);  
} 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;  
  }  
}  
};
```

- Consulte detalhes da API em [GetBucketWebsite](#) na Referência da APIs do AWS SDK para JavaScript.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna os detalhes das configurações do site estático do bucket do S3 em questão.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetBucketWebsite](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna os detalhes das configurações do site estático do bucket do S3 em questão.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetBucketWebsite](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetObject** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o `GetObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Conheça os conceitos básicos](#)
- [Obter um objeto de um bucket que foi modificado](#)
- [Obter um objeto de um ponto de acesso multirregional](#)
- [Conceitos básicos de criptografia](#)
- [Introdução ao armazenamento de objetos](#)
- [Fazer solicitações condicionais](#)
- [Monitorar uploads e downloads](#)

.NET

SDK para .NET (v4)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [GetObject](#) na Referência da API AWS SDK para .NET.

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha um objeto usando uma solicitação 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;
    }
}

```

- Consulte detalhes da API em [GetObject](#) na Referência da API AWS SDK para .NET.

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

#####
# 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
}
```

- Consulte detalhes da API em [GetObject](#) na Referência de comandos da AWS CLI.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();
}
```

- Consulte detalhes da API em [GetObject](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte exemplo usa o comando `get-object` para fazer download de um objeto do Amazon S3:

```
aws s3api get-object --bucket text-content --key dir/  
my_images.tar.bz2 my_images.tar.bz2
```

Observe que o parâmetro `outfile` é especificado sem um nome de opção, como `--outfile`. O nome do arquivo de saída deve ser o último parâmetro no comando.

O exemplo abaixo demonstra o uso de `--range` para fazer download de um intervalo de bytes específico de um objeto. Observe que os intervalos de bytes precisam ser prefixados com `"bytes="`:

```
aws s3api get-object --bucket text-content --key dir/my_data --  
range bytes=8888-9999 my_data_range
```

Para obter mais informações sobre como recuperar objetos, consulte [Obter um objeto](#) no Guia do desenvolvedor do Amazon S3.

- Consulte detalhes da API em [GetObject](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

- Consulte detalhes da API em [GetObject](#) na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Leia dados como uma matriz de bytes usando um [S3Client](#).

```
/**
 * Asynchronously retrieves the bytes of an object from an Amazon S3 bucket
 * and writes them to a local file.
 *
 * @param bucketName the name of the S3 bucket containing the object
 * @param keyName    the key (or name) of the S3 object to retrieve
 * @param path       the local file path where the object's bytes will be
 * written
 * @return a {@link CompletableFuture} that completes when the object bytes
 * have been written to the local file
 */
public CompletableFuture<Void> getObjectBytesAsync(String bucketName, String
keyName, String path) {
    GetObjectRequest objectRequest = GetObjectRequest.builder()
        .key(keyName)
        .bucket(bucketName)
        .build();

    CompletableFuture<ResponseBytes<GetObjectResponse>> response =
getAsyncClient().getObject(objectRequest, AsyncResponseTransformer.toBytes());
    return response.thenAccept(objectBytes -> {
        try {
            byte[] data = objectBytes.asByteArray();
            Path filePath = Paths.get(path);
            Files.write(filePath, data);
            logger.info("Successfully obtained bytes from an S3 object");
        } catch (IOException ex) {
            throw new RuntimeException("Failed to write data to file", ex);
        }
    }).whenComplete((resp, ex) -> {
```

```
        if (ex != null) {
            throw new RuntimeException("Failed to get object bytes from S3",
ex);
        }
    });
}
```

Use [S3TransferManager](#) para [fazer download de um objeto](#) em um bucket do S3 para um arquivo local. Veja o [arquivo completo](#) e [teste](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String
bucketName,

                                String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFilePath))
            .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);
```

```

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentType());
        return downloadResult.response().contentType();
    }

```

Leia etiquetas que pertencem a um objeto usando um [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);
    }
}
}
```

Obtenha um URL para um objeto usando um [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);
    }
}
}

```

Obtenha um objeto usando o objeto de cliente S3Presigner usando um [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.setRequestProperty(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();
    }
}
}

```

Obtenha um objeto usando um objeto `ResponseTransformer` e o [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);
    }
}
```

- Consulte detalhes da API em [GetObject](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Baixe o 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;
    }
}
};
```

Baixe o objeto com a condição de que sua ETag corresponda à fornecida.

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

Baixe o objeto com a condição de que sua ETag não corresponda à fornecida.

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

```
}  
}
```

Baixe o objeto usando a condição de que ele tenha sido criado ou modificado em determinado período.

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

Baixe o objeto usando a condição de que ele não tenha sido criado ou modificado em determinado período.

```
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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [GetObject](#) na Referência da API do AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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")  
        }  
    }  
}
```

- Consulte detalhes da API em [GetObject](#) na Referência da API AWS SDK for Kotlin.

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha um 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.");
}
```

- Consulte detalhes da API em [GetObject](#) na Referência da API AWS SDK para PHP.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando recupera o item “sample.txt” do bucket “amzn-s3-demo-bucket” e o salva em um arquivo chamado “local-sample.txt” no local atual. O arquivo “local-sample.txt” não precisa existir antes de esse comando ser chamado.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-
sample.txt
```

Exemplo 2: este comando recupera o diretório virtual “DIR” do bucket “amzn-s3-demo-bucket” e o salva em uma pasta chamada “Local-DIR” no local atual. O arquivo “Local-DIR” não precisa existir antes de esse comando ser chamado.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Exemplo 3: baixa todos os objetos com chaves terminadas em “.json” de buckets com “config” no respectivo nome em arquivos na pasta especificada. As chaves do objeto são usadas para definir o nome dos arquivos.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\Config0bjects
```

- Consulte detalhes da API em [GetObject](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando recupera o item “sample.txt” do bucket “amzn-s3-demo-bucket” e o salva em um arquivo chamado “local-sample.txt” no local atual. O arquivo “local-sample.txt” não precisa existir antes de esse comando ser chamado.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Exemplo 2: este comando recupera o diretório virtual “DIR” do bucket “amzn-s3-demo-bucket” e o salva em uma pasta chamada “Local-DIR” no local atual. O arquivo “Local-DIR” não precisa existir antes de esse comando ser chamado.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Exemplo 3: baixa todos os objetos com chaves terminadas em “.json” de buckets com “config” no respectivo nome em arquivos na pasta especificada. As chaves do objeto são usadas para definir o nome dos arquivos.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\Config0bjects
```

Exemplo 4: baixa um único objeto usando o download paralelo de várias partes para melhorar o desempenho em arquivos grandes.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt -UseMultipartDownload
```

Exemplo 5: baixa um objeto grande usando o modo RANGE com um tamanho de parte personalizado de 64 MB. O modo RANGE usa solicitações HTTP de intervalo de bytes e funciona para todos os objetos, independentemente de como eles foram carregados.

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

Exemplo 6: baixa todos os objetos com o prefixo de chave "data/" para uma pasta local usando o download de várias partes com downloads simultâneos de arquivos.

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

Exemplo 7: baixa um único objeto usando o download de várias partes com 20 conexões simultâneas para obter o throughput máximo em redes de alta largura de banda.

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

Exemplo 8: baixa todos os objetos com o prefixo de chave "data/" para uma pasta local usando o download de várias partes, continuando com as falhas de arquivos individuais em vez de interrompê-las.

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

- Consulte detalhes da API em [GetObject](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

Obtenha um objeto usando uma solicitação 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
```

- Consulte detalhes da API em [GetObject](#) na Referência da API do AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha um 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__

```

Obtenha um objeto e relate seu estado de criptografia do lado do 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

# 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

  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__
```

- Consulte detalhes da API em [GetObject](#) na Referência da API AWS SDK para Ruby.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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)
}
```

- Consulte detalhes da API em [GetObject](#) na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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.
```

- Consulte detalhes da API em [GetObject](#) na Referência da API AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
    }
}
```

- Consulte detalhes da API em [GetObject](#) na Referência da API do AWS SDK para Swift.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetObjectAcl** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o `GetObjectAcl`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Gerenciar listas de controle de acesso \(ACLs\)](#)

C++

SDK para C++

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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";
        }
    }
}

```

- Consulte detalhes da API em [GetObjectAcl](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte comando recupera a lista de controle de acesso de um objeto em um bucket chamado `amzn-s3-demo-bucket`:

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key index.html
```

Resultado:

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

- Consulte detalhes da API em [GetObjectAcl](#) na Referência de comandos da AWS CLI.

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [GetObjectAcl](#) na Referência da API AWS SDK para Kotlin.

PowerShell

Ferramentas para PowerShell V5

Exemplo 1: o comando obtém os detalhes do proprietário do objeto do S.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
*).Owner
```

Saída:

```

DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544

```

- Para obter detalhes da API, consulte [GetObjectAcl](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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

```

- Consulte detalhes da API em [GetObjectAcl](#) na Referência da API AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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 obter detalhes da API, consulte [GetObjectAcl](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetObjectAttributes** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o `GetObjectAttributes`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Trabalhar com a integridade de objetos do Amazon S3](#)

C++

SDK para C++

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
// ! 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;
}

```

- Para receber detalhes da API, consulte [GetObjectAttributes](#) na Referência da API do AWS SDK para C++.

CLI

AWS CLI

Como recuperar metadados de um objeto sem exibir o próprio objeto

O exemplo de `get-object-attributes` a seguir recupera metadados do objeto `doc1.rtf`.

```

aws s3api get-object-attributes \
  --bucket amzn-s3-demo-bucket \
  --key doc1.rtf \
  --object-attributes "StorageClass" "ETag" "ObjectSize"

```

Resultado:

```

{
  "LastModified": "2022-03-15T19:37:31+00:00",
  "VersionId": "IuCPjXTDzHNfIdAuitVBIKJpF2p1fg4P",

```

```
"ETag": "b662d79adeb7c8d787ea7eafb9ef6207",  
"StorageClass": "STANDARD",  
"ObjectSize": 405  
}
```

Para receber mais informações, consulte [GetObjectAttributes](#) na Referência da API do Amazon S3.

- Para receber detalhes da API, consulte [GetObjectAttributes](#) na Referência de comandos da AWS CLI.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetObjectLegalHold** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o `GetObjectLegalHold`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- [Bloquear objetos do Amazon S3](#)

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [GetObjectLegalHold](#) na Referência de API do AWS SDK para .NET.

CLI

AWS CLI

Recupera o status de “Retenção jurídica” de um objeto

O exemplo de `get-object-legal-hold` a seguir recupera o status de “Retenção jurídica” do objeto especificado.

```
aws s3api get-object-legal-hold \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf
```

Resultado:

```
{
  "LegalHold": {
    "Status": "ON"
  }
}
```

- Consulte detalhes da API em [GetObjectLegalHold](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

- Consulte detalhes da API em [GetObjectLegalHold](#) na Referência de API do AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
// 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;
}
```

- Consulte detalhes da API em [GetObjectLegalHold](#) na Referência de API do AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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



```
}
```

- Consulte detalhes da API em [GetObjectLegalHold](#) na Referência de API do AWS SDK para JavaScript.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina a retenção legal de objetos.

```
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 receber detalhes da API, consulte [GetObjectLegalHold](#) na Referência da API do AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 ver detalhes da API, consulte [GetObjectLegalHold](#) na Referência de API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetObjectLockConfiguration** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o `GetObjectLockConfiguration`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- [Bloquear objetos do Amazon S3](#)

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
/// <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($"Bucket object lock config for {bucketName} in
{bucketName}: " +
                        $"Enabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                        $"Rule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");
    }
}
```

```
        return response.ObjectLockConfiguration;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}'");
        return new ObjectLockConfiguration();
    }
}
```

- Consulte detalhes da API em [GetObjectLockConfiguration](#) na Referência de API do AWS SDK para .NET.

CLI

AWS CLI

Como recuperar uma configuração de bloqueio de objetos de um bucket

O exemplo de `get-object-lock-configuration` a seguir recupera a configuração de bloqueio de objetos do bucket especificado.

```
aws s3api get-object-lock-configuration \
    --bucket amzn-s3-demo-bucket-with-object-lock
```

Resultado:

```
{
  "ObjectLockConfiguration": {
    "ObjectLockEnabled": "Enabled",
    "Rule": {
      "DefaultRetention": {
        "Mode": "COMPLIANCE",
        "Days": 50
      }
    }
  }
}
```

- Consulte detalhes da API em [GetObjectLockConfiguration](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

- Consulte detalhes da API em [GetObjectLockConfiguration](#) na Referência de API do AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
```

```
GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .build();

GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
System.out.println("Bucket object lock config for "+bucketName+": ");
System.out.println("\tEnabled:
"+response.getObjectLockConfiguration().objectLockEnabled());
System.out.println("\tRule: "+
response.getObjectLockConfiguration().rule().defaultRetention());
}
```

- Consulte detalhes da API em [GetObjectLockConfiguration](#) na Referência de API do AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [GetObjectLockConfiguration](#) na Referência de API do AWS SDK para JavaScript.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: se a configuração do Bloqueio de Objetos estiver habilitada para o bucket do S3 em questão, este comando retornará o valor “Habilitado”.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Saída:

```
Value  
-----  
Enabled
```

- Consulte detalhes da API em [GetObjectLockConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: se a configuração do Bloqueio de Objetos estiver habilitada para o bucket do S3 em questão, este comando retornará o valor “Habilitado”.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Saída:

```
Value  
-----  
Enabled
```

- Consulte detalhes da API em [GetObjectLockConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina a configuração de bloqueio 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 receber detalhes da API, consulte [GetObjectLockConfiguration](#) na Referência da API do AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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 obter detalhes da API, consulte [GetObjectLockConfiguration](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetObjectRetention** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o `GetObjectRetention`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- [Bloquear objetos do Amazon S3](#)

.NET

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
/// <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($"{Object retention for {objectKey} in
{bucketName}: " +
                        $"{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}");
        return response.Retention;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"{Unable to fetch object lock retention:
'{ex.Message}'");
        return new ObjectLockRetention();
    }
}
```

- Consulte detalhes da API em [GetObjectRetention](#) na Referência de API do AWS SDK para .NET.

CLI

AWS CLI

Como recuperar a configuração de retenção de objetos de um objeto

O exemplo de `get-object-retention` a seguir recupera a configuração de retenção de objetos do objeto especificado.

```
aws s3api get-object-retention \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf
```

Resultado:

```
{
  "Retention": {
    "Mode": "GOVERNANCE",
    "RetainUntilDate": "2025-01-01T00:00:00.000Z"
  }
}
```

- Consulte detalhes da API em [GetObjectRetention](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

- Consulte detalhes da API em [GetObjectRetention](#) na Referência de API do AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
// 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;
    }
}
```

- Consulte detalhes da API em [GetObjectRetention](#) na Referência de API do AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [GetObjectRetention](#) na Referência de API do AWS SDK para JavaScript.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando retorna o modo e a data até a qual o objeto ficará retido.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Consulte detalhes da API em [GetObjectRetention](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).


```
{
  "Value": "confidential",
  "Key": "designation"
}
]
```

O exemplo `get-object-tagging` a seguir tenta recuperar os conjuntos de tags do objeto `doc2.rtf`, o qual não tem tags.

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc2.rtf
```

Resultado:

```
{
  "TagSet": []
}
```

O exemplo `get-object-tagging` a seguir recupera os conjuntos de tags do objeto `doc3.rtf`, o qual não tem tags.

```
aws s3api get-object-tagging \
  --bucket amzn-s3-demo-bucket \
  --key doc3.rtf
```

Resultado:

```
{
  "TagSet": [
    {
      "Value": "confidential",
      "Key": "designation"
    },
    {
      "Value": "finance",
      "Key": "department"
    },
    {
      "Value": "payroll",
```

```
        "Key": "team"
      }
    ]
  }
```

- Consulte detalhes da API em [GetObjectTagging](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o exemplo retorna as tags associadas ao objeto presente no bucket do S3 em questão.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Key  Value
---  -
test value
```

- Consulte detalhes da API em [GetObjectTagging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: o exemplo retorna as tags associadas ao objeto presente no bucket do S3 em questão.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Saída:

```
Key  Value
---  -
test value
```

- Consulte detalhes da API em [GetObjectTagging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetPublicAccessBlock** com uma CLI

Os exemplos de código a seguir mostram como usar o `GetPublicAccessBlock`.

CLI

AWS CLI

Para definir ou modificar a configuração do Bloqueio de Acesso Público de um bucket

O exemplo `get-public-access-block` a seguir exibe a configuração do Bloqueio de Acesso Público do bucket especificado.

```
aws s3api get-public-access-block \
  --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "PublicAccessBlockConfiguration": {
    "IgnorePublicAcls": true,
    "BlockPublicPolicy": true,
    "BlockPublicAcls": true,
    "RestrictPublicBuckets": true
  }
}
```

- Consulte detalhes da API em [GetPublicAccessBlock](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando retorna a configuração do Bloqueio de Acesso Público do bucket do S3 em questão.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetPublicAccessBlock](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando retorna a configuração do Bloqueio de Acesso Público do bucket do S3 em questão.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [GetPublicAccessBlock](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **HeadBucket** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o HeadBucket.

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
#####  
# 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
}
```

- Consulte detalhes da API em [HeadBucket](#) na Referência de comandos da AWS CLI.

CLI

AWS CLI

O seguinte comando verifica o acesso ao bucket `amzn-s3-demo-bucket`:

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

Se o bucket existir e você tiver acesso a ele, nenhuma saída será retornada. Caso contrário, uma mensagem de erro será exibida. Por exemplo:

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- Consulte detalhes da API em [HeadBucket](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

- Consulte detalhes da API em [HeadBucket](#) na Referência da API AWS SDK para Go.

PowerShell

Ferramentas para PowerShell V5

Exemplo 1: esse comando retorna a saída com o código de status HTTP 200 OK para o bucket existente quando o usuário tem permissão para acessá-lo. O parâmetro BucketArn só é compatível com buckets de diretório do S.

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

Saída:

```
AccessPointAlias    : False
BucketArn           :
BucketLocationName  :
BucketLocationType  :
BucketRegion        : us-east-2
ResponseMetadata    : Amazon.Runtime.ResponseMetadata
ContentLength       : 0
HttpStatusCode       : OK
```

Exemplo 2: esse comando gera um erro com o código de status HTTP NotFound para um bucket inexistente.

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

Saída:

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status
Code NotFound. No further error information was returned by the service.
```

Exemplo 3: esse comando gera um erro com o código de status HTTP Proibido para um bucket existente em que o usuário não tem permissão para acessá-lo.

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

Saída:

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status
Code Forbidden. No further error information was returned by the service.
```

- Consulte detalhes da API em [HeadBucket](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

- Consulte detalhes da API em [HeadBucket](#) na Referência da API AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [HeadBucket](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **HeadObject** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o HeadObject.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Verificar se um objeto existe](#)
- [Introdução ao armazenamento de objetos](#)

CLI

AWS CLI

O seguinte comando recupera metadados de um objeto no bucket `amzn-s3-demo-bucket`:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

Resultado:

```
{
  "AcceptRanges": "bytes",
  "ContentType": "text/html",
  "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
  "ContentLength": 77,
  "VersionId": "null",
  "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
  "Metadata": {}
}
```

- Consulte detalhes da API em [HeadObject](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Determine o tipo de conteúdo de um 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);
        }
    }
}

```

Obtenha o status de restauração de um 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);
    }
}
}
```

- Consulte detalhes da API em [HeadObject](#) em Referência da API AWS SDK for Java 2.x.

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object
    #{@object.bucket.name}:#{@object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"
```

```
wrapper = ObjectExistsWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
exists = wrapper.exists?

puts "Object #{object_key} #{exists ? 'does' : 'does not'} exist."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Consulte detalhes da API em [HeadObject](#) em Referência da API AWS SDK para Ruby.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListBucketAnalyticsConfigurations** com uma CLI

Os exemplos de código a seguir mostram como usar o ListBucketAnalyticsConfigurations.

CLI

AWS CLI

Como recuperar uma lista de configurações de análise de um bucket

O exemplo list-bucket-analytics-configurations a seguir recupera uma lista de configurações de análise do bucket especificado.

```
aws s3api list-bucket-analytics-configurations \
  --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "AnalyticsConfigurationList": [
    {
      "StorageClassAnalysis": {},
      "Id": "1"
    }
  ],
  "IsTruncated": false
}
```

- Consulte detalhes da API em [ListBucketAnalyticsConfigurations](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna as cem primeiras análises do bucket do S3 em questão.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [ListBucketAnalyticsConfigurations](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna as cem primeiras análises do bucket do S3 em questão.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [ListBucketAnalyticsConfigurations](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListBucketInventoryConfigurations** com uma CLI

Os exemplos de código a seguir mostram como usar o ListBucketInventoryConfigurations.

CLI

AWS CLI

Como recuperar uma lista de configurações de inventário de um bucket

O exemplo list-bucket-inventory-configurations a seguir recupera as configurações de inventário do bucket especificado.

```
aws s3api list-bucket-inventory-configurations \
```

```
--bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "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
}
```

- Consulte detalhes da API em [ListBucketInventoryConfigurations](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna as cem primeiras configurações de inventário do bucket do S3 em questão.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [ListBucketInventoryConfigurations](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna as cem primeiras configurações de inventário do bucket do S3 em questão.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Consulte detalhes da API em [ListBucketInventoryConfigurations](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListBuckets** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o ListBuckets.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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);
    }
}
}
```

- Consulte detalhes da API em [ListBuckets](#) na Referência da API AWS SDK para .NET.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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;
}
```


- Consulte detalhes da API em [ListBuckets](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte comando usa o comando `list-buckets` para exibir os nomes de todos os buckets do Amazon S3 (em todas as regiões):

```
aws s3api list-buckets --query "Buckets[].Name"
```

A opção de consulta filtra a saída de `list-buckets` até sobrar apenas os nomes dos buckets.

Para obter mais informações sobre buckets, consulte [Working with Amazon S3 Buckets](#) no Guia do desenvolvedor do Amazon S3.

- Consulte detalhes da API em [ListBuckets](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

```
}
```

- Consulte detalhes da API em [ListBuckets](#) na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter informações da API, consulte [ListBuckets](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Liste os 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [ListBuckets](#) na Referência da API do AWS SDK para JavaScript.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando retorna todos os buckets do S.

```
Get-S3Bucket
```

Exemplo 2: este comando retorna um bucket denominado "test-files"

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Consulte detalhes da API em [ListBuckets](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando retorna todos os buckets do S.

```
Get-S3Bucket
```

Exemplo 2: este comando retorna um bucket denominado "test-files"

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Consulte detalhes da API em [ListBuckets](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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

```

- Consulte detalhes da API em [ListBuckets](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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__
```

- Consulte detalhes da API em [ListBuckets](#) na Referência da API AWS SDK para Ruby.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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(())
}
```

- Consulte detalhes da API em [ListBuckets](#) na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [ListBuckets](#) na Referência da API do SDK da AWS para SAP ABAP.

Swift

SDK para Swift

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [ListBuckets](#) na Referência de API do AWS SDK para Swift.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListMultipartUploads** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o ListMultipartUploads.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Excluir uploads multiparte incompletos](#)

CLI

AWS CLI

O seguinte comando lista todos os uploads fracionados ativos do bucket `amzn-s3-demo-bucket`:

```
aws s3api list-multipart-uploads --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "Uploads": [
    {
      "Initiator": {
        "DisplayName": "username",
        "ID": "arn:aws:iam::0123456789012:user/username"
      },
      "Initiated": "2015-06-02T18:01:30.000Z",
      "UploadId":
        "dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3",
      "StorageClass": "STANDARD",
      "Key": "multipart/01",
      "Owner": {
        "DisplayName": "aws-account-name",
        "ID":
          "100719349fc3b6dcd7c820a124bf7aecd408092c3d7b51b38494939801fc248b"
      }
    }
  ],
  "CommonPrefixes": []
}
```

Os uploads fracionados em andamento incorrem em custos de armazenamento no Amazon S3. Conclua ou cancele um upload fracionado ativo para remover suas partes da conta.

- Consulte detalhes da API em [ListMultipartUploads](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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);
    }
}
}
```

- Consulte detalhes da API em [ListMultipartUploads](#) na Referência da API AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListObjectVersions** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o ListObjectVersions.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Introdução ao armazenamento de objetos](#)
- [Trabalhar com objetos versionados](#)

.NET

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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.USSouth2);
        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}'");
}
}
```

- Consulte detalhes da API em [ListObjectVersions](#) na Referência da API AWS SDK para .NET.

CLI

AWS CLI

O seguinte comando recupera as informações da versão de um objeto do bucket `amzn-s3-demo-bucket`:

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix index.html
```

Resultado:

```
{
  "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_l2T8UHDkFEwCgJjhl9P0ZC0qJ.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
    }
]
}

```

- Consulte detalhes da API em [ListObjectVersions](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

```
}
```

- Consulte detalhes da API em [ListObjectVersions](#) na Referência da API AWS SDK para Go.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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(())  
}
```

- Consulte detalhes da API em [ListObjectVersions](#) na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [ListObjectVersions](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListObjects** com uma CLI

Os exemplos de código a seguir mostram como usar o ListObjects.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Criar uma página da web que oferece uma lista de objetos do Amazon S3](#)

CLI

AWS CLI

O seguinte exemplo usa o comando `list-objects` para exibir os nomes de todos os objetos presentes no bucket especificado:

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

O exemplo usa o argumento `--query` para filtrar a saída de `list-objects` até encontrar o valor e o tamanho da chave de cada objeto

Para obter mais informações sobre objetos, consulte [Working with Amazon S3 Buckets](#) no Guia do desenvolvedor do Amazon S3.

- Consulte detalhes da API em [ListObjects](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando recupera as informações sobre todos os itens no bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Exemplo 2: este comando recupera as informações sobre o item “sample.txt” do bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemplo 3: este comando recupera as informações sobre todos os itens com prefixo “sample” do bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Para ver detalhes da API, consulte [ListObjects](#) na Ferramentas da AWS para PowerShell Cmdlet Reference (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando recupera as informações sobre todos os itens no bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Exemplo 2: este comando recupera as informações sobre o item “sample.txt” do bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Exemplo 3: este comando recupera as informações sobre todos os itens com prefixo “sample” do bucket “test-files”.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Para ver detalhes da API, consulte [ListObjects](#) na Ferramentas da AWS para PowerShell Cmdlet Reference (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListObjectsV2** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o ListObjectsV2.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Conheça os conceitos básicos](#)
- [Excluir todos os objetos em um bucket](#)
- [Introdução ao armazenamento de objetos](#)

.NET

SDK para .NET (v4)

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).


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

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK para .NET. SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Liste objetos com um 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}");
            }
        }
    }
}

```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK para .NET.

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
#####
# 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
}
```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência de comandos da AWS CLI.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);
```

```

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: listObjects: " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    } else {
        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();

        allObjects.insert(allObjects.end(), objects.begin(), objects.end());
        continuationToken = outcome.GetResult().GetNextContinuationToken();
    }
} while (!continuationToken.empty());

std::cout << allObjects.size() << " object(s) found:" << std::endl;

for (const auto &object: allObjects) {
    std::cout << " " << object.GetKey() << std::endl;
    keysResult.push_back(object.GetKey());
}

return true;
}

```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

Como obter uma lista dos objetos em um bucket

O exemplo de `list-objects-v2` a seguir lista os objetos no bucket especificado.

```
aws s3api list-objects-v2 \
    --bucket amzn-s3-demo-bucket
```

Resultado:

```
{
  "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
}
]
```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

```

Liste objetos usando paginação.

```

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);
    }
}
}
```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Liste todos os objetos no bucket. Se houver mais de um objeto, `IsTruncated` e `NextContinuationToken` serão usados para iterar a 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 (err) {
    if (err instanceof S3ServiceException) {
      console.error("S3ServiceException: ", err.message);
    } else {
      console.error("Unknown error: ", err);
    }
  }
};
```

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

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 { calcKb(it) }} KBs")
        println("The owner is ${myObject.owner}")
    }
}

private fun calcKb(intValue: Long): Long = intValue / 1024

```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Liste objetos em um 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.");
}

```

```
}
```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK para PHP.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
```



```
# @return [Integer] The number of objects listed.
def list_objects(max_objects)
  count = 0
  puts "The objects in #{@bucket.name} are:"
  @bucket.objects.each do |obj|
    puts "\t#{obj.key}"
    count += 1
    break if count == max_objects
  end
  count
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list objects in bucket #{bucket.name}. Here's why:
#{e.message}"
  0
end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

  wrapper = BucketListObjectsWrapper.new(Aws::S3::Bucket.new(bucket_name))
  count = wrapper.list_objects(25)
  puts "Listed #{count} objects."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK para Ruby.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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(())
}
```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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.

```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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 obter detalhes da API, consulte [ListObjectsV2](#) na Referência da API do AWS SDK para Swift.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketAccelerateConfiguration** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutBucketAccelerateConfiguration.

.NET

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [PutBucketAccelerateConfiguration](#) na Referência da API do AWS SDK para .NET.

CLI

AWS CLI

Para definir a configuração de aceleração de um bucket

O exemplo `put-bucket-accelerate-configuration` a seguir habilita a configuração de aceleração do bucket especificado.

```
aws s3api put-bucket-accelerate-configuration \
    --bucket amzn-s3-demo-bucket \
    --accelerate-configuration Status=Enabled
```

Este comando não produz saída.

- Consulte detalhes da API em [PutBucketAccelerateConfiguration](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando habilita a aceleração de transferência do bucket do S3 em questão.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Consulte detalhes da API em [PutBucketAccelerateConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando habilita a aceleração de transferência do bucket do S3 em questão.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Consulte detalhes da API em [PutBucketAccelerateConfiguration](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketAc1** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutBucketAc1.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- [Gerenciar listas de controle de acesso \(ACLs\)](#)

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [PutBucketAcl](#) na Referência da API AWS SDK para .NET.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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;
}

```

- Consulte detalhes da API em [PutBucketAcl](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

Este exemplo concede `full control` a dois usuários da AWS (`user1@example.com` e `user2@example.com`) e a permissão `read` a todos:

```
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 obter detalhes sobre ACLs personalizadas (os comandos da ACL s3api, como put-bucket-acl, usam a mesma notação abreviada de argumentos).

- Consulte detalhes da API em [PutBucketAcl](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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);
}
}
```

- Consulte detalhes da API em [PutBucketAcl](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Coloque a ACL do 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [PutBucketAcl](#) na Referência da API do AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 = ownerOb
            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")
    }
}

```

- Consulte detalhes da API em [PutBucketAcl](#) na Referência da API AWS SDK para Kotlin.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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

```

- Consulte detalhes da API em [PutBucketAcl](#) na Referência da API AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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 obter detalhes da API, consulte [PutBucketAcl](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketCors** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutBucketCors.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

/// <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);
}
```

- Consulte detalhes da API em [PutBucketCors](#) na Referência da API AWS SDK para .NET.

CLI

AWS CLI

O seguinte exemplo habilita solicitações PUT, POST e DELETE de `www.example.com` e solicitações GET de qualquer domínio:

```
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
    }
  ]
}
```

- Consulte detalhes da API em [PutBucketCors](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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);
    }
}
```

```
}  
}
```

- Consulte detalhes da API em [PutBucketCors](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Adicione uma regra de 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).

- Consulte detalhes da API em [PutBucketCors](#) na Referência da API do AWS SDK para JavaScript.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

- Consulte detalhes da API em [PutBucketCors](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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

```

- Consulte detalhes da API em [PutBucketCors](#) na Referência da API AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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 obter detalhes da API, consulte [PutBucketCors](#) na Referência de API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketEncryption** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutBucketEncryption.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- [Introdução ao armazenamento de objetos](#)

.NET

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
/// <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 ter detalhes da API, consulte [PutBucketEncryption](#) na Referência da API do AWS SDK para .NET.

CLI

AWS CLI

Para configurar a criptografia do lado do servidor para um bucket

O exemplo `put-bucket-encryption` a seguir define a criptografia AES256 como padrão para o bucket especificado.

```
aws s3api put-bucket-encryption \  
  --bucket amzn-s3-demo-bucket \  
  --server-side-encryption-configuration '{"Rules":  
  [{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}}]'
```

Este comando não produz saída.

- Consulte detalhes da API em [PutBucketEncryption](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando habilita a criptografia padrão do lado do servidor AES256 com chaves gerenciadas do Amazon S3 (SSE-S3) no bucket em questão.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
  @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
  ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Consulte detalhes da API em [PutBucketEncryption](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando habilita a criptografia padrão do lado do servidor AES256 com chaves gerenciadas do Amazon S3 (SSE-S3) no bucket em questão.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
  @{ServerSideEncryptionAlgorithm = "AES256"}}
```

```
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Consulte detalhes da API em [PutBucketEncryption](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketLifecycleConfiguration** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutBucketLifecycleConfiguration.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Excluir uploads multipartes incompletos](#)
- [Gerenciar mensagens grandes usando o S3](#)
- [Trabalhar com objetos versionados](#)

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
/// <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 obter informações da API, consulte [PutBucketLifecycleConfiguration](#) em Referência da API AWS SDK para .NET .

CLI

AWS CLI

O seguinte comando aplica a configuração de ciclo de vida ao bucket `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --  
lifecycle-configuration file://lifecycle.json
```

O arquivo `lifecycle.json` é um documento JSON na pasta atual que especifica duas regras:

```
{  
  "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"
  }
]
```

A primeira regra move arquivos com o prefixo `rotated` para o Glacier na data especificada. A segunda regra move versões antigas de objetos para o Glacier quando elas não estão mais atualizadas. Para obter mais informações sobre formatos de data/hora aceitáveis, consulte [Especificar valores de parâmetro](#) Guia do usuário da AWS CLI.

- Consulte detalhes da API em [PutBucketLifecycleConfiguration](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter informações da API, consulte [PutBucketLifecycleConfiguration](#) em Referência da API AWS SDK for Java 2.x .

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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 obter informações da API, consulte [PutBucketLifecycleConfiguration](#) na Referência da API do AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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 obter informações da API, consulte [PutBucketLifecycleConfiguration](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketLogging** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutBucketLogging.

.NET

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();
}
}
```

- Consulte detalhes da API em [PutBucketLogging](#) na Referência da API do AWS SDK para .NET.

CLI

AWS CLI

Exemplo 1: definir o registro em log da política de bucket

O exemplo de `put-bucket-logging` a seguir define a política de registro em log para `amzn-s3-demo-bucket`. Primeiro, conceda permissão à entidade principal do serviço de registro em log na política de bucket usando o comando `put-bucket-policy`.

```
aws s3api put-bucket-policy \
    --bucket amzn-s3-demo-bucket \
    --policy file://policy.json
```


Conteúdo 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 a política de registro em log, use `put-bucket-logging`.

```
aws s3api put-bucket-logging \
  --bucket amzn-s3-demo-bucket \
  --bucket-logging-status file://logging.json
```

Conteúdo de logging.json:

```
{
  "LoggingEnabled": {
    "TargetBucket": "amzn-s3-demo-bucket",
    "TargetPrefix": "Logs/"
  }
}
```

O comando `put-bucket-policy` é necessário para conceder as permissões `s3:PutObject` à entidade principal do serviço de registro em log.

Consulte mais informações em [Registrar em log as solicitações com registro em log de acesso ao servidor](#) no Guia do usuário do Amazon S3.

Exemplo 2: definir uma política de bucket para registrar em log o acesso a um único usuário

O exemplo de `put-bucket-logging` a seguir define a política de registro em log para `amzn-s3-demo-bucket`. O usuário `bob@example.com` da AWS terá controle total sobre os arquivos de log e ninguém mais terá acesso. Primeiro, conceda permissão ao S3 com `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
```

Depois, aplique a política de registro em log usando `put-bucket-logging`.

```
aws s3api put-bucket-logging \
  --bucket amzn-s3-demo-bucket \
  --bucket-logging-status file://logging.json
```

Conteúdo 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"
      }
    ]
  }
}
```

O comando `put-bucket-acl` é necessário para conceder as permissões necessárias ao sistema de entrega de log do S3 (permissões de gravação e `read-acp`).

Consulte mais informações em [Habilitar o log de acesso ao servidor do Amazon S3](#) no Guia do desenvolvedor do Amazon S3.

- Consulte detalhes da API em [PutBucketLogging](#) na Referência de comandos da AWS CLI.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketNotification** com uma CLI

Os exemplos de código a seguir mostram como usar o PutBucketNotification.

CLI

AWS CLI

O exemplo aplica uma configuração de notificação ao bucket `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-configuration file://notification.json
```

O arquivo `notification.json` é um documento JSON na pasta atual que especifica um tópico do SNS e o tipo de evento a ser monitorado:

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

O tópico do SNS deve ter uma política do IAM anexada que permita que o Amazon S3 publique nele:

```
{
  "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"
        }
    }
}
]
}

```

- Consulte detalhes da API em [PutBucketNotification](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este exemplo define a configuração do tópico do SNS para o evento ObjectRemovedDelete do S3 e habilita a notificação para o bucket do S3 em questão

```

$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

```

Exemplo 2: este exemplo habilita notificações de ObjectCreatedAll para o bucket em questão, enviando-as à função do 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

```

Exemplo 3: este exemplo cria duas configurações diferentes do Lambda com base em diferentes sufixos de chave e configura ambas em um único 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
```

- Consulte detalhes da API em [PutBucketNotification](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este exemplo define a configuração do tópico do SNS para o evento ObjectRemovedDelete do S3 e habilita a notificação para o bucket do S3 em questão

```
$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
```

Exemplo 2: este exemplo habilita notificações de ObjectCreatedAll para o bucket em questão, enviando-as à função do 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
```

Exemplo 3: este exemplo cria duas configurações diferentes do Lambda com base em diferentes sufixos de chave e configura ambas em um único 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
```

- Consulte detalhes da API em [PutBucketNotification](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketNotificationConfiguration** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o `PutBucketNotificationConfiguration`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Processar notificações de eventos do S3](#)
- [Enviar notificações de eventos ao EventBridge](#)

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [PutBucketNotificationConfiguration](#) na Referência da API do AWS SDK para .NET.

CLI

AWS CLI

Como habilitar as notificações especificadas em um bucket

O exemplo de `put-bucket-notification-configuration` a seguir aplica uma configuração de notificação a um bucket chamado `amzn-s3-demo-bucket`. O arquivo `notification.json` é um documento JSON na pasta atual que especifica um tópico do SNS e o tipo de evento a ser monitorado.

```
aws s3api put-bucket-notification-configuration \
    --bucket amzn-s3-demo-bucket \
    --notification-configuration file://notification.json
```

Conteúdo de `notification.json`:

```
{
  "TopicConfigurations": [
    {
```

```
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-
topic",
        "Events": [
            "s3:ObjectCreated:*"
        ]
    }
]
```

O tópico do SNS deve ter uma política do IAM anexada que permita que o Amazon S3 publique nele.

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

- Consulte detalhes da API em [PutBucketNotificationConfiguration](#) na Referência de comandos da AWS CLI.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketPolicy** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutBucketPolicy.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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\"\"\"      }, \n"
        "      \"Action\": [ \"s3:getObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3::\"
        + bucketName +
        \"/*\" ] \n"
        "    } \n"
        "  ] \n"
        "}";
}

```

- Para obter informações da API, consulte [PutBucketPolicy](#) na Referência da API do AWS SDK para C++.

CLI

AWS CLI

Esse exemplo permite que todos os usuários recuperem qualquer objeto em `amzn-s3-demo-bucket`, exceto os objetos em `MySecretFolder`. Ele também concede permissões `put` e `delete` ao usuário-raiz da conta da 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/*"
    }
  ]
}
```

- Consulte detalhes da API em [PutBucketPolicy](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter informações da API, consulte [PutBucketPolicy](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Adicione a 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [PutBucketPolicy](#) na Referência da API AWS SDK para JavaScript.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter informações da API, consulte [PutBucketPolicy](#) na Referência da API AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
# 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 obter informações da API, consulte [PutBucketPolicy](#) na Referência da API do AWS SDK para Ruby.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [PutBucketPolicy](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketReplication** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutBucketReplication.

CLI

AWS CLI

Como configurar a replicação para um bucket do S

O exemplo `put-bucket-replication` a seguir aplica uma configuração de replicação ao bucket do S3 especificado.

```
aws s3api put-bucket-replication \  
  --bucket amzn-s3-demo-bucket1 \  
  --replication-configuration file://replication.json
```

Conteúdo 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"  
      }  
    }  
  ]  
}
```

O bucket de destino deve ter o versionamento habilitado. A função especificada deve ter permissão para gravar no bucket de destino e ter uma relação de confiança que permita que o Amazon S3 assuma a função.

Exemplo de política de permissão de função:

```
{  
  "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/*"
    }
  ]
}
```

Exemplo de política de relação de confiança:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Este comando não produz saída.

Para obter mais informações, consulte [This is the topic title](#) no Guia do usuário do Amazon Simple Storage Service.

- Consulte detalhes da API em [PutBucketReplication](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [PutBucketReplication](#) na Referência de API do AWS SDK for Java 2.x.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: esse exemplo define uma configuração de replicação com uma única regra que permite a replicação para o bucket “amzn-s3-demo-bucket” de qualquer novo objeto criado com o prefixo de nome de chave “TaxDocs” no 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
```

Exemplo 2: este exemplo define uma configuração de replicação com várias regras que permitem a replicação para o bucket “amzn-s3-demo-bucket” de qualquer novo objeto criado com o prefixo de nome de chave “TaxDocs” ou “OtherDocs”. Os prefixos de chave não devem se sobrepor.

```
$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
```

Exemplo 3: este exemplo atualiza a configuração de replicação no bucket especificado para desabilitar a regra que controla a replicação de objetos com o prefixo de nome de chave “TaxDocs” para o 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
```

- Consulte detalhes da API em [PutBucketReplication](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: esse exemplo define uma configuração de replicação com uma única regra que permite a replicação para o bucket “amzn-s3-demo-bucket” de qualquer novo objeto criado com o prefixo de nome de chave “TaxDocs” no 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
```

Exemplo 2: este exemplo define uma configuração de replicação com várias regras que permitem a replicação para o bucket “amzn-s3-demo-bucket” de qualquer novo objeto criado com o prefixo de nome de chave “TaxDocs” ou “OtherDocs”. Os prefixos de chave não devem se sobrepor.

```
$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
```

Exemplo 3: este exemplo atualiza a configuração de replicação no bucket especificado para desabilitar a regra que controla a replicação de objetos com o prefixo de nome de chave "TaxDocs" para o 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
```

- Consulte detalhes da API em [PutBucketReplication](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketRequestPayment** com uma CLI

Os exemplos de código a seguir mostram como usar o PutBucketRequestPayment.

CLI

AWS CLI

Exemplo 1: para habilitar a configuração “requester pays” para um bucket

O exemplo `put-bucket-request-payment` a seguir habilita `requester pays` do bucket especificado.

```
aws s3api put-bucket-request-payment \
  --bucket amzn-s3-demo-bucket \
  --request-payment-configuration '{"Payer":"Requester"}'
```

Este comando não produz saída.

Exemplo 2: para desabilitar a configuração “requester pays” para um bucket

O exemplo `put-bucket-request-payment` a seguir desabilita `requester pays` do bucket especificado.

```
aws s3api put-bucket-request-payment \
  --bucket amzn-s3-demo-bucket \
  --request-payment-configuration '{"Payer":"BucketOwner"}'
```

Este comando não produz saída.

- Consulte detalhes da API em [PutBucketRequestPayment](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: atualiza a configuração de pagamento de solicitação do bucket denominado “amzn-s3-demo-bucket” para que a pessoa que está solicitando downloads pelo bucket seja cobrada pelo download. Por padrão, o proprietário do bucket paga pelos downloads. Para definir o pagamento de solicitação de volta ao padrão, use “BucketOwner” para o parâmetro `RequestPaymentConfiguration_Payer`.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -
RequestPaymentConfiguration_Payer Requester
```

- Consulte detalhes da API em [PutBucketRequestPayment](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: atualiza a configuração de pagamento de solicitação do bucket denominado “amzn-s3-demo-bucket” para que a pessoa que está solicitando downloads pelo bucket seja cobrada pelo download. Por padrão, o proprietário do bucket paga pelos downloads. Para definir o pagamento de solicitação de volta ao padrão, use “BucketOwner” para o parâmetro RequestPaymentConfiguration_Payer.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- Consulte detalhes da API em [PutBucketRequestPayment](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketTagging** com uma CLI

Os exemplos de código a seguir mostram como usar o PutBucketTagging.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Introdução ao armazenamento de objetos](#)

CLI

AWS CLI

O seguinte comando aplica a configuração de marcação a um bucket denominado amzn-s3-demo-bucket:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://  
tagging.json
```

O arquivo tagging.json é um documento JSON na pasta atual que especifica duas tags:


```
{
  "TagSet": [
    {
      "Key": "organization",
      "Value": "marketing"
    }
  ]
}
```

Ou aplique uma configuração de marcação a `amzn-s3-demo-bucket` diretamente da linha de comando:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging
'TagSet=[{Key=organization,Value=marketing}]'
```

- Consulte detalhes da API em [PutBucketTagging](#) na Referência de comandos da AWS CLI.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando aplica duas tags a um bucket denominado **cloudtrail-test-2018** (uma tag com uma chave de Stage e um valor de Test e uma tag com uma chave de Environment e um valor de Alpha). Para verificar se as tags foram adicionadas ao bucket, execute **Get-S3BucketTagging -BucketName bucket_name**. Os resultados devem mostrar as tags que você aplicou ao bucket no primeiro comando. Observe que **Write-S3BucketTagging** substitui todo o conjunto de tags existente em um bucket. Para adicionar ou excluir tags individuais, execute os cmdlets da API de grupos de recursos e marcação: **Add-RGTResourceTag** e **Remove-RGTResourceTag**. Também é possível usar o editor de tags no Console de Gerenciamento da AWS para gerenciar as tags de bucket do S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
  Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Exemplo 2: este comando envia um bucket denominado **cloudtrail-test-2018** para o cmdlet **Write-S3BucketTagging**. Ele aplica as tags Stage:Production e Department:Finance ao bucket. Observe que **Write-S3BucketTagging** substitui todo o conjunto de tags existente em um bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
Value="Finance" } )
```

- Consulte detalhes da API em [PutBucketTagging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando aplica duas tags a um bucket denominado **cloudtrail-test-2018** (uma tag com uma chave de Stage e um valor de Test e uma tag com uma chave de Environment e um valor de Alpha). Para verificar se as tags foram adicionadas ao bucket, execute **Get-S3BucketTagging -BucketName bucket_name**. Os resultados devem mostrar as tags que você aplicou ao bucket no primeiro comando. Observe que **Write-S3BucketTagging** substitui todo o conjunto de tags existente em um bucket. Para adicionar ou excluir tags individuais, execute os cmdlets da API de grupos de recursos e marcação: **Add-RGTResourceTag** e **Remove-RGTResourceTag**. Também é possível usar o editor de tags no Console de Gerenciamento da AWS para gerenciar as tags de bucket do S3.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Exemplo 2: este comando envia um bucket denominado **cloudtrail-test-2018** para o cmdlet **Write-S3BucketTagging**. Ele aplica as tags Stage:Production e Department:Finance ao bucket. Observe que **Write-S3BucketTagging** substitui todo o conjunto de tags existente em um bucket.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
Value="Finance" } )
```

- Consulte detalhes da API em [PutBucketTagging](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketVersioning** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutBucketVersioning.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Introdução ao armazenamento de objetos](#)

CLI

AWS CLI

O comando a seguir habilita o versionamento em um bucket denominado `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled
```

O comando a seguir habilita o versionamento e usa um código mfa.

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled --mfa "SERIAL 123456"
```

- Consulte detalhes da API em [PutBucketVersioning](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [PutBucketVersioning](#) na Referência de API do AWS SDK for Java 2.x.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando habilita o versionamento do bucket do S3 em questão.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- Consulte detalhes da API em [PutBucketVersioning](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando habilita o versionamento do bucket do S3 em questão.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -
VersioningConfig_Status Enabled
```

- Consulte detalhes da API em [PutBucketVersioning](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [PutBucketVersioning](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketWebsite** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutBucketWebsite.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
// Put the website configuration.
PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
{
    BucketName = bucketName,
    WebsiteConfiguration = new WebsiteConfiguration()
    {
        IndexDocumentSuffix = indexDocumentSuffix,
        ErrorDocument = errorDocument,
    },
};
PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- Consulte detalhes da API em [PutBucketWebsite](#) na Referência da API AWS SDK para .NET.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
```

```

        const Aws::String &indexPage, const Aws::String
&errorPage,
        const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
        client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
            << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
            << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- Consulte detalhes da API em [PutBucketWebsite](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

Aplica uma configuração de site estático ao bucket `amzn-s3-demo-bucket`:

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-configuration file://website.json
```

O arquivo `website.json` é um documento JSON na pasta atual que especifica páginas de índice e erro para o site:

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- Consulte detalhes da API em [PutBucketWebsite](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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);
}
}
```

- Consulte detalhes da API em [PutBucketWebsite](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina a configuração do site.

```
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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [PutBucketWebsite](#) na Referência da API do AWS SDK para JavaScript.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando habilita a hospedagem de sites para o bucket em questão com o documento de índice como “index.html” e o documento de erro como “error.html”.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Consulte detalhes da API em [PutBucketWebsite](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando habilita a hospedagem de sites para o bucket em questão com o documento de índice como “index.html” e o documento de erro como “error.html”.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Consulte detalhes da API em [PutBucketWebsite](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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__
```

- Consulte detalhes da API em [PutBucketWebsite](#) na Referência da API AWS SDK para Ruby.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutObject** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o `PutObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Conheça os conceitos básicos](#)
- [Introdução ao armazenamento de objetos](#)
- [Fazer solicitações condicionais](#)
- [Monitorar uploads e downloads](#)
- [Trabalhar com a integridade de objetos do Amazon S3](#)

.NET

SDK para .NET (v4)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [PutObject](#) na Referência da API AWS SDK para .NET.

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Faça upload de um objeto com criptografia do lado do 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");
    }
}
```

```
    }  
  }  
}
```

Coloque um objeto usando uma solicitação 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;
    }
}

```

- Consulte detalhes da API em [PutObject](#) na Referência da API AWS SDK para .NET.

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

#####
# 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
}
```

- Consulte detalhes da API em [PutObject](#) na Referência de comandos da AWS CLI.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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();
}

```

- Consulte detalhes da API em [PutObject](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

Exemplo 1: fazer upload de um objeto no Amazon S3

O exemplo do comando `put-object` a seguir faz upload de um objeto no Amazon S3:

```
aws s3api put-object \
```

```
--bucket amzn-s3-demo-bucket \  
--key my-dir/MySampleImage.png \  
--body MySampleImage.png
```

Consulte informações sobre o upload de objetos em “Fazer upload de objetos” < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html> > no Guia do desenvolvedor do Amazon S3.

Exemplo 2: fazer upload de um arquivo de vídeo no Amazon S3

O exemplo do comando `put-object` a seguir faz upload de um arquivo 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
```

Consulte informações sobre o upload de objetos em “Fazer upload de objetos” < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html> > no Guia do desenvolvedor do Amazon S3.

- Consulte detalhes da API em [PutObject](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Coloque um objeto em um bucket usando a API de baixo nível.

```
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
}
```

Faça upload de um objeto em um bucket usando um gerenciador de transferência.

```
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
}
```

- Consulte detalhes da API em [PutObject](#) na Referência da API do AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Faça upload de um arquivo em um bucket usando um [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);
        }
    });
}
```

Use [S3TransferManager](#) para [fazer upload de um arquivo](#) em um bucket. Veja o [arquivo completo](#) e [teste](#).

```
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();
    }
```

Faça upload de um objeto em um bucket e defina etiquetas usando um [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;
}
```

```
}  
}
```

Faça upload de um objeto em um bucket e defina metadados usando um [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);
    }
}
}
```

Faça upload de um objeto em um bucket e defina um valor de retenção de objetos usando um [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);
    }
}
}
```

- Consulte detalhes da API em [PutObject](#) na Referência da API AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Carregue o 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;
    }
  }
};

```

Carregue o objeto com a condição de que sua ETag corresponda à fornecida.

```

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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).
- Consulte detalhes da API em [PutObject](#) na Referência da API do AWS SDK para JavaScript.

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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}")
    }
}
```

- Consulte detalhes da API em [PutObject](#) na Referência da API AWS SDK para Kotlin.

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Carregue um objeto em um 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.");
}
```

- Consulte detalhes da API em [PutObject](#) na Referência da API AWS SDK para PHP.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: este comando carrega o arquivo único “local-sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```


Exemplo 2: este comando carrega o arquivo único “sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”. Quando o parâmetro -Key não é fornecido, usa-se o nome do arquivo como chave do objeto do S.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Exemplo 3: este comando carrega o arquivo único “local-sample.txt” no Amazon S3, criando um objeto com a chave “prefix/to/sample.txt” no bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -  
File .\local-sample.txt
```

Exemplo 4: este comando carrega todos os arquivos no subdiretório “Scripts” no bucket “test-files” e aplica o prefixo de chave comum “SampleScripts” a cada objeto. Cada arquivo carregado terá uma chave “SampleScripts/filename”, em que “filename” varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\
```

Exemplo 5: este comando carrega todos os arquivos *.ps1 no diretório local “Scripts” no bucket “test-files” e aplica o prefixo de chave comum “SampleScripts” a cada objeto. Cada arquivo carregado terá uma chave “SampleScripts/filename.ps1”, em que “filename” varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Exemplo 6: este comando cria um objeto do S3 contendo a string de conteúdo especificada com a chave “sample.txt”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Exemplo 7: este comando carrega o arquivo especificado (o nome do arquivo é usado como chave) e aplica as tags especificadas ao novo objeto.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Exemplo 8: este comando carrega recursivamente a pasta especificada e aplica as tags especificadas a todos os novos objetos.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles" -Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Consulte detalhes da API em [PutObject](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: este comando carrega o arquivo único “local-sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-sample.txt
```

Exemplo 2: este comando carrega o arquivo único “sample.txt” no Amazon S3, criando um objeto com a chave “sample.txt” no bucket “test-files”. Quando o parâmetro -Key não é fornecido, usa-se o nome do arquivo como chave do objeto do S.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Exemplo 3: este comando carrega o arquivo único “local-sample.txt” no Amazon S3, criando um objeto com a chave “prefix/to/sample.txt” no bucket “test-files”.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -File .\local-sample.txt
```

Exemplo 4: este comando carrega todos os arquivos no subdiretório “Scripts” no bucket “test-files” e aplica o prefixo de chave comum “SampleScripts” a cada objeto. Cada arquivo carregado terá uma chave “SampleScripts/filename”, em que “filename” varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix SampleScripts\
```

Exemplo 5: este comando carrega todos os arquivos *.ps1 no diretório local “Scripts” no bucket “test-files” e aplica o prefixo de chave comum “SampleScripts” a cada objeto. Cada arquivo carregado terá uma chave “SampleScripts/filename.ps1”, em que “filename” varia.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Exemplo 6: este comando cria um objeto do S3 contendo a string de conteúdo especificada com a chave "sample.txt".

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Exemplo 7: este comando carrega o arquivo especificado (o nome do arquivo é usado como chave) e aplica as tags especificadas ao novo objeto.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Exemplo 8: este comando carrega recursivamente a pasta especificada e aplica as tags especificadas a todos os novos objetos.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"  
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Consulte detalhes da API em [PutObject](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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()
```

Faça upload de um objeto usando uma solicitação 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
```

- Consulte detalhes da API em [PutObject](#) na Referência da API do AWS SDK para Python (Boto3).

Ruby

SDK para Ruby

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Carregue um arquivo usando um carregador gerenciado (`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__

```

Carregue um arquivo usando `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
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__
```

Carregue um arquivo usando `Object.put` e adicione criptografia do lado do 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__
```

- Consulte detalhes da API em [PutObject](#) na Referência da API AWS SDK para Ruby.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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)  
}
```

- Consulte detalhes da API em [PutObject](#) na Referência da API AWS SDK para Rust.

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
"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.
```

- Consulte detalhes da API em [PutObject](#) na Referência da API AWS SDK para SAP ABAP.

Swift

SDK para Swift

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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  
}  
}
```

- Consulte detalhes da API em [PutObject](#) na Referência da API AWS SDK para Swift.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutObjectAcl** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutObjectAcl.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Gerenciar listas de controle de acesso \(ACLs\)](#)

C++

SDK para C++

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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;
}
```

- Consulte detalhes da API em [PutObjectAcl](#) na Referência da API AWS SDK para C++.

CLI

AWS CLI

O seguinte comando concede `full control` a dois usuários da AWS (`user1@example.com` e `user2@example.com`) e a permissão `read` a todos:

```
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 obter detalhes sobre ACLs personalizadas (os comandos da ACL `s3api`, como `put-object-acl`, usam a mesma notação abreviada de argumentos).

- Consulte detalhes da API em [PutObjectAcl](#) na Referência de comandos da AWS CLI.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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

```

- Consulte detalhes da API em [PutObjectAcl](#) na Referência da API AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [PutObjectAcl](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutObjectLegalHold** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutObjectLegalHold.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- [Bloquear objetos do Amazon S3](#)

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

```
}
```

- Consulte detalhes da API em [PutObjectLegalHold](#) na Referência de API do AWS SDK para .NET.

CLI

AWS CLI

Como aplicar uma retenção legal a um objeto

O exemplo de `put-object-legal-hold` a seguir define uma retenção legal no 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 não produz saída.

- Consulte detalhes da API em [PutObjectLegalHold](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

- Consulte detalhes da API em [PutObjectLegalHold](#) na Referência de API do AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
// 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 + ".");
}
```

- Consulte detalhes da API em [PutObjectLegalHold](#) na Referência de API do AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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"));
  }
}

```

- Consulte detalhes da API em [PutObjectLegalHold](#) na Referência de API do AWS SDK para JavaScript.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina a retenção legal de objetos.

```

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 receber detalhes da API, consulte [PutObjectLegalHold](#) na Referência da API do AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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 obter detalhes da API, consulte [PutObjectLegalHold](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutObjectLockConfiguration** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutObjectLockConfiguration.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- [Bloquear objetos do Amazon S3](#)

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina a configuração de Bloqueio de Objetos de um 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($"\\tAdded an object lock policy to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
        return false;
    }
}
```

Defina o período de retenção padrão de um 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;
}
}
```

- Consulte detalhes da API em [PutObjectLockConfiguration](#) na Referência de API do AWS SDK para .NET.

CLI

AWS CLI

Como definir a configuração de bloqueio de objetos em um bucket

O exemplo de `put-object-lock-configuration` a seguir define um bloqueio de objetos de 50 dias no 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 não produz saída.

- Consulte detalhes da API em [PutObjectLockConfiguration](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina a configuração de Bloqueio de Objetos de um 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
}

```

Defina o período de retenção padrão de um 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
}
```

- Consulte detalhes da API em [PutObjectLockConfiguration](#) na Referência de API do AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina a configuração de Bloqueio de Objetos de um 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()
+ "'");
    }
}
```

Defina o período de retenção padrão de um 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
+ ".");
    }
```

- Consulte detalhes da API em [PutObjectLockConfiguration](#) na Referência de API do AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina a configuração de Bloqueio de Objetos de um 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"));
    }
  }
}
```

Defina o período de retenção padrão de um 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),
            },
        },
    },
    }),
);
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"));
    }
}
```

- Consulte detalhes da API em [PutObjectLockConfiguration](#) na Referência de API do AWS SDK para JavaScript.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina a configuração de bloqueio de objetos.

```
s3_client.put_object_lock_configuration(  
    Bucket=bucket,  
    ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":  
{}},  
)
```

- Para receber detalhes da API, consulte [PutObjectLockConfiguration](#) na Referência da API do AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter informações da API, consulte [PutObjectLockConfiguration](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutObjectRetention** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o PutObjectRetention.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação no contexto no seguinte exemplo de código:

- [Bloquear objetos do Amazon S3](#)

.NET

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [PutObjectRetention](#) na Referência de API do AWS SDK para .NET.

CLI

AWS CLI

Como definir uma configuração de retenção de objetos para um objeto

O exemplo de `put-object-retention` a seguir define uma configuração de retenção de objetos para o objeto especificado até 2025-01-01.

```
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 não produz saída.

- Consulte detalhes da API em [PutObjectRetention](#) na Referência de comandos da AWS CLI.

Go

SDK para Go V2

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
}
```

- Consulte detalhes da API em [PutObjectRetention](#) na Referência de API do AWS SDK para Go.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
// 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 +".");
}
```

- Consulte detalhes da API em [PutObjectRetention](#) na Referência de API do AWS SDK for Java 2.x.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [PutObjectRetention](#) na Referência de API do AWS SDK para JavaScript.

PowerShell

Ferramentas para PowerShell V4

Exemplo 1: o comando habilita o modo de retenção de governança até a data “31st Dec 2019 00:00:00” para o objeto “testfile.txt” no bucket do S3 em questão.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Consulte detalhes da API em [PutObjectRetention](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V4).

Ferramentas para PowerShell V5

Exemplo 1: o comando habilita o modo de retenção de governança até a data “31st Dec 2019 00:00:00” para o objeto “testfile.txt” no bucket do S3 em questão.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Consulte detalhes da API em [PutObjectRetention](#) na Referência de cmdlet do Ferramentas da AWS para PowerShell (V5).

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina uma retenção de objetos.

```
s3_client.put_object_retention(  
    Bucket=bucket,  
    Key=key,  
    VersionId=version_id,  
    Retention={"Mode": "GOVERNANCE", "RetainUntilDate":  
far_future_date},  
    BypassGovernanceRetention=True,  
)
```

- Para receber detalhes da API, consulte [PutObjectRetention](#) na Referência da API do AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [PutObjectRetention](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **RestoreObject** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o RestoreObject.

.NET

SDK para .NET

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 { }
        }
    }

```

```

    }
    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}");
}
}

```

- Consulte detalhes da API em [RestoreObject](#) em Referência da API AWS SDK para .NET.

CLI

AWS CLI

Como criar uma solicitação de restauração para um objeto

O exemplo de `restore-object` a seguir restaura o objeto Amazon S3 Glacier especificado do bucket `my-glacier-bucket` por 10 dias.

```
aws s3api restore-object \  
  --bucket my-glacier-bucket \  
  --key doc1.rtf \  
  --restore-request Days=10
```

Este comando não produz saída.

- Consulte detalhes da API em [RestoreObject](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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);
        }
    }
}

```

- Consulte detalhes da API em [RestoreObject](#) em Referência da API AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **SelectObjectContent** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o `SelectObjectContent`.

CLI

AWS CLI

Como filtrar o conteúdo de um objeto do Amazon S3 com base em uma instrução SQL

O exemplo de `select-object-content` a seguir filtra o objeto `my-data-file.csv` com a instrução SQL especificada e envia a saída para um arquivo.

```
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 não produz saída.

- Consulte detalhes da API em [SelectObjectContent](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

O exemplo a seguir mostra uma consulta usando um objeto JSON. O [exemplo completo](#) também mostra o uso de um 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;
    }
}
```

- Consulte detalhes da API em [SelectObjectContent](#) na Referência de API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **UploadPart** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o UploadPart.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Usar somas de verificação](#)
- [Trabalhar com a integridade de objetos do Amazon S3](#)

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
//! 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 receber detalhes da API, consulte [UploadPart](#) na Referência da API do AWS SDK para C++.

CLI

AWS CLI

O seguinte comando faz o upload da primeira parte de um upload fracionado iniciado com o 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'
```

A opção `body` usa o nome ou o caminho de um arquivo local para upload (não use o prefixo `file://`). O tamanho mínimo de uma parte é de 5 MB. O ID de upload é retornado por `create-multipart-upload` e também pode ser recuperado com `list-multipart-uploads`. O bucket e a chave são especificados ao criar o upload fracionado.

Resultado:

```
{
  "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}
```

Salve o valor do ETag de cada parte para etapas posteriores. Eles são necessários para concluir o upload fracionado.

- Consulte detalhes da API em [UploadPart](#) na Referência de comandos da AWS CLI.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [UploadPart](#) na Referência da API do AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **UploadPartCopy** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o UploadPartCopy.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Executar uma cópia multipart](#)

CLI

AWS CLI

Como fazer upload de parte de um objeto copiando dados de um objeto existente como a fonte de dados

O exemplo upload-part-copy a seguir faz upload de uma parte copiando dados de um objeto existente como uma fonte de dados.

```
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_RiBy0gg0H6b13pVRY7QjvL1f75iFdJqp_2wztk5hvpUM2SesXgrz"
```

Resultado:

```
{  
  "CopyPartResult": {  
    "LastModified": "2019-12-13T23:16:03.000Z",  
    "ETag": "\"711470fc377698c393d94aed6305e245\""  
  }  
}
```

- Para ver detalhes da API, consulte [UploadPartCopy](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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;  
}
```

- Consulte detalhes da API em [UploadPartCopy](#) na Referência de API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Cenários do Amazon S3 usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns no Amazon S3 com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções dentro do Amazon S3 ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- [Verificar se um bucket existe](#)
- [Verificar se um objeto existe](#)
- [Converter texto em fala e de volta em texto usando um AWS SDK](#)
- [Criar um URL pré-assinado para o Amazon S3 usando um AWS SDK](#)
- [Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos](#)
- [Uma página da web que oferece uma lista de objetos do Amazon S3 usando um AWS SDK](#)
- [Criar uma aplicação de exploração do Amazon Textract](#)
- [Excluir todos os objetos de um bucket específico do Amazon S3 utilizando um SDK da AWS](#)

- [Excluir uploads multiparte incompletos para o Amazon S3 usando um AWS SDK](#)
- [Detectar EPI em imagens com o Amazon Rekognition usando um AWS SDK](#)
- [Detectar entidades em texto extraído de uma imagem usando um AWS SDK](#)
- [Detectar faces em uma imagem usando um AWS SDK](#)
- [Detectar objetos em imagens com o Amazon Rekognition usando um AWS SDK](#)
- [Detectar pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK](#)
- [Baixar “diretórios” do S3 de um bucket do Amazon Simple Storage Service \(Amazon S3\)](#)
- [Fazer download de todos os objetos de um bucket do Amazon Simple Storage Service \(Amazon S3\) em um diretório local](#)
- [Baixar um fluxo de tamanho desconhecido em um objeto do Amazon S3 usando um AWS SDK](#)
- [Obter um objeto do Amazon S3 de um ponto de acesso multirregional usando um AWS SDK](#)
- [Obter um objeto de um bucket do Amazon S3 usando um AWS SDK e especificando um cabeçalho If-Modified-Since](#)
- [Conceitos básicos de criptografia para objetos do Amazon S3 usando um AWS SDK](#)
- [Conceitos básicos de etiquetas para objetos do Amazon S3 usando um AWS SDK](#)
- [Introdução aos clusters de processamento de big data](#)
- [Conceitos básicos do gerenciamento de configurações](#)
- [Introdução à extração de texto de documentos](#)
- [Conceitos básicos de arquivos de atributos de machine learning](#)
- [Introdução ao armazenamento de objetos](#)
- [Introdução à análise de consulta](#)
- [Trabalhar com os recursos de bloqueio de objetos do Amazon S3 usando um SDK da AWS](#)
- [Fazer solicitações condicionais do Amazon S3 usando um AWS SDK](#)
- [Gerenciar listas de controle de acesso \(ACLs\) para buckets do Amazon S3 usando um AWS SDK](#)
- [Gerenciar mensagens grandes do Amazon SQS usando o Amazon S3 com um AWS SDK](#)
- [Gerenciar objetos do Amazon S3 versionados em lotes com uma função do Lambda usando um AWS SDK](#)
- [Analisar URIs do Amazon S3 usando um AWS SDK](#)
- [Executar uma cópia multipart de um objeto do Amazon S3 usando um AWS SDK](#)
- [Receber e processar notificações de eventos do Amazon S3 usando um AWS SDK](#)
- [Salvar o EXIF e outras informações de imagem usando um AWS SDK](#)

- [Enviar notificações de eventos do S3 ao Amazon EventBridge usando um AWS SDK](#)
- [Monitorar o upload ou download de um objeto do Amazon S3 usando um SDK da AWS](#)
- [Como transformar dados para sua aplicação com o S3 Object Lambda](#)
- [Exemplos de abordagens para testes de unidade e integração com SDK da AWS](#)
- [Fazer upload recursivo de um diretório local em um bucket do Amazon Simple Storage Service \(Amazon S3\)](#)
- [Fazer upload ou download de arquivos grandes de e para o Amazon S3 usando um AWS SDK](#)
- [Faça upload de um fluxo de tamanho desconhecido em um objeto do Amazon S3 usando um AWS SDK](#)
- [Usar somas de verificação para trabalhar com um objeto do Amazon S3 usando um AWS SDK](#)
- [Trabalhar com os recursos de integridade de objetos do Amazon S3 usando um AWS SDK](#)
- [Trabalhar com objetos versionados do Amazon S3 usando um AWS SDK](#)

Verificar se um bucket existe

O exemplo de código a seguir mostra como verificar se um bucket existe.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Você pode usar o método `doesBucketExists` a seguir como substituto do método [AmazonS3Client#doesBucketExistV2\(String\)](#) do 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.statusCode() == HttpStatus.NOT_FOUND) {
            return false;
        }
        throw ase;
    }
}
```

- Consulte detalhes da API em [GetBucketAcl](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Verificar se um objeto existe

O exemplo de código a seguir mostra como verificar se um objeto existe.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Você pode usar o método `doesObjectExist` a seguir como substituto do método [AmazonS3Client#doesObjectExist\(String, String\)](#) do 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 obter detalhes da API, consulte [HeadObject](#) em Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Converter texto em fala e de volta em texto usando um AWS SDK

O código de exemplo abaixo mostra como:

- Usar o Amazon Polly para sintetizar um arquivo de entrada de texto simples (UTF-8) para um arquivo de áudio.
- Fazer upload do arquivo de áudio para um bucket do Amazon S3.
- Usar o Amazon Transcribe para converter o arquivo de áudio em texto.
- Exibir o texto.

Rust

SDK para Rust

Use o Amazon Polly para sintetizar um arquivo de texto simples (UTF-8) para um arquivo de áudio, fazer upload do arquivo de áudio para um bucket do Amazon S3, usar o Amazon Transcribe para converter esse arquivo de áudio em texto e exibir o texto.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços utilizados neste exemplo

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Criar um URL pré-assinado para o Amazon S3 usando um AWS SDK

Os exemplos de código a seguir mostram como criar um URL pré-assinado para o Amazon S3 e carregar um objeto.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Gere um URL pré-assinado que possa executar uma ação do Amazon S3 por tempo 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;
    }
}
```

Gere um URL pré-assinado e faça um upload usando esse 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie e use URLs POST pré-assinados para uploads diretos do 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Gere um URL pré-assinado para baixar um 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);
    }
```

Baixe usando a 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;
}

```

Gere um URL pré-assinado para carregar um 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);
}

```

Carregue usando a 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie funções que envolvam ações de pré-assinatura do 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
}
```

Execute um exemplo interativo que gera e usa URLs pré-assinados para fazer upload, fazer download e excluir um objeto do S3.

```
import (
    "bytes"
    "context"
    "io"
    "log"
    "mime/multipart"
    "net/http"
    "os"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunPresigningScenario is an interactive example that shows you how to get
// presigned
// HTTP requests that you can use to move data into and out of Amazon Simple
// Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and
// can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local
//    file to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the
//    object to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
```



```
// This example creates an Amazon S3 presign client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
// mocked
// during testing.
func RunPresigningScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner, httpRequester IHttpRequester) {
defer func() {
if r := recover(); r != nil {
log.Println("Something went wrong with the demo.")
_, isMock := questioner.(*demotools.MockQuestioner)
if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
log.Println(r)
}
}
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 presigning demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}
presignClient := s3.NewPresignClient(s3Client)
presigner := actions.Presigner{PresignClient: presignClient}

bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
"you own or one you want to create:", demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
panic(err)
}
if !bucketExists {
err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
if err != nil {
panic(err)
} else {
```

```
    log.Println("Bucket created.")
}
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
    demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
    presignedPutRequest.Method,
    presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()
if err != nil {
    panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
    uploadFile)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
    presignedPutRequest.Method,
    uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
```

```
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Now we'll create a new request to put the same object using a
presigned post request")
questioner.Ask("Press Enter when you're ready.")
presignPostRequest, err := presigner.PresignPostObject(ctx, bucketName,
uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned post request to url %v with values %v\n",
presignPostRequest.URL, presignPostRequest.Values)
log.Println("Using net/http multipart to send the request...")
uploadFile, err = os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
multiPartResponse, err := sendMultipartRequest(presignPostRequest.URL,
presignPostRequest.Values, uploadFile, uploadKey, httpRequester)
if err != nil {
    panic(err)
}
```

```

log.Printf("Presign post object %v with presigned URL returned %v.", uploadKey,
multiPartResponse.StatusCode)

log.Println("Let's presign a request to delete the object.")
questioner.Ask("Press Enter when you're ready.")
presignedDelRequest, err := presigner.DeleteObject(ctx, bucketName, uploadKey)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedDelRequest.Method,
presignedDelRequest.URL)
log.Println("Using net/http to send the request...")
delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.\n",
presignedDelRequest.Method,
uploadKey, delResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Defina um wrapper de solicitação HTTP usado pelo exemplo para fazer solicitações HTTP.

```

// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
// during
// unit testing.
type IHttpRequester interface {
    Get(url string) (resp *http.Response, err error)
    Post(url, contentType string, body io.Reader) (resp *http.Response, err error)
    Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err error)
    Delete(url string) (resp *http.Response, err error)
}

```

```
// HttpRequester uses the net/http package to make HTTP requests during the
// scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
    return http.Get(url)
}

func (httpReq HttpRequester) Post(url, contentType string, body io.Reader) (resp
    *http.Response, err error) {
    postRequest, err := http.NewRequest("POST", url, body)
    if err != nil {
        return nil, err
    }
    postRequest.Header.Set("Content-Type", contentType)
    return http.DefaultClient.Do(postRequest)
}

func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
    (resp *http.Response, err error) {
    putRequest, err := http.NewRequest("PUT", url, body)
    if err != nil {
        return nil, err
    }
    putRequest.ContentLength = contentLength
    return http.DefaultClient.Do(putRequest)
}

func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error)
    {
    delRequest, err := http.NewRequest("DELETE", url, nil)
    if err != nil {
        return nil, err
    }
    return http.DefaultClient.Do(delRequest)
}
```

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Veja a seguir três exemplos de como criar URLs pré-assinados e usar os URLs com bibliotecas de cliente HTTP:

- Uma solicitação HTTP GET que usa o URL com três bibliotecas de cliente HTTP
- Uma solicitação HTTP PUT com metadados nos cabeçalhos que usa o URL com três bibliotecas de cliente HTTP
- Uma solicitação HTTP PUT com parâmetros de consulta que usa o URL com uma biblioteca cliente HTTP

Gere um URL pré-assinado para um objeto e faça download dele (solicitação GET).

Importações.

```
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;
```

Gere o 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();
    }
}
```

Faça o download do objeto usando uma das três abordagens a seguir.

Use a classe `URLConnection` do JDK (desde a v1.1) para fazer o download.

```
/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useURLConnectionToGet(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();
}
```

Use a classe `HttpClient` do JDK (desde a v11) para fazer o download.

```
/* 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();
}

```

Use a classe `SdkHttpClient` do AWS SDK para Java para fazer o download.

```

/* 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();
}
```

Gere um URL pré-assinado com metadados em cabeçalhos para um upload e faça upload de um arquivo (solicitação PUT).

Importações.

```
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;
```

Gere o 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();
    }
}
```

Faça upload de um objeto de arquivo usando uma das três abordagens a seguir.

Use a classe `URLConnection` do JDK (desde a v1.1) para fazer o upload.

```

/* 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);
    }
}

```

Use a classe `HttpClient` do JDK (desde a v11) para fazer o upload.

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

Use a classe `SdkHttpClient` do AWS para Java V2 para fazer o upload.

```
/* 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);
    }
}

```

Gere um URL pré-assinado com parâmetros de consulta para um upload e faça upload de um arquivo (solicitação PUT).

Importações.

```

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;
```

Gere o 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();
    }
}

```

Use a classe `SdkHttpClient` do AWS para Java V2 para fazer o upload.

```

/**
 * 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 uplaod
 */
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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie um URL pré-assinado para carregar um objeto em um 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;
    }
  }
};

```

Crie um URL pré-assinado para baixar um objeto de um 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para JavaScript](#).

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie uma solicitação pré-assinada de `GetObject` e use o URL para fazer download de um 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
}
```

Crie uma solicitação `GetObject` atribuída previamente com opções avançadas.

```
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
}
```

Crie uma solicitação pré-assinada de `PutObject` e use-a para fazer upload de um 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 ter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para Kotlin](#).

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Gere um URL pré-assinado que possa realizar uma ação do S3 por tempo limitado. Use o pacote Requests para fazer uma solicitação com o 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()

```

Gere uma solicitação POST pré-assinada para carregar um arquivo.

```
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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
require 'aws-sdk-s3'
require 'net/http'

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's
  why: #{e.message}"
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-file.txt"
  object_content = "This is the content of my-file.txt."

  bucket = Aws::S3::Bucket.new(bucket_name)
  presigned_url = get_presigned_url(bucket, object_key)
  return unless presigned_url

  response = Net::HTTP.start(presigned_url.host) do |http|
```

```
    http.send_request('PUT', presigned_url.request_uri, object_content,
'content_type' => '')
end

case response
when Net::HTTPSuccess
    puts 'Content uploaded!'
else
    puts response.value
end
end

run_demo if $PROGRAM_NAME == __FILE__
```

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie solicitações de pré-assinatura para objetos GET do 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(())
}
```

Crie solicitações de pré-assinatura para objetos PUT do 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie solicitações pré-assinadas para objetos GET do 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 ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Criar uma aplicação de gerenciamento de ativos de fotos que permita que os usuários gerenciem fotos usando rótulos

O exemplo de código a seguir mostra como criar uma aplicação com tecnologia sem servidor que permite que os usuários gerenciem fotos usando rótulos.

.NET

SDK para .NET

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK para C++

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK para Java 2.x

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços usados neste exemplo

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK para JavaScript (v3)

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK para Kotlin

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK para PHP

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK para Rust

Mostra como desenvolver uma aplicação de gerenciamento de ativos fotográficos que detecta rótulos em imagens usando o Amazon Rekognition e os armazena para recuperação posterior.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Para uma análise detalhada da origem desse exemplo, veja a publicação na [Comunidade da AWS](#).

Serviços usados neste exemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Para obter uma lista completa dos Guias do desenvolvedor do AWS SDK e exemplos de código, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Uma página da web que oferece uma lista de objetos do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como listar objetos do Amazon S3 em uma página da web.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

O código a seguir é o componente relevante do React que faz chamadas para o AWS SDK. Uma versão executável da aplicação que contém esse componente pode ser encontrada no link anterior do 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;
```

- Consulte detalhes da API em [ListObjects](#) na Referência da API AWS SDK para JavaScript.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Criar uma aplicação de exploração do Amazon Textract

Os exemplos de código a seguir mostram como explorar a saída do Amazon Textract por meio de uma aplicação interativa.

JavaScript

SDK para JavaScript (v3)

Mostra como usar o AWS SDK para JavaScript para criar uma aplicação React que usa o Amazon Textract para extrair dados de uma imagem de documento e exibi-los em uma página da Web interativa. Este exemplo é executado em um navegador da Web e requer uma identidade autenticada do Amazon Cognito como credenciais. Ele usa o Amazon Simple Storage Service (Amazon S3) para armazenamento e, para notificações, pesquisa uma fila do Amazon Simple Queue Service (Amazon SQS) que está inscrita em um tópico do Amazon Simple Notification Service (Amazon SNS).

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK para Python (Boto3).

Mostra como usar o AWS SDK para Python (Boto3) com o Amazon Textract para detectar elementos de texto, formulários e tabelas em uma imagem de documento. A imagem de entrada e a saída do Amazon Textract são mostradas em um aplicativo Tkinter que permite explorar os elementos detectados.

- Envie uma imagem de documento para o Amazon Textract e explore a saída dos elementos detectados.
- Envie imagens diretamente para o Amazon Textract ou por meio de um bucket do Amazon Simple Storage Service (Amazon S3).
- Use APIs assíncronas para iniciar um trabalho que publica uma notificação em um tópico do Amazon Simple Notification Service (Amazon SNS) quando o trabalho for concluído.

- Faça uma pesquisa em uma fila do Amazon Simple Queue Service (Amazon SQS) para obter uma mensagem de conclusão do trabalho e exiba os resultados.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Para obter uma lista completa dos Guias do desenvolvedor do SDK da AWS e exemplos de código, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Excluir todos os objetos de um bucket específico do Amazon S3 utilizando um SDK da AWS

Os exemplos de código a seguir mostram como excluir todos os objetos de um bucket do Amazon S3.

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua todos os objetos de um bucket específico do Amazon S3.

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript.
 - [DeleteObjects](#)
 - [ListObjectsV2](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Excluir uploads multiparte incompletos para o Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como excluir ou interromper uploads multiparte incompletos do Amazon S3.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Para interromper uploads multiparte que estão em andamento ou incompletos por qualquer motivo, você pode obter uma lista de uploads para excluí-los, conforme mostrado no exemplo a seguir.

```

/**
 * 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 excluir uploads multiparte incompletos que foram iniciados antes ou depois de uma data, você pode excluir seletivamente os uploads multiparte com base em um momento específico, conforme mostrado no exemplo a seguir.

```

/**
 * 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);
                }
            }
        }
    }
}

```

Se você tiver acesso ao ID de upload depois de iniciar um upload multiparte, poderá excluir o upload em andamento usando esse 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 excluir uploads multiparte incompletos após determinado número de dias de maneira consistente, defina uma configuração de ciclo de vida para o bucket. O exemplo a seguir mostra como criar uma regra para excluir uploads multiparte existentes há mais de 7 dias.

```

/**
 * 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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [PutBucketLifecycleConfiguration](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detectar EPI em imagens com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como criar uma aplicação que usa o Amazon Rekognition para detectar equipamentos de proteção individual (EPI) em imagens.

Java

SDK para Java 2.x

Mostra como criar uma função do AWS Lambda que detecta imagens com equipamento de proteção individual.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detectar entidades em texto extraído de uma imagem usando um AWS SDK

O exemplo de código a seguir mostra como usar o Amazon Comprehend para detectar entidades em texto extraído pelo Amazon Textract de uma imagem armazenada no Amazon S3.

Python

SDK para Python (Boto3).

Mostra como usar o AWS SDK para Python (Boto3) em um caderno Jupyter para detectar entidades em texto extraído de uma imagem. Este exemplo usa o Amazon Textract para extrair texto de uma imagem armazenada no Amazon Simple Storage Service (Amazon S3) e no Amazon Comprehend para detectar entidades no texto extraído.

Este exemplo é um caderno Jupyter e deve ser executado em um ambiente que possa hospedar blocos de anotações. Consulte instruções de como executar o exemplo usando o Amazon SageMaker AI em [TextractAndComprehendNotebook.ipynb](#).

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços utilizados neste exemplo

- Amazon Comprehend
- Amazon S3
- Amazon Textract

Para obter uma lista completa dos Guias do desenvolvedor do SDK da AWS e exemplos de código, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detectar faces em uma imagem usando um AWS SDK

O código de exemplo abaixo mostra como:

- Salvar uma imagem em um bucket do Amazon S3.
- Usar o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.).
- Exibir esses detalhes.

Rust

SDK para Rust

Salve a imagem em um bucket do Amazon S3 com um prefixo uploads, use o Amazon Rekognition para detectar detalhes faciais, como faixa etária, gênero e emoções (sorriso, etc.), e exiba esses detalhes.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detectar objetos em imagens com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como construir uma aplicação que usa o Amazon Rekognition para detectar objetos por categoria em imagens.

.NET

SDK para .NET

Mostra como usar a API .NET do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

JavaScript

SDK para JavaScript (v3)

Mostra como usar o Amazon Rekognition com o AWS SDK para JavaScript para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Aprenda como:

- Criar um usuário não autenticado usando o Amazon Cognito.
- Analisar imagens em busca de objetos usando o Amazon Rekognition.
- Verificar um endereço de e-mail para o Amazon SES.
- Enviar uma notificação por e-mail usando o Amazon SES.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK para Kotlin

Mostra como usar a API Kotlin do Amazon Rekognition para construir uma aplicação que usa o Amazon Rekognition para identificar objetos por categoria em imagens localizadas em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK para Python (Boto3).

Mostra como usar o AWS SDK para Python (Boto3) para criar uma aplicação Web que permite a você fazer o seguinte:

- Carregar fotos em um bucket do Amazon Simple Storage Service (Amazon S3).
- Usar o Amazon Rekognition para analisar e rotular as fotos.
- Usar o Amazon Simple Email Service (Amazon SES) para enviar relatórios de análise da imagem por e-mail.

Este exemplo contém dois componentes principais: uma página da Web escrita em JavaScript que é construída com o React e um serviço REST escrito em Python que é construído com o Flask-RESTful.

Você pode usar a página da Web do React para:

- Exibir uma lista de imagens que estão armazenadas no bucket do S3.
- Carregar imagens do computador para o bucket do S3.
- Exibir imagens e rótulos que identificam os itens detectados na imagem.
- Obter um relatório de todas as imagens no bucket do S3 e enviar um relatório por e-mail.

A página da Web chama o serviço REST. O serviço envia solicitações à AWS para realizar as seguintes ações:

- Obter e filtrar a lista de imagens no bucket do S3.
- Carregar fotos no bucket do S3.
- Usar o Amazon Rekognition para analisar fotos individuais e obter uma lista dos rótulos que identifiquem os itens detectados nas fotos.
- Analisar todas as fotos no bucket do S3 e usar o Amazon SES para enviar um relatório por e-mail.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Detectar pessoas e objetos em um vídeo com o Amazon Rekognition usando um AWS SDK

Os exemplos de código a seguir mostram como detectar pessoas e objetos em um vídeo com o Amazon Rekognition.

Java

SDK para Java 2.x

Mostra como usar a API Java do Amazon Rekognition a fim de construir uma aplicação para detectar faces e objetos em vídeos localizados em um bucket do Amazon Simple Storage Service (Amazon S3). A aplicação envia uma notificação por e-mail ao administrador com os resultados usando o Amazon Simple Email Service (Amazon SES).

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK para Python (Boto3).

Use o Amazon Rekognition para detectar faces, objetos e pessoas em vídeos iniciando trabalhos de detecção assíncrona. Este exemplo também configura o Amazon Rekognition para notificar um tópico do Amazon Simple Notification Service (Amazon SNS) quando os trabalhos são concluídos e inscreve uma fila do Amazon Simple Queue Service (Amazon SQS) no tópico. Quando a fila recebe uma mensagem sobre um trabalho, o trabalho é recuperado e os resultados são apresentados.

Este exemplo é melhor visualizado no GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Para obter uma lista completa dos Guias do desenvolvedor do SDK da AWS e exemplos de código, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Baixar “diretórios” do S3 de um bucket do Amazon Simple Storage Service (Amazon S3)

O exemplo de código a seguir mostra como baixar e filtrar o conteúdo de “diretórios” de bucket do Amazon S3.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Esse exemplo mostra como usar o [S3TransferManager](#) no AWS SDK for Java 2.x para baixar “diretórios” de um bucket do Amazon S3. Também demonstra como usar [DownloadFilters](#) na solicitação.

```
/**
 * 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 ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Fazer download de todos os objetos de um bucket do Amazon Simple Storage Service (Amazon S3) em um diretório local

O exemplo de código a seguir mostra como fazer download de todos os objetos de um bucket do Amazon Simple Storage Service (Amazon S3) em um diretório local.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Use [S3TransferManager](#) para [fazer download de todos os objetos do S3](#) no mesmo bucket do S3. Veja o [arquivo completo](#) e [teste](#).

```
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 obter detalhes da API, consulte [DownloadDirectory](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Baixar um fluxo de tamanho desconhecido em um objeto do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como baixar um fluxo de tamanho desconhecido de um objeto do Amazon S3.

Swift

SDK para Swift

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Obter um objeto do Amazon S3 de um ponto de acesso multirregional usando um AWS SDK

O exemplo de código a seguir demonstra como obter um objeto de um ponto de acesso multirregional.

Kotlin

SDK para Kotlin

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Configure o cliente do S3 para usar o algoritmo de assinatura Asymmetric 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  
}
```

Use o ARN do ponto de acesso multirregional em vez de um nome de bucket para recuperar o 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 ter mais informações, consulte o [Guia do desenvolvedor do AWS SDK para Kotlin](#).
- Consulte detalhes da API em [GetObject](#) na Referência da API do AWS SDK for Kotlin.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Obter um objeto de um bucket do Amazon S3 usando um AWS SDK e especificando um cabeçalho If-Modified-Since

O exemplo de código a seguir mostra como ler dados de um objeto em um bucket do S3, mas somente se esse bucket não tiver sido modificado desde a última recuperação.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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(())
}

```

- Consulte detalhes da API em [GetObject](#) na Referência da API AWS SDK para Rust.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Conceitos básicos de criptografia para objetos do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como começar a usar a criptografia de objetos do Amazon S3.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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>

```



```
call    /// <param name="client">The initialized Amazon S3 client object used to
        /// 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    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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET.
 - [CopyObject](#)
 - [GetObject](#)
 - [GetObjectMetadata](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Conceitos básicos de etiquetas para objetos do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como começar a usar etiquetas de objetos do Amazon S3.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

/// <summary>
/// This example shows how to work with tags in Amazon Simple Storage
/// Service (Amazon S3) objects.
/// </summary>
public class ObjectTag
{
    public static async Task Main()
```

```
{
    string bucketName = "amzn-s3-demo-bucket";
    string keyName = "newobject.txt";
    string filePath = @"*** file path ***";

    // Specify your bucket region (an example region is shown).
    RegionEndpoint bucketRegion = RegionEndpoint.USWest2;

    var client = new AmazonS3Client(bucketRegion);
    await PutObjectsWithTagsAsync(client, bucketName, keyName, filePath);
}

/// <summary>
/// This method uploads an object with tags. It then shows the tag
/// values, changes the tags, and shows the new tags.
/// </summary>
/// <param name="client">The Initialized Amazon S3 client object used
/// to call the methods to create and change an objects tags.</param>
/// <param name="bucketName">A string representing the name of the
/// bucket where the object will be stored.</param>
/// <param name="keyName">A string representing the key name of the
/// object to be tagged.</param>
/// <param name="filePath">The directory location and file name of the
/// object to be uploaded to the Amazon S3 bucket.</param>
public static async Task PutObjectsWithTagsAsync(IAmazonS3 client, string
bucketName, string keyName, string filePath)
{
    try
    {
        // Create an object with tags.
        var putRequest = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
            FilePath = filePath,
            TagSet = new List<Tag>
            {
                new Tag { Key = "Keyx1", Value = "Value1" },
                new Tag { Key = "Keyx2", Value = "Value2" },
            },
        };

        PutObjectResponse response = await
client.PutObjectAsync(putRequest);
```

```
// Now retrieve the new object's tags.
GetObjectTaggingRequest getTagsRequest = new
GetObjectTaggingRequest()
{
    BucketName = bucketName,
    Key = keyName,
};

GetObjectTaggingResponse objectTags = await
client.GetObjectTaggingAsync(getTagsRequest);

// Display the tag values.
objectTags.Tagging
    .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));

Tagging newTagSet = new Tagging()
{
    TagSet = new List<Tag>
    {
        new Tag { Key = "Key3", Value = "Value3" },
        new Tag { Key = "Key4", Value = "Value4" },
    },
};

PutObjectTaggingRequest putObjTagsRequest = new
PutObjectTaggingRequest()
{
    BucketName = bucketName,
    Key = keyName,
    Tagging = newTagSet,
};

PutObjectTaggingResponse response2 = await
client.PutObjectTaggingAsync(putObjTagsRequest);

// Retrieve the tags again and show the values.
GetObjectTaggingRequest getTagsRequest2 = new
GetObjectTaggingRequest()
{
    BucketName = bucketName,
    Key = keyName,
};
```

```
GetObjectTaggingResponse objectTags2 = await
client.GetObjectTaggingAsync(getTagsRequest2);

    objectTags2.Tagging
        .ForEach(t => Console.WriteLine($"Key: {t.Key}, Value:
{t.Value}"));
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine(
            $"Error: '{ex.Message}'");
    }
}
```

- Para obter detalhes da API, consulte [GetObjectTagging](#) na Referência da API do AWS SDK para .NET.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Introdução aos clusters de processamento de big data

O código de exemplo abaixo mostra como:

- Criar um par de chaves do EC2.
- Configurar o armazenamento e preparar uma aplicação.
- Limpar os recursos

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório [Sample developer tutorials](#).

```
#!/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."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI.
 - [AddSteps](#)
 - [Cp](#)
 - [CreateCluster](#)
 - [CreateDefaultRoles](#)
 - [CreateKeyPair](#)
 - [DescribeCluster](#)
 - [DescribeKeyPairs](#)
 - [DescribeStep](#)
 - [Ls](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)

- [ssh](#)
- [TerminateClusters](#)
- [Wait](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Conceitos básicos do gerenciamento de configurações

O código de exemplo abaixo mostra como:

- Criar um bucket do Amazon S3.
- Criar um tópico do Amazon SNS.
- Criar um perfil do IAM para o Config.
- Configurar o gravador de configuração do Config.
- Configurar o canal de entrega do Config.
- Iniciar o gravador de configuração.
- Verificar a configuração do Config

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório [Sample developer tutorials](#).

```
#!/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
}
```

```
fi

if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
"true" ]; then
    echo "Deleting configuration recorder..."
    aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
fi

if [ -n "$ROLE_NAME" ]; then
    if [ -n "$POLICY_NAME" ]; then
        echo "Detaching custom policy from role..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" 2>/dev/null || true
    fi

    if [ -n "$MANAGED_POLICY_ARN" ]; then
        echo "Detaching managed policy from role..."
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN" 2>/dev/null || true
    fi

    echo "Deleting IAM role..."
    aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
fi

if [ -n "$SNS_TOPIC_ARN" ]; then
    echo "Deleting SNS topic..."
    aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
fi

if [ -n "$S3_BUCKET_NAME" ]; then
    echo "Emptying S3 bucket..."
    aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

    echo "Deleting S3 bucket..."
    if [ "$BUCKET_IS_SHARED" = "false" ]; then
        aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
true
    fi
fi
}

# Function to display created resources
```

```

display_resources() {
    echo ""
    echo "=====
    echo "CREATED RESOURCES"
    echo "=====
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "=====
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"

```

```

else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-`${RANDOM_ID}`"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1" # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"

    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
"BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets=true"
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-`${RANDOM_ID}`"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"

```

```
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Action": [
      "s3:PutObject"
    ],
    "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
    "Condition": {
      "StringLike": {
        "s3:x-amz-acl": "bucket-owner-full-control"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "s3:GetBucketAcl"
    ],
    "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
  },
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "${SNS_TOPIC_ARN}"
  }
]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"

```

```
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-records 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
    "name": "${CONFIG_RECORDER_NAME}",
    "roleARN": "${ROLE_ARN}",
    "recordingMode": {
        "recordingFrequency": "CONTINUOUS"
    }
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
    "allSupported": true,
    "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
```

```
aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
  Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
  echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
  echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
  echo "Delivery channel already exists."
  # Get the name of the existing channel
  DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
| head -1 | cut -d'"' -f4)
  echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

  # Update the existing delivery channel
  echo "Creating delivery channel configuration for update..."
  cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

  echo "Updating delivery channel..."
  CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
  check_command "$CHANNEL_RESULT"
  echo "Delivery channel updated"
else
  echo "No existing delivery channel found. Will create a new one."
  CREATED_NEW_DELIVERY_CHANNEL="true"

  echo "Creating delivery channel configuration..."
  cat > deliveryChannel.json << EOF
```



```
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

echo "Creating delivery channel..."
CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
file://deliveryChannel.json)
check_command "$CHANNEL_RESULT"
echo "Delivery channel created"

aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
  echo "Configuration recorder is already running."
else
  echo "Starting configuration recorder..."
  START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
  check_command "$START_RESULT"
  echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-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!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da 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 ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Introdução à extração de texto de documentos

O código de exemplo abaixo mostra como:

- Criar um bucket do S3.
- Fazer upload de um documento no S3.
- Limpar os recursos

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório [Sample developer tutorials](#).

```
#!/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***/'
        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***/'
    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 $AWS_REGION..."
TEXTTRACT_CHECK=$(aws textract help 2>&1)
TEXTTRACT_CHECK_STATUS=$?
if [ $TEXTTRACT_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: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUhEUgAAAAEAAAABCAyAAAFfcSjAAAAADULEQVR42mNk
+M9QDwAdhgGAWjR9awAAAABJRU5ErkJggg==" | base64 -d > "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Created minimal sample document: $DOCUMENT_NAME"
    fi
fi

# Step 3: Upload document to S3
echo "Uploading document to S3..."
UPLOAD_OUTPUT=$(aws s3 cp "$DOCUMENT_NAME" "s3://$BUCKET_NAME/" --sse AES256
2>&1)
UPLOAD_STATUS=$?
echo "$UPLOAD_OUTPUT"
check_error $UPLOAD_STATUS "$UPLOAD_OUTPUT" "aws s3 cp "$DOCUMENT_NAME" s3://
$BUCKET_NAME/"
RESOURCES_CREATED+=("S3 Object: s3://$BUCKET_NAME/$DOCUMENT_NAME")

# Step 4: Analyze document with Amazon Textract
echo "Analyzing document with Amazon Textract..."
echo "This may take a few seconds..."

# Create a JSON file for the document parameter to avoid shell escaping issues

```

```
cat > document.json << 'EOF'
{
  "S3Object": {
    "Bucket": "BUCKET_PLACEHOLDER",
    "Name": "DOCUMENT_PLACEHOLDER"
  }
}
EOF

sed -i.bak "s|BUCKET_PLACEHOLDER|$BUCKET_NAME|g; s|DOCUMENT_PLACEHOLDER|
$DOCUMENT_NAME|g" document.json
rm -f document.json.bak
chmod 600 document.json

# Create a JSON file for the feature types parameter
cat > features.json << 'EOF'
["TABLES", "FORMS", "SIGNATURES"]
EOF
chmod 600 features.json

ANALYZE_OUTPUT=$(aws textract analyze-document --document file://document.json --
feature-types file://features.json 2>&1)
ANALYZE_STATUS=$?

echo "Analysis complete."
if [ $ANALYZE_STATUS -ne 0 ]; then
  echo "ERROR: Document analysis failed"
  echo "$ANALYZE_OUTPUT" | sed 's/\(aws_secret_access_key\|Authorization\|Token
\).*\/\1=***REDACTED***\/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

```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI.
 - [AnalyzeDocument](#)
 - [Cp](#)
 - [Help](#)
 - [Mb](#)

- [Rb](#)
- [Rm](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Conceitos básicos de arquivos de atributos de machine learning

O código de exemplo abaixo mostra como:

- Configurar permissões do IAM.
- Criar um perfil de execução do SageMaker.
- Criar grupo de atributos.
- Limpar os recursos

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório [Sample developer tutorials](#).

```
#!/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 [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$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
fi

```

```
echo "Script completed at $(date)"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI.
 - [AttachRolePolicy](#)
 - [CreateBucket](#)
 - [CreateFeatureGroup](#)
 - [CreateRole](#)
 - [DeleteBucket](#)
 - [DeleteFeatureGroup](#)
 - [DeleteRole](#)
 - [DescribeFeatureGroup](#)
 - [DetachRolePolicy](#)
 - [GetCallerIdentity](#)
 - [GetRole](#)
 - [ListFeatureGroups](#)
 - [PutPublicAccessBlock](#)
 - [Rm](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Introdução ao armazenamento de objetos

O código de exemplo abaixo mostra como:

- Criar seu primeiro bucket do S3.
- Fazer upload de um objeto.
- Habilitar o versionamento.
- Configurar a criptografia padrão.
- Adicionar tags a um bucket.
- Listar objetos e versões.
- Limpar os recursos

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório [Sample developer tutorials](#).

```
#!/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"
    fi
fi
```

```
    }
    }' >/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."
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [HeadObject](#)
 - [ListObjectVersions](#)
 - [ListObjectsV2](#)
 - [PutBucketEncryption](#)
 - [PutBucketTagging](#)
 - [PutBucketVersioning](#)
 - [PutObject](#)
 - [PutPublicAccessBlock](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Introdução à análise de consulta

O código de exemplo abaixo mostra como:

- Criar um bucket do S3 para resultados de consulta.
- Criar um banco de dados.

- Criar uma tabela.
- Executar uma consulta.
- Criar e usar consultas nomeadas.
- Limpar os recursos

Bash

AWS CLI com script Bash

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório [Sample developer tutorials](#).

```
#!/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,
    BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
    \"input.regex\" = \"^(?!#)([ ]+\\\\\\\\s+([ ]+\\\\\\\\s+([ ]+\\\\\\\\s+([ ]+\\\\\\\\s+
+([ ]+\\\\\\\\s+([ ]+\\\\\\\\s+([ ]+\\\\\\\\s+([ ]+\\\\\\\\s+([ ]+\\\\\\\\s+([ ]+\\\\\\\\s+([ ]+\\\\\\\\s+([ ]+\\\\\\\\s+
\\\\\\\\(]+\\\\\\\\(\\([\\\\\\\\\\\\\\\\;+).*\\\\\\\\%20(\\\\\\\\\\\\\\\\/+)\\\\\\\\\\\\\\\\/)(.*)$\"
) LOCATION 's3://athena-examples-us-east-1/cloudfront/plaintext/';"

CREATE_TABLE_RESULT=$(aws athena start-query-execution \\
    --query-string \"$CREATE_TABLE_QUERY\" \\
    --result-configuration \"OutputLocation=s3://$S3_BUCKET/output/\" \\
    --region \"$AWS_REGION\" 2>&1)

if echo \"$CREATE_TABLE_RESULT\" | grep -qi \"error\\|failed\"; then
    handle_error \"Failed to create table: $CREATE_TABLE_RESULT\"
fi

QUERY_ID=$(echo \"$CREATE_TABLE_RESULT\" | jq -r '.QueryExecutionId // empty' 2>/
dev/null || echo \"$CREATE_TABLE_RESULT\" | grep -o '\"QueryExecutionId\": \"[^\"]*\"' |
    cut -d'\"' -f4)
if [ -z \"$QUERY_ID\" ]; then
    handle_error \"Failed to extract Query ID from table creation response\"
fi
echo \"Table creation query ID: $QUERY_ID\"

# Wait for table creation to complete
echo \"Waiting for table creation to complete...\"
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
\"$QUERY_ID\" \\
        --query \"QueryExecution.Status.State\" --output text --region
\"$AWS_REGION\" 2>&1)
    if [ \"$QUERY_STATUS\" = \"SUCCEEDED\" ]; then
        echo \"Table creation completed successfully.\"
        break
    elif [ \"$QUERY_STATUS\" = \"FAILED\" ] || [ \"$QUERY_STATUS\" = \"CANCELLED\" ];
then
        handle_error \"Table creation failed with status: $QUERY_STATUS\"
    fi

```

```
    echo "Table creation in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Table creation timed out"
fi

# Verify the table was created
echo "Verifying table creation..."
LIST_TABLE_RESULT=$(aws athena list-table-metadata \
    --catalog-name AwsDataCatalog \
    --database-name "$DATABASE_NAME" \
    --region "$AWS_REGION" 2>&1)
if echo "$LIST_TABLE_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list tables: $LIST_TABLE_RESULT"
fi
echo "$LIST_TABLE_RESULT"

# Step 3: Query data
echo "Step 3: Running a query on the table..."
QUERY="SELECT os, COUNT(*) count
FROM $DATABASE_NAME.$TABLE_NAME
WHERE date BETWEEN date '2014-07-05' AND date '2014-08-05'
GROUP BY os"

QUERY_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to run query: $QUERY_RESULT"
fi

QUERY_ID=$(echo "$QUERY_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
|| echo "$QUERY_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from query execution response"
fi
echo "Query execution ID: $QUERY_ID"

# Wait for query to complete
```

```
echo "Waiting for query to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Query completed successfully."
        break
    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
        handle_error "Query failed with status: $QUERY_STATUS"
    fi
    echo "Query in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Query execution timed out"
fi

# Get query results
echo "Getting query results..."
RESULTS=$(aws athena get-query-results --query-execution-id "$QUERY_ID" --region
"$AWS_REGION" 2>&1)
if echo "$RESULTS" | grep -qi "error\|failed"; then
    handle_error "Failed to get query results: $RESULTS"
fi
echo "$RESULTS"

# Download results from S3
echo "Downloading query results from S3..."
S3_PATH=$(aws athena get-query-execution --query-execution-id "$QUERY_ID" \
    --query "QueryExecution.ResultConfiguration.OutputLocation" --output text \
    --region "$AWS_REGION" 2>&1)
if echo "$S3_PATH" | grep -qi "error\|failed"; then
    handle_error "Failed to get S3 path for results: $S3_PATH"
fi

if [ -z "$S3_PATH" ] || [ "$S3_PATH" = "None" ]; then
    handle_error "S3 path for query results is empty"
fi
```

```
DOWNLOAD_RESULT=$(aws s3 cp "$S3_PATH" "./query-results.csv" 2>&1)
if echo "$DOWNLOAD_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to download query results: $DOWNLOAD_RESULT"
fi

# Security: Secure the downloaded file
chmod 600 "./query-results.csv"
echo "Query results downloaded to query-results.csv (permissions: 600)"

# Step 4: Create a named query
echo "Step 4: Creating a named query..."
NAMED_QUERY_RESULT=$(aws athena create-named-query \
    --name "OS Count Query" \
    --description "Count of operating systems in CloudFront logs" \
    --database "$DATABASE_NAME" \
    --query-string "$QUERY" \
    --region "$AWS_REGION" 2>&1)

if echo "$NAMED_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to create named query: $NAMED_QUERY_RESULT"
fi

NAMED_QUERY_ID=$(echo "$NAMED_QUERY_RESULT" | jq -r '.NamedQueryId // empty' 2>/
dev/null || echo "$NAMED_QUERY_RESULT" | grep -o '"NamedQueryId": "[^"]*' | cut -
d'"' -f4)
if [ -z "$NAMED_QUERY_ID" ]; then
    handle_error "Failed to extract Named Query ID from response"
fi
echo "Named query created with ID: $NAMED_QUERY_ID"

# List named queries
echo "Listing named queries..."
LIST_QUERIES_RESULT=$(aws athena list-named-queries --region "$AWS_REGION" 2>&1)
if echo "$LIST_QUERIES_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to list named queries: $LIST_QUERIES_RESULT"
fi
echo "$LIST_QUERIES_RESULT"

# Get the named query details
echo "Getting named query details..."
GET_QUERY_RESULT=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --region "$AWS_REGION" 2>&1)
```

```

if echo "$GET_QUERY_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to get named query: $GET_QUERY_RESULT"
fi
echo "$GET_QUERY_RESULT"

# Execute the named query
echo "Executing the named query..."
QUERY_STRING=$(aws athena get-named-query --named-query-id "$NAMED_QUERY_ID" \
    --query "NamedQuery.QueryString" --output text --region "$AWS_REGION" 2>&1)
if echo "$QUERY_STRING" | grep -qi "error\|failed"; then
    handle_error "Failed to get query string: $QUERY_STRING"
fi

if [ -z "$QUERY_STRING" ] || [ "$QUERY_STRING" = "None" ]; then
    handle_error "Query string is empty"
fi

EXEC_RESULT=$(aws athena start-query-execution \
    --query-string "$QUERY_STRING" \
    --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
    --region "$AWS_REGION" 2>&1)

if echo "$EXEC_RESULT" | grep -qi "error\|failed"; then
    handle_error "Failed to execute named query: $EXEC_RESULT"
fi

QUERY_ID=$(echo "$EXEC_RESULT" | jq -r '.QueryExecutionId // empty' 2>/dev/null
|| echo "$EXEC_RESULT" | grep -o '"QueryExecutionId": "[^"]*"' | cut -d'"' -f4)
if [ -z "$QUERY_ID" ]; then
    handle_error "Failed to extract Query ID from named query execution response"
fi
echo "Named query execution ID: $QUERY_ID"

# Wait for named query to complete
echo "Waiting for named query execution to complete..."
ELAPSED=0
while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
    QUERY_STATUS=$(aws athena get-query-execution --query-execution-id
"$QUERY_ID" \
    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
    if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    fi
done

```

```

    elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" = "CANCELLED" ];
then
    handle_error "Named query execution failed with status: $QUERY_STATUS"
fi
    echo "Named query execution in progress, status: $QUERY_STATUS"
    sleep 2
    ((ELAPSED+=2))
done

if [ $ELAPSED -ge $WAIT_TIMEOUT ]; then
    handle_error "Named query execution timed out"
fi

# Summary of resources created
echo ""
echo "=====
echo "RESOURCES CREATED"
echo "=====
echo "- S3 Bucket: $S3_BUCKET"
echo "- Database: $DATABASE_NAME"
echo "- Table: $TABLE_NAME"
echo "- Named Query: $NAMED_QUERY_ID"
echo "- Query results saved to: query-results.csv"
echo "=====

# Auto-confirm cleanup
echo ""
echo "=====
echo "CLEANUP CONFIRMATION"
echo "=====
echo "Starting cleanup..."
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup..."

    # Delete named query
    echo "Deleting named query: $NAMED_QUERY_ID"
    DELETE_QUERY_RESULT=$(aws athena delete-named-query --named-query-id
"$NAMED_QUERY_ID" \
        --region "$AWS_REGION" 2>&1)
    if echo "$DELETE_QUERY_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete named query: $DELETE_QUERY_RESULT"
    else

```



```

        echo "Named query deleted successfully."
    fi

    # Drop table
    echo "Dropping table: $TABLE_NAME"
    DROP_TABLE_RESULT=$(aws athena start-query-execution \
        --query-string "DROP TABLE IF EXISTS $DATABASE_NAME.$TABLE_NAME" \
        --result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
        --region "$AWS_REGION" 2>&1)

    if echo "$DROP_TABLE_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to drop table: $DROP_TABLE_RESULT"
    else
        QUERY_ID=$(echo "$DROP_TABLE_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_TABLE_RESULT" | grep -o '"QueryExecutionId": "[^"]*'
| cut -d'"' -f4)
        if [ -n "$QUERY_ID" ]; then
            echo "Waiting for table deletion to complete..."

            ELAPSED=0
            while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
                QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                    --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
                if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                    echo "Table dropped successfully."
                    break
                elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
                    echo "Warning: Table deletion failed with status:
$QUERY_STATUS"
                    break
                fi
                echo "Table deletion in progress, status: $QUERY_STATUS"
                sleep 2
                ((ELAPSED+=2))
            done
        fi
    fi

    # Drop database
    echo "Dropping database: $DATABASE_NAME"
    DROP_DB_RESULT=$(aws athena start-query-execution \

```

```

--query-string "DROP DATABASE IF EXISTS $DATABASE_NAME" \
--result-configuration "OutputLocation=s3://$S3_BUCKET/output/" \
--region "$AWS_REGION" 2>&1)

if echo "$DROP_DB_RESULT" | grep -qi "error\|failed"; then
    echo "Warning: Failed to drop database: $DROP_DB_RESULT"
else
    QUERY_ID=$(echo "$DROP_DB_RESULT" | jq -r '.QueryExecutionId // empty'
2>/dev/null || echo "$DROP_DB_RESULT" | grep -o '"QueryExecutionId": "[^"]*' |
cut -d'"' -f4)
    if [ -n "$QUERY_ID" ]; then
        echo "Waiting for database deletion to complete..."

        ELAPSED=0
        while [ $ELAPSED -lt $WAIT_TIMEOUT ]; do
            QUERY_STATUS=$(aws athena get-query-execution --query-execution-
id "$QUERY_ID" \
                --query "QueryExecution.Status.State" --output text --region
"$AWS_REGION" 2>&1)
            if [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
                echo "Database dropped successfully."
                break
            elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
                echo "Warning: Database deletion failed with status:
$QUERY_STATUS"
                break
            fi
            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
fi

```

```
    echo "Deleting S3 bucket: $S3_BUCKET"
    DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
    if echo "$DELETE_BUCKET_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to delete S3 bucket: $DELETE_BUCKET_RESULT"
    else
        echo "S3 bucket deleted successfully."
    fi
else
    echo "Skipping S3 bucket deletion (shared resource)"
fi

# Security: Remove downloaded query results
if [ -f "./query-results.csv" ]; then
    if command -v shred &>/dev/null; then
        shred -vfz -n 3 "./query-results.csv" 2>/dev/null || rm -f "./query-
results.csv"
    else
        rm -f "./query-results.csv"
    fi
    echo "Query results file securely removed."
fi

echo "Cleanup completed."
fi

echo "Tutorial completed successfully!"
```

- Consulte detalhes da API nos tópicos a seguir na Referência de comandos da AWS CLI.
 - [Cp](#)
 - [CreateNamedQuery](#)
 - [DeleteNamedQuery](#)
 - [GetNamedQuery](#)
 - [GetQueryExecution](#)
 - [GetQueryResults](#)
 - [ListDatabases](#)
 - [ListNamedQueries](#)
 - [ListTableMetadata](#)
 - [Mb](#)

- [Rb](#)
- [Rm](#)
- [StartQueryExecution](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Trabalhar com os recursos de bloqueio de objetos do Amazon S3 usando um SDK da AWS

Os exemplos de código a seguir mostram como trabalhar com os recursos do Bloqueio de Objetos do S3.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Execute um cenário interativo demonstrando os recursos de bloqueio de objetos do 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;
    }
}
```

Uma classe de wrapper para funções do S3.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Configuration;

namespace S3ObjectLockScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, IConfiguration configuration)
    {
        _amazonS3 = amazonS3;
    }

    /// <summary>
    /// Create a new Amazon S3 bucket with object lock actions.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to create.</param>
    /// <param name="enableObjectLock">True to enable object lock on the
    bucket.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> CreateBucketWithObjectLock(string bucketName, bool
    enableObjectLock)
    {

```



```

        Console.WriteLine($"\\tCreating bucket {bucketName} with object lock
{enableObjectLock}.");
        try
        {
            var request = new PutBucketRequest
            {
                BucketName = bucketName,
                UseClientRegion = true,
                ObjectLockEnabledForBucket = enableObjectLock,
            };

            var response = await _amazonS3.PutBucketAsync(request);

            return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error creating bucket: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Enable object lock on an existing bucket.
    /// </summary>
    /// <param name="bucketName">The name of the bucket to modify.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> EnableObjectLockOnBucket(string bucketName)
    {
        try
        {
            // First, enable Versioning on the bucket.
            await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
            {
                BucketName = bucketName,
                VersioningConfig = new S3BucketVersioningConfig()
                {
                    EnableMfaDelete = false,
                    Status = VersionStatus.Enabled
                }
            });

            var request = new PutObjectLockConfigurationRequest()

```

```

        {
            BucketName = bucketName,
            ObjectLockConfiguration = new ObjectLockConfiguration()
            {
                ObjectLockEnabled = new ObjectLockEnabled("Enabled"),
            },
        };

        var response = await
        _amazonS3.PutObjectLockConfigurationAsync(request);
        Console.WriteLine($"{bucketName}\tAdded an object lock policy to bucket
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error modifying object lock: '{ex.Message}'");
        return false;
    }
}

/// <summary>
/// Set or modify a retention period on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="retention">The retention mode.</param>
/// <param name="retainUntilDate">The date retention expires.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectRetentionPeriod(string bucketName,
    string objectKey, ObjectLockRetentionMode retention, DateTime
retainUntilDate)
{
    try
    {
        var request = new PutObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            Retention = new ObjectLockRetention()
            {
                Mode = retention,
                RetainUntilDate = retainUntilDate
            }
        }
    }
}

```

```

    };

    var response = await _amazonS3.PutObjectRetentionAsync(request);
    Console.WriteLine($"\\tSet retention for {objectKey} in {bucketName}
until {retainUntilDate:d}.");
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tError modifying retention period:
'{ex.Message}'");
    return false;
}
}

/// <summary>
/// Set or modify a retention period on an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket to modify.</param>
/// <param name="retention">The retention mode.</param>
/// <param name="retainUntilDate">The date for retention until.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyBucketDefaultRetention(string bucketName, bool
enableObjectLock, ObjectLockRetentionMode retention, DateTime retainUntilDate)
{
    var enabledString = enableObjectLock ? "Enabled" : "Disabled";
    var timeDifference = retainUntilDate.Subtract(DateTime.Now);
    try
    {
        // First, enable Versioning on the bucket.
        await _amazonS3.PutBucketVersioningAsync(new
PutBucketVersioningRequest()
        {
            BucketName = bucketName,
            VersioningConfig = new S3BucketVersioningConfig()
            {
                EnableMfaDelete = false,
                Status = VersionStatus.Enabled
            }
        });

        var request = new PutObjectLockConfigurationRequest()
        {
            BucketName = bucketName,

```

```

        ObjectLockConfiguration = new ObjectLockConfiguration()
        {
            ObjectLockEnabled = new ObjectLockEnabled(enabledString),
            Rule = new ObjectLockRule()
            {
                DefaultRetention = new DefaultRetention()
                {
                    Mode = retention,
                    Days = timeDifference.Days // Can be specified in
days or years but not both.
                }
            }
        }
    };

    var response = await
_amazonS3.PutObjectLockConfigurationAsync(request);
    Console.WriteLine($"\\tAdded a default retention to bucket
{bucketName}.");
    return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"\\tError modifying object lock: '{ex.Message}'");
    return false;
}
}

/// <summary>
/// Get the retention period for an S3 object.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The object key.</param>
/// <returns>The object retention details.</returns>
public async Task<ObjectLockRetention> GetObjectRetention(string bucketName,
    string objectKey)
{
    try
    {
        var request = new GetObjectRetentionRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };
    }

```

```

        var response = await _amazonS3.GetObjectRetentionAsync(request);
        Console.WriteLine($"\\tObject retention for {objectKey} in
{bucketName}: " +
                        $"\\n\\t{response.Retention.Mode} until
{response.Retention.RetainUntilDate:d}.");
        return response.Retention;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to fetch object lock retention:
'{ex.Message}'");
        return new ObjectLockRetention();
    }
}

/// <summary>
/// Set or modify a legal hold on an object in an S3 bucket.
/// </summary>
/// <param name="bucketName">The bucket of the object.</param>
/// <param name="objectKey">The key of the object.</param>
/// <param name="holdStatus">The On or Off status for the legal hold.</param>
/// <returns>True if successful.</returns>
public async Task<bool> ModifyObjectLegalHold(string bucketName,
    string objectKey, ObjectLockLegalHoldStatus holdStatus)
{
    try
    {
        var request = new PutObjectLegalHoldRequest()
        {
            BucketName = bucketName,
            Key = objectKey,
            LegalHold = new ObjectLockLegalHold()
            {
                Status = holdStatus
            }
        };

        var response = await _amazonS3.PutObjectLegalHoldAsync(request);
        Console.WriteLine($"\\tModified legal hold for {objectKey} in
{bucketName}.");
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)

```

```

        {
            Console.WriteLine($"\\tError modifying legal hold: '{ex.Message}'");
            return false;
        }
    }

    /// <summary>
    /// Get the legal hold details for an S3 object.
    /// </summary>
    /// <param name="bucketName">The bucket of the object.</param>
    /// <param name="objectKey">The object key.</param>
    /// <returns>The object legal hold details.</returns>
    public async Task<ObjectLockLegalHold> GetObjectLegalHold(string bucketName,
        string objectKey)
    {
        try
        {
            var request = new GetObjectLegalHoldRequest()
            {
                BucketName = bucketName,
                Key = objectKey
            };

            var response = await _amazonS3.GetObjectLegalHoldAsync(request);
            Console.WriteLine($"\\tObject legal hold for {objectKey} in
{bucketName}: " +
                $"\\n\\tStatus: {response.LegalHold.Status}");
            return response.LegalHold;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tUnable to fetch legal hold: '{ex.Message}'");
            return new ObjectLockLegalHold();
        }
    }

    /// <summary>
    /// Get the object lock configuration details for an S3 bucket.
    /// </summary>
    /// <param name="bucketName">The bucket to get details.</param>
    /// <returns>The bucket's object lock configuration details.</returns>
    public async Task<ObjectLockConfiguration>
    GetBucketObjectLockConfiguration(string bucketName)
    {

```

```

        try
        {
            var request = new GetObjectLockConfigurationRequest()
            {
                BucketName = bucketName
            };

            var response = await
            _amazonS3.GetObjectLockConfigurationAsync(request);
            Console.WriteLine($"\\tBucket object lock config for {bucketName} in
{bucketName}: " +
                $"\\n\\tEnabled:
{response.ObjectLockConfiguration.ObjectLockEnabled}" +
                $"\\n\\tRule:
{response.ObjectLockConfiguration.Rule?.DefaultRetention}");

            return response.ObjectLockConfiguration;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tUnable to fetch object lock config:
'{ex.Message}'");
            return new ObjectLockConfiguration();
        }
    }

    /// <summary>
    /// Upload a file from the local computer to an Amazon S3 bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectName">The object to upload.</param>
    /// <param name="filePath">The path, including file name, of the object to
upload.</param>
    /// <returns>True if success.</returns>
    public async Task<bool> UploadFileAsync(string bucketName, string objectName,
string filePath)
    {
        var request = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = objectName,
            FilePath = filePath,
            ChecksumAlgorithm = ChecksumAlgorithm.SHA256
        };
    }

```

```

        var response = await _amazonS3.PutObjectAsync(request);
        if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
        {
            Console.WriteLine($"\\tSuccessfully uploaded {objectName} to
{bucketName}.");
            return true;
        }
        else
        {
            Console.WriteLine($"\\tCould not upload {objectName} to
{bucketName}.");
            return false;
        }
    }

    /// <summary>
    /// List bucket objects and versions.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <returns>The list of objects and versions.</returns>
    public async Task<ListVersionsResponse> ListBucketObjectsAndVersions(string
bucketName)
    {
        var request = new ListVersionsRequest()
        {
            BucketName = bucketName
        };

        var response = await _amazonS3.ListVersionsAsync(request);
        return response;
    }

    /// <summary>
    /// Delete an object from a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="hasRetention">True if the object has retention settings.</
param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey, bool hasRetention, string? versionId = null)

```



```

    {
        try
        {
            var request = new DeleteObjectRequest()
            {
                BucketName = bucketName,
                Key = objectKey,
                VersionId = versionId,
            };
            if (hasRetention)
            {
                // Set the BypassGovernanceRetention header
                // if the file has retention settings.
                request.BypassGovernanceRetention = true;
            }
            await _amazonS3.DeleteObjectAsync(request);
            Console.WriteLine(
                $"Deleted {objectKey} in {bucketName}.");
            return true;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Unable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteBucketByName(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"Delete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
    }

```

```
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
ex.Message);
            return false;
        }
    }
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Go

SDK para Go V2

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Execute um cenário interativo demonstrando os recursos de bloqueio de objetos do Amazon S3.

```
import (
    "context"
    "fmt"
    "log"
```

```
"strings"

"s3_object_lock/actions"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
    questioner demotools.IQuestioner
    resources  Resources
    s3Actions  *actions.S3Actions
    sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner
    demotools.IQuestioner) ObjectLockScenario {
    scenario := ObjectLockScenario{
        questioner: questioner,
        resources:  Resources{},
        s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
        sdkConfig:  sdkConfig,
    }
    scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
    scenario.resources.init(scenario.s3Actions, questioner)
    return scenario
}

type nameLocked struct {
    name    string
    locked  bool
}

var createInfo = []nameLocked{
    {"standard-bucket", false},
    {"lock-bucket", true},
    {"retention-bucket", false},
}
```

```
// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
    log.Println("Let's create some S3 buckets to use for this workflow.")
    success := false
    for !success {
        prefix := scenario.questioner.Ask(
            "This example creates three buckets. Enter a prefix to name your buckets\n(remember bucket names must be globally unique):")

        for _, info := range createInfo {
            log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
            bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx,
                fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
            if err != nil {
                switch err.(type) {
                    case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
                        log.Printf("Couldn't create bucket %s.\n", bucketName)
                    default:
                        panic(err)
                }
                break
            }
            scenario.resources.demoBuckets[info.name] = &DemoBucket{
                name:      bucketName,
                objectKeys: []string{},
            }
            log.Printf("Created bucket %s.\n", bucketName)
        }

        if len(scenario.resources.demoBuckets) < len(createInfo) {
            scenario.resources.deleteBuckets(ctx)
        } else {
            success = true
        }
    }

    log.Println("S3 buckets created.")
    log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
    log.Println("\nA bucket can be configured to use object locking.")
    scenario.questioner.Ask("Press Enter to continue.")
}
```

```
var err error
bucket := scenario.resources.demoBuckets["retention-bucket"]
err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
        log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
    default:
        panic(err)
    }
} else {
    log.Printf("Object locking enabled on bucket %s.", bucket.name)
}

log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a
// bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx
context.Context) {
    log.Println("\nA bucket can be configured to use object locking with a default
retention period.")

    bucket := scenario.resources.demoBuckets["retention-bucket"]
    retentionPeriod := scenario.questioner.AskInt("Enter the default retention
period in days: ")
    err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
types.ObjectLockRetentionModeGovernance)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("Couldn't configure a default retention period on bucket %s.\n",
bucket.name)
        default:
            panic(err)
        }
    } else {
        log.Printf("Default retention policy set on bucket %s with %d day retention
period.", bucket.name, retentionPeriod)
        bucket.retentionEnabled = true
    }
}
```

```

    log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
    log.Println("Uploading test objects to S3 buckets.")

    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        for i := 0; i < 2; i++ {
            key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
                fmt.Sprintf("example-%d", i),
                fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
            if err != nil {
                switch err.(type) {
                case *types.NoSuchBucket:
                    log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
                default:
                    panic(err)
                }
            } else {
                log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
                bucket.objectKeys = append(bucket.objectKeys, key)
            }
        }
    }

    scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
    log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test
// objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx
    context.Context) {
    log.Println("Now let's set object lock configurations on individual objects.")

    buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
        scenario.resources.demoBuckets["retention-bucket"]}
    for _, bucket := range buckets {
        for index, objKey := range bucket.objectKeys {
            switch index {
            case 0:

```

```

    if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold
to %s in %s (y/n)? ", objKey, bucket.name), "y") {
        err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
types.ObjectLockLegalHoldStatusOn)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Couldn't set legal hold on %s.\n", objKey)
            default:
                panic(err)
            }
        } else {
            log.Printf("Legal hold set on %s.\n", objKey)
        }
    }
    case 1:
        q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to
%s in %s?\n"+
            "Reminder: Only a user with the s3:BypassGovernanceRetention permission is
able to delete this object\n"+
            "or its bucket until the retention period has expired. (y/n) ", objKey,
bucket.name)
        if scenario.questioner.AskBool(q, "y") {
            err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
types.ObjectLockRetentionModeGovernance, 1)
            if err != nil {
                switch err.(type) {
                case *types.NoSuchKey:
                    log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
bucket.name)
                default:
                    panic(err)
                }
            } else {
                log.Printf("Retention period set to 1 for %s.", objKey)
                bucket.retentionEnabled = true
            }
        }
    }
}
log.Println(strings.Repeat("-", 88))
}

```

```
const (
    ListAll = iota
    DeleteObject
    DeleteRetentionObject
    OverwriteObject
    ViewRetention
    ViewLegalHold
    Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
// object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
    log.Println("Now you can interact with the objects to explore the object lock
    configurations.")
    interactiveChoices := []string{
        "List all objects and buckets.",
        "Attempt to delete an object.",
        "Attempt to delete an object with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the retention settings for an object.",
        "View the legal hold settings for an object.",
        "Finish the workflow."}

    choice := ListAll
    for choice != Finish {
        objList := scenario.GetAllObjects(ctx)
        objChoices := scenario.makeObjectChoiceList(objList)
        choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
        interactiveChoices)
        switch choice {
        case ListAll:
            log.Println("The current objects in the example buckets are:")
            for _, objChoice := range objChoices {
                log.Println("\t", objChoice)
            }
        case DeleteObject, DeleteRetentionObject:
            objChoice := scenario.questioner.AskChoice("Enter the number of the object to
            delete:\n", objChoices)
            obj := objList[objChoice]
            deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
            obj.versionId, choice == DeleteRetentionObject)
            if err != nil {
                switch err.(type) {
```



```
    case *types.NoSuchKey:
        log.Println("Nothing to delete.")
    default:
        panic(err)
    }
} else if deleted {
    log.Printf("Object %s deleted.\n", obj.key)
}
case OverwriteObject:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
    obj := objList[objChoice]
    _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Println("Couldn't upload to nonexistent bucket.")
        default:
            panic(err)
        }
    } else {
        log.Printf("Uploaded new content to object %s.\n", obj.key)
    }
case ViewRetention:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
    obj := objList[objChoice]
    retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket,
obj.key)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Printf("Can't get retention configuration for %s.\n", obj.key)
        default:
            panic(err)
        }
    } else if retention != nil {
        log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
    } else {
        log.Printf("Object %s does not have object retention configured.\n", obj.key)
    }
case ViewLegalHold:
```

```

    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
    obj := objList[objChoice]
    legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket,
obj.key, obj.versionId)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
        default:
            panic(err)
        }
    } else if legalHold != nil {
        log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
    } else {
        log.Printf("Object %s does not have legal hold configured.", obj.key)
    }
case Finish:
    log.Println("Let's clean up.")
}
log.Println(strings.Repeat("-", 88))
}
}

type BucketKeyVersionId struct {
    bucket    string
    key       string
    versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns
them in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
[]BucketKeyVersionId {
    var objectList []BucketKeyVersionId
    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Printf("Couldn't get object versions for %s.\n", bucket.name)
            default:
                panic(err)
            }
        }
    }
}

```

```

    }
    } else {
        for _, version := range versions {
            objectList = append(objectList,
                BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
*version.VersionId})
        }
    }
}
return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
// are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
[]BucketKeyVersionId) []string {
    choices := make([]string, len(bucketObjects))
    for i := 0; i < len(bucketObjects); i++ {
        choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
            bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
    }
    return choices
}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := scenario.questioner.(*demotools.MockQuestioner)
            if isMock || scenario.questioner.AskBool("Do you want to see the full error
message (y/n)?", "y") {
                log.Println(r)
            }
            scenario.resources.Cleanup(ctx)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
    log.Println(strings.Repeat("-", 88))

    scenario.CreateBuckets(ctx)

```

```
scenario.EnableLockOnBucket(ctx)
scenario.SetDefaultRetentionPolicy(ctx)
scenario.UploadTestObjects(ctx)
scenario.SetObjectLockConfigurations(ctx)
scenario.InteractWithObjects(ctx)

scenario.resources.Cleanup(ctx)

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

Defina um struct que envolva as ações do S3 usadas neste exemplo.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking
// enabled
```

```
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    }

    if enableObjectLock {
        input.ObjectLockEnabledForBucket = aws.Bool(true)
    }

    _, err := actor.S3Client.CreateBucket(ctx, input)
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", bucket)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", bucket)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
        }
    }

    return bucket, err
}

// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
    var status *types.ObjectLockLegalHoldStatus
    input := &s3.GetObjectLegalHoldInput{
        Bucket:    aws.String(bucket),
```

```

    Key:      aws.String(key),
    VersionId: aws.String(versionId),
}

output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
if err != nil {
    var noSuchKeyErr *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noSuchKeyErr) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noSuchKeyErr
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            log.Printf("Object %s does not have an object lock configuration.\n", key)
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    }
} else {
    status = &output.LegalHold.Status
}

return status, err
}

// GetObjectLockConfiguration retrieves the object lock configuration for an S3
// bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
    string) (*types.ObjectLockConfiguration, error) {
    var lockConfig *types.ObjectLockConfiguration
    input := &s3.GetObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
    }

    output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noBucket) {

```

```
    log.Printf("Bucket %s does not exist.\n", bucket)
    err = noBucket
} else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
"ObjectLockConfigurationNotFoundError" {
    log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
    err = nil
}
} else {
    lockConfig = output.ObjectLockConfiguration
}

return lockConfig, err
}

// GetObjectRetention retrieves the object retention configuration for an S3
object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
string) (*types.ObjectLockRetention, error) {
    var retention *types.ObjectLockRetention
    input := &s3.GetObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }

    output, err := actor.S3Client.GetObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "NoSuchObjectLockConfiguration":
                err = nil
            case "InvalidRequest":
                log.Printf("Bucket %s does not have locking enabled.", bucket)
                err = nil
            }
        }
    } else {
        retention = output.Retention
    }
}
```

```
}

return retention, err
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }

    _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}

// ModifyDefaultBucketRetention modifies the default retention period of an
existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
```



```

ObjectLockConfiguration: &types.ObjectLockConfiguration{
    ObjectLockEnabled: lockMode,
    Rule: &types.ObjectLockRule{
        DefaultRetention: &types.DefaultRetention{
            Days: aws.Int32(retentionPeriod),
            Mode: retentionMode,
        },
    },
},
},
}
_, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
}

return err
}

// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
string) error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
            MFADelete: types.MFADeleteDisabled,
            Status:    types.BucketVersioningStatusEnabled,
        },
    }
    _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
        return err
    }
}

```

```
input := &s3.PutObjectLockConfigurationInput{
    Bucket: aws.String(bucket),
    ObjectLockConfiguration: &types.ObjectLockConfiguration{
        ObjectLockEnabled: types.ObjectLockEnabledEnabled,
    },
}
_, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
}

return err
}

// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
error {
input := &s3.PutObjectRetentionInput{
    Bucket: aws.String(bucket),
    Key:    aws.String(key),
    Retention: &types.ObjectLockRetention{
        Mode:          retentionMode,
        RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
    },
    BypassGovernanceRetention: aws.Bool(true),
}

_, err := actor.S3Client.PutObjectRetention(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    }
}
}
```

```
    return err
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
string, contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:      aws.String(bucket),
        Key:         aws.String(key),
        Body:        bytes.NewReader([]byte(contents)),
        ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
    }
    output, err := actor.S3Manager.Upload(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    } else {
        err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
            Bucket: aws.String(bucket),
            Key:    aws.String(key),
        }, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
bucket)
        } else {
            outKey = *output.Key
        }
    }
    return outKey, err
}

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
```

```

var versions []types.ObjectVersion
input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
for versionPaginator.HasMorePages() {
    output, err = versionPaginator.NextPage(ctx)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
        break
    } else {
        versions = append(versions, output.Versions...)
    }
}
return versions, err
}

```

```

// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {

```

```

    case "AccessDenied":
        log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
        err = nil
    case "InvalidArgument":
        if bypassGovernance {
            log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
            err = nil
        }
    }
} else {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
        time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
\n", key, bucket)
    } else {
        deleted = true
    }
}
return deleted, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &types.Delete{
            Objects: objects,
            Quiet:   aws.Bool(true),
        },
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
}

```

```

delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
if err != nil || len(delOut.Errors) > 0 {
    log.Printf("Error deleting objects from bucket %s.\n", bucket)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    } else if len(delOut.Errors) > 0 {
        for _, outErr := range delOut.Errors {
            log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
        }
        err = fmt.Errorf("%s", *delOut.Errors[0].Message)
    }
} else {
    for _, delObjs := range delOut.Deleted {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                *delObjs.Key)
        } else {
            log.Printf("Deleted %s.\n", *delObjs.Key)
        }
    }
}
return err
}

```

Limpe 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)
        }
    }
    delObjects := make([]types.ObjectIdentifier, len(versions))
    for i, version := range versions {
        if lockConfig != nil && lockConfig.ObjectLockEnabled ==
            types.ObjectLockEnabledEnabled {
            status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
                *version.Key, *version.VersionId)
            if err != nil {
                switch err.(type) {
                case *types.NoSuchKey:
                    log.Printf("Couldn't determine legal hold status for %s in %s.\n",
                        *version.Key, bucket.name)
                default:
                    panic(err)
                }
            } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
                err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
                    *version.VersionId, types.ObjectLockLegalHoldStatusOff)
                if err != nil {
                    switch err.(type) {
                    case *types.NoSuchKey:
                        log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
                            bucket.name)
                    default:
                        panic(err)
                    }
                }
            }
        }
    }
}

```

```
    del0objects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.DeleteObjects(ctx, bucket.name, del0objects,
bucket.retentionEnabled)
if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
        log.Println("Nothing to delete.")
    default:
        panic(err)
    }
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Go.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Execute um cenário interativo demonstrando os recursos de bloqueio de objetos do 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;
}
```

Uma classe de wrapper para funções do S3.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
```

```
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
    software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {

    private static S3Client getClient() {
        return S3Client.builder()
            .region(Region.US_EAST_1)
            .build();
    }

    // Set or modify a retention period on an object in an S3 bucket.
    public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
    {
        // Calculate the instant one day from now.
        Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

        // Convert the Instant to a ZonedDateTime object with a specific time
        zone.
        ZonedDateTime zonedDateTime =
        futureInstant.atZone(ZoneId.systemDefault());
```

```
// Define a formatter for human-readable output.
DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

// Format the ZonedDateTime object to a human-readable date string.
String humanReadableDate = formatter.format(zonedDateTime);

// Print the formatted date string.
System.out.println("Formatted Date: " + humanReadableDate);
ObjectLockRetention retention = ObjectLockRetention.builder()
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .retainUntilDate(futureInstant)
    .build();

PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
    .bucket(bucketName)
    .key(objectKey)
    .retention(retention)
    .build();

getClient().putObjectRetention(retentionRequest);
System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}

// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();
    } catch (S3Exception ex) {
```

```
        System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
    }

    return null;
}

// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}

public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
    AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
    return bucketNames.stream()
        .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
        .map(version -> {
            S3InfoObject s3InfoObject = new S3InfoObject();
            s3InfoObject.setBucketName(bucketName);
            s3InfoObject.setVersion(version.versionId());
            s3InfoObject.setKeyName(version.key());
            return s3InfoObject;
        }))
        .peek(s3InfoObject -> {
            int i = counter.incrementAndGet(); // Increment and get the
updated value.
            if (interactive) {
                System.out.println(i + ": " + s3InfoObject.getKeyName());
            }
        })
}
```

```
        System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
        System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
    }
})
.collect(Collectors.toList());
}

public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
    ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
        .bucket(bucketName)
        .build();

    return getClient().listObjectVersions(versionsRequest);
}

// Set or modify a retention period on an S3 bucket.
public void modifyBucketDefaultRetention(String bucketName) {
    VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
        .mfaDelete(MFADelete.DISABLED)
        .status(BucketVersioningStatus.ENABLED)
        .build();

    PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
        .bucket(bucketName)
        .versioningConfiguration(versioningConfiguration)
        .build();

    getClient().putBucketVersioning(versioningRequest);
    DefaultRetention rention = DefaultRetention.builder()
        .days(1)
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .build();

    ObjectLockRule lockRule = ObjectLockRule.builder()
        .defaultRetention(rention)
        .build();
}
```



```
        ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
        .objectLockEnabled(ObjectLockEnabled.ENABLED)
        .rule(lockRule)
        .build();

        PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .objectLockConfiguration(objectLockConfiguration)
        .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
        System.out.println("Added a default retention to bucket "+bucketName
+ ".");
    }

    // Enable object lock on an existing bucket.
    public void enableObjectLockOnBucket(String bucketName) {
        try {
            VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
                .status(BucketVersioningStatus.ENABLED)
                .build();

            PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
                .bucket(bucketName)
                .versioningConfiguration(versioningConfiguration)
                .build();

            // Enable versioning on the bucket.
            getClient().putBucketVersioning(putBucketVersioningRequest);
            PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
                .bucket(bucketName)
                .objectLockConfiguration(ObjectLockConfiguration.builder()
                    .objectLockEnabled(ObjectLockEnabled.ENABLED)
                    .build())
                .build();

            getClient().putObjectLockConfiguration(request);
```

```
        System.out.println("Successfully enabled object lock on  
"+bucketName);  
  
        } catch (S3Exception ex) {  
            System.out.println("Error modifying object lock: '" + ex.getMessage()  
+ "'");  
        }  
    }  
  
    public void uploadFile(String bucketName, String objectName, String filePath)  
{  
        Path file = Paths.get(filePath);  
        PutObjectRequest request = PutObjectRequest.builder()  
            .bucket(bucketName)  
            .key(objectName)  
            .checksumAlgorithm(ChecksumAlgorithm.SHA256)  
            .build();  
  
        PutObjectResponse response = getClient().putObject(request, file);  
        if (response != null) {  
            System.out.println("\tSuccessfully uploaded " + objectName + " to " +  
bucketName + ".");  
        } else {  
            System.out.println("\tCould not upload " + objectName + " to " +  
bucketName + ".");  
        }  
    }  
  
    // Set or modify a legal hold on an object in an S3 bucket.  
    public void modifyObjectLegalHold(String bucketName, String objectKey,  
boolean legalHoldOn) {  
        ObjectLockLegalHold legalHold ;  
        if (legalHoldOn) {  
            legalHold = ObjectLockLegalHold.builder()  
                .status(ObjectLockLegalHoldStatus.ON)  
                .build();  
        } else {  
            legalHold = ObjectLockLegalHold.builder()  
                .status(ObjectLockLegalHoldStatus.OFF)  
                .build();  
        }  
  
        PutObjectLegalHoldRequest legalHoldRequest =  
PutObjectLegalHoldRequest.builder()
```

```
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

        getClient().putObjectLegalHold(legalHoldRequest) ;
        System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
    }

    // Delete an object from a specific bucket.
    public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
        try {
            DeleteObjectRequest objectRequest;
            if (hasRetention) {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .bypassGovernanceRetention(true)
                    .build();
            } else {
                objectRequest = DeleteObjectRequest.builder()
                    .bucket(bucketName)
                    .key(objectKey)
                    .versionId(versionId)
                    .build();
            }

            getClient().deleteObject(objectRequest) ;
            System.out.println("The object was successfully deleted");

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

    // Get the retention period for an S3 object.
    public ObjectLockRetention getObjectRetention(String bucketName, String key){
        try {
            GetObjectRetentionRequest retentionRequest =
            GetObjectRetentionRequest.builder()
                .bucket(bucketName)
```

```
        .key(key)
        .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("\tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

public void deleteBucketByName(String bucketName) {
    try {
        DeleteBucketRequest request = DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        getClient().deleteBucket(request);
        System.out.println(bucketName +" was deleted.");

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

// Get the object lock configuration details for an S3 bucket.
public void getBucketObjectLockConfiguration(String bucketName) {
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =
GetObjectLockConfigurationRequest.builder()
        .bucket(bucketName)
        .build();

    GetObjectLockConfigurationResponse response =
getClient().getObjectLockConfiguration(objectLockConfigurationRequest);
    System.out.println("Bucket object lock config for "+bucketName +": ");
    System.out.println("\tEnabled:
"+response.objectLockConfiguration().objectLockEnabled());
    System.out.println("\tRule: "+
response.objectLockConfiguration().rule().defaultRetention());
}
```

```
}  
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Ponto de entrada do cenário (index.js). Isso orquestra todas as etapas. Acesse o GitHub para ver os detalhes de implementação de Scenario, ScenarioInput, ScenarioOutput e ScenarioAction.

```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";  
import {  
  exitOnFalse,  
  loadState,  
  saveState,  
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";  
  
import { welcome, welcomeContinue } from "../welcome.steps.js";  
import {  
  confirmCreateBuckets,  
  confirmPopulateBuckets,  
  confirmSetLegalHoldFileEnabled,
```

```
confirmSetLegalHoldFileRetention,  
confirmSetRetentionPeriodFileEnabled,  
confirmSetRetentionPeriodFileRetention,  
confirmUpdateLockPolicy,  
confirmUpdateRetention,  
createBuckets,  
createBucketsAction,  
getBucketPrefix,  
populateBuckets,  
populateBucketsAction,  
setLegalHoldFileEnabledAction,  
setLegalHoldFileRetentionAction,  
setRetentionPeriodFileEnabledAction,  
setRetentionPeriodFileRetentionAction,  
updateLockPolicy,  
updateLockPolicyAction,  
updateRetention,  
updateRetentionAction,  
} from "./setup.steps.js";  
  
/**  
 * @param {Scenarios} scenarios  
 * @param {Record<string, any>} initialState  
 */  
export const getWorkflowStages = (scenarios, initialState = {}) => {  
  const client = new S3Client({});  
  
  return {  
    deploy: new scenarios.Scenario(  
      "S3 Object Locking - Deploy",  
      [  
        welcome(scenarios),  
        welcomeContinue(scenarios),  
        exitOnFalse(scenarios, "welcomeContinue"),  
        getBucketPrefix(scenarios),  
        createBuckets(scenarios),  
        confirmCreateBuckets(scenarios),  
        exitOnFalse(scenarios, "confirmCreateBuckets"),  
        createBucketsAction(scenarios, client),  
        updateRetention(scenarios),  
        confirmUpdateRetention(scenarios),  
        exitOnFalse(scenarios, "confirmUpdateRetention"),  
        updateRetentionAction(scenarios, client),  
        populateBuckets(scenarios),
```

```

        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        updateLockPolicy(scenarios),
        confirmUpdateLockPolicy(scenarios),
        exitOnFalse(scenarios, "confirmUpdateLockPolicy"),
        updateLockPolicyAction(scenarios, client),
        confirmSetLegalHoldFileEnabled(scenarios),
        setLegalHoldFileEnabledAction(scenarios, client),
        confirmSetRetentionPeriodFileEnabled(scenarios),
        setRetentionPeriodFileEnabledAction(scenarios, client),
        confirmSetLegalHoldFileRetention(scenarios),
        setLegalHoldFileRetentionAction(scenarios, client),
        confirmSetRetentionPeriodFileRetention(scenarios),
        setRetentionPeriodFileRetentionAction(scenarios, client),
        saveState,
    ],
    initialState,
),
demo: new scenarios.Scenario(
    "S3 Object Locking - Demo",
    [loadState, replAction(scenarios, client)],
    initialState,
),
clean: new scenarios.Scenario(
    "S3 Object Locking - Destroy",
    [
        loadState,
        confirmCleanup(scenarios),
        exitOnFalse(scenarios, "confirmCleanup"),
        cleanupAction(scenarios, client),
    ],
    initialState,
),
};
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {

```

```

const objectLockingScenarios = getWorkflowStages(Scenarios);
Scenarios.parseScenarioArgs(objectLockingScenarios, {
  name: "Amazon S3 object locking workflow",
  description:
    "Work with Amazon Simple Storage Service (Amazon S3) object locking
features.",
  synopsis:
    "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
  });
}

```

Envie mensagens de boas-vindas para o console (welcome.steps.js).

```

/**
 * @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
 */

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "Welcome to the Amazon Simple Storage Service (S3) Object Locking Feature
Scenario. For this workflow, we will use the AWS SDK for JavaScript to create
several S3 buckets and files to demonstrate working with S3 locking features.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };

```


Implante buckets, objetos e configurações de arquivos (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,  
};
```

Visualize e exclua arquivos nos 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 scenarios.ScenarioAction(  
    "replAction",  
    async (state) => {  
      const files = await getAllFiles(client, [  
        state.noLockBucketName,  
        state.lockEnabledBucketName,  
        state.retentionBucketName,  
      ]);  
  
      const fileInput = new scenarios.ScenarioInput(  
        "selectedFile",  
        "Select a file:",  
        {  
          type: "select",  
          choices: files.map((file, index) => ({  
            name: `${index + 1}: ${file.bucket}: ${file.key} (version: ${  
              file.version  
            })`,  
            value: index,  
          })),  
        },  
      );  
  
      const { replChoice } = state;  
  
      switch (replChoice) {  
        case choices.LIST_ALL_FILES: {  
          const files = await getAllFiles(client, [  
            state.noLockBucketName,  
            state.lockEnabledBucketName,  
            state.retentionBucketName,  
          ]);  
          state.replOutput = files
```

```
        .map(
          (file) =>
            `${file.bucket}: ${file.key} (version: ${file.version})`,
        )
        .join("\n");
    break;
}

case choices.DELETE_FILE: {
  /** @type {number} */
  const fileToDelete = await fileInput.handle(state);
  const selectedFile = files[fileToDelete];
  try {
    await client.send(
      new DeleteObjectCommand({
        Bucket: selectedFile.bucket,
        Key: selectedFile.key,
        VersionId: selectedFile.version,
      }),
    );
    state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
  } catch (err) {
    state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
  }
  break;
}

case choices.DELETE_FILE_WITH_RETENTION: {
  /** @type {number} */
  const fileToDelete = await fileInput.handle(state);
  const selectedFile = files[fileToDelete];
  try {
    await client.send(
      new DeleteObjectCommand({
        Bucket: selectedFile.bucket,
        Key: selectedFile.key,
        VersionId: selectedFile.version,
        BypassGovernanceRetention: true,
      }),
    );
    state.replOutput = `Deleted ${selectedFile.key} in
${selectedFile.bucket}.`;
  } catch (err) {
```

```
        state.replOutput = `Unable to delete object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
    }
    break;
}
case choices.OVERWRITE_FILE: {
    /** @type {number} */
    const fileToOverwrite = await fileInput.handle(state);
    const selectedFile = files[fileToOverwrite];
    try {
        await client.send(
            new PutObjectCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                Body: "New content",
                ChecksumAlgorithm: ChecksumAlgorithm.SHA256,
            }),
        );
        state.replOutput = `Overwrote ${selectedFile.key} in
${selectedFile.bucket}.`;
    } catch (err) {
        state.replOutput = `Unable to overwrite object ${selectedFile.key} in
bucket ${selectedFile.bucket}: ${err.message}`;
    }
    break;
}
case choices.VIEW_RETENTION_SETTINGS: {
    /** @type {number} */
    const fileToView = await fileInput.handle(state);
    const selectedFile = files[fileToView];
    try {
        const retention = await client.send(
            new GetObjectRetentionCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
            }),
        );
        const bucketConfig = await client.send(
            new GetObjectLockConfigurationCommand({
                Bucket: selectedFile.bucket,
            }),
        );
    }
}
```



```

        state.replOutput = `Object retention for ${selectedFile.key}
in ${selectedFile.bucket}: ${retention.Retention?.Mode} until
${retention.Retention?.RetainUntilDate?.toISOString()}.
Bucket object lock config for ${selectedFile.bucket} in ${selectedFile.bucket}:
Enabled: ${bucketConfig.ObjectLockConfiguration?.ObjectLockEnabled}
Rule:
${JSON.stringify(bucketConfig.ObjectLockConfiguration?.Rule?.DefaultRetention)}`;
    } catch (err) {
        state.replOutput = `Unable to fetch object lock retention:
'${err.message}'`;
    }
    break;
}
case choices.VIEW_LEGAL_HOLD_SETTINGS: {
    /** @type {number} */
    const fileToView = await fileInput.handle(state);
    const selectedFile = files[fileToView];
    try {
        const legalHold = await client.send(
            new GetObjectLegalHoldCommand({
                Bucket: selectedFile.bucket,
                Key: selectedFile.key,
                VersionId: selectedFile.version,
            }),
        );
        state.replOutput = `Object legal hold for ${selectedFile.key} in
${selectedFile.bucket}: Status: ${legalHold.LegalHold?.Status}`;
    } catch (err) {
        state.replOutput = `Unable to fetch legal hold: '${err.message}'`;
    }
    break;
}
default:
    throw new Error(`Invalid replChoice: ${replChoice}`);
}
},
{
    whileConfig: {
        whileFn: ({ replChoice }) => replChoice !== choices.EXIT,
        input: replInput(scenarios),
        output: new scenarios.ScenarioOutput(
            "REPL output",
            (state) => state.replOutput,
            { preformatted: true },
        ),
    },
}

```

```

    },
  },
},
);

export { replInput, replAction, choices };

```

Destrúa todos os recursos criados (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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Fazer solicitações condicionais do Amazon S3 usando um AWS SDK

Os exemplos de código a seguir mostram como adicionar pré-condições a solicitações do Amazon S3.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Execute um cenário interativo que demonstre os recursos de solicitação condicional do 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;
}
}
```

Uma classe de wrapper para funções do S3.

```
using System.Net;
using Amazon.S3;
using Amazon.S3.Model;
using Microsoft.Extensions.Logging;

namespace S3ConditionalRequestsScenario;

/// <summary>
/// Encapsulate the Amazon S3 operations.
/// </summary>
public class S3ActionsWrapper
{
    private readonly IAmazonS3 _amazonS3;
    private readonly ILogger<S3ActionsWrapper> _logger;

    /// <summary>
    /// Constructor for the S3ActionsWrapper.
    /// </summary>
    /// <param name="amazonS3">The injected S3 client.</param>
    /// <param name="logger">The class logger.</param>
    public S3ActionsWrapper(IAmazonS3 amazonS3, ILogger<S3ActionsWrapper> logger)
    {
        _amazonS3 = amazonS3;
        _logger = logger;
    }

    /// <summary>
    /// Retrieves an object from Amazon S3 with a conditional request.
    /// </summary>
    /// <param name="objectKey">The key of the object to retrieve.</param>
    /// <param name="sourceBucket">The source bucket of the object.</param>
    /// <param name="conditionType">The type of condition: 'IfMatch',
    'IfNoneMatch', 'IfModifiedSince', 'IfUnmodifiedSince'.</param>
```

```
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional read is successful, False otherwise.</
returns>
public async Task<bool> GetObjectConditional(string objectKey, string
sourceBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)
{
    try
    {
        var getObjectRequest = new GetObjectRequest
        {
            BucketName = sourceBucket,
            Key = objectKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                getObjectRequest.EtagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                getObjectRequest.EtagToNotMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfModifiedSince:
                getObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            case S3ConditionType.IfUnmodifiedSince:
                getObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            default:
                throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
        }

        var response = await _amazonS3.GetObjectAsync(getObjectRequest);
        var sampleBytes = new byte[20];
        await response.ResponseStream.ReadAsync(sampleBytes, 0, 20);
    }
}
```



```

        _logger.LogInformation($"Conditional read
        successful. Here are the first 20 bytes of the object:
        \n{System.Text.Encoding.UTF8.GetString(sampleBytes)}");
        return true;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional read failed: Precondition failed");
        }
        else if (e.ErrorCode == "NotModified")
        {
            _logger.LogError("Conditional read failed: Object not modified");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

/// <summary>
/// Uploads an object to Amazon S3 with a conditional request. Prevents
/// overwrite using an IfNoneMatch condition for the object key.
/// </summary>
/// <param name="objectKey">The key of the object to upload.</param>
/// <param name="bucket">The source bucket of the object.</param>
/// <param name="content">The content to upload as a string.</param>
/// <returns>The ETag if the conditional write is successful, empty
/// otherwise.</returns>
public async Task<string> PutObjectConditional(string objectKey, string
bucket, string content)
{
    try
    {
        var putObjectRequest = new PutObjectRequest
        {
            BucketName = bucket,
            Key = objectKey,
            ContentBody = content,
            IfNoneMatch = "*"
        }
    }
}

```

```

    };

    var putResult = await _amazonS3.PutObjectAsync(putObjectRequest);
    _logger.LogInformation($"Conditional write successful for key
{objectKey} in bucket {bucket}.");
    return putResult.ETag;
}
catch (AmazonS3Exception e)
{
    if (e.ErrorCode == "PreconditionFailed")
    {
        _logger.LogError("Conditional write failed: Precondition
failed");
    }
    else
    {
        _logger.LogError($"Unexpected error: {e.ErrorCode}");
        throw;
    }
    return string.Empty;
}
}

/// <summary>
/// Copies an object from one Amazon S3 bucket to another with a conditional
request.
/// </summary>
/// <param name="sourceKey">The key of the source object to copy.</param>
/// <param name="destKey">The key of the destination object.</param>
/// <param name="sourceBucket">The source bucket of the object.</param>
/// <param name="destBucket">The destination bucket of the object.</param>
/// <param name="conditionType">The type of condition to apply, e.g.
'CopySourceIfMatch', 'CopySourceIfNoneMatch', 'CopySourceIfModifiedSince',
'CopySourceIfUnmodifiedSince'.</param>
/// <param name="conditionDateValue">The value to use for the condition for
dates.</param>
/// <param name="etagConditionalValue">The value to use for the condition for
etags.</param>
/// <returns>True if the conditional copy is successful, False otherwise.</
returns>
public async Task<bool> CopyObjectConditional(string sourceKey, string
destKey, string sourceBucket, string destBucket,
    S3ConditionType conditionType, DateTime? conditionDateValue = null,
string? etagConditionalValue = null)

```

```
{
    try
    {
        var copyObjectRequest = new CopyObjectRequest
        {
            DestinationBucket = destBucket,
            DestinationKey = destKey,
            SourceBucket = sourceBucket,
            SourceKey = sourceKey
        };

        switch (conditionType)
        {
            case S3ConditionType.IfMatch:
                copyObjectRequest.ETagToMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfNoneMatch:
                copyObjectRequest.ETagToNotMatch = etagConditionalValue;
                break;
            case S3ConditionType.IfModifiedSince:
                copyObjectRequest.ModifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            case S3ConditionType.IfUnmodifiedSince:
                copyObjectRequest.UnmodifiedSinceDateUtc =
conditionDateValue.GetValueOrDefault();
                break;
            default:
                throw new ArgumentOutOfRangeException(nameof(conditionType),
conditionType, null);
        }

        await _amazonS3.CopyObjectAsync(copyObjectRequest);
        _logger.LogInformation($"Conditional copy successful for key
{destKey} in bucket {destBucket}.");
        return true;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "PreconditionFailed")
        {
            _logger.LogError("Conditional copy failed: Precondition failed");
        }
        else if (e.ErrorCode == "304")
    }
```

```
        {
            _logger.LogError("Conditional copy failed: Object not modified");
        }
        else
        {
            _logger.LogError($"Unexpected error: {e.ErrorCode}");
            throw;
        }
        return false;
    }
}

/// <summary>
/// Create a new Amazon S3 bucket with a specified name and check that the
bucket is ready.
/// </summary>
/// <param name="bucketName">The name of the bucket to create.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CreateBucketWithName(string bucketName)
{
    Console.WriteLine($"\\tCreating bucket {bucketName}.");
    try
    {
        var request = new PutBucketRequest
        {
            BucketName = bucketName,
            UseClientRegion = true
        };

        await _amazonS3.PutBucketAsync(request);
        var bucketReady = false;
        var retries = 5;
        while (!bucketReady && retries > 0)
        {
            Thread.Sleep(5000);
            bucketReady = await
Amazon.S3.Util.AmazonS3Util.DoesS3BucketExistV2Async(_amazonS3, bucketName);
            retries--;
        }

        return bucketReady;
    }
    catch (BucketAlreadyExistsException ex)
    {

```

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

/// <summary>
/// Cleans up objects and deletes the bucket by name.
/// </summary>
/// <param name="bucketName">The name of the bucket.</param>
/// <returns>Async task.</returns>
public async Task CleanupBucketByName(string bucketName)
{
    try
    {
        var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
        foreach (var obj in listObjectsResponse.S3Objects)
        {
            await _amazonS3.DeleteObjectAsync(new DeleteObjectRequest
{ BucketName = bucketName, Key = obj.Key });
        }
        await _amazonS3.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        Console.WriteLine($"Cleaned up bucket: {bucketName}.");
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "NoSuchBucket")
        {
            Console.WriteLine($"Bucket {bucketName} does not exist, skipping
cleanup.");
        }
        else
        {
            Console.WriteLine($"Error deleting bucket: {e.ErrorCode}");
            throw;
        }
    }
}
```

```
/// <summary>
/// List the contents of the bucket with their ETag.
/// </summary>
/// <param name="bucketName">The name of the bucket.</param>
/// <returns>Async task.</returns>
public async Task<List<S3Object>> ListBucketContentsByName(string bucketName)
{
    var results = new List<S3Object>();
    try
    {
        Console.WriteLine($"\\t Items in bucket {bucketName}");
        var listObjectsResponse = await _amazonS3.ListObjectsV2Async(new
ListObjectsV2Request { BucketName = bucketName });
        if (listObjectsResponse.S3Objects.Count == 0)
        {
            Console.WriteLine("\\t\\tNo objects found.");
        }
        else
        {
            foreach (var obj in listObjectsResponse.S3Objects)
            {
                Console.WriteLine($"\\t\\t object: {obj.Key} ETag {obj.ETag}");
            }
        }
        results = listObjectsResponse.S3Objects;
    }
    catch (AmazonS3Exception e)
    {
        if (e.ErrorCode == "NoSuchBucket")
        {
            _logger.LogError($"Bucket {bucketName} does not exist.");
        }
        else
        {
            _logger.LogError($"Error listing bucket and objects:
{e.ErrorCode}");
            throw;
        }
    }

    return results;
}
```

```
/// <summary>
/// Delete an object from a specific bucket.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteObjectFromBucket(string bucketName, string
objectKey)
{
    try
    {
        var request = new DeleteObjectRequest()
        {
            BucketName = bucketName,
            Key = objectKey
        };
        await _amazonS3.DeleteObjectAsync(request);
        Console.WriteLine($"Deleted {objectKey} in {bucketName}.");
        return true;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Unable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
        return false;
    }
}

/// <summary>
/// Delete a specific bucket by deleting the objects and then the bucket
itself.
/// </summary>
/// <param name="bucketName">The Amazon S3 bucket to use.</param>
/// <param name="objectKey">The key of the object to delete.</param>
/// <param name="versionId">Optional versionId.</param>
/// <returns>True if successful.</returns>
public async Task<bool> CleanUpBucketByName(string bucketName)
{
    try
    {
        var allFiles = await ListBucketContentsByName(bucketName);

        foreach (var fileInfo in allFiles)
```

```
        {
            await DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key);
        }

        var request = new DeleteBucketRequest() { BucketName = bucketName, };
        var response = await _amazonS3.DeleteBucketAsync(request);
        Console.WriteLine($"\\tDelete for {bucketName} complete.");
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
            ex.Message);
        return false;
    }
}
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET.
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

JavaScript

SDK para JavaScript (v3)

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Ponto de entrada do fluxo de trabalho (index.js). Isso orquestra todas as etapas. Acesse o GitHub para ver os detalhes de implementação de Scenario, ScenarioInput, ScenarioOutput e ScenarioAction.


```
import * as Scenarios from "@aws-doc-sdk-examples/lib/scenario/index.js";
import {
  exitOnFalse,
  loadState,
  saveState,
} from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";

import { welcome, welcomeContinue } from "../welcome.steps.js";
import {
  confirmCreateBuckets,
  confirmPopulateBuckets,
  createBuckets,
  createBucketsAction,
  getBucketPrefix,
  populateBuckets,
  populateBucketsAction,
} from "../setup.steps.js";

/**
 * @param {Scenarios} scenarios
 * @param {Record<string, any>} initialState
 */
export const getWorkflowStages = (scenarios, initialState = {}) => {
  const client = new S3Client({});

  return {
    deploy: new scenarios.Scenario(
      "S3 Conditional Requests - Deploy",
      [
        welcome(scenarios),
        welcomeContinue(scenarios),
        exitOnFalse(scenarios, "welcomeContinue"),
        getBucketPrefix(scenarios),
        createBuckets(scenarios),
        confirmCreateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmCreateBuckets"),
        createBucketsAction(scenarios, client),
        populateBuckets(scenarios),
        confirmPopulateBuckets(scenarios),
        exitOnFalse(scenarios, "confirmPopulateBuckets"),
        populateBucketsAction(scenarios, client),
        saveState,
      ],
    ),
  },
}
```

```

        initialState,
    ),
    demo: new scenarios.Scenario(
        "S3 Conditional Requests - Demo",
        [loadState, welcome(scenarios), replAction(scenarios, client)],
        initialState,
    ),
    clean: new scenarios.Scenario(
        "S3 Conditional Requests - Destroy",
        [
            loadState,
            confirmCleanup(scenarios),
            exitOnFalse(scenarios, "confirmCleanup"),
            cleanupAction(scenarios, client),
        ],
        initialState,
    ),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { S3Client } from "@aws-sdk/client-s3";
import { cleanupAction, confirmCleanup } from "./clean.steps.js";
import { replAction } from "./repl.steps.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
    const objectLockingScenarios = getWorkflowStages(Scenarios);
    Scenarios.parseScenarioArgs(objectLockingScenarios, {
        name: "Amazon S3 object locking workflow",
        description:
            "Work with Amazon Simple Storage Service (Amazon S3) object locking features.",
        synopsis:
            "node index.js --scenario <deploy | demo | clean> [-h|--help] [-y|--yes] [-v|--verbose]",
    });
}

```

Envie mensagens de boas-vindas para o console (welcome.steps.js).

```
/**
```

```
* @typedef {import("@aws-doc-sdk-examples/lib/scenario/index.js")} Scenarios
*/

/**
 * @param {Scenarios} scenarios
 */
const welcome = (scenarios) =>
  new scenarios.ScenarioOutput(
    "welcome",
    "This example demonstrates the use of conditional requests for S3
operations." +
    " You can use conditional requests to add preconditions to S3 read requests
to return " +
    "or copy an object based on its Entity tag (ETag), or last modified
date.You can use " +
    "a conditional write requests to prevent overwrites by ensuring there is no
existing " +
    "object with the same key.\n" +
    "This example will enable you to perform conditional reads and writes that
will succeed " +
    "or fail based on your selected options.\n" +
    "Sample buckets and a sample object will be created as part of the example.
\n" +
    "Some steps require a key name prefix to be defined by the user. Before you
begin, you can " +
    "optionally edit this prefix in ./object_name.json. If you do so, please
reload the scenario before you begin.",
    { header: true },
  );

/**
 * @param {Scenarios} scenarios
 */
const welcomeContinue = (scenarios) =>
  new scenarios.ScenarioInput(
    "welcomeContinue",
    "Press Enter when you are ready to start.",
    { type: "confirm" },
  );

export { welcome, welcomeContinue };
```

Implante buckets e 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,
};
```

Obtenha, copie e coloque objetos usando solicitações condicionais do 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 };

```

Destrua todos os recursos criados (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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para JavaScript.
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Execute um cenário interativo que demonstre as solicitações condicionais do Amazon S3.

```
"""
Purpose

Shows how to use AWS SDK for Python (Boto3) to get started using conditional
requests for
Amazon Simple Storage Service (Amazon S3).

"""

import logging
import random
import sys
import datetime

import boto3
from botocore.exceptions import ClientError

from s3_conditional_requests import S3ConditionalRequests
```

```
# Add relative path to include demo_tools in this code example without need for
setup.
sys.path.append(".././.././..")
import demo_tools.question as q # noqa

# Constants
FILE_CONTENT = "This is a test file for S3 conditional requests."
RANDOM_SUFFIX = str(random.randint(100, 999))

logger = logging.getLogger(__name__)

class ConditionalRequestsScenario:
    """Runs a scenario that shows how to use S3 Conditional Requests."""

    def __init__(self, conditional_requests, s3_client):
        """
        :param conditional_requests: An object that wraps S3 conditional request
        actions.
        :param s3_client: A Boto3 S3 client for setup and cleanup operations.
        """
        self.conditional_requests = conditional_requests
        self.s3_client = s3_client

    def setup_scenario(self, source_bucket: str, dest_bucket: str, object_key:
str):
        """
        Sets up the scenario by creating a source and destination bucket.
        Prompts the user to provide a bucket name prefix.

        :param source_bucket: The name of the source bucket.
        :param dest_bucket: The name of the destination bucket.
        :param object_key: The name of a test file to add to the source bucket.
        """

        # Create the buckets.
        try:
            self.s3_client.create_bucket(Bucket=source_bucket)
            self.s3_client.create_bucket(Bucket=dest_bucket)
            print(
                f"Created source bucket: {source_bucket} and destination bucket:
{dest_bucket}"
            )
        except ClientError as e:
```

```

        error_code = e.response["Error"]["Code"]
        logger.error(f"Error creating buckets: {error_code}")
        raise

# Upload test file into the source bucket.
try:
    print(f"Uploading file {object_key} to bucket {source_bucket}")
    response = self.s3_client.put_object(
        Bucket=source_bucket, Key=object_key, Body=FILE_CONTENT
    )
    object_etag = response["ETag"]
    return object_etag

except Exception as e:
    logger.error(
        f"Failed to upload file {object_key} to bucket {source_bucket}:
{e}"
    )

def cleanup_scenario(self, source_bucket: str, dest_bucket: str):
    """
    Cleans up the scenario by deleting the source and destination buckets.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    """
    self.cleanup_bucket(source_bucket)
    self.cleanup_bucket(dest_bucket)

def cleanup_bucket(self, bucket_name: str):
    """
    Cleans up the bucket by deleting all objects and then the bucket itself.

    :param bucket_name: The name of the bucket.
    """
    try:
        # Get list of all objects in the bucket.
        list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
        objs = list_response.get("Contents", [])
        for obj in objs:
            key = obj["Key"]
            self.s3_client.delete_object(Bucket=bucket_name, Key=key)
        self.s3_client.delete_bucket(Bucket=bucket_name)

```

```
        print(f"Cleaned up bucket: {bucket_name}.")
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "NoSuchBucket":
            logger.info(f"Bucket {bucket_name} does not exist, skipping
cleanup.")
        else:
            logger.error(f"Error deleting bucket: {error_code}")
            raise

def display_buckets(self, source_bucket: str, dest_bucket: str):
    """
    Display a list of the objects in the test buckets.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    """
    self.list_bucket_contents(source_bucket)
    self.list_bucket_contents(dest_bucket)

def list_bucket_contents(self, bucket_name):
    """
    Display a list of the objects in the bucket.

    :param bucket_name: The name of the bucket.
    """
    try:
        # Get list of all objects in the bucket.
        print(f"\t Items in bucket {bucket_name}")
        list_response = self.s3_client.list_objects_v2(Bucket=bucket_name)
        objs = list_response.get("Contents", [])
        if not objs:
            print("\t\tNo objects found.")
        for obj in objs:
            key = obj["Key"]
            print(f"\t\t object: {key} ETag {obj['ETag']}")
        return objs
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "NoSuchBucket":
            logger.info(f"Bucket {bucket_name} does not exist.")
        else:
            logger.error(f"Error listing bucket and objects: {error_code}")
```

```
        raise

def display_menu(
    self, source_bucket: str, dest_bucket: str, object_key: str, etag: str
):
    """
    Displays the menu of conditional request options for the user.

    :param source_bucket: The name of the source bucket.
    :param dest_bucket: The name of the destination bucket.
    :param object_key: The key of the test object in the source bucket.
    :param etag: The etag of the test object in the source bucket.
    """

    actions = [
        "Print list of bucket items.",
        "Perform a conditional read.",
        "Perform a conditional copy.",
        "Perform a conditional write.",
        "Clean up and exit.",
    ]

    conditions = [
        "If-Match: using the object's ETag. This condition should succeed.",
        "If-None-Match: using the object's ETag. This condition should",
fail.",
        "If-Modified-Since: using yesterday's date. This condition should",
succeed.",
        "If-Unmodified-Since: using yesterday's date. This condition should",
fail.",
    ]

    condition_types = [
        "IfMatch",
        "IfNoneMatch",
        "IfModifiedSince",
        "IfUnmodifiedSince",
    ]

    copy_condition_types = [
        "CopySourceIfMatch",
        "CopySourceIfNoneMatch",
        "CopySourceIfModifiedSince",
        "CopySourceIfUnmodifiedSince",
    ]
```



```

    ]

    yesterday_date = datetime.datetime.utcnow() - datetime.timedelta(days=1)

    choice = 0
    while choice != 4:
        print("-" * 88)
        print("Choose an action to explore some example conditional
requests.")
        choice = q.choose("Which action would you like to take? ", actions)
        if choice == 0:
            print("Listing the objects and buckets.")
            self.display_buckets(source_bucket, dest_bucket)
        elif choice == 1:
            print("Perform a conditional read.")
            condition_type = q.choose("Enter the condition type : ",
conditions)
            if condition_type == 0 or condition_type == 1:
                self.conditional_requests.get_object_conditional(
                    object_key, source_bucket,
condition_types[condition_type], etag
                )
            elif condition_type == 2 or condition_type == 3:
                self.conditional_requests.get_object_conditional(
                    object_key,
                    source_bucket,
                    condition_types[condition_type],
                    yesterday_date,
                )
        elif choice == 2:
            print("Perform a conditional copy.")
            condition_type = q.choose("Enter the condition type : ",
conditions)
            dest_key = q.ask("Enter an object key: ", q.non_empty)
            if condition_type == 0 or condition_type == 1:
                self.conditional_requests.copy_object_conditional(
                    object_key,
                    dest_key,
                    source_bucket,
                    dest_bucket,
                    copy_condition_types[condition_type],
                    etag,
                )
            elif condition_type == 2 or condition_type == 3:

```

```

        self.conditional_requests.copy_object_conditional(
            object_key,
            dest_key,
            copy_condition_types[condition_type],
            yesterday_date,
        )
    elif choice == 3:
        print(
            "Perform a conditional write using IfNoneMatch condition on
the object key."
        )
        print("If the key is a duplicate, the write will fail.")
        object_key = q.ask("Enter an object key: ", q.non_empty)
        self.conditional_requests.put_object_conditional(
            object_key, source_bucket, b"Conditional write example data."
        )
    elif choice == 4:
        print("Proceeding to cleanup.")

def run_scenario(self):
    """
    Runs the interactive scenario.
    """
    print("-" * 88)
    print("Welcome to the Amazon S3 conditional requests example.")
    print("-" * 88)

    print(
        f"""\
    This example demonstrates the use of conditional requests for S3
operations.
    You can use conditional requests to add preconditions to S3 read requests
to return or copy
    an object based on its Entity tag (ETag), or last modified date.
    You can use a conditional write requests to prevent overwrites by
ensuring
    there is no existing object with the same key.

    This example will allow you to perform conditional reads
and writes that will succeed or fail based on your selected options.

    Sample buckets and a sample object will be created as part of the
example.
```

```

        """
    )

    bucket_prefix = q.ask("Enter a bucket name prefix: ", q.non_empty)
    source_bucket_name = f"{bucket_prefix}-source-{RANDOM_SUFFIX}"
    dest_bucket_name = f"{bucket_prefix}-dest-{RANDOM_SUFFIX}"
    object_key = "test-upload-file.txt"

    try:
        etag = self.setup_scenario(source_bucket_name, dest_bucket_name,
                                object_key)
        self.display_menu(source_bucket_name, dest_bucket_name, object_key,
                        etag)
    finally:
        self.cleanup_scenario(source_bucket_name, dest_bucket_name)

    print("-" * 88)
    print("Thanks for watching.")
    print("-" * 88)

if __name__ == "__main__":
    scenario = ConditionalRequestsScenario(
        S3ConditionalRequests.from_client(), boto3.client("s3")
    )
    scenario.run_scenario()

```

Uma classe de wrapper que define as operações de solicitação 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\n"
                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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
 - [CopyObject](#)
 - [GetObject](#)

- [PutObject](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Gerenciar listas de controle de acesso (ACLs) para buckets do Amazon S3 usando um AWS SDK

Os exemplos de código a seguir mostram como gerenciar listas de controle de acesso (ACLs) para buckets do Amazon S3.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
    /// 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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET.
 - [GetBucketAcl](#)
 - [GetObjectAcl](#)
 - [PutBucketAcl](#)
 - [PutObjectAcl](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Gerenciar mensagens grandes do Amazon SQS usando o Amazon S3 com um AWS SDK

O exemplo de código a seguir mostra como usar a biblioteca do cliente em versão ampliada do Amazon SQS para trabalhar com mensagens grandes do SQS.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK for Java 2.x](#).
- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
 - [CreateBucket](#)
 - [PutBucketLifecycleConfiguration](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Gerenciar objetos do Amazon S3 versionados em lotes com uma função do Lambda usando um AWS SDK

O exemplo de código a seguir mostra como gerenciar objetos do S3 versionados em lotes com uma função do Lambda.

Python

SDK para Python (Boto3).

Mostra como manipular objetos do Amazon Simple Storage Service (Amazon S3) versionados em lotes criando trabalhos que chamam funções do AWS Lambda para executar o processamento. Este exemplo cria um bucket habilitado para versão, carrega as estrofes do poema Você é velho, padre William, de Lewis Carroll, e usa trabalhos em lote do Amazon S3 para distorcer o poema de várias maneiras.

Aprenda como:

- Criar funções do Lambda que operam em objetos versionados.
- Criar um manifesto de objetos para atualizar.

- Cria trabalhos em lote que invocam funções do Lambda para atualizar objetos.
- Excluir funções do Lambda.
- Esvaziar e excluir um bucket versionado.

Este exemplo é mais bem visualizado no GitHub. Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- Amazon S3

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Analisar URIs do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como usar analisar URIs do Amazon S3 para extrair componentes importantes como o nome do bucket e a chave do objeto.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Analise um URI do Amazon S3 usando a classe [S3Uri](#).

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.S3Uri;
import software.amazon.awssdk.services.s3.S3Utilities;

import java.net.URI;
```

```
import java.util.List;
import java.util.Map;

/**
 *
 * @param s3Client    - An S3Client through which you acquire an S3Uri
instance.
 * @param s3objectUrl - A complex URL (String) that is used to demonstrate
S3Uri
 *
 *                      capabilities.
 */
public static void parseS3UriExample(S3Client s3Client, String s3objectUrl) {
    logger.info(s3objectUrl);
    // Console output:
    // 'https://s3.us-west-1.amazonaws.com/myBucket/resources/doc.txt?
versionId=abc123&partNumber=77&partNumber=88'.

    // Create an S3Utilities object using the configuration of the s3Client.
    S3Utilities s3Utilities = s3Client.utilities();

    // From a String URL create a URI object to pass to the parseUri()
method.
    URI uri = URI.create(s3objectUrl);
    S3Uri s3Uri = s3Utilities.parseUri(uri);

    // If the URI contains no value for the Region, bucket or key, the SDK
returns
    // an empty Optional.
    // The SDK returns decoded URI values.

    Region region = s3Uri.region().orElse(null);
    log("region", region);
    // Console output: 'region: us-west-1'.

    String bucket = s3Uri.bucket().orElse(null);
    log("bucket", bucket);
    // Console output: 'bucket: myBucket'.

    String key = s3Uri.key().orElse(null);
    log("key", key);
    // Console output: 'key: resources/doc.txt'.

    Boolean isPathStyle = s3Uri.isPathStyle();
    log("isPathStyle", isPathStyle);
}
```

```

// Console output: 'isPathStyle: true'.

// If the URI contains no query parameters, the SDK returns an empty map.
Map<String, List<String>> queryParams = s3Uri.rawQueryParameters();
log("rawQueryParameters", queryParams);
// Console output: 'rawQueryParameters: {versionId=[abc123],
partNumber=[77,
// 88]}'.

// Retrieve the first or all values for a query parameter as shown in the
// following code.
String versionId =
s3Uri.firstMatchingRawQueryParameter("versionId").orElse(null);
log("firstMatchingRawQueryParameter-versionId", versionId);
// Console output: 'firstMatchingRawQueryParameter-versionId: abc123'.

String partNumber =
s3Uri.firstMatchingRawQueryParameter("partNumber").orElse(null);
log("firstMatchingRawQueryParameter-partNumber", partNumber);
// Console output: 'firstMatchingRawQueryParameter-partNumber: 77'.

List<String> partNumbers =
s3Uri.firstMatchingRawQueryParameters("partNumber");
log("firstMatchingRawQueryParameter", partNumbers);
// Console output: 'firstMatchingRawQueryParameter: [77, 88]'.

/*
 * Object keys and query parameters with reserved or unsafe characters,
must be
 * URL-encoded.
 * For example replace whitespace " " with "%20".
 * Valid:
 * "https://s3.us-west-1.amazonaws.com/myBucket/object%20key?query=
%5Bbrackets%5D"
 * Invalid:
 * "https://s3.us-west-1.amazonaws.com/myBucket/object key?
query=[brackets]"
 *
 * Virtual-hosted-style URIs with bucket names that contain a dot, ".",
the dot
 * must not be URL-encoded.
 * Valid: "https://my.Bucket.s3.us-west-1.amazonaws.com/key"
 * Invalid: "https://my%2EBucket.s3.us-west-1.amazonaws.com/key"
 */

```

```
}

private static void log(String s3UriElement, Object element) {
    if (element == null) {
        logger.info("{}: {}", s3UriElement, "null");
    } else {
        logger.info("{}: {}", s3UriElement, element);
    }
}
```

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Executar uma cópia multipart de um objeto do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como executar uma cópia multipart de um objeto do Amazon S3.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para .NET.
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [GetObjectMetadata](#)
 - [UploadPartCopy](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Receber e processar notificações de eventos do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como trabalhar com notificações de eventos do S3 de uma forma orientada a objetos.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Esse exemplo mostra como processar o evento de notificação do S3 usando o 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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
 - [DeleteMessageBatch](#)
 - [GetQueueAttributes](#)
 - [PutBucketNotificationConfiguration](#)
 - [ReceiveMessage](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Salvar o EXIF e outras informações de imagem usando um AWS SDK

O código de exemplo abaixo mostra como:

- Obter informações de EXIF de um arquivo JPG, JPEG ou PNG.
- Fazer upload do arquivo de imagem para um bucket do Amazon S3.
- Usar o Amazon Rekognition para identificar os três principais atributos (rótulos) no arquivo.
- Adicione as informações de EXIF e rótulo a uma tabela do Amazon DynamoDB na região.

Rust

SDK para Rust

Obtenha informações de EXIF de um arquivo JPG, JPEG ou PNG, faça upload do arquivo de imagem para um bucket do Amazon S3, use o Amazon Rekognition para identificar os três principais atributos (rótulos no Amazon Rekognition) no arquivo e adicione as informações de EXIF e de rótulo a uma tabela do Amazon DynamoDB na região.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços usados neste exemplo

- DynamoDB
- Amazon Rekognition

- Amazon S3

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Enviar notificações de eventos do S3 ao Amazon EventBridge usando um AWS SDK

O exemplo de código a seguir mostra como habilitar um bucket para enviar notificações de eventos do S3 ao EventBridge e direcionar notificações para um tópico do Amazon SNS e uma fila do Amazon SQS.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
/** 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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
 - [PutBucketNotificationConfiguration](#)
 - [PutRule](#)
 - [PutTargets](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Monitorar o upload ou download de um objeto do Amazon S3 usando um SDK da AWS

O exemplo de código a seguir mostra como usar o Amazon S3 para carregar ou baixar arquivos grandes.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Acompanhe o andamento do carregamento de um arquivo.

```
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!

    */
}

```

Acompanhe o andamento do download de um arquivo.

```

public void trackDownloadFile(S3TransferManager transferManager, String
bucketName,
                                String key, String downloadedFilePath) {
    DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
        .getObjectRequest(b -> b.bucket(bucketName).key(key))
        .addTransferListener(LoggingTransferListener.create()) // Add
listener.
        .destination(Paths.get(downloadedFilePath))
        .build();

    FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

    CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
    /*
        The SDK provides a LoggingTransferListener implementation of the
TransferListener interface.
        You can also implement the interface to provide your own logic.

        Configure log4J2 with settings such as the following.
        <Configuration status="WARN">
            <Appenders>
                <Console name="AlignedConsoleAppender"
target="SYSTEM_OUT">
                    <PatternLayout pattern="%m%n"/>
                </Console>
            </Appenders>

            <Loggers>
                <logger
name="software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener"
level="INFO" additivity="false">
                    <AppenderRef ref="AlignedConsoleAppender"/>
                </logger>

```



```

        </Loggers>
    </Configuration>

    Log4J2 logs the progress. The following is example output for a 21.3
    MB file download.

        Transfer initiated...
        |=====| 39.4%
        |=====| 78.8%
        |=====| 100.0%
        Transfer complete!

    */
}
```

- Para obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
 - [GetObject](#)
 - [PutObject](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Como transformar dados para sua aplicação com o S3 Object Lambda

O exemplo de código a seguir mostra como transformar dados para sua aplicação com o S3 Object Lambda.

.NET

SDK para .NET

Mostra como adicionar código personalizado a solicitações GET padrão do S3 para modificar o objeto solicitado e recuperado do S3 e possibilitar que o objeto atenda às necessidades do cliente ou aplicação solicitante.

Para obter o código-fonte completo e instruções sobre como configurar e executar o exemplo, consulte o exemplo completo no [GitHub](#).

Serviços utilizados neste exemplo

- Lambda

- Amazon S3

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos de abordagens para testes de unidade e integração com SDK da AWS

O exemplo de código a seguir mostra técnicas de práticas recomendadas ao escrever testes de unidade e de integração usando SDK da AWS.

Rust

SDK para Rust

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Cargo.toml para exemplos de testes.

```
[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"
```

Exemplo de teste de unidade usando automock e um wrapper de serviço.

```
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);
}
}

```

Exemplo de teste de integração usando 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 ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Fazer upload recursivo de um diretório local em um bucket do Amazon Simple Storage Service (Amazon S3)

O exemplo de código a seguir mostra como fazer upload recursivo de um diretório local em um bucket do Amazon Simple Storage Service (Amazon S3).

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Use [S3TransferManager](#) para [fazer upload de um diretório local](#). Veja o [arquivo completo](#) e [teste](#).

```
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 obter detalhes da API, consulte [UploadDirectory](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Fazer upload ou download de arquivos grandes de e para o Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como fazer upload e download de arquivos grandes de e para o Amazon S3.

Para obter mais informações, consulte [Carregar um objeto usando carregamento fracionado](#).

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Chame funções que transferem arquivos de e para um bucket do S3 usando o 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($"\\n{sepBar}");
    }

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
    }
}
```

```
    }  
}  
  
// Displays a list of the files in the specified S3 bucket and prefix.  
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string  
    s3Path)  
{  
    ListObjectsV2Request request = new()  
    {  
        BucketName = bucketName,  
        Prefix = s3Path,  
        MaxKeys = 5,  
    };  
  
    var response = new ListObjectsV2Response();  
  
    do  
    {  
        response = await client.ListObjectsV2Async(request);  
  
        response.S3Objects  
            .ForEach(obj => Console.WriteLine($"{obj.Key}"));  
  
        // If the response is truncated, set the request ContinuationToken  
        // from the NextContinuationToken property of the response.  
        request.ContinuationToken = response.NextContinuationToken;  
    } while (response.IsTruncated);  
}
```

Carregue um único arquivo.

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

Carregue um diretório local inteiro.

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

Baixe um único arquivo.

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

Baixe o conteúdo de um bucket do 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;
}
```

Acompanhe o progresso de um upload usando `TransferUtility`.

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

/// <summary>
/// This example shows how to track the progress of a multipart upload
/// using the Amazon Simple Storage Service (Amazon S3) TransferUtility to
/// upload to an Amazon S3 bucket.
/// </summary>
public class TrackMPUUsingHighLevelAPI
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "sample_pic.png";
        string path = "filepath/directory/";
        string filePath = $"{path}{keyName}";

        // If the AWS Region defined for your default user is different
        // from the Region where your Amazon S3 bucket is located,
        // pass the Region name to the Amazon S3 client object's constructor.
        // For example: RegionEndpoint.USSouth1 or RegionEndpoint.USEast2.
        IAmazonS3 client = new AmazonS3Client();

        await TrackMPUAsync(client, bucketName, filePath, keyName);
    }
}
```

```
/// <summary>
/// Starts an Amazon S3 multipart upload and assigns an event handler to
/// track the progress of the upload.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
/// perform the multipart upload.</param>
/// <param name="bucketName">The name of the bucket to which to upload
/// the file.</param>
/// <param name="filePath">The path, including the file name of the
/// file to be uploaded to the Amazon S3 bucket.</param>
/// <param name="keyName">The file name to be used in the
/// destination Amazon S3 bucket.</param>
public static async Task TrackMPUAsync(
    IAmazonS3 client,
    string bucketName,
    string filePath,
    string keyName)
{
    try
    {
        var fileTransferUtility = new TransferUtility(client);

        // Use TransferUtilityUploadRequest to configure options.
        // In this example we subscribe to an event.
        var uploadRequest =
            new TransferUtilityUploadRequest
            {
                BucketName = bucketName,
                FilePath = filePath,
                Key = keyName,
            };

        uploadRequest.UploadProgressEvent +=
            new EventHandler<UploadProgressArgs>(
                UploadRequest_UploadPartProgressEvent);

        await fileTransferUtility.UploadAsync(uploadRequest);
        Console.WriteLine("Upload completed");
    }
    catch (AmazonS3Exception ex)
    {
        Console.WriteLine($"Error:: {ex.Message}");
    }
}
```

```
/// <summary>
/// Event handler to check the progress of the multipart upload.
/// </summary>
/// <param name="sender">The object that raised the event.</param>
/// <param name="e">The object that contains multipart upload
/// information.</param>
public static void UploadRequest_UploadPartProgressEvent(object sender,
UploadProgressArgs e)
{
    // Process event.
    Console.WriteLine($"{e.TransferredBytes}/{e.TotalBytes}");
}
}
```

Faça upload de um objeto com criptografia.

```
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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie funções que usam gerenciadores de upload e download para dividir os dados em partes e transferi-los simultaneamente.

```
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
}
```

Execute um cenário interativo que mostre como usar os gerenciadores de upload e download em 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Chame funções que transferem arquivos de e para um bucket do S3 usando o `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();
}
```


Carregue um diretório local inteiro.

```
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();
}
```

Carregue um único arquivo.

```
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();
}
```

Os exemplos de código usam as importações a seguir.

```
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;
```

Use o [Gerenciador de Transferências do S3](#) na parte superior do [cliente do S3 baseado no AWS CRT](#) para realizar de forma transparente um carregamento fracionado quando o tamanho do conteúdo exceder um limite. O limite padrão é 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();
}

```

Use a [API S3Client](#) para realizar um carregamento fracionado.

```

/**
 * 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()));
}

```

Use a [API S3AsyncClient](#) com o carregamento fracionado habilitado para realizar um carregamento fracionado.

```

/**
 * 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Faça upload de um arquivo 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;
    }
  }
};
```

Baixe um arquivo 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 and 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie funções que transfiram arquivos usando várias configurações do gerenciador de transferências disponíveis. Use uma classe de retorno de chamada para gravar o andamento do retorno de chamada durante a transferência de arquivos.

```
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
```

Demonstre as funções do gerenciador de transferências e relate 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 {} {}"
        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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Faça upload de um fluxo de tamanho desconhecido em um objeto do Amazon S3 usando um AWS SDK

Os exemplos de código a seguir mostram como fazer upload de um fluxo de tamanho desconhecido em um objeto do Amazon S3.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Use o [Cliente do S3 baseado em CRT da 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;
    }
}

```

Use o [cliente do S3 assíncrono padrão com o upload de várias partes ativado](#).

```
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;
    }
}

```

Use o [Gerenciador de transferências do 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).


```
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 ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar somas de verificação para trabalhar com um objeto do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como usar somas de verificação para trabalhar com um objeto do Amazon S3.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Os exemplos de código usam um subconjunto das importações a seguir.

```
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 um algoritmo de soma de verificação para o método `putObject` ao [criar 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"));
    }
```

Verifique a soma de verificação do método `getObject` ao [criar o `GetObjectRequest`](#).

```
public GetObjectResponse getObjectWithChecksum(String bucketName, String key)
{
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

Calcule previamente uma soma de verificação para o método `putObject` ao [criar o `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)));
}
```

Use o [Gerenciador de Transferências do S3](#) na parte superior do [cliente do S3 baseado no AWS CRT](#) para realizar de forma transparente um carregamento fracionado quando o tamanho do conteúdo exceder um limite. O limite padrão é 8 MB.

Você pode especificar um algoritmo de soma de verificação para o SDK usar. Por padrão, o SDK usa o 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();
    }

```

Use a [API S3Client](#) ou (API S3AsyncClient) para realizar um carregamento fracionado. Se você especificar uma soma de verificação adicional, deverá indicar o algoritmo a ser usado no início do carregamento. Você também deve especificar o algoritmo para a solicitação de cada parte e fornecer a soma de verificação calculada para cada parte após o carregamento.

```

    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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Trabalhar com os recursos de integridade de objetos do Amazon S3 usando um AWS SDK

O exemplo de código a seguir mostra como trabalhar com os recursos de integridade de objetos do S3.

C++

SDK para C++

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Execute um cenário interativo que demonstra os recursos de integridade do 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.getByteBufferHash();

    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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para C++.
- [AbortMultipartUpload](#)
- [CompleteMultipartUpload](#)
- [CreateMultipartUpload](#)
- [DeleteObject](#)
- [GetObjectAttributes](#)
- [PutObject](#)
- [UploadPart](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Trabalhar com objetos versionados do Amazon S3 usando um AWS SDK

O código de exemplo abaixo mostra como:

- Criar um bucket do S3 versionado.
- Obter todas as versões de um objeto.
- Reverter um objeto para uma versão anterior.
- Excluir e restaurar um objeto versionado.
- Excluir, permanentemente, todas as versões de um objeto.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie funções que envolvam ações do 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

```

Carregue uma estrofe de poema para um objeto versionado e realize uma série de ações nela.

```

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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
 - [CreateBucket](#)

- [DeleteObject](#)
- [ListObjectVersions](#)
- [PutBucketLifecycleConfiguration](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos de tecnologia sem servidor para o Amazon S3

Os exemplos de código a seguir mostram como usar o Amazon S3 com SDKs da AWS.

Exemplos

- [Invocar uma função do Lambda em um acionador do Amazon S3](#)

Invocar uma função do Lambda em um acionador do Amazon S3

Os exemplos de código a seguir mostram como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto em um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama a API do Amazon S3 para recuperar e registrar em log o tipo de conteúdo do objeto.

.NET

SDK para .NET

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do S3 com o Lambda usando .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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do S3 com o Lambda usando 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do S3 com o Lambda usando 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do S3 com o Lambda usando 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);
    }
};
```

Consumir um evento do S3 com o Lambda usando 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Como consumir um evento do S3 com o Lambda usando 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do S3 com o Lambda usando 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Como consumir um evento do S3 com o Lambda usando 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

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do S3 com o Lambda usando 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 ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos de código do Amazon S3 Control usando AWS SDKs

Os exemplos de código a seguir mostram como usar o Amazon S3 Control com um kit de desenvolvimento de software (SDK) da AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, é possível ver as ações no contexto em seus cenários relacionados.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Amazon S3 Control

- [Exemplos básicos do Amazon S3 Control usando AWS SDKs](#)
 - [Olá, Amazon S3 Control](#)
 - [Conheça os conceitos básicos do Amazon S3 Control com um SDK da AWS](#)
 - [Ações do Amazon S3 Control usando AWS SDKs](#)
 - [Usar CreateJob com o AWS SDK ou a CLI](#)
 - [Usar DeleteJobTagging com um SDK da AWS](#)
 - [Usar DescribeJob com o AWS SDK ou a CLI](#)
 - [Usar GetJobTagging com um SDK da AWS](#)
 - [Usar PutJobTagging com um SDK da AWS](#)
 - [Usar UpdateJobPriority com o AWS SDK ou a CLI](#)
 - [Usar UpdateJobStatus com o AWS SDK ou a CLI](#)

Exemplos básicos do Amazon S3 Control usando AWS SDKs

Os exemplos de código a seguir mostram como usar os conceitos básicos do Amazon S3 Control com AWS SDKs.

Exemplos

- [Olá, Amazon S3 Control](#)

- [Conheça os conceitos básicos do Amazon S3 Control com um SDK da AWS](#)
- [Ações do Amazon S3 Control usando AWS SDKs](#)
 - [Usar CreateJob com o AWS SDK ou a CLI](#)
 - [Usar DeleteJobTagging com um SDK da AWS](#)
 - [Usar DescribeJob com o AWS SDK ou a CLI](#)
 - [Usar GetJobTagging com um SDK da AWS](#)
 - [Usar PutJobTagging com um SDK da AWS](#)
 - [Usar UpdateJobPriority com o AWS SDK ou a CLI](#)
 - [Usar UpdateJobStatus com o AWS SDK ou a CLI](#)

Olá, Amazon S3 Control

Os exemplos de código a seguir mostram como começar a usar o Amazon S3 Control.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [ListJobs](#) na Referência da API AWS SDK for Java 2.x.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 ver detalhes da API, consulte [ListJobs](#) na Referência de API do SDK da AWS para Python (Boto3).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Conheça os conceitos básicos do Amazon S3 Control com um SDK da AWS

Os exemplos de código a seguir mostram como aprender as principais operações do Amazon S3 Control.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Conheça as principais operações.

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

```

    }
}

}

```

Uma classe de ação que envolve operações.

```

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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
 - [CreateJob](#)
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)
 - [UpdateJobStatus](#)

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Conheça o cenário básico do S3 Batch.

```
class S3BatchWrapper:
    """Wrapper class for managing S3 Batch Operations."""

    def __init__(self, s3_client: Any, s3control_client: Any, sts_client: Any) ->
    None:
        """
        Initializes the S3BatchWrapper with AWS service clients.

        :param s3_client: A Boto3 Amazon S3 client. This client provides low-
        level
                           access to AWS S3 services.
        :param s3control_client: A Boto3 Amazon S3 Control client. This client
        provides
                           low-level access to AWS S3 Control services.
        :param sts_client: A Boto3 AWS STS client. This client provides low-level
```

```

        access to AWS STS services.

    """
    self.s3_client = s3_client
    self.s3control_client = s3control_client
    self.sts_client = sts_client
    # Get region from the client for bucket creation logic
    self.region_name = self.s3_client.meta.region_name

def get_account_id(self) -> str:
    """
    Get AWS account ID.

    Returns:
        str: AWS account ID
    """
    return self.sts_client.get_caller_identity()["Account"]

def create_bucket(self, bucket_name: str) -> None:
    """
    Create an S3 bucket.

    Args:
        bucket_name (str): Name of the bucket to create

    Raises:
        ClientError: If bucket creation fails
    """
    try:
        if self.region_name and self.region_name != 'us-east-1':
            self.s3_client.create_bucket(
                Bucket=bucket_name,
                CreateBucketConfiguration={
                    'LocationConstraint': self.region_name
                }
            )
        else:
            self.s3_client.create_bucket(Bucket=bucket_name)
            print(f"Created bucket: {bucket_name}")
    except ClientError as e:
        print(f"Error creating bucket: {e}")
        raise

def upload_files_to_bucket(self, bucket_name: str, file_names: List[str]) ->
str:

```

```
"""
Upload files to S3 bucket including manifest file.

Args:
    bucket_name (str): Target bucket name
    file_names (list): List of file names to upload

Returns:
    str: ETag of the manifest file

Raises:
    ClientError: If file upload fails
"""
try:
    for file_name in file_names:
        if file_name != "job-manifest.csv":
            content = f"Content for {file_name}"
            self.s3_client.put_object(
                Bucket=bucket_name,
                Key=file_name,
                Body=content.encode('utf-8')
            )
            print(f"Uploaded {file_name} to {bucket_name}")

    manifest_content = ""
    for file_name in file_names:
        if file_name != "job-manifest.csv":
            manifest_content += f"{bucket_name},{file_name}\n"

    manifest_response = self.s3_client.put_object(
        Bucket=bucket_name,
        Key="job-manifest.csv",
        Body=manifest_content.encode('utf-8')
    )
    print(f"Uploaded manifest file to {bucket_name}")
    print(f"Manifest content:\n{manifest_content}")
    return manifest_response['ETag'].strip('')

except ClientError as e:
    print(f"Error uploading files: {e}")
    raise

def create_s3_batch_job(self, account_id: str, role_arn: str,
    manifest_location: str,
```



```

        report_bucket_name: str) -> str:
    """
    Create an S3 batch operation job.

    Args:
        account_id (str): AWS account ID
        role_arn (str): IAM role ARN for batch operations
        manifest_location (str): Location of the manifest file
        report_bucket_name (str): Bucket for job reports

    Returns:
        str: Job ID

    Raises:
        ClientError: If job creation fails
    """
    try:
        bucket_name = manifest_location.split(':::')[1].split('/')[0]
        manifest_key = 'job-manifest.csv'
        manifest_obj = self.s3_client.head_object(
            Bucket=bucket_name,
            Key=manifest_key
        )
        etag = manifest_obj['ETag'].strip('\"')

        response = self.s3control_client.create_job(
            AccountId=account_id,
            Operation={
                'S3PutObjectTagging': {
                    'TagSet': [
                        {
                            'Key': 'BatchTag',
                            'Value': 'BatchValue'
                        },
                    ],
                }
            },
            Report={
                'Bucket': report_bucket_name,
                'Format': 'Report_CSV_20180820',
                'Enabled': True,
                'Prefix': 'batch-op-reports',
                'ReportScope': 'AllTasks'
            },

```

```

        Manifest={
            'Spec': {
                'Format': 'S3BatchOperations_CSV_20180820',
                'Fields': ['Bucket', 'Key']
            },
            'Location': {
                'ObjectArn': manifest_location,
                'ETag': etag
            }
        },
        Priority=10,
        RoleArn=role_arn,
        Description='Batch job for tagging objects',
        ConfirmationRequired=True
    )
    job_id = response['JobId']
    print(f"The Job id is {job_id}")
    return job_id
except ClientError as e:
    print(f"Error creating batch job: {e}")
    if 'Message' in str(e):
        print(f"Detailed error message: {e.response['Message']}")
    raise

def check_job_failure_reasons(self, job_id: str, account_id: str) ->
List[Dict[str, Any]]:
    """
    Check for any failure reasons of a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID

    Returns:
        list: List of failure reasons

    Raises:
        ClientError: If checking job failure reasons fails
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )

```

```

        if 'FailureReasons' in response['Job']:
            for reason in response['Job']['FailureReasons']:
                print(f"- {reason}")
            return response['Job'].get('FailureReasons', [])
    except ClientError as e:
        print(f"Error checking job failure reasons: {e}")
        raise

    def wait_for_job_ready(self, job_id: str, account_id: str, desired_status:
    str = 'Ready') -> bool:
        """
        Wait for a job to reach the desired status.

        Args:
            job_id (str): ID of the batch job
            account_id (str): AWS account ID
            desired_status (str): Target status to wait for

        Returns:
            bool: True if desired status is reached, False otherwise

        Raises:
            ClientError: If checking job status fails
        """
        print(f"Waiting for job to become {desired_status}...")
        max_attempts = 60
        attempt = 0
        while attempt < max_attempts:
            try:
                response = self.s3control_client.describe_job(
                    AccountId=account_id,
                    JobId=job_id
                )
                current_status = response['Job']['Status']
                print(f"Current job status: {current_status}")
                if current_status == desired_status:
                    return True
                if current_status == 'Suspended':
                    print("Job is in Suspended state, can proceed with
activation")
                    return True
                if current_status in ['Active', 'Failed', 'Cancelled',
'Complete']:

```

```

        print(f"Job is in {current_status} state, cannot reach
{desired_status} status")
        if 'FailureReasons' in response['Job']:
            print("Failure reasons:")
            for reason in response['Job']['FailureReasons']:
                print(f"- {reason}")
            return False

        time.sleep(20)
        attempt += 1
    except ClientError as e:
        print(f"Error checking job status: {e}")
        raise
print(f"Timeout waiting for job to become {desired_status}")
return False

def update_job_priority(self, job_id: str, account_id: str) -> None:
    """
    Update the priority of a batch job and start it.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended']:
            self.s3control_client.update_job_priority(
                AccountId=account_id,
                JobId=job_id,
                Priority=60
            )
            print("The job priority was updated")

        try:
            self.s3control_client.update_job_status(
                AccountId=account_id,
                JobId=job_id,

```

```

        RequestedJobStatus='Ready'
    )
    print("Job activated successfully")
except ClientError as activation_error:
    print(f"Note: Could not activate job automatically:
{activation_error}")
    print("Job priority was updated successfully. Job may need
manual activation in the console.")
    elif current_status in ['Active', 'Completing', 'Complete']:
        print(f"Job is in '{current_status}' state - priority cannot be
updated")
        if current_status == 'Completing':
            print("Job is finishing up and will complete soon.")
        elif current_status == 'Complete':
            print("Job has already completed successfully.")
        else:
            print("Job is currently running.")
    else:
        print(f"Job is in '{current_status}' state - priority update not
allowed")

except ClientError as e:
    print(f"Error updating job priority: {e}")
    print("Continuing with the scenario...")
    return

def cancel_job(self, job_id: str, account_id: str) -> None:
    """
    Cancel an S3 batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        current_status = response['Job']['Status']
        print(f"Current job status: {current_status}")

        if current_status in ['Ready', 'Suspended', 'Active']:
            self.s3control_client.update_job_status(

```

```

        AccountId=account_id,
        JobId=job_id,
        RequestedJobStatus='Cancelled'
    )
    print(f"Job {job_id} was successfully canceled.")
elif current_status in ['Completing', 'Complete']:
    print(f"Job is in '{current_status}' state - cannot be
cancelled")
    if current_status == 'Completing':
        print("Job is finishing up and will complete soon.")
    elif current_status == 'Complete':
        print("Job has already completed successfully.")
else:
    print(f"Job is in '{current_status}' state - cancel not allowed")
except ClientError as e:
    print(f"Error canceling job: {e}")
    raise

def describe_job_details(self, job_id: str, account_id: str) -> None:
    """
    Describe detailed information about a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.describe_job(
            AccountId=account_id,
            JobId=job_id
        )
        job = response['Job']
        print(f"Job ID: {job['JobId']}")
        print(f"Description: {job.get('Description', 'N/A')}")
        print(f"Status: {job['Status']}")
        print(f"Role ARN: {job['RoleArn']}")
        print(f"Priority: {job['Priority']}")
        if 'ProgressSummary' in job:
            progress = job['ProgressSummary']
            print(f"Progress Summary:
Total={progress.get('TotalNumberOfTasks', 0)}, "
                  f"Succeeded={progress.get('NumberOfTasksSucceeded', 0)}, "
                  f"Failed={progress.get('NumberOfTasksFailed', 0)}")
    except ClientError as e:

```

```
        print(f"Error describing job: {e}")
        raise

def get_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Get tags associated with a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.get_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        tags = response.get('Tags', [])
        if tags:
            print(f"Tags for job {job_id}:")
            for tag in tags:
                print(f"  {tag['Key']}: {tag['Value']}")
        else:
            print(f"No tags found for job ID: {job_id}")
    except ClientError as e:
        print(f"Error getting job tags: {e}")
        raise

def put_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Add tags to a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.put_job_tagging(
            AccountId=account_id,
            JobId=job_id,
            Tags=[
                {'Key': 'Environment', 'Value': 'Development'},
                {'Key': 'Team', 'Value': 'DataProcessing'}
            ]
        )
```

```
        print(f"Additional tags were added to job {job_id}")
    except ClientError as e:
        print(f"Error adding job tags: {e}")
        raise

def list_jobs(self, account_id: str) -> None:
    """
    List all batch jobs for the account.

    Args:
        account_id (str): AWS account ID
    """
    try:
        response = self.s3control_client.list_jobs(
            AccountId=account_id,
            JobStatuses=['Active', 'Complete', 'Cancelled', 'Failed', 'New',
'Paused', 'Pausing', 'Preparing', 'Ready', 'Suspended']
        )
        jobs = response.get('Jobs', [])
        for job in jobs:
            print(f"The job id is {job['JobId']}")
            print(f"The job priority is {job['Priority']}")
    except ClientError as e:
        print(f"Error listing jobs: {e}")
        raise

def delete_job_tags(self, job_id: str, account_id: str) -> None:
    """
    Delete all tags from a batch job.

    Args:
        job_id (str): ID of the batch job
        account_id (str): AWS account ID
    """
    try:
        self.s3control_client.delete_job_tagging(
            AccountId=account_id,
            JobId=job_id
        )
        print(f"You have successfully deleted {job_id} tagging.")
    except ClientError as e:
        print(f"Error deleting job tags: {e}")
        raise
```



```
def cleanup_resources(self, bucket_name: str, file_names: List[str]) -> None:
    """
    Clean up all resources created during the scenario.

    Args:
        bucket_name (str): Name of the bucket to clean up
        file_names (list): List of files to delete

    Raises:
        ClientError: If cleanup fails
    """
    try:
        for file_name in file_names:
            self.s3_client.delete_object(Bucket=bucket_name, Key=file_name)
            print(f"Deleted {file_name}")

        response = self.s3_client.list_objects_v2(
            Bucket=bucket_name,
            Prefix='batch-op-reports/'
        )
        if 'Contents' in response:
            for obj in response['Contents']:
                self.s3_client.delete_object(
                    Bucket=bucket_name,
                    Key=obj['Key']
                )
                print(f"Deleted {obj['Key']}")

        self.s3_client.delete_bucket(Bucket=bucket_name)
        print(f"Deleted bucket {bucket_name}")
    except ClientError as e:
        print(f"Error in cleanup: {e}")
        raise
```

- Para ver detalhes da API, consulte os tópicos a seguir na Referência da API do SDK da AWS para Python (Boto3).
 - [CreateJob](#)
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)

- [ListJobs](#)
- [PutJobTagging](#)
- [UpdateJobPriority](#)
- [UpdateJobStatus](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Ações do Amazon S3 Control usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações específicas do Amazon S3 Control com AWS SDKs. Cada exemplo inclui um link para o GitHub, em que você pode encontrar instruções para configurar e executar o código.

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na [Referência de API do Amazon S3 Control](#).

Exemplos

- [Usar CreateJob com o AWS SDK ou a CLI](#)
- [Usar DeleteJobTagging com um SDK da AWS](#)
- [Usar DescribeJob com o AWS SDK ou a CLI](#)
- [Usar GetJobTagging com um SDK da AWS](#)
- [Usar PutJobTagging com um SDK da AWS](#)
- [Usar UpdateJobPriority com o AWS SDK ou a CLI](#)
- [Usar UpdateJobStatus com o AWS SDK ou a CLI](#)

Usar **CreateJob** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o CreateJob.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

CLI

AWS CLI

Para criar um novo trabalho de Operações em Lote do Amazon S

O exemplo `create-job` a seguir cria um trabalho de Operações em Lote do Amazon S3 para marcar 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
```

Resultado:

```
{
  "JobId": "93735294-df46-44d5-8638-6356f335324e"
}
```

- Consulte detalhes da API em [CreateJob](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie um trabalho assíncrono do S3.

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

```

Crie um trabalho de retenção de conformidade.

```

/**
 * 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();
}

```

Crie uma suspensão legal do trabalho.

```

/**
 * 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();
}

```

Crie um trabalho de retenção de governança.

```

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

- Consulte detalhes da API em [CreateJob](#) na Referência da API AWS SDK for Java 2.x.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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

```

- Consulte detalhes da API em [CreateJob](#) na Referência da API AWS SDK para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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.

```

- Consulte detalhes da API em [CreateJob](#) na Referência da API AWS SDK para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteJobTagging** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o DeleteJobTagging.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [DeleteJobTagging](#) na Referência de API do AWS SDK for Java 2.x.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [DeleteJobTagging](#) na Referência da API do SDK da AWS para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [DeleteJobTagging](#) na Referência de APIs do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DescribeJob** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o DescribeJob.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

CLI

AWS CLI

Para descrever um novo trabalho de Operações em Lote do Amazon S

O exemplo `describe-job` a seguir fornece os parâmetros de configuração e o status do trabalho de operações em lote especificado.

```
aws s3control describe-job \  
  --account-id 123456789012 \  
  --job-id 93735294-df46-44d5-8638-6356f335324e
```

Resultado:

```
{  
  "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"
}
}

```

- Consulte detalhes da API em [DescribeJob](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

/**
 * 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);
    });
}

```

- Consulte detalhes da API em [DescribeJob](#) na Referência de API do AWS SDK for Java 2.x.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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 ver detalhes da API, consulte [DescribeJob](#) na Referência de APIs do SDK da AWS para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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.
```

- Consulte detalhes da API em [DescribeJob](#) na Referência da API do AWS SDK para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetJobTagging** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o GetJobTagging.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
/**
```

```
* 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
        });
}
```

- Consulte detalhes da API em [GetJobTagging](#), na Referência de API do AWS SDK for Java 2.x.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [GetJobTagging](#) na Referência da API do SDK da AWS para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes sobre a API, consulte [GetJobTagging](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutJobTagging** com um SDK da AWS

Os exemplos de código a seguir mostram como usar o PutJobTagging.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
/**
 * 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
    });
}

```

- Consulte detalhes da API em [PutJobTagging](#) na Referência de API do AWS SDK for Java 2.x.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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 obter detalhes da API, consulte [PutJobTagging](#) na Referência da API do SDK da AWS para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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 obter detalhes da API, consulte [PutJobTagging](#) na Referência de API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **UpdateJobPriority** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o UpdateJobPriority.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

CLI

AWS CLI

Para atualizar a prioridade de um trabalho de Operações em Lote do Amazon S

O exemplo update-job-priority a seguir atualiza o trabalho especificado para uma nova prioridade.

```
aws s3control update-job-priority \
```



```
--account-id 123456789012 \  
--job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \  
--priority 52
```

Resultado:

```
{  
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",  
  "Priority": 52  
}
```

- Para ver detalhes da API, consulte [UpdateJobPriority](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
/**  
 * 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;
}

```

- Consulte detalhes da API em [UpdateJobPriority](#) na Referência de API do AWS SDK for Java 2.x.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```

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 obter detalhes da API, consulte [UpdateJobPriority](#) na Referência da API do SDK da AWS para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
TRY.
    oo_result = lo_s3c->updatejobpriority(    " oo_result is returned for
testing purposes.
    iv_accountid = iv_account_id
    iv_jobid      = iv_job_id
    iv_priority   = 60
    ).
    MESSAGE |Job { oo_result->get_jobid( ) } priority updated to { oo_result-
>get_priority( ) }| TYPE 'I'.
    CATCH /aws1/cx_s3cnotfoundexception INTO DATA(lo_ex_nf).
    MESSAGE lo_ex_nf->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_nf.
    CATCH /aws1/cx_s3cbadrequestex INTO DATA(lo_ex_bad).
    MESSAGE lo_ex_bad->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_bad.
    CATCH /aws1/cx_s3cclientexc INTO DATA(lo_ex_cli).
    MESSAGE lo_ex_cli->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_cli.
    CATCH /aws1/cx_s3cserverexc INTO DATA(lo_ex_srv).
    MESSAGE lo_ex_srv->get_text( ) TYPE 'I'.
    RAISE EXCEPTION TYPE /aws1/cx_rt_generic
    EXPORTING previous = lo_ex_srv.
```

```
ENDTRY.
```

- Para obter os detalhes da API, consulte [UpdateJobPriority](#) na Referência da API do AWS SDK para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **UpdateJobStatus** com o AWS SDK ou a CLI

Os exemplos de código a seguir mostram como usar o UpdateJobStatus.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

CLI

AWS CLI

Para atualizar o status de um trabalho de Operações em Lote do Amazon S

O exemplo update-job-status a seguir cancela o trabalho especificado que está aguardando aprovação.

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
  --requested-job-status Cancelled
```

Resultado:

```
{
  "Status": "Cancelled",
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"
}
```

O exemplo `update-job-status` a seguir confirma e executa o trabalho especificado que está aguardando aprovação.

```
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"  
}
```

O exemplo `update-job-status` a seguir cancela o trabalho especificado que em execução.

```
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 ver detalhes da API, consulte [UpdateJobStatus](#) na Referência de comandos da AWS CLI.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

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

- Consulte detalhes da API em [UpdateJobStatus](#) na Referência de API do AWS SDK for Java 2.x.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [UpdateJobStatus](#) na Referência da API do SDK da AWS para Python (Boto3).

SAP ABAP

SDK para SAP ABAP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte [UpdateJobStatus](#) na Referência da API do SDK da AWS para SAP ABAP.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Exemplos de código para buckets de diretório do S3 usando AWS SDKs

Os exemplos de código a seguir mostram como usar os buckets de diretório do S3 com um kit de desenvolvimento de software (SDK) da AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar perfis de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Para ver uma lista completa dos Guias do desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Buckets de diretório do S3

- [Exemplos básicos de buckets de diretório do S3 usando AWS SDKs](#)
 - [Olá, Amazon S3 directory buckets](#)
 - [Aprender as noções básicas dos buckets de diretório do S3 com um SDK AWS](#)

- [Ações para buckets de diretório do S3 usando AWS SDKs](#)
 - [Usar AbortMultipartUpload com um SDK da AWS](#)
 - [Usar CompleteMultipartUpload com um SDK da AWS](#)
 - [Usar CopyObject com um SDK da AWS](#)
 - [Usar CreateBucket com um SDK da AWS](#)
 - [Usar CreateMultipartUpload com um SDK da AWS](#)
 - [Usar CreateSession com um SDK da AWS](#)
 - [Usar DeleteBucket com um SDK da AWS](#)
 - [Usar DeleteBucketEncryption com um SDK da AWS](#)
 - [Usar DeleteBucketPolicy com um SDK da AWS](#)
 - [Usar DeleteObject com um SDK da AWS](#)
 - [Usar DeleteObjects com um SDK da AWS](#)
 - [Usar GetBucketEncryption com um SDK da AWS](#)
 - [Usar GetBucketPolicy com um SDK da AWS](#)
 - [Usar GetObject com um SDK da AWS](#)
 - [Usar GetObjectAttributes com um SDK da AWS](#)
 - [Usar HeadBucket com um SDK da AWS](#)
 - [Usar HeadObject com um SDK da AWS](#)
 - [Usar ListDirectoryBuckets com um SDK da AWS](#)
 - [Usar ListMultipartUploads com um SDK da AWS](#)
 - [Usar ListObjectsV2 com um SDK da AWS](#)
 - [Usar ListParts com um SDK da AWS](#)
 - [Usar PutBucketEncryption com um SDK da AWS](#)
 - [Usar PutBucketPolicy com um SDK da AWS](#)
 - [Usar PutObject com um SDK da AWS](#)
 - [Usar UploadPart com um SDK da AWS](#)
 - [Usar UploadPartCopy com um SDK da AWS](#)
- [Cenários para buckets de diretório do S3 usando AWS SDKs](#)
 - [Criar um URL pré-assinado para os buckets de diretório do Amazon S3 para obter um objeto usando um AWS SDK](#)

Exemplos básicos de buckets de diretório do S3 usando AWS SDKs

Os exemplos de código a seguir mostram como usar os elementos básicos de buckets de diretório do Amazon S3 com AWS SDKs.

Exemplos

- [Olá, Amazon S3 directory buckets](#)
- [Aprender as noções básicas dos buckets de diretório do S3 com um SDK AWS](#)
- [Ações para buckets de diretório do S3 usando AWS SDKs](#)
 - [Usar AbortMultipartUpload com um SDK da AWS](#)
 - [Usar CompleteMultipartUpload com um SDK da AWS](#)
 - [Usar CopyObject com um SDK da AWS](#)
 - [Usar CreateBucket com um SDK da AWS](#)
 - [Usar CreateMultipartUpload com um SDK da AWS](#)
 - [Usar CreateSession com um SDK da AWS](#)
 - [Usar DeleteBucket com um SDK da AWS](#)
 - [Usar DeleteBucketEncryption com um SDK da AWS](#)
 - [Usar DeleteBucketPolicy com um SDK da AWS](#)
 - [Usar DeleteObject com um SDK da AWS](#)
 - [Usar DeleteObjects com um SDK da AWS](#)
 - [Usar GetBucketEncryption com um SDK da AWS](#)
 - [Usar GetBucketPolicy com um SDK da AWS](#)
 - [Usar GetObject com um SDK da AWS](#)
 - [Usar GetObjectAttributes com um SDK da AWS](#)
 - [Usar HeadBucket com um SDK da AWS](#)
 - [Usar HeadObject com um SDK da AWS](#)
 - [Usar ListDirectoryBuckets com um SDK da AWS](#)
 - [Usar ListMultipartUploads com um SDK da AWS](#)
 - [Usar ListObjectsV2 com um SDK da AWS](#)
 - [Usar ListParts com um SDK da AWS](#)
 - [Usar PutBucketEncryption com um SDK da AWS](#)
 - [Usar PutBucketPolicy com um SDK da AWS](#)

- [Usar PutObject com um SDK da AWS](#)
- [Usar UploadPart com um SDK da AWS](#)
- [Usar UploadPartCopy com um SDK da AWS](#)

Olá, Amazon S3 directory buckets

O exemplo de código a seguir mostra como começar a usar buckets de diretório do Amazon S3.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
 - [CreateBucket](#)
 - [ListDirectoryBuckets](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Aprender as noções básicas dos buckets de diretório do S3 com um SDK AWS

Os exemplos de código a seguir mostram como:

- Configure uma VPC e um endpoint da VPC.
- Configure as políticas, os perfis e o usuário para trabalhar com os buckets de diretório do S3 e a classe de armazenamento S3 Express One Zone.
- Crie dois clientes do S3.
- Crie dois buckets.
- Crie um objeto e copie-o.
- Demonstre a diferença de desempenho.
- Preencha os buckets para mostrar a diferença lexicográfica.
- Confirme se o usuário deseja limpar os recursos.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Execute um cenário interativo que demonstre os recursos do 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 = createSessionAdd0bjects();

// 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.deleteBucketAnd0bjectsAsync(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
store
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
more
            differently, which can cause unexpected results. Let's add a few
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.");
        }
    }
}
}
}

```

Uma classe de wrapper para os métodos de SDK do 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 {
                // 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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK for Java 2.x.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

PHP

SDK para PHP

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Execute um cenário demonstrando os conceitos básicos sobre os buckets de diretório do Amazon S3 e a classe 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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para PHP.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)

- [PutObject](#)

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Execute um cenário demonstrando os conceitos básicos sobre os buckets de diretório do Amazon S3 e a classe 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_ynsno,
    ):
        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_ynsno):
            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"],
        )

```

Uma classe de wrapper para funções de SDK do 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 obter detalhes da API, consulte os tópicos a seguir na Referência da API AWS SDK para Python (Boto3).
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Ações para buckets de diretório do S3 usando AWS SDKs

Os exemplos de código a seguir demonstram como realizar ações individuais de buckets de diretório do S3 com AWS SDKs. Cada exemplo inclui um link para o GitHub, em que é possível encontrar instruções para configurar e executar o código.

Esses trechos chamam a API de buckets de diretório do S3 e são trechos de código de programas maiores que devem ser executados no contexto. É possível ver as ações em contexto em [Cenários para buckets de diretório do S3 usando AWS SDKs](#).

Os exemplos a seguir incluem apenas as ações mais utilizadas. Consulte uma lista completa na [Referência de API de buckets de diretório do Amazon S3](#).

Exemplos

- [Usar AbortMultipartUpload com um SDK da AWS](#)
- [Usar CompleteMultipartUpload com um SDK da AWS](#)
- [Usar CopyObject com um SDK da AWS](#)
- [Usar CreateBucket com um SDK da AWS](#)
- [Usar CreateMultipartUpload com um SDK da AWS](#)
- [Usar CreateSession com um SDK da AWS](#)
- [Usar DeleteBucket com um SDK da AWS](#)
- [Usar DeleteBucketEncryption com um SDK da AWS](#)
- [Usar DeleteBucketPolicy com um SDK da AWS](#)
- [Usar DeleteObject com um SDK da AWS](#)
- [Usar DeleteObjects com um SDK da AWS](#)
- [Usar GetBucketEncryption com um SDK da AWS](#)
- [Usar GetBucketPolicy com um SDK da AWS](#)
- [Usar GetObject com um SDK da AWS](#)
- [Usar GetObjectAttributes com um SDK da AWS](#)
- [Usar HeadBucket com um SDK da AWS](#)

- [Usar HeadObject com um SDK da AWS](#)
- [Usar ListDirectoryBuckets com um SDK da AWS](#)
- [Usar ListMultipartUploads com um SDK da AWS](#)
- [Usar ListObjectsV2 com um SDK da AWS](#)
- [Usar ListParts com um SDK da AWS](#)
- [Usar PutBucketEncryption com um SDK da AWS](#)
- [Usar PutBucketPolicy com um SDK da AWS](#)
- [Usar PutObject com um SDK da AWS](#)
- [Usar UploadPart com um SDK da AWS](#)
- [Usar UploadPartCopy com um SDK da AWS](#)

Usar **AbortMultipartUpload** com um SDK da AWS

O código de exemplo a seguir mostra como usar `AbortMultipartUpload`.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Anular um upload fracionado em um bucket de diretórios.

```
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 receber detalhes da API, consulte [AbortMultipartUpload](#) na Referência da API AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **CompleteMultipartUpload** com um SDK da AWS

O código de exemplo a seguir mostra como usar CompleteMultipartUpload.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Concluir um upload fracionado em um bucket de diretórios.

```
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 receber detalhes da API, consulte [CompleteMultipartUpload](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **CopyObject** com um SDK da AWS

O código de exemplo a seguir mostra como usar CopyObject.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Copie um objeto de um bucket de diretório para um bucket de diretório.

```
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;
    }
}
```

- Consulte detalhes da API em [CopyObject](#) em Referência da API AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **CreateBucket** com um SDK da AWS

O código de exemplo a seguir mostra como usar CreateBucket.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie um bucket de diretório do 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;
    }
}
```

```
}  
}
```

- Consulte detalhes da API em [CreateBucket](#) na Referência da API AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **CreateMultipartUpload** com um SDK da AWS

O código de exemplo a seguir mostra como usar `CreateMultipartUpload`.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie um upload fracionado em um bucket de diretório.

```
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 receber detalhes da API, consulte [CreateMultipartUpload](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **CreateSession** com um SDK da AWS

O código de exemplo a seguir mostra como usar `CreateSession`.

Python

SDK para Python (Boto3).

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

```
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
```

- Consulte detalhes da API em [CreateSession](#) na Referência de API do AWS SDK para Python (Boto3).

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucket** com um SDK da AWS

O código de exemplo a seguir mostra como usar DeleteBucket.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua um bucket de diretório do 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;  
    }  
}
```

- Consulte detalhes da API em [DeleteBucket](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketEncryption** com um SDK da AWS

O código de exemplo a seguir mostra como usar DeleteBucketEncryption.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua a configuração de criptografia de um bucket de diretório.

```
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;
    }
}
```

- Consulte detalhes da API em [DeleteBucketEncryption](#) na Referência de API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteBucketPolicy** com um SDK da AWS

O código de exemplo a seguir mostra como usar DeleteBucketPolicy.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua uma política de um bucket de diretório.

```
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;
    }
}
```

- Consulte detalhes da API em [DeleteBucketPolicy](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteObject** com um SDK da AWS

O código de exemplo a seguir mostra como usar DeleteObject.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua um objeto em um bucket de diretório.

```
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;
    }
}
```

- Consulte detalhes da API em [DeleteObject](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **DeleteObjects** com um SDK da AWS

O código de exemplo a seguir mostra como usar `DeleteObjects`.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Exclua vários objetos em um bucket de diretório.

```
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;
    }
}
```

- Consulte detalhes da API em [DeleteObjects](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketEncryption** com um SDK da AWS

O código de exemplo a seguir mostra como usar `GetBucketEncryption`.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha a configuração de criptografia de um bucket de diretório.

```
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 ter detalhes da API, consulte [GetBucketEncryption](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetBucketPolicy** com um SDK da AWS

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha a política de um bucket de diretório.

```
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;
    }
}
```

- Consulte detalhes da API em [GetBucketPolicy](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetObject** com um SDK da AWS

O código de exemplo a seguir mostra como usar `GetObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto nos seguintes exemplos de código:

- [Conheça os conceitos básicos](#)
- [Criar um URL pré-assinado para obter um objeto](#)

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha um objeto de um bucket de diretório.

```
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;
    }
}
```

```
}
```

- Consulte detalhes da API em [GetObject](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **GetObjectAttributes** com um SDK da AWS

O código de exemplo a seguir mostra como usar `GetObjectAttributes`.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha atributos de um objeto de um bucket de diretório.

```
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 receber detalhes da API, consulte [GetObjectAttributes](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **HeadBucket** com um SDK da AWS

O código de exemplo a seguir mostra como usar HeadBucket.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Verifica se o bucket de diretório do S3 especificado existe e está acessível.

```
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;
    }
}
```

- Consulte detalhes da API em [HeadBucket](#) na Referência da API AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **HeadObject** com um SDK da AWS

O código de exemplo a seguir mostra como usar `HeadObject`.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Obtenha metadados de um objeto em um bucket de diretório.

```
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;
}
```

- Consulte detalhes da API em [HeadObject](#) em Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListDirectoryBuckets** com um SDK da AWS

O código de exemplo a seguir mostra como usar `ListDirectoryBuckets`.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Liste todos os buckets de diretório.

```
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;
}
}
```

- Consulte detalhes da API em [ListDirectoryBuckets](#) na Referência de API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListMultipartUploads** com um SDK da AWS

O código de exemplo a seguir mostra como usar ListMultipartUploads.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Liste os uploads fracionados em um bucket de diretório.

```
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
    }
}
```

- Consulte detalhes da API em [ListMultipartUploads](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListObjectsV2** com um SDK da AWS

O código de exemplo a seguir mostra como usar ListObjectsV2.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Liste os objetos em um bucket de diretório.

```
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;
    }
}
```

- Consulte detalhes da API em [ListObjectsV2](#) na Referência da API AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **ListParts** com um SDK da AWS

O código de exemplo a seguir mostra como usar `ListParts`.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Liste as partes de um upload fracionado em um bucket de diretório.

```
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
        }
    }
}

```

- Consulte detalhes da API em [ListParts](#) na Referência de API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketEncryption** com um SDK da AWS

O código de exemplo a seguir mostra como usar `PutBucketEncryption`.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Defina a criptografia em um bucket de diretório.

```
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 ter detalhes da API, consulte [PutBucketEncryption](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutBucketPolicy** com um SDK da AWS

O código de exemplo a seguir mostra como usar PutBucketPolicy.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Aplique uma política a um bucket de diretório.

```
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 obter informações da API, consulte [PutBucketPolicy](#) na Referência da API do AWS SDK for Java 2.x.

Para obter uma lista completa dos Guias do desenvolvedor do SDK da AWS e exemplos de código, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **PutObject** com um SDK da AWS

O código de exemplo a seguir mostra como usar `PutObject`.

Exemplos de ações são trechos de código de programas maiores e devem ser executados em contexto. É possível ver essa ação em contexto no seguinte exemplo de código:

- [Conheça os conceitos básicos](#)

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Coloque um objeto em um bucket de diretório.

```
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;
    }
}
```

- Consulte detalhes da API em [PutObject](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **UploadPart** com um SDK da AWS

O código de exemplo a seguir mostra como usar UploadPart.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Faça upload de uma parte de um upload fracionado em um bucket de diretório.

```
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 receber detalhes da API, consulte [UploadPart](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Usar **UploadPartCopy** com um SDK da AWS

O código de exemplo a seguir mostra como usar UploadPartCopy.

Java

SDK para Java 2.x

 Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Crie partes de cópia com base no tamanho do objeto de origem e copie partes individuais para um bucket de diretório.

```
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;
    }
```

- Consulte detalhes da API em [UploadPartCopy](#) na Referência de API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Cenários para buckets de diretório do S3 usando AWS SDKs

Os exemplos de código a seguir mostram como implementar cenários comuns em buckets de diretório do S3 com AWS SDKs. Esses casos mostram como realizar tarefas específicas chamando várias funções em buckets de diretório do S3 ou combinadas com outros Serviços da AWS. Cada cenário inclui um link para o código-fonte completo, onde podem ser encontradas instruções sobre como configurar e executar o código.

Os cenários têm como alvo um nível intermediário de experiência para ajudar você a compreender ações de serviço em contexto.

Exemplos

- [Criar um URL pré-assinado para os buckets de diretório do Amazon S3 para obter um objeto usando um AWS SDK](#)

Criar um URL pré-assinado para os buckets de diretório do Amazon S3 para obter um objeto usando um AWS SDK

O exemplo de código a seguir mostra como criar um URL pré-assinado para buckets de diretório do S3 e obter um objeto.

Java

SDK para Java 2.x

Note

Há mais no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [AWSCode Examples Repository](#).

Gere um URL GET pré-assinado para acessar um objeto em um bucket de diretório do S3.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;

import java.nio.file.Path;
import java.time.Duration;

import static com.example.s3.util.S3DirectoryBucketUtils.createDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Client;
import static com.example.s3.util.S3DirectoryBucketUtils.createS3Presigner;
import static
    com.example.s3.util.S3DirectoryBucketUtils.deleteAllObjectsInDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.deleteDirectoryBucket;
import static com.example.s3.util.S3DirectoryBucketUtils.getFilePath;
import static
    com.example.s3.util.S3DirectoryBucketUtils.putDirectoryBucketObject;

/**
 * Generates a presigned URL for accessing an object in the specified S3
 * directory bucket.
 */
```

```
    * @param s3Presigner The S3 presigner client used to generate the presigned
URL
    * @param bucketName The name of the directory bucket
    * @param objectKey The key (name) of the object to access
    * @return A presigned URL for accessing the specified object
    */
    public static String generatePresignedGetURLForDirectoryBucket(S3Presigner
s3Presigner, String bucketName,
        String objectKey) {
        logger.info("Generating presigned URL for object: {} in bucket: {}",
objectKey, bucketName);

        try {
            // Create a GetObjectRequest
            GetObjectRequest getObjectRequest = GetObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .build();

            // Create a GetObjectPresignRequest
            GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
                .signatureDuration(Duration.ofMinutes(10)) // Presigned URL
valid for 10 minutes
                .getObjectRequest(getObjectRequest)
                .build();

            // Generate the presigned URL
            PresignedGetObjectRequest presignedGetObjectRequest =
s3Presigner.presignGetObject(getObjectPresignRequest);

            // Get the presigned URL
            String presignedURL = presignedGetObjectRequest.url().toString();
            logger.info("Presigned URL: {}", presignedURL);
            return presignedURL;

        } catch (S3Exception e) {
            logger.error("Failed to generate presigned URL: {} - Error code: {}",
e.awsErrorDetails().errorMessage(),
                e.awsErrorDetails().errorCode(), e);
            throw e;
        }
    }
}
```

- Consulte detalhes da API em [GetObject](#) na Referência da API do AWS SDK for Java 2.x.

Para ver uma lista completa dos guias de desenvolvedor e exemplos de código do SDK da AWS, consulte [Desenvolvimento com o Amazon S3 usando AWS SDKs](#). Este tópico também inclui informações sobre como começar e detalhes sobre versões anteriores do SDK.

Autenticação de solicitações (AWS Signature Version 4)

Tópicos

- [Métodos de autenticação](#)
- [Introdução à assinatura de solicitações](#)
- [Autenticação de solicitações: usando o cabeçalho de autorização \(AWSSignature Version 4\)](#)
- [Autenticação de solicitações: uso de parâmetros de consulta \(AWS Signature Version 4\)](#)
- [Exemplos: cálculos de assinatura no AWS Signature Version 4](#)
- [Autenticação de solicitações: uploads baseados no navegador usando POST \(AWS Signature Version 2\)](#)
- [Chaves de política específicas de autenticação do Amazon S3 Signature Version 4.](#)

Cada interação com o Amazon S3 é autenticada ou anônima. Esta seção explica a autenticação de solicitações com o algoritmo AWS Signature Version 4.

Note

Caso use os SDKs da AWS (consulte [Código de exemplo e bibliotecas](#)) para enviar solicitações, você não precisa ler esta seção, pois os clientes do SDK autenticam as solicitações usando as chaves de acesso fornecidas por você. Você deve usar sempre os SDKs da AWS, a menos que tenha um bom motivo para não fazer isso. Nas regiões que oferecem suporte às duas versões de assinatura, você pode solicitar que os SDKs da AWS usem uma versão de assinatura específica. Para obter mais informações, consulte [Enviar solicitações autenticadas usando os AWS SDKs](#). Leia esta seção somente se estiver implementando o algoritmo AWS Signature Version 4 em seu cliente personalizado.

A autenticação com o AWS Signature Version 4 fornece alguns ou todos os itens a seguir, dependendo de como você decidir assinar sua solicitação:

- **Verificação da identidade do solicitante:** as solicitações autenticadas exigem uma assinatura que você cria usando as chaves de acesso (ID da chave de acesso, chave de acesso secreta). Para obter informações sobre como obter chaves de acesso, consulte [Compreender e obter as](#)

[credenciais de segurança](#) na Referência geral da AWS. Se você usar credenciais de segurança temporárias, os cálculos da assinatura também exigirão um token de segurança. Para obter mais informações, consulte [Solicitação de credenciais de segurança temporárias](#) no Manual do usuário do IAM.

- **Proteção de dados em trânsito:** para evitar a adulteração de uma solicitação enquanto ela está em trânsito, você usa alguns dos elementos da solicitação para calcular a assinatura. Ao receber a solicitação, o Amazon S3 calcula a assinatura usando os mesmos elementos da solicitação. Se algum componente de solicitação recebido pelo Amazon S3 não corresponder ao componente usado para calcular a assinatura, o Amazon S3 rejeitará a solicitação.
- **Proteção contra a reutilização das partes assinadas da solicitação:** as partes assinadas (usando AWS Signatures) das solicitações são válidas dentro de 15 minutos a partir da data e da hora da solicitação. Uma parte não autorizada que tenha acesso a uma solicitação assinada pode modificar as partes não assinadas da solicitação sem afetar a validade da solicitação no período de 15 minutos. Por isso, recomendamos que você maximize a proteção assinando os cabeçalhos e o corpo da solicitação, fazendo solicitações HTTPS para o Amazon S3 e usando a chave de condição `s3:x-amz-content-sha256` (consulte [Chaves de política específicas de autenticação do Amazon S3 Signature Version 4.](#)) nas políticas da AWS para exigir que os usuários assinem corpos de solicitação do Amazon S3.

Note

O Amazon S3 é compatível com o Signature Version 4, um protocolo para autenticação de solicitações de API de entrada para produtos da AWS, em todas as Regiões da AWS. Neste momento, as Regiões da AWS criadas antes de 30 de janeiro de 2014 continuarão oferecendo suporte ao protocolo anterior, Signature Version 2. Todas as regiões novas a partir de 30 de janeiro de 2014 oferecerão suporte apenas ao Signature versão 4. Portanto, todas as solicitações para essas regiões devem ser feitas com o Signature versão 4. Para obter mais informações sobre o AWS Signature Version 2, consulte [Assinatura e autenticação de solicitações REST](#) no Guia do usuário do Amazon Simple Storage Service.

Métodos de autenticação

É possível expressar informações de autenticação usando um dos métodos a seguir:

- **Cabeçalho HTTP Authorization:** usar o cabeçalho HTTP Authorization é o método mais comum para autenticar uma solicitação do Amazon S3. Todas as operações REST do Amazon S3 (exceto para carregamentos baseados em navegador que usam solicitações POST) requerem esse cabeçalho. Para obter mais informações sobre o valor do cabeçalho Authorization e como calcular a assinatura e as opções relacionadas, consulte [Autenticação de solicitações: usando o cabeçalho de autorização \(AWSSignature Version 4\)](#).
- **Parâmetros da string de consulta:** é possível usar uma string de consulta para expressar uma solicitação completa em um só URL. Nesse caso, utilize parâmetros de consulta para fornecer informações de solicitação, inclusive as informações de autenticação. Como a solicitação de assinatura faz parte do URL, esse tipo de URL muitas vezes é chamado de URL pré-assinado. É possível usar URLs pré-assinados para incorporar links clicáveis em HTML, que podem ser válidos por até sete dias. Para obter mais informações, consulte [Autenticação de solicitações: uso de parâmetros de consulta \(AWS Signature Version 4\)](#).

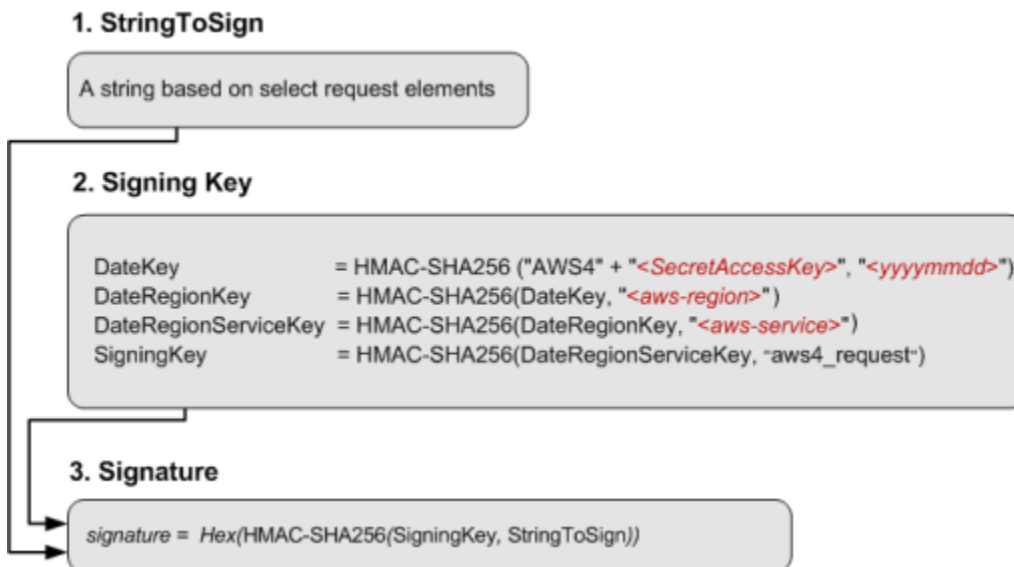
O Amazon S3 também oferece suporte a uploads baseados no navegador que usam solicitações HTTP POST. Com uma solicitação HTTP POST, você pode fazer upload de conteúdo para o Amazon S3 diretamente do navegador. Para obter informações sobre autenticação de solicitações POST, consulte [Uploads baseados no navegador usando POST \(AWS Signature Version 4\)](#).

Introdução à assinatura de solicitações

As informações de autenticação enviadas em uma solicitação devem incluir uma assinatura. Para calcular uma assinatura, primeiro concatene os elementos de solicitação selecionados para formar uma string, chamada de string a ser assinada. Em seguida, use uma chave de assinatura para calcular o código de autenticação de mensagens por hash (HMAC) da string para assinar.

No AWS Signature Version 4, você não usa sua chave de acesso secreta apenas para assinar a solicitação. Em vez disso, use primeiro sua chave de acesso secreta para derivar uma chave de assinatura. A chave de assinatura derivada é específica para data, serviço e região. Para obter mais informações sobre como derivar uma chave de assinatura em linguagens de programação diferentes, consulte [Exemplos de como derivar uma chave de assinatura para o Signature Version 4](#).

O diagrama a seguir ilustra o processo geral de computação de uma assinatura.



A string para assinar depende do tipo de solicitação. Por exemplo, ao usar o cabeçalho HTTP Authorization ou os parâmetros de consulta para autenticação, utilize uma combinação variável de elementos de solicitação para criar a string a ser assinada. Para uma solicitação HTTP POST, a política de POST na solicitação é a string que você assina. Para obter mais informações sobre o cálculo da string a ser assinada, siga os links fornecidos no fim desta seção.

Para chave de assinatura, o diagrama mostra uma série de cálculos, em que o resultado de cada etapa é inserido na etapa seguinte. A etapa final é a chave de assinatura.

Ao receber uma solicitação autenticada, os servidores do Amazon S3 recriam a assinatura usando as informações de autenticação contidas na solicitação. Se as assinaturas coincidirem, o Amazon S3 processará sua solicitação; caso contrário, a solicitação será rejeitada.

Para obter mais informações sobre autenticação de solicitações, consulte os tópicos abaixo:

- [Autenticação de solicitações: usando o cabeçalho de autorização \(AWSSignature Version 4\)](#)
- [Autenticação de solicitações: uso de parâmetros de consulta \(AWS Signature Version 4\)](#)
- [Uploads baseados no navegador usando POST \(AWS Signature Version 4\)](#)

Autenticação de solicitações: usando o cabeçalho de autorização (AWSSignature Version 4)

Tópicos

- [Visão geral](#)

- [Cálculos de assinatura para o cabeçalho de autorização: transferência de carga útil em uma única parte \(AWS Signature Version 4\)](#)
- [Cálculos de assinatura para o cabeçalho de autorização: transferência de carga útil em vários fragmentos \(upload fragmentado\) \(AWS Signature Version 4\)](#)
- [Cálculos de assinatura para cabeçalhos finais \(uploads fragmentados\) \(AWS Signature Version 4\)](#)

Visão geral

Usar o cabeçalho HTTP Authorization é o método mais usado para fornecer informações de autenticação. Exceto para [solicitações POST](#) e solicitações assinadas usando parâmetros de consulta, todas as operações do Amazon S3 usam o cabeçalho de solicitação Authorization para fornecer informações de autenticação.

Veja a seguir um exemplo de um valor de cabeçalho Authorization. As quebras de linha foram adicionadas a este exemplo somente para facilitar a leitura:

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

A tabela a seguir descreve os vários componentes do valor do cabeçalho Authorization do exemplo anterior:

Componente	Descrição
AWS4-HMAC-SHA256	O algoritmo que foi usado para calcular a assinatura. É necessário fornecer esse valor ao usar o AWS Signature Version 4 para autenticação. A string especifica o AWS Signature Version 4 (AWS4) e o algoritmo de assinatura (HMAC-SHA256).
Credential	

Componente	Descrição
	Seu ID de chave de acesso e as informações do escopo, como a data, a região e o serviço que foram usados para calcular a assinatura. Essa string tem a seguinte forma: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Em que: • O valor de <i><data></i> é especificado usando o formato YYYYMMDD. • O valor de <i><serviço da aws></i> é s3 ao enviar uma solicitação ao Amazon S3.
SignedHeaders	Uma lista separada por ponto e vírgula dos cabeçalhos de solicitação usados para computar a Signature . A lista contém somente nomes de cabeçalho, e os nomes dos cabeçalhos devem estar em letras minúsculas. Por exemplo: <pre>host;range;x-amz-date</pre>

Componente	Descrição
Signature	A assinatura de 256 bits expressa como 64 caracteres hexadecimais minúsculos. Por exemplo: <pre>fe5f80f77d5fa3beca038a248ff027d0445342fe2855d dc963176630326f1024</pre> Os cálculos da assinatura variam de acordo com a opção escolhida para transferência de carga útil.

Os cálculos da assinatura variam de acordo com o método escolhido para transferência da carga útil da solicitação. O S3 oferece suporte às seguintes opções:

- Transferir carga útil em um único bloco: nesse caso, você tem as seguintes opções de cálculo de assinatura:
 - Opção de carga útil assinada: se desejar, você pode calcular toda a soma de verificação da carga útil e incluí-la no cálculo da assinatura. Isso fornece segurança adicional, mas você precisa ler sua carga útil duas vezes ou armazená-la na memória.

Por exemplo, para fazer upload de um arquivo, você precisa primeiro ler o arquivo para calcular um hash de carga útil para o cálculo da assinatura e novamente para transmissão ao criar a solicitação. Para cargas úteis menores, essa abordagem pode ser preferível. No entanto, para arquivos grandes, ler o arquivo duas vezes pode ser ineficaz, então talvez você queira carregar os dados em partes.

Recomendamos que você inclua a soma de verificação da carga útil para aumentar a segurança.

- Opção de carga útil não assinada: não inclua a soma de verificação da carga útil no cálculo da assinatura.

Para obter instruções passo a passo sobre como calcular a assinatura e criar o valor do cabeçalho de autorização, consulte [Cálculos de assinatura para o cabeçalho de autorização: transferência de carga útil em uma única parte \(AWS Signature Version 4\)](#).

- Transferir carga útil em vários blocos (upload em partes): nesse caso, você transfere a carga útil em partes. Você pode transferir uma carga útil em partes, independentemente do tamanho.

Você pode dividir sua carga útil em partes. Podem ser partes de tamanho fixo ou variável. Ao carregar dados em partes, você evita ler toda a carga útil para calcular a assinatura. Em vez disso, para a primeira parte, você calcula uma assinatura inicial que usa somente os cabeçalhos da solicitação. A segunda parte contém a assinatura da primeira, e cada parte subsequente contém a assinatura da parte anterior. Ao final do upload, você envia uma parte final com 0 bytes de dados que contém a assinatura da última parte da carga útil. Para obter mais informações, consulte [Cálculos de assinatura para o cabeçalho de autorização: transferência de carga útil em vários fragmentos \(upload fragmentado\) \(AWS Signature Version 4\)](#).

Ao assinar suas solicitações, você pode usar o AWS Signature Version 4 ou o AWS Signature Version 4A. A principal diferença entre os dois está na forma como a assinatura é calculada. Com o AWS Signature Version 4A, a assinatura não inclui informações específicas da região e é calculada usando o algoritmo AWS4-ECDsa-P256-SHA256.

Além dessas opções, você pode incluir um finalizador com sua solicitação. Para incluir um finalizador em sua solicitação, você precisa especificar isso no cabeçalho definindo `x-amz-content-sha256` como o valor apropriado. Se você estiver usando um cabeçalho à direita, deverá incluir `x-amz-trailer` no cabeçalho e especificar os nomes dos cabeçalhos à direita como uma string em uma lista separada por vírgulas. Todos os cabeçalhos finais são gravados após a parte final. Se você estiver carregando os dados em várias partes, deverá enviar uma parte final com 0 bytes de dados antes de enviar o cabeçalho final.

Ao enviar uma solicitação, você deve informar ao Amazon S3 qual das opções anteriores foi escolhida no cálculo da assinatura, adicionando o cabeçalho `x-amz-content-sha256` com um dos seguintes valores:

Valor de cabeçalho	Descrição
Valor real da soma de verificação da carga útil	Esse valor é a soma de verificação real do objeto e só é possível quando você está carregando os dados em uma única parte.
UNSIGNED-PAYLOAD	Use isso ao fazer upload do objeto como uma única parte não assinada.

Valor de cabeçalho	Descrição
STREAMING-UNSIGNED-PAYLOAD-TRAILER	Use isso ao enviar uma carga útil não assinada em várias partes. Nesse caso, você também tem um cabeçalho final após o upload da parte.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD	Use isso ao enviar uma carga útil em várias partes, e as partes são assinadas usando AWS4-HMAC-SHA256 . Isso produz uma assinatura SigV4.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER	Use isso ao enviar uma carga útil em várias partes, e as partes são assinadas usando AWS4-HMAC-SHA256 . Isso produz uma assinatura SigV4. Além disso, o resumo das partes é incluído como cabeçalho final.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD	Use isso ao enviar uma carga útil em várias partes, e as partes são assinadas usando AWS4-ECDSA-P256-SHA256 . Isso produz uma assinatura SigV4A.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD-TRAILER	Use isso ao enviar uma carga útil em várias partes, e as partes são assinadas usando AWS4-ECDSA-P256-SHA256 . Isso produz uma assinatura SigV4A. Além disso, o resumo das partes é incluído como cabeçalho final.

Ao receber a solicitação, o Amazon S3 recria a string a ser assinada usando as informações no cabeçalho `Authorization` e no cabeçalho `date`. Em seguida, ele verifica com o serviço de autenticação se as assinaturas coincidem. A data da solicitação pode ser especificada usando o cabeçalho `HTTP Date` ou o cabeçalho `x-amz-date`. Se os dois cabeçalhos estiverem presentes, `x-amz-date` terá precedência.

Se as assinaturas coincidirem, o Amazon S3 processará sua solicitação; caso contrário, a solicitação falhará.

Para saber mais, consulte os seguintes tópicos:

[Cálculos de assinatura para o cabeçalho de autorização: transferência de carga útil em uma única parte \(AWS Signature Version 4\)](#)

[Cálculos de assinatura para o cabeçalho de autorização: transferência de carga útil em vários fragmentos \(upload fragmentado\) \(AWS Signature Version 4\)](#)

[Cálculos de assinatura para cabeçalhos finais \(uploads fragmentados\) \(AWS Signature Version 4\)](#)

Cálculos de assinatura para o cabeçalho de autorização: transferência de carga útil em uma única parte (AWS Signature Version 4)

Ao usar o cabeçalho `Authorization` para autenticar solicitações, o valor do cabeçalho inclui, entre outras coisas, uma assinatura. Os cálculos da assinatura variam de acordo com a escolha de transferência da carga útil ([Visão geral](#)). Esta seção explica cálculos de assinatura quando você decide transferir a carga útil em uma única parte. A seção de exemplo (consulte [Exemplos: cálculos de assinatura](#)) mostra cálculos de assinatura e cabeçalhos `Authorization` resultantes que você pode usar como um conjunto de testes para verificar seu código.

Important

Ao transferir a carga útil em uma única parte, você pode optar por incluir o hash da carga útil nos cálculos da assinatura, chamado de carga útil assinada (se você não a incluir, a carga útil será considerada não assinada). O procedimento de assinatura discutido na seção a seguir se aplica a ambos, mas observe as seguintes diferenças:

- Opção de carga útil assinada: você inclui o hash da carga útil ao criar a solicitação canônica (que então se torna parte de `StringToSign`, conforme explicado na seção de cálculo da assinatura). Você também especifica o mesmo valor como o valor do cabeçalho `x-amz-content-sha256` ao enviar a solicitação para o S3.
- Opção de carga útil não assinada: inclua a string literal `UNSIGNED-PAYLOAD` ao construir uma solicitação canônica e defina o mesmo valor como o cabeçalho `x-amz-content-sha256` ao enviar a solicitação para o Amazon S3.

Quando você envia a solicitação para o Amazon S3, o valor do cabeçalho `x-amz-content-sha256` informa ao Amazon S3 se a carga útil está assinada ou não. O Amazon S3 pode então criar a assinatura adequadamente para verificação.

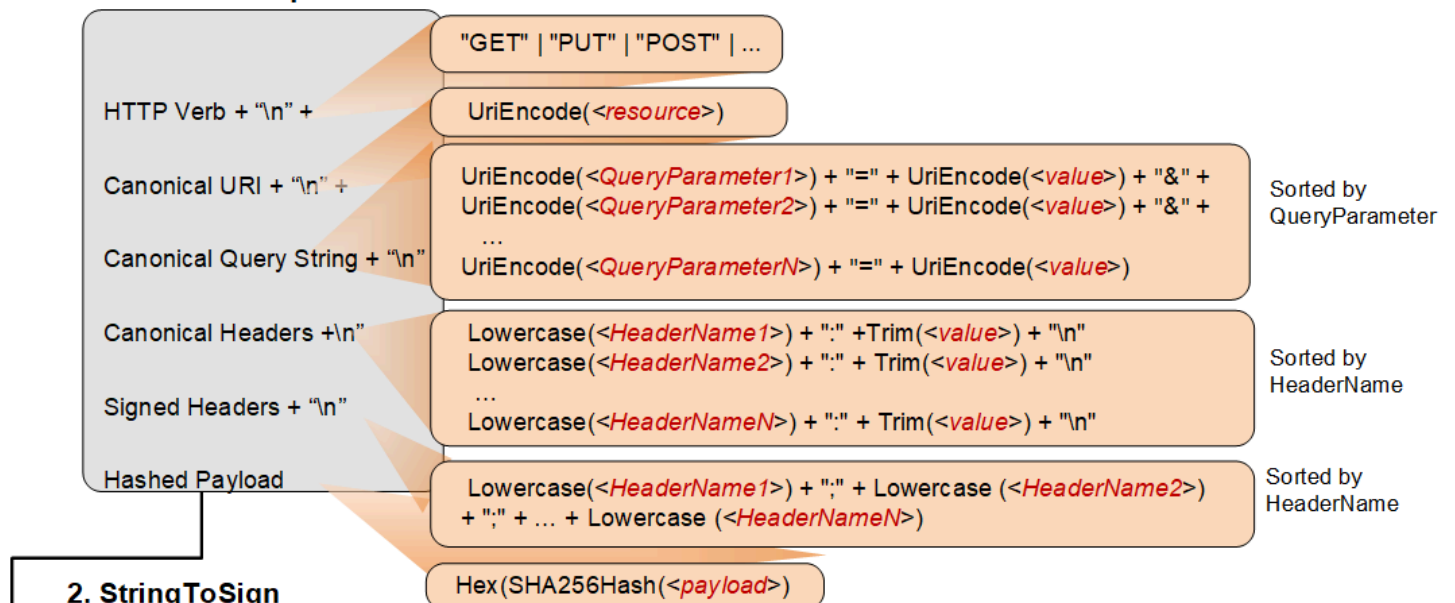
Em ambos os casos, como o valor do cabeçalho `x-amz-content-sha256` já faz parte da `HashedPayload`, você não precisa incluir o cabeçalho `x-amz-content-sha256` como cabeçalho canônico.

Cálculo de uma assinatura

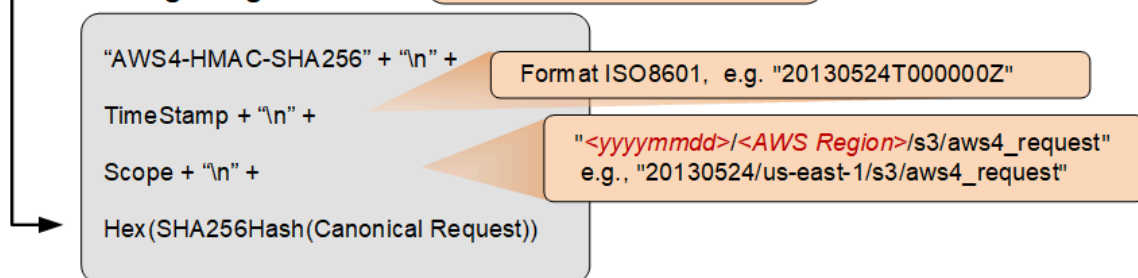
Para calcular uma assinatura, primeiro é necessário usar uma string para assinar. Em seguida, calcule um hash HMAC-SHA256 da string a ser assinada usando uma chave de assinatura. O diagrama a seguir ilustra o processo, incluindo os vários componentes da string criada para assinatura

Ao receber uma solicitação autenticada, o Amazon S3 calcula a assinatura e a compara com a assinatura que você forneceu na solicitação. Por esse motivo, é necessário calcular a assinatura usando o mesmo método usado pelo Amazon S3. O processo de colocar uma solicitação em um formulário estabelecido para assinatura se chama canonização.

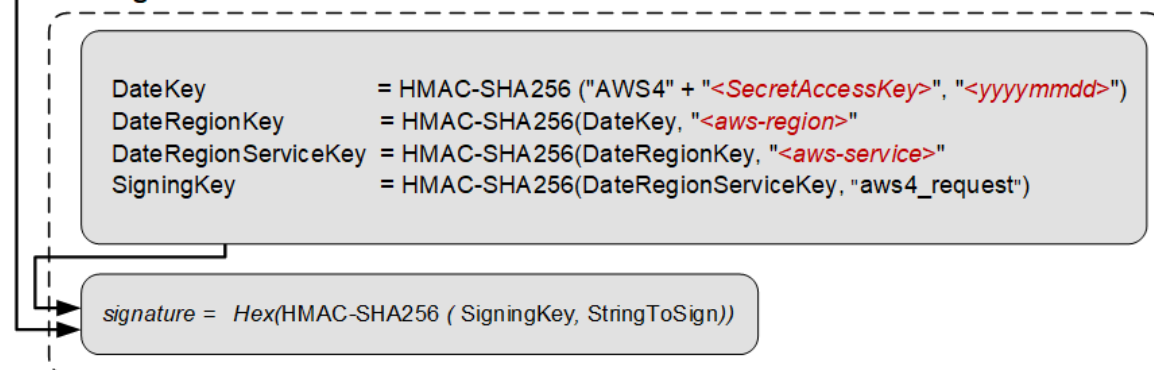
1. Canonical Request



2. StringToSign



3. Signature



A tabela a seguir descreve as funções exibidas no diagrama. É necessário implementar o código para essas funções.

Função	Descrição
Lowercase()	Converte a string em letras minúsculas.

Função	Descrição
Hex()	Codificação de base 16 em letras minúsculas.
SHA256Hash()	Função de hash criptográfico do Secure Hash Algorithm (SHA).
HMAC-SHA256()	Calcula o HMAC usando o algoritmo SHA256 com a chave de assinatura fornecida. Essa é a assinatura final.
Trim()	Remova qualquer espaço em branco inicial e final.

Função	Descrição
<code>UriEncode()</code>	O URI codifica cada byte. O <code>UriEncode()</code> deve aplicar as seguintes regras: • O URI codifica cada byte, exceto os caracteres não reservados: “A”-“Z”, “a”-“z”, “0”-“9”, “-”, “.”, “_” e “~”. • O caractere de espaço é um caractere reservado e deve ser codificado como “%20” (e não como “+”). • Cada byte codificado por URI é formado por um “%” e o valor hexadecimal de dois dígitos do byte. • As letras no valor hexadecimal devem estar em maiúsculas; por exemplo, “%1A”. • Codifique o caractere de barra, “/”, em todos os lugares, exceto no nome da chave do objeto. Por exemplo, se o nome da chave do objeto for <code>photos/Jan/sample.jpg</code>, a barra no nome da chave não está codificada.
	 Important As funções <code>UriEncode</code> padrão fornecidas por sua plataforma de desenvolvimento podem não funcionar devido às diferenças na implementação e à ambiguidade relacionada nos RFCs subjacentes. É recomendável escrever sua própria função <code>UriEncode</code> personalizada para garantir que a codificação funcione.
	Veja a seguir um exemplo da função <code>UriEncode()</code> em Java. <pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBu ilder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i);</pre>

Função	Descrição
	<pre> if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Tarefa 1: Criar uma solicitação canônica

Esta seção fornece uma visão geral da criação de uma solicitação canônica.

Confira a seguir o formato de solicitação canônica que o Amazon S3 usa para calcular uma assinatura. Para que as assinaturas coincidam, você deve criar uma solicitação canônica neste formato:

```

<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>

```

Em que:

- **HTTPMethod** é um dos métodos HTTP, por exemplo, GET, PUT, HEAD e DELETE.
- **CanonicalUri** é a versão codificada por URI do componente de caminho absoluto do URI, começando pelo "/" após o nome do domínio e seguindo até o final da string ou até o caractere de ponto de interrogação ("?"), se houver parâmetros de string de consulta. O URI no seguinte exemplo, /examplebucket/myphoto.jpg, é o caminho absoluto, e você não codifica o "/" no caminho absoluto:

```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

 Note

Não normalize caminhos de URI para solicitações ao Amazon S3. Por exemplo, você pode ter um bucket com um objeto chamado "my-object//example//photo.user". A normalização do caminho altera o nome do objeto na solicitação para "my-object/example/photo.user". Esse é um caminho incorreto para esse objeto.

- **CanonicalQueryString** especifica os parâmetros da string de consulta codificados por URI. Codifique por URI o nome e os valores individualmente. Também é necessário classificar os parâmetros na string de consulta canônica em ordem alfabética pelo nome da chave. A classificação ocorre após a codificação. A string de consulta no seguinte exemplo de URI é `prefix=somePrefix&marker=someMarker&max-keys=20`:

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

A string de consulta canônica é como este exemplo (quebras de linha foram adicionadas ao exemplo para facilitar a leitura):

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&" +  
UriEncode("max-keys")+"="+UriEncode("20") + "&" +  
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

Quando uma solicitação se destina a um sub-recurso, o valor do parâmetro de consulta correspondente é uma string vazia (""). Por exemplo, o URI a seguir identifica o sub-recurso ACL no bucket examplebucket:

```
http://s3.amazonaws.com/examplebucket?acl
```

A CanonicalQueryString nesse caso é a seguinte:

```
UriEncode("acl") + "=" + ""
```

Se o URI não contém um “?”, não há strings de consulta na solicitação, e você define a string de consulta canônica como uma string vazia (“”). Você ainda precisará incluir o “\n”.

- *CanonicalHeaders* é uma lista de cabeçalhos de solicitação com os respectivos valores. Os pares individuais de nome e valor do cabeçalho são separados pelo caractere de nova linha (“\n”). Os nomes de cabeçalho devem estar em letras minúsculas. É necessário classificar esses nomes em ordem alfabética para criar a string, conforme mostrado no exemplo a seguir:

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"  
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"  
...  
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

As funções Lowercase() e Trim() usadas neste exemplo estão descritas na seção anterior.

A lista *CanonicalHeaders* deve conter:

- Cabeçalho HTTP host
- Se o cabeçalho Content-MD5 estiver presente na solicitação, você deverá adicioná-lo à lista *CanonicalHeaders*.
- Também deverão ser adicionados todos os cabeçalhos x-amz-* que você pretende incluir na solicitação. Por exemplo, se você estiver usando credenciais de segurança temporárias, precisará incluir o x-amz-security-token na solicitação. É necessário adicionar esse cabeçalho à lista de *CanonicalHeaders*.

Note

- O cabeçalho x-amz-content-sha256 é obrigatório para todas as solicitações do AWS Signature Version 4. Ele fornece um hash da carga da solicitação. No entanto, você não precisa incluir o x-amz-content-sha256 como cabeçalho canônico porque o S3 usará automaticamente seu valor ao calcular o hash de carga útil enviado na solicitação.
- Se não houver carga útil, será necessário fornecer o hash de uma string vazia.
- Se você não quiser que o S3 verifique o hash da solicitação, use a string literal "UNSIGNED-PAYLOAD".

Veja a seguir um exemplo de strings `CanonicalHeaders`. O nomes dos cabeçalhos estão em letras minúsculas e são classificados.

Example Exemplo 1

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

Example Exemplo 2

```
host:s3.amazonaws.com
x-amz-content-sha256:UNSIGNED-PAYLOAD
x-amz-date:20130708T220855Z
```

Example Exemplo 3

```
host:s3.amazonaws.com
x-amz-date:20130708T220855Z
```

Note

Para fins de cálculo de uma assinatura de autorização, o cabeçalho do host e todos os cabeçalhos `x-amz-*`, exceto `x-amz-content-sha256`, são necessários. No entanto, para evitar a adulteração de dados, convém incluir todos os cabeçalhos no cálculo da assinatura.

Assinar o cabeçalho `x-amz-content-sha256` é opcional porque o S3 usará seu valor ao calcular o hash da carga útil da solicitação recebida.

- ***SignedHeaders*** é uma lista em ordem alfabética, separada por ponto e vírgula, de nomes de cabeçalhos de solicitação em letras minúsculas. Os cabeçalhos de solicitação da lista são os mesmos cabeçalhos que você incluiu na string `CanonicalHeaders`. Nos exemplos anteriores, o valor de ***SignedHeaders*** seria:

Example Exemplos 1 e 2

```
host;x-amz-content-sha256;x-amz-date
```

Example Exemplo 3

```
host;x-amz-date
```

- *HashedPayload* é o valor hexadecimal do hash SHA256 da carga útil da solicitação.

```
Hex(SHA256Hash(<payload>))
```

Se não houver carga útil na solicitação, calcule um hash da string vazia da seguinte forma:

```
Hex(SHA256Hash(""))
```

O hash retorna o seguinte valor:

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Por exemplo, ao carregar um objeto usando uma solicitação PUT, forneça dados do objeto no corpo. Ao recuperar um objeto usando uma solicitação GET, calcule o hash da string vazia.

Tarefa 2: Criar uma string para assinar

Esta seção fornece uma visão geral da criação de uma string a ser assinada. Para obter instruções passo a passo, consulte [Tarefa 2: criar uma string a ser assinada](#) na Referência geral da AWS.

A string a ser assinada é uma concatenação destas strings:

```
"AWS4-HMAC-SHA256" + "\n" +  
timestampISO8601Format + "\n" +  
<Scope> + "\n" +  
Hex(SHA256Hash(<CanonicalRequest>))
```

A string constante AWS4-HMAC-SHA256 especifica o algoritmo de hash que você está usando, HMAC-SHA256. `timeStamp` é a hora UTC atual no formato ISO 8601 (por exemplo, 20130524T000000Z).

Scope vincula a assinatura resultante a uma data específica, uma Região da AWS e um serviço. Assim, sua assinatura resultante funcionará somente na região específica e para um serviço específico. A assinatura é válida por sete dias após a data especificada.

```
date.Format(<YYYYMMDD>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

Para o Amazon S3, a string de serviço é `s3`. Para obter uma lista de strings de *região*, consulte [Regiões e endpoints](#) na Referência geral da AWS. A coluna Região nesta tabela fornece a lista de strings de região válidas.

O escopo a seguir restringe a assinatura resultante à região `us-east-1` e ao Amazon S3.

```
20130606/us-east-1/s3/aws4_request
```

Note

Scope deve usar a mesma data que você usa para calcular a chave de assinatura, conforme discutido na seção a seguir.

Tarefa 3: calcular a assinatura

No AWS Signature Version 4, em vez de usar suas chaves de acesso da AWS para assinar uma solicitação, primeiro crie uma chave de assinatura que tenha como escopo uma região e um serviço específicos. Para obter mais informações sobre chaves de assinatura, consulte [Introdução à assinatura de solicitações](#).

```
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

Alguns casos de uso podem processar chaves de assinatura por até sete dias. Para obter mais informações, consulte [Compartilhar um objeto com outras pessoas](#).

Para obter uma lista de strings de região, consulte [Regiões e endpoints](#) na Referência geral da AWS.

O uso de uma chave de assinatura permite que você mantenha suas credenciais da AWS em um local seguro. Por exemplo, se você tiver vários servidores que se comunicam com o Amazon S3, compartilhe a chave de assinatura com esses servidores; você não precisa manter uma cópia da sua chave de acesso secreta em cada servidor. A chave de assinatura é válida por até sete dias. Portanto, toda vez que você calcular a chave de assinatura, precisará compartilhá-la com os servidores. Para obter mais informações, consulte [Autenticação de solicitações \(AWS Signature Version 4\)](#).

A assinatura final é o hash HMAC-SHA256 da string a ser assinada, usando a chave de assinatura como chave.

```
HMAC-SHA256(SigningKey, StringToSign)
```

Para obter instruções passo a passo sobre como criar uma assinatura, consulte [Tarefa 3: criar uma assinatura](#) na Referência geral da AWS.

Exemplos: cálculos de assinatura

Você pode usar os exemplos desta seção como uma referência para verificar os cálculos de assinatura no seu código. Os cálculos mostrados nos exemplos usam os seguintes dados:

- Exemplo de chaves de acesso.

Parâmetro	Valor
AWSSecretAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- Solicite o carimbo de data/hora de 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT).
- Nome do bucket: examplebucket.
- Presume-se que o bucket esteja na região Leste dos EUA (Norte da Virgínia). A credencial Scope e os cálculos de Signing Key usam us-east-1 como especificador de região. Para obter informações sobre outras regiões, consulte [Regiões e endpoints](#) na Referência geral da AWS.
- Você pode usar solicitações no estilo de caminho ou no estilo hospedado virtual. Os exemplos a seguir mostram como assinar uma solicitação no estilo hospedado virtual, por exemplo:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Para obter mais informações, consulte [Hospedagem virtual de buckets](#) no Guia do usuário do Amazon Simple Storage Service.

Exemplo: objeto GET

O exemplo a seguir obtém os primeiros 10 bytes de um objeto (test.txt) de examplebucket. Para obter mais informações sobre a ação da 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 essa solicitação GET não fornece nenhum conteúdo no corpo, o valor x-amz-content-sha256 pode ser o hash do corpo vazio da solicitação ou a string literal "UNSIGNED-PAYLOAD". As etapas a seguir mostram os cálculos de assinatura e a construção do cabeçalho Authorization usando o hash de uma string vazia.

1. StringToSign

a. CanonicalRequest

```
GET
/test.txt
```

```
host:examplebucket.s3.amazonaws.com
range:bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z
```

```
host;range;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

Na string de solicitação canônica, a última linha é o hash do corpo da solicitação vazio. A terceira linha está vazia porque não há parâmetros de consulta na solicitação.

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. Cabeçalho de autorização

O cabeçalho Authorization resultante é o seguinte:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;range;x-amz-content-sha256;x-amz-
date,Signature=f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

Exemplo: objeto PUT

Este exemplo de solicitação PUT cria um objeto (test\$file.text) em examplebucket. O exemplo supõe o seguinte:

- Você está solicitando REDUCED_REDUNDANCY como classe de armazenamento adicionando o cabeçalho de solicitação x-amz-storage-class. Para obter informações sobre classes de armazenamento, consulte [Classes de armazenamento](#) no Guia do usuário do Amazon Simple Storage Service.
- O conteúdo do arquivo enviado é uma string, "Welcome to Amazon S3." O valor de x-amz-content-sha256 na solicitação é baseado nessa string.

Para obter informações sobre a ação da 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>
```

As etapas a seguir mostram cálculos de assinatura.

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
```

Na solicitação canônica, a terceira linha está vazia, pois não há parâmetros de consulta na solicitação. O cabeçalho canônico `x-amz-content-sha256` pode ser assinado opcionalmente, pois o hash de carga útil já é fornecido na parte inferior da solicitação. A última linha é o hash do corpo, que deve ser igual ao valor `x-amz-content-sha256` header enviado ao S3 na solicitação 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. Cabeçalho de autorização

O cabeçalho `Authorization` resultante é o seguinte:

```
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
```

Exemplo: ciclo de vida do bucket GET

A solicitação GET a seguir recupera a configuração de ciclo de vida de `examplebucket`. Para obter informações sobre a ação da API, consulte [GetBucketLifecycleConfiguration](#).

```
GET ?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256: e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

Como a solicitação não fornece nenhum conteúdo no corpo, o valor do cabeçalho `x-amz-content-sha256` é o hash do corpo vazio da solicitação. As etapas a seguir mostram cálculos de assinatura.

1. StringToSign

a. CanonicalRequest

```
GET
/
lifecycle=
host:examplebucket.s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

Na solicitação canônica, a última linha é o hash do corpo da solicitação vazio.

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. Cabeçalho de autorização

O cabeçalho Authorization resultante é o seguinte:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/  
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-  
date,Signature=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

Exemplo: obter bucket (listar objetos)

O exemplo a seguir recupera uma lista de objetos do bucket `examplebucket`. Para obter informações sobre a ação da 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: e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

Como a solicitação não fornece um corpo, o valor de `x-amz-content-sha256` é o hash do corpo da solicitação vazio. As etapas a seguir mostram cálculos de assinatura.

1. StringToSign

a. CanonicalRequest

```
GET  
/
```

```
max-keys=2&prefix=J
host:examplebucket.s3.amazonaws.com
x-amz-content-
sha256:e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

Na string canônica, a última linha é o hash do corpo da solicitação vazio.

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. Cabeçalho de autorização

O cabeçalho Authorization resultante é o seguinte:

```
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 assinatura para o cabeçalho de autorização: transferência de carga útil em vários fragmentos (upload fragmentado) (AWS Signature Version 4)

Conforme descrito na [Visão geral](#), ao autenticar solicitações usando o cabeçalho Authorization, você tem a opção de fazer upload da carga útil em fragmentos. Você pode enviar dados em fragmentos de tamanho fixo ou variável. Esta seção descreve o processo de cálculo da assinatura no upload fragmentado, como você cria o corpo do fragmento e como a assinatura atrasada funciona quando você faz upload do fragmento pela primeira vez e envia a assinatura no fragmento subsequente. A seção de exemplo (consulte [Exemplo: objeto PUT](#)) mostra cálculos de assinatura e cabeçalhos Authorization resultantes que você pode usar como um conjunto de testes para verificar seu código.

Note

Ao transferir dados em uma série de fragmentos, você deve fazer o seguinte:

- Especifique explicitamente o tamanho total do conteúdo (tamanho do objeto em bytes mais metadados em cada parte) usando o cabeçalho HTTP Content-Length. Para fazer isso, você deve pré-calcular o tamanho total da carga útil, incluindo os metadados enviados em cada parte, antes de iniciar sua solicitação.
- Especifique o cabeçalho HTTP Transfer-Encoding. Se você incluir o cabeçalho Transfer-Encoding e especificar qualquer valor diferente de identity, deverá omitir o cabeçalho Content-Length.

Para todas as solicitações, você deve incluir o cabeçalho x-amz-decoded-content-length, especificando o tamanho do objeto em bytes.

Cada cálculo de assinatura de fragmento inclui a assinatura do fragmento anterior. Para começar, você cria uma assinatura inicial usando somente os cabeçalhos. Você usa a assinatura inicial no cálculo da assinatura do primeiro fragmento. Para cada fragmento subsequente, você cria uma assinatura que inclui a assinatura do fragmento anterior. Assim, as assinaturas de fragmento são encadeadas; ou seja, a assinatura do fragmento n é uma função $F(\text{fragmento } n, \text{assinatura}(\text{fragmento } n-1))$. O encadeamento garante que você envie os fragmentos na ordem correta.

Para realizar um upload fragmentado, realize estas etapas:

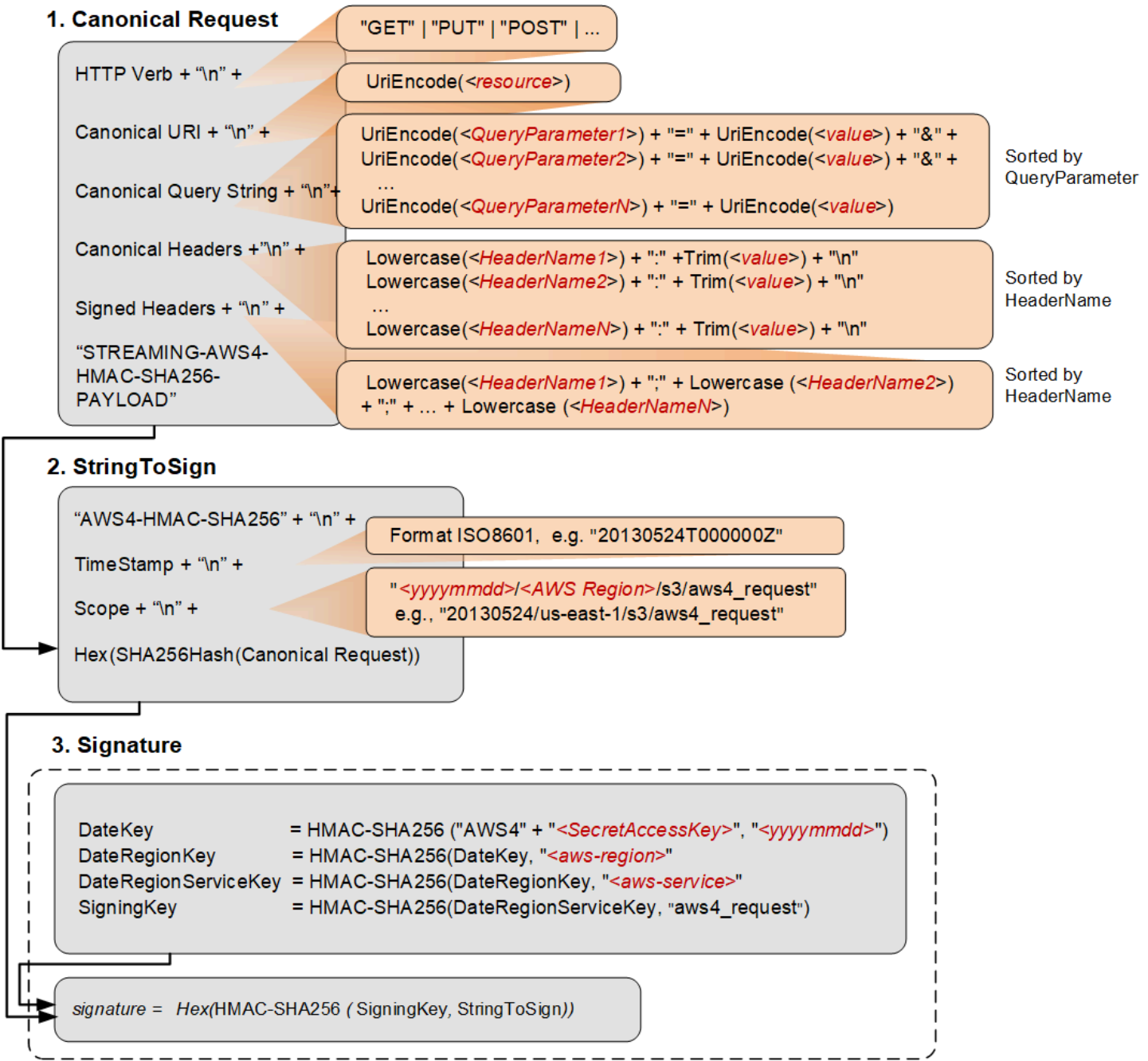
1. Decida o tamanho do fragmento de carga útil. Você precisa disso ao gravar o código.

O tamanho do fragmento deve ser de pelo menos 8 KB. Recomendamos um tamanho de fragmento de pelo menos 64 KB para melhor desempenho. Esse tamanho de fragmento se aplica a todos os fragmentos, exceto o último. O último fragmento enviado pode ter menos de 8 KB. Se a carga útil for pequena e couber em um fragmento, ela poderá ser menor que 8 KB.

2. Crie a assinatura inicial para inclusão no primeiro fragmento. Para obter mais informações, consulte [Cálculo da assinatura inicial](#).
3. Crie o primeiro fragmento e o transmita. Para obter mais informações, consulte [Definição do corpo do fragmento](#).
4. Para cada fragmento subsequente, calcule a assinatura que inclui a assinatura anterior na string que você assina, construa o fragmento e o envie. Para obter mais informações, consulte [Definição do corpo do fragmento](#).
5. Envie o fragmento adicional final, que é igual aos outros fragmentos na construção, mas tem zero bytes de dados. Para obter mais informações, consulte [Definição do corpo do fragmento](#).

Cálculo da assinatura inicial

O diagrama a seguir ilustra o processo de cálculo da assinatura inicial.



A tabela a seguir descreve as funções exibidas no diagrama. É necessário implementar o código para essas funções.

Função	Descrição
Lowercase()	Converta a string em letras minúsculas.
Hex()	Codificação de base 16 em letras minúsculas.

Função	Descrição
SHA256Hash()	Função de hash criptográfico do Secure Hash Algorithm (SHA).
HMAC-SHA256()	Calcula o HMAC usando o algoritmo SHA256 com a chave de assinatura fornecida. Essa é a assinatura final.
Trim()	Remova qualquer espaço em branco inicial e final.

Função	Descrição
<code>UriEncode()</code>	O URI codifica cada byte. O <code>UriEncode()</code> deve aplicar as seguintes regras: • O URI codifica cada byte, exceto os caracteres não reservados: “A”-“Z”, “a”-“z”, “0”-“9”, “-”, “.”, “_” e “~”. • O caractere de espaço é um caractere reservado e deve ser codificado como “%20” (e não como “+”). • Cada byte codificado por URI é formado por um “%” e o valor hexadecimal de dois dígitos do byte. • As letras no valor hexadecimal devem estar em maiúsculas; por exemplo, “%1A”. • Codifique o caractere de barra, “/”, em todos os lugares, exceto no nome da chave do objeto. Por exemplo, se o nome da chave do objeto for <code>photos/Jan/sample.jpg</code>, a barra no nome da chave não está codificada.
	 Important As funções <code>UriEncode</code> padrão fornecidas por sua plataforma de desenvolvimento podem não funcionar devido às diferenças na implementação e à ambiguidade relacionada nos RFCs subjacentes. É recomendável escrever sua própria função <code>UriEncode</code> personalizada para garantir que a codificação funcione.
	Veja a seguir um exemplo da função <code>UriEncode()</code> em Java. <pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBu ilder(); for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i);</pre>

Função	Descrição
	<pre> if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Para obter informações sobre o processo de assinatura, consulte [Cálculos de assinatura para o cabeçalho de autorização: transferência de carga útil em uma única parte \(AWS Signature Version 4\)](#). O processo é o mesmo, mas a criação de CanonicalRequest difere da seguinte forma:

- Além dos cabeçalhos de solicitação que pretende adicionar, você deve incluir os seguintes cabeçalhos:

Cabeçalho	Descrição
x-amz-content-sha256	Esse cabeçalho é obrigatório para todas as solicitações do AWS Signature Version 4. Defina o valor como STREAMING-AWS4-HMAC-SHA256-PAYLOAD para indicar que a assinatura cobre somente cabeçalhos e que não há carga útil.

Cabeçalho	Descrição
Content-Encoding	Defina o valor como <code>aws-chunked</code> . O Amazon S3 oferece suporte a vários valores de codificação de conteúdo. Você pode especificar a codificação de conteúdo personalizada ao usar a API de streaming do Signature Version 4. Por exemplo: Content-Encoding : aws-chunked,gzip O Amazon S3 armazena o objeto resultante sem o valor <code>aws-chunked</code> no cabeçalho <code>content-encoding</code> . Se <code>aws-chunked</code> for o único valor transmitido no cabeçalho <code>content-encoding</code> , o S3 vai considerar o cabeçalho <code>content-encoding</code> vazio e não vai retornar esse cabeçalho quando você recuperar o objeto.
x-amz-decoded-content-length	Defina o valor como o tamanho, em bytes, dos dados a serem fragmentados, sem contar nenhum metadado. Por exemplo, se você estiver carregando um arquivo de 4 GB, defina o valor como 4294967296. Esse é o tamanho bruto do objeto a ser carregado (dados que você deseja armazenar no Amazon S3).
Content-Length	Defina o valor como o tamanho real do corpo HTTP transmitido, que inclui o tamanho dos seus dados (valor definido para <code>x-amz-decoded-content-length</code>), além dos metadados de fragmento . Cada fragmento tem metadados, como a assinatura do fragmento anterior. Os cálculos de fragmentos serão discutidos na seção a seguir. Se você incluir o cabeçalho <code>Transfer-Encoding</code> e especificar qualquer valor diferente de <code>identity</code>, não inclua o cabeçalho <code>Content-Length</code> .

Envie o primeiro fragmento com a assinatura inicial. Você deve construir o fragmento conforme descrito na seção a seguir.

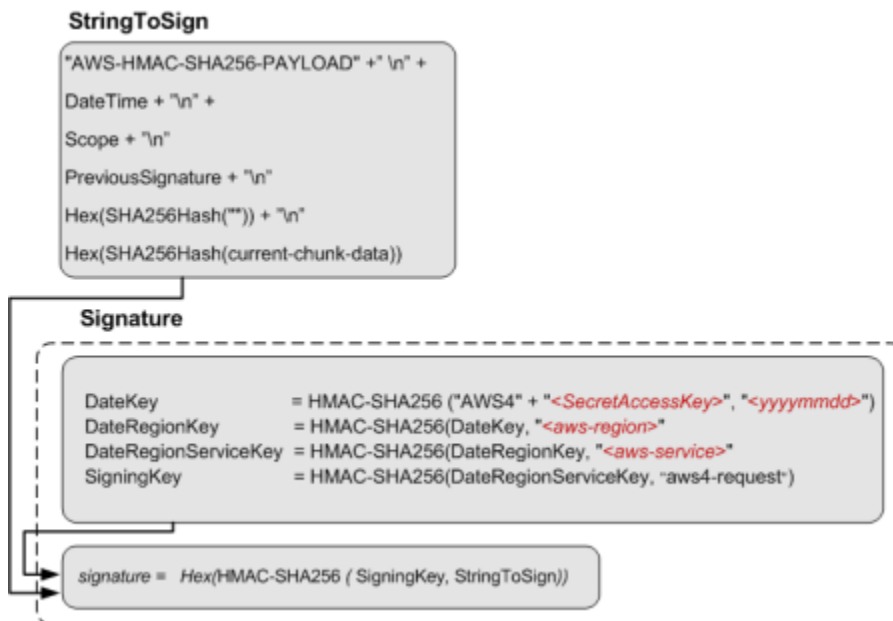
Definição do corpo do fragmento

Todos os fragmentos incluem alguns metadados. Cada fragmento deve estar de acordo com a seguinte estrutura:

```
string(IntHexBase(chunk-size)) + ";chunk-signature=" + signature + \r\n + chunk-data + \r\n
```

Em que:

- `IntHexBase()` é uma função gravada para converter um tamanho de fragmento inteiro em hexadecimal. Por exemplo, se o tamanho do fragmento for 65536, a string hexadecimal será "10000".
- *chunk-size* é o tamanho, em bytes, dos dados do fragmento, sem metadados. Por exemplo, se estiver carregando um objeto de 65 KB e usando um tamanho de fragmento de 64 KB, você carregará os dados em três fragmentos: o primeiro teria 64 KB, o segundo 1 KB e o fragmento final com 0 bytes.
- *signature* Para cada fragmento, você calcula a assinatura usando a seguinte string a ser assinada. Para o primeiro fragmento, você usa a assinatura inicial como a assinatura anterior.



O tamanho dos dados do fragmento final que você envia é 0, embora o corpo do fragmento ainda contenha metadados, incluindo a assinatura do fragmento anterior.

Exemplo: objeto PUT

Você pode usar os exemplos desta seção como uma referência para verificar os cálculos de assinatura no seu código. Antes de analisar os exemplos, observe o seguinte:

- Os cálculos de assinatura nesses exemplos usam os seguintes exemplos de credenciais de segurança.

Parâmetro	Valor
AWSSecretAccessKey	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- Todos os exemplos usam o carimbo de data/hora de solicitação 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT).
- Todos os exemplos usam `examplebucket` como nome do bucket.
- Presume-se que o bucket esteja na região Leste dos EUA (Norte da Virgínia) e a credencial Scope e os cálculos de Signing Key usem `us-east-1` como especificador da região. Para obter mais informações, consulte [Regiões e endpoints](#) na Referência geral da Amazon Web Services.
- Você pode usar solicitações no estilo de caminho ou no estilo hospedado virtual. Os exemplos a seguir usam solicitações de estilo hospedado virtual, por exemplo:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Para obter mais informações, consulte [Hospedagem virtual de buckets](#) no Guia do usuário do Amazon Simple Storage Service.

O exemplo a seguir envia uma solicitação PUT para fazer upload de um objeto. Os cálculos de assinatura presumem o seguinte:

- Você está fazendo o upload de um arquivo de texto de 65 KB, e o conteúdo do arquivo é uma string de um único caractere composta pela letra 'a'.
- O fragmento tem 64 KB. Como resultado, a carga útil é carregada em três fragmentos: 64 KB, 1 KB e o fragmento final com 0 bytes de dados.
- O objeto resultante tem o nome de chave `chunkObject.txt`.
- Você está solicitando `REDUCED_REDUNDANCY` como classe de armazenamento adicionando o cabeçalho de solicitação `x-amz-storage-class`.

Para obter informações sobre a ação da API, consulte [PutObject](#). A sintaxe de solicitação geral é a seguinte:

```
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>
```

As etapas a seguir mostram cálculos de assinatura.

1. Assinatura inicial — Criar uma string a ser assinada
 - 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
```

Na solicitação canônica, a terceira linha está vazia, pois não há parâmetros de consulta na solicitação. A última linha é a string constante fornecida como o valor da carga útil com hash, que deve ser igual ao valor de `x-amz-content-sha256` header.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
cee3fed04b70f867d036f722359b0b1f2f0e5dc0efadbc082b76c4c60e316455
```

Note

Para obter informações sobre cada linha na string a ser assinada, consulte o diagrama que explica o cálculo da assinatura inicial.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Assinatura inicial

```
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

4. Cabeçalho de autorização

O cabeçalho de autorização resultante é o seguinte:

```
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: (65536 bytes, com valor 97 para a letra 'a')

a. String do fragmento a ser assinada:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

Para obter informações sobre cada linha na string a ser assinada, consulte o diagrama anterior que mostra vários componentes da string a ser assinada (por exemplo, as últimas três linhas são `previous-signature`, `hash("")` e `hash(current-chunk-data)`).

b. Assinatura do fragmento:

```
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
```

c. Dados do fragmento enviados:

```
10000;chunk-
signature=ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
<65536-bytes>
```

6. Fragmento 2: (1024 bytes, com valor 97 para a letra 'a')

a. String do fragmento a ser assinada:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

```
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Assinatura do fragmento:

```
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
```

c. Dados do fragmento enviados:

```
400; chunk-  
signature=0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
<1024 bytes>
```

7. Fragmento 3: (0 bytes de dados)

a. String do fragmento a ser assinada:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855  
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Assinatura do fragmento:

```
b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

c. Dados do fragmento enviados:

```
0; chunk-  
signature=b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

Cálculos de assinatura para cabeçalhos finais (uploads fragmentados) (AWS Signature Version 4)

Ao autenticar solicitações usando o cabeçalho `Authorization`, você também pode fazer upload da carga útil em fragmentos. Ao enviar os dados do objeto em fragmentos, você também tem a opção de incluir cabeçalhos finais. (Para ter mais informações, consulte [Cálculos de assinatura para o cabeçalho de autorização: transferência de carga útil em vários fragmentos \(upload fragmentado\) \(AWS Signature Version 4\)](#).) Esta seção descreve as etapas que você precisa seguir quando quiser incluir um cabeçalho final ao término do upload de vários fragmentos.

Important

Ao incluir cabeçalhos finais, você deve enviar o seguinte no cabeçalho inicial:

- Você deve definir `x-amz-content-sha256` como um valor apropriado que indique que um finalizador será incluído. Para ver os valores aceitáveis para `x-amz-content-sha256`, consulte [Autenticação de solicitações: usando o cabeçalho de autorização \(AWSSignature Version 4\)](#).
- Você deve definir `x-amz-trailer` para indicar o conteúdo que está incluindo no cabeçalho final.

Os cabeçalhos finais só são enviados após o upload dos fragmentos. Os fragmentos anteriores são enviados normalmente e assinados conforme descrito nas seções anteriores, incluindo o envio do fragmento final com uma carga útil de 0 bytes. Os cabeçalhos finais são incluídos como seu próprio fragmento e enviados após o fragmento final com uma carga útil de 0 bytes. Por exemplo, se os dados terminassem com um fragmento de 100 KB, você enviaria o seguinte:

- Fragmentos de dados anteriores
- Fragmento final de 100 KB do objeto
- Fragmento de 0 bytes que indica o fim do objeto
- Fragmento de cabeçalhos finais

Exemplos: verificação de cálculos de assinatura

Você pode usar os exemplos desta seção como uma referência para verificar os cálculos de assinatura no seu código. Antes de analisar os exemplos, observe o seguinte:

- Os cálculos de assinatura nesses exemplos usam os seguintes exemplos de credenciais de segurança.

Parâmetro	Valor
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- Todos os exemplos usam o carimbo de data/hora de solicitação 20130524T000000Z (Fri, 24 May 2013 00:00:00 GMT).
- Todos os exemplos usam `examplebucket` como nome do bucket.
- Presume-se que o bucket esteja na região Leste dos EUA (Norte da Virgínia) e a credencial `Scope` e os cálculos de `Signing Key` use `us-east-1` como especificador da região. Para obter mais informações, consulte [Regiões e endpoints do](#) na Referência geral do Amazon Web Services.
- Você pode usar solicitações no estilo de caminho ou no estilo hospedado virtual. Os exemplos a seguir usam solicitações de estilo hospedado virtual, por exemplo:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Para obter mais informações, consulte [Hospedagem virtual de buckets](#) no Guia do usuário do Amazon Simple Storage Service.

O exemplo a seguir envia uma solicitação PUT para fazer upload de um objeto. Os cálculos de assinatura presumem o seguinte:

- Você está fazendo o upload de um arquivo de texto de 65 KB, e o conteúdo do arquivo é uma string de um único caractere composta pela letra 'a'.
- O fragmento tem 64 KB. Como resultado, a carga útil é carregada em três fragmentos: 64 KB, 1 KB e o fragmento final com 0 bytes de dados.

- O objeto resultante tem o nome de chave `chunkObject.txt`.
- Você está solicitando `REDUCED_REDUNDANCY` como classe de armazenamento adicionando o cabeçalho de solicitação `x-amz-storage-class`.
- A transferência inclui um valor de soma de verificação CRC32C como cabeçalho final.

Para obter informações sobre a ação da API, consulte [PutObject](#). A sintaxe de solicitação geral é a seguinte:

```
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>
```

As etapas a seguir mostram cálculos de assinatura.

1. Assinatura inicial — Criar uma string a ser assinada
 - 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

Na solicitação canônica, a terceira linha está vazia, pois não há parâmetros de consulta na solicitação. A última linha é a string constante fornecida como o valor da carga útil com hash, que deve ser igual ao valor de `x-amz-content-sha256` header.

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
44d48b8c2f70eae815a0198cc73d7a546a73a93359c070abbaa5e6c7de112559
```

 Note

Para obter informações sobre cada linha na string a ser assinada, consulte o diagrama que explica o cálculo da assinatura inicial.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Assinatura inicial

```
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

4. Cabeçalho de autorização

O cabeçalho de autorização resultante é o seguinte:

```
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: (65536 bytes, com valor 97 para a letra 'a')

a. String do fragmento a ser assinada:

```
AWS4-HMAC-SHA256-PAYLOAD  
20130524T000000Z  
20130524/us-east-1/s3/aws4_request  
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e  
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855  
bf718b6f653bebc184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

 Note

Para obter informações sobre cada linha na string a ser assinada, consulte o diagrama no tópico anterior ([Cálculo da assinatura inicial](#)) que mostra vários componentes da string a ser assinada. Por exemplo, as últimas três linhas consistem no seguinte:

- *previous-signature*
- `hash("")`
- `hash(current-chunk-data)`

b. Assinatura do fragmento:

```
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
```

c. Dados do fragmento enviados:

```
10000;chunk-  
signature=b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2  
<65536-bytes>
```

6. Fragmento 2: (1024 bytes, com valor 97 para a letra 'a')

a. String do fragmento a ser assinada:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Assinatura do fragmento:

```
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
```

c. Dados do fragmento enviados:

```
400;chunk-
signature=1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
<1024-bytes>
```

7. Fragmento 3: (0 bytes de dados)

a. String do fragmento a ser assinada:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Assinatura do fragmento:

```
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

c. Dados do fragmento enviados:

```
0;chunk-
signature=2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

8. Fragmento 4: cabeçalhos finais

a. String do fragmento final a ser assinada:

```

AWS4-HMAC-SHA256-TRAILER
20130524T000000Z
20130524/us-east-1/s3/aws4_request
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
1e376db7e1a34a8ef1c4bcee131a2d60a1cb62503747488624e10995f448d774

```

 Note

As duas últimas linhas são `previous-signature` (a assinatura do fragmento de dados de 0 bytes), `hash(trailing-checksum-header-name: base64-encoded-trailing-checksum-value\n)`.

O hash é calculado da seguinte forma, sem espaços em branco:

- O nome do cabeçalho de soma de verificação final
- Dois pontos (:)
- O valor da soma de verificação final com codificação base64
- Um caractere de nova linha (\n).

Neste exemplo, onde estamos usando o algoritmo de hash `crc32c`, com o valor da soma de verificação com codificação base64 `s008/Q==`, podemos representar o cálculo da seguinte forma: `hash('x-amz-checksum-crc32c:s008/Q==\n')`.

b. Assinatura do fragmento:

```
d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

c. Dados do fragmento enviados:

```

x-amz-checksum-crc32c:s008/Q==
x-amz-trailer-
signature:d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435

```

Autenticação de solicitações: uso de parâmetros de consulta (AWS Signature Version 4)

Conforme descrito na visão geral da autenticação (consulte [Métodos de autenticação](#)), você pode fornecer informações de autenticação usando parâmetros de string de consulta. Usar parâmetros de consulta para autenticar solicitações é útil quando você deseja expressar uma solicitação inteiramente em um URL. Esse método também é conhecido como pré-assinatura de um URL.

Em um cenário de caso de uso para URLs pré-assinados, você pode conceder acesso temporário aos recursos do Amazon S3. Por exemplo, você pode incorporar um URL pré-assinado em seu site ou, se preferir, usá-lo em um cliente de linha de comando (como Curl) para baixar objetos.

Note

Você também pode usar a AWS CLI para criar URLs pré-assinados. Para ter mais informações, consulte [presign](#) na Referência de comandos da AWS CLI.

O exemplo a seguir é um URL pré-assinado.

```
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>
```

No URL de exemplo, observe:

- Os feeds de linha são adicionados para legibilidade.
- O valor de `X-Amz-Credential` no URL exibe o caractere “/” somente para facilitar a leitura. Na prática, deve ser codificado como `%2F`. Por exemplo:

```
&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request
```

A tabela a seguir descreve os parâmetros de consulta no URL que fornecem informações de autenticação.

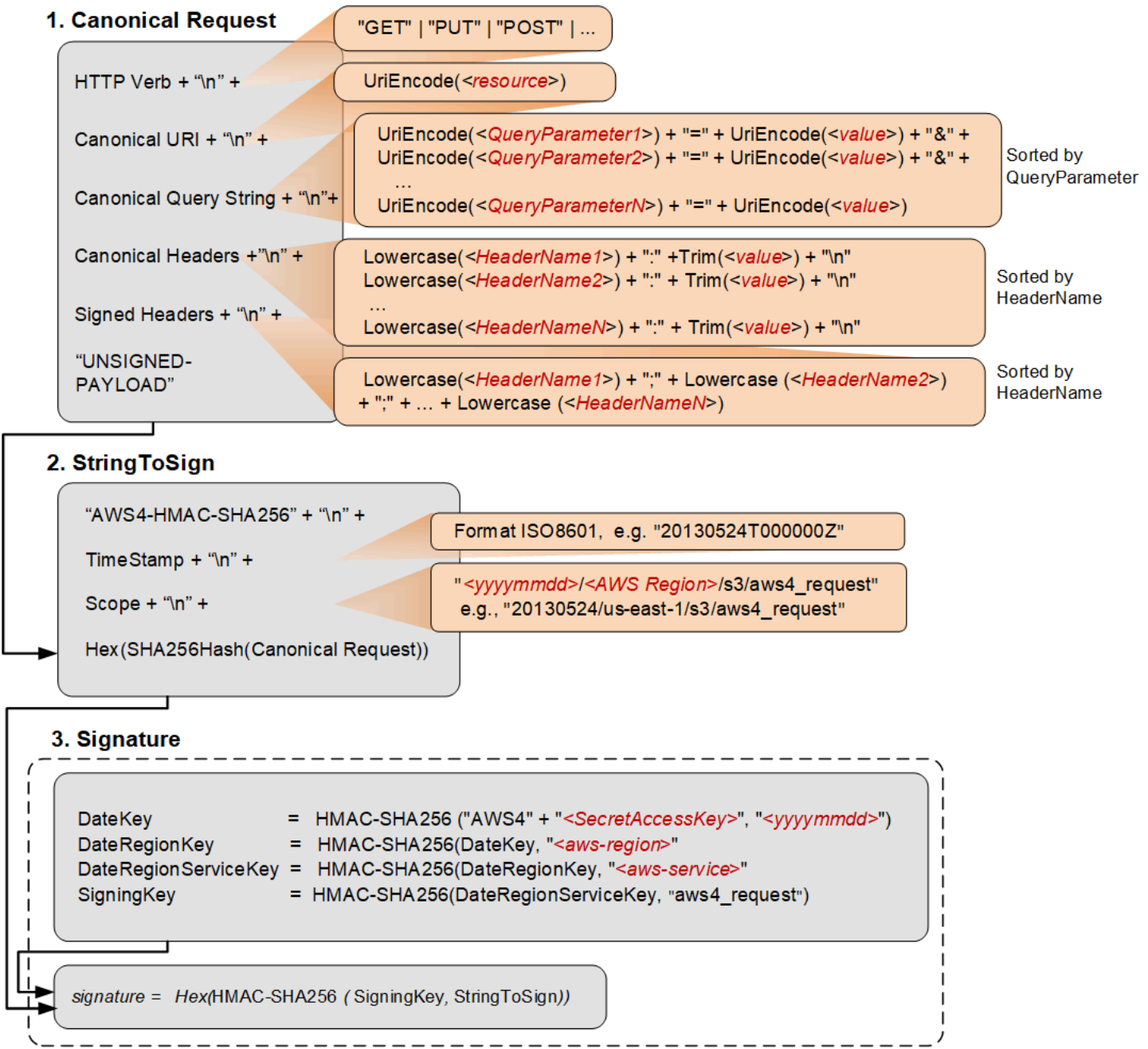
Nome de parâmetro de query string	Valor de exemplo
X-Amz-Algorithm	Identifica a versão da assinatura da AWS e o algoritmo que você usou para calcular a assinatura. Para o AWS Signature Version 4, você define esse valor de parâmetro como <code>AWS4-HMAC-SHA256</code>. Essa string identifica o AWS Signature Version 4 (AWS4) e o algoritmo HMAC-SHA256 (HMAC-SHA256).
X-Amz-Credential	Além do ID da chave de acesso, esse parâmetro também fornece o escopo (Região da AWS e serviço) para o qual a assinatura é válida. O valor deve corresponder ao escopo usado nos cálculos de assinatura, abordados na seção a seguir. A forma geral para esse valor de parâmetro é: <pre><your-access-key-id> /<date>/<Região da AWS>/<AWS-service> /aws4_request</pre> Por exemplo: <pre>AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request</pre> Para o Amazon S3, a string <code>AWS-service</code> é <code>s3</code>. Para conferir a lista de strings da <code>AWS-region</code> do S3, consulte Regiões e endpoints na Referência geral da AWS.
X-Amz-Date	O formato de data e hora deve seguir o padrão ISO 8601 e deve ter a formatação "<code>aaaaMMddTHHmssZ</code>". Por exemplo,

Nome de parâmetro de query string	Valor de exemplo
	se a data e a hora forem “08/01/2016 15:32:41.982-700”, elas primeiro deverão ser convertidas em UTC (Tempo Universal Coordenado) e depois enviadas como “20160801T223241Z”.
X-Amz-Expires	Fornece o período, em segundos, de validade do URL pré-assinado gerado. Por exemplo, 86400 (24 horas). Esse valor é um inteiro. O valor mínimo que você pode definir é 1 e o máximo é 604800 (sete dias). O URL pré-assinado pode ser válido por no máximo sete dias, pois a chave de assinatura usada no cálculo da assinatura é válida por até sete dias.
X-Amz-SignedHeaders	Lista os cabeçalhos usados para calcular a assinatura. Os seguintes cabeçalhos são obrigatórios para os cálculos da assinatura: • O cabeçalho HTTP host. • Todo cabeçalho x-amz-* que você pretende adicionar à solicitação.  Note Para maior segurança, é necessário assinar todos os cabeçalhos de solicitação que pretende incluir na solicitação.

Nome de parâmetro de query string	Valor de exemplo
X-Amz-Signature	Fornece a assinatura para autenticar a solicitação. Essa assinatura deve corresponder à assinatura calculada pelo Amazon S3; caso contrário, o Amazon S3 negará a solicitação. Por exemplo, .. 733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7 Os cálculos de assinatura serão descritos na seção a seguir.
X-Amz-Security-Token	Parâmetro de credencial opcional ao usar credenciais provenientes do serviço STS.

Cálculo de uma assinatura

O diagrama a seguir ilustra o processo de cálculo da assinatura.



Função	Descrição
SHA256Hash()	Função de hash criptográfico do Secure Hash Algorithm (SHA).
HMAC-SHA256()	Calcula o HMAC usando o algoritmo SHA256 com a chave de assinatura fornecida. Essa é a assinatura final.
Trim()	Remova qualquer espaço em branco inicial e final.

Função	Descrição
<code>UriEncode()</code>	O URI codifica cada byte. O <code>UriEncode()</code> deve aplicar as seguintes regras: • O URI codifica cada byte, exceto os caracteres não reservados: “A”-“Z”, “a”-“z”, “0”-“9”, “-”, “.”, “_” e “~”. • O caractere de espaço é um caractere reservado e deve ser codificado como “%20” (e não como “+”). • Cada byte codificado por URI é formado por um “%” e o valor hexadecimal de dois dígitos do byte. • As letras no valor hexadecimal devem estar em maiúsculas; por exemplo, “%1A”. • Codifique o caractere de barra, “/”, em todos os lugares, exceto no nome da chave do objeto. Por exemplo, se o nome da chave do objeto for <code>photos/Jan/sample.jpg</code>, a barra no nome da chave não está codificada.

 **Important**

As funções `UriEncode` padrão fornecidas por sua plataforma de desenvolvimento podem não funcionar devido às diferenças na implementação e à ambiguidade relacionada nos RFCs subjacentes. É recomendável escrever sua própria função `UriEncode` personalizada para garantir que a codificação funcione.

Veja a seguir um exemplo da função `UriEncode()` em Java.

```
public static String UriEncode(CharSequence input,
    boolean encodeSlash) {
    StringBuilder result = new StringBu
    ilder();
    for (int i = 0; i < input.length(); i++) {
        char ch = input.charAt(i);
```

Função	Descrição
	<pre> if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Para obter mais informações sobre o processo de assinatura (detalhes da criação de uma solicitação canônica, string a ser assinada e cálculos de assinatura), consulte [Cálculos de assinatura para o cabeçalho de autorização: transferência de carga útil em uma única parte \(AWS Signature Version 4\)](#). O processo geralmente é o mesmo, mas a criação de CanonicalRequest em um URL pré-assinado é diferente da seguinte forma:

- Não inclua um hash de carga útil na solicitação canônica porque, ao criar um URL pré-assinado, você não conhece o conteúdo da carga útil porque o URL é usado para carregar uma carga útil arbitrária. Em vez disso, você usa uma string constante UNSIGNED-PAYLOAD.
- A string de consulta canônica deve incluir todos os parâmetros de consulta da tabela anterior, exceto X-Amz-Signature.
- Para o S3, é necessário incluir o parâmetro de consulta X-Amz-Security-Token no URL ao usar credenciais provenientes do serviço STS.
- Os cabeçalhos canônicos devem incluir o cabeçalho HTTP host. Se você pretende incluir qualquer um dos cabeçalhos x-amz-*, esses cabeçalhos também deverão ser adicionados para o cálculo da assinatura. Se desejar, adicione todos os outros cabeçalhos que pretende incluir na solicitação. Para maior segurança, você deve assinar o máximo de cabeçalhos possível. Se você adicionar um cabeçalho assinado que também seja um parâmetro de consulta assinado e o valor deles for diferente, será gerado um erro InvalidRequest, pois a entrada é conflitante.

Um exemplo

Suponha que você tenha um objeto `test.txt` em seu bucket `examplebucket`. Você deseja compartilhar esse objeto com outras pessoas por um período de 24 horas (86400 segundos) criando um URL pré-assinado.

```
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>
```

As etapas a seguir ilustram primeiro os cálculos da assinatura e, em seguida, a construção do URL pré-assinado. O exemplo faz as seguintes suposições adicionais:

- O carimbo de data/hora da solicitação é `Fri, 24 May 2013 00:00:00 GMT`.
- O bucket está na região Leste dos EUA (Norte da Virgínia) e a credencial `Scope` e os cálculos de `Signing Key` usam `us-east-1` como especificador da região. Para obter mais informações, consulte [Regiões e endpoints](#) no Referência geral da AWS.

Você pode usar esse exemplo como um caso de teste para verificar a assinatura que seu código calcula. No entanto, você deve usar o mesmo nome de bucket, chave de objeto, carimbo de data/hora e as seguintes credenciais de exemplo:

Parâmetro	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

```

aeeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404

```

Agora você tem todas as informações para criar um URL pré-assinado. O URL resultante desse exemplo é mostrado a seguir (você pode usá-lo para comparar seu URL pré-assinado):

```

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

```


- Objeto PUT: exemplos separados ilustram o upload da carga útil completa de uma vez e o upload da carga útil em fragmentos. Para obter informações sobre como usar o cabeçalho de autorização para autenticação, consulte [Autenticação de solicitações: usando o cabeçalho de autorização \(AWSSignature Version 4\)](#).
- Objeto GET: este exemplo gera um URL pré-assinado para obter um objeto. Os parâmetros de consulta fornecem a assinatura e outras informações de autenticação. Os usuários podem colar um URL pré-assinado no navegador para recuperar o objeto, ou você pode usar o URL para criar um link clicável. Para obter informações sobre o uso de parâmetros de consulta para autenticação, consulte [Autenticação de solicitações: uso de parâmetros de consulta \(AWS Signature Version 4\)](#).

O restante desta seção descreve os exemplos em Java e C#. Os tópicos incluem instruções para baixar os exemplos e executá-los.

Exemplos de cálculo de assinatura usando Java (AWS Signature Version 4)

O exemplo Java que mostra o cálculo da assinatura pode ser baixado em <https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AWSS3SigV4JavaSamples.zip>. Esses exemplos usam o SDK da AWS para Java v1, e recomendamos o uso do [SDK da AWS para Java v2](#) para novos aplicativos. Em `RunAllSamples.java`, a função `main()` executa solicitações de exemplo para criar um objeto, recuperar um objeto e criar um URL pré-assinado para o objeto. O exemplo cria um objeto a partir da string de texto fornecida no 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);
```

Para testar os exemplos em um computador Linux

As instruções a seguir servem para o sistema operacional Linux.

1. Em um terminal, navegue até o diretório que contém `AWSS3SigV4JavaSamples.zip`.
2. Extraia o arquivo `.zip`.
3. Em um editor de texto, abra o arquivo `./com/amazonaws/services/s3/samples/RunAllSamples.java`. Atualize o código com as seguintes informações:

- O nome de um bucket em que o novo objeto pode ser criado.

 Note

Os exemplos usam uma solicitação de estilo hospedado virtualmente para acessar o bucket. Para evitar possíveis erros, o nome do bucket deve estar de acordo com as regras de nomenclatura do bucket, conforme explicado em [Restrições e limitações do bucket](#) no Guia do usuário do Amazon Simple Storage Service.

- Região da AWS onde o bucket reside.

Se o bucket estiver na região Leste dos EUA (Norte da Virgínia), use us-east-1 para especificar a região. Para obter uma lista de outras Regiões da AWS, acesse [Amazon Simple Storage Service \(S3\)](#) na Referência geral da AWS.

4. Compile o código-fonte e armazene as classes compiladas no diretório bin/.

```
javac -d bin -source 6 -verbose com
```

5. Altere o diretório para bin/ e, em seguida, execute RunAllSamples.

```
java com.amazonaws.services.s3.sample.RunAllSamples
```

O código executa todos os métodos em `main()`. Para cada solicitação, a saída mostrará a solicitação canônica, a string a ser assinada e a assinatura.

Exemplos de cálculos de assinatura usando C# (AWS Signature Version 4)

O exemplo de C# que mostra o cálculo da assinatura pode ser baixado em https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip. Esses exemplos usam o SDK da AWS para .NET v1, e recomendamos o uso do [SDK da AWS para .NET v3](#) para novos aplicativos. Em `Program.cs`, a função `main()` executa solicitações de exemplo para criar um objeto, recuperar um objeto e criar um URL pré-assinado para o objeto. O código para o cálculo da assinatura está na pasta `\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");
```

Para testar os exemplos com o Microsoft Visual Studio 2010 ou posterior

1. Extraia o arquivo .zip.
2. Inicie o Visual Studio e, em seguida, abra o arquivo .sln.
3. Atualize o arquivo App.config com credenciais de segurança válidas.
4. Atualize o código da seguinte maneira:
 - Em Program.cs, forneça o nome do bucket e a Região da AWS onde o bucket reside. O exemplo cria um objeto nesse bucket.
5. Execute o código.
6. Para verificar se o objeto foi criado, copie o URL pré-assinado criado pelo programa e cole-o em uma janela do navegador.

Autenticação de solicitações: uploads baseados no navegador usando POST (AWS Signature Version 2)

O Amazon S3 é compatível com solicitações HTTP POST para que os usuários possam fazer upload de conteúdo diretamente no Amazon S3. Usar HTTP POST para upload de conteúdo simplifica os uploads e reduz a latência do upload em que os usuários carregam dados para armazenar no Amazon S3. Esta seção descreve como autenticar solicitações HTTP POST. Para obter mais informações sobre solicitações HTTP POST, como criar um formulário, criar uma política de POST e um exemplo, consulte [Uploads baseados no navegador usando POST \(AWS Signature Version 4\)](#).

Para autenticar uma solicitação HTTP POST, faça o seguinte:

1. O formulário deve incluir os seguintes campos para fornecer a assinatura e as informações relevantes que o Amazon S3 pode usar para recalcular a assinatura ao receber a solicitação:

Nome do elemento	Descrição
<code>policy</code>	Política de segurança com codificação Base64 que descreve o que é permitido na solicitação. Para o cálculo da assinatura, essa política é a string que você assina. O Amazon S3 deve obter essa política para poder recalcular a assinatura.
<code>x-amz-algorithm</code>	O algoritmo de assinatura usado. Para AWS Signature Version 4, o valor é <code>AWS4-HMAC-SHA256</code> .
<code>x-amz-credential</code>	Além do ID da chave de acesso, isso fornece as informações do escopo que você usou para calcular a chave para o cálculo da assinatura. É uma string da seguinte forma: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Por exemplo: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . . .</pre> Para o Amazon S3, a string <code>aws-service</code> é <code>s3</code>. Para conferir a lista de strings <code>aws-region</code> do Amazon S3, consulte Regiões e endpoints na Referência geral da AWS.

Nome do elemento	Descrição
x-amz-date	É o valor de data no formato ISO8601. Por exemplo, 20130728T000000Z . É a mesma data que você usou para criar a chave de assinatura. Também deve ser o mesmo valor fornecido na política (x-amz-date) que você assinou.
x-amz-signature	(AWS Signature Version 4) O hash HMAC-SHA256 da política de segurança. Para obter mais informações sobre opções de assinatura, consulte Adicionar a assinatura à solicitação HTTP na Referência geral da AWS.

2. A política de POST deve incluir os seguintes elementos:

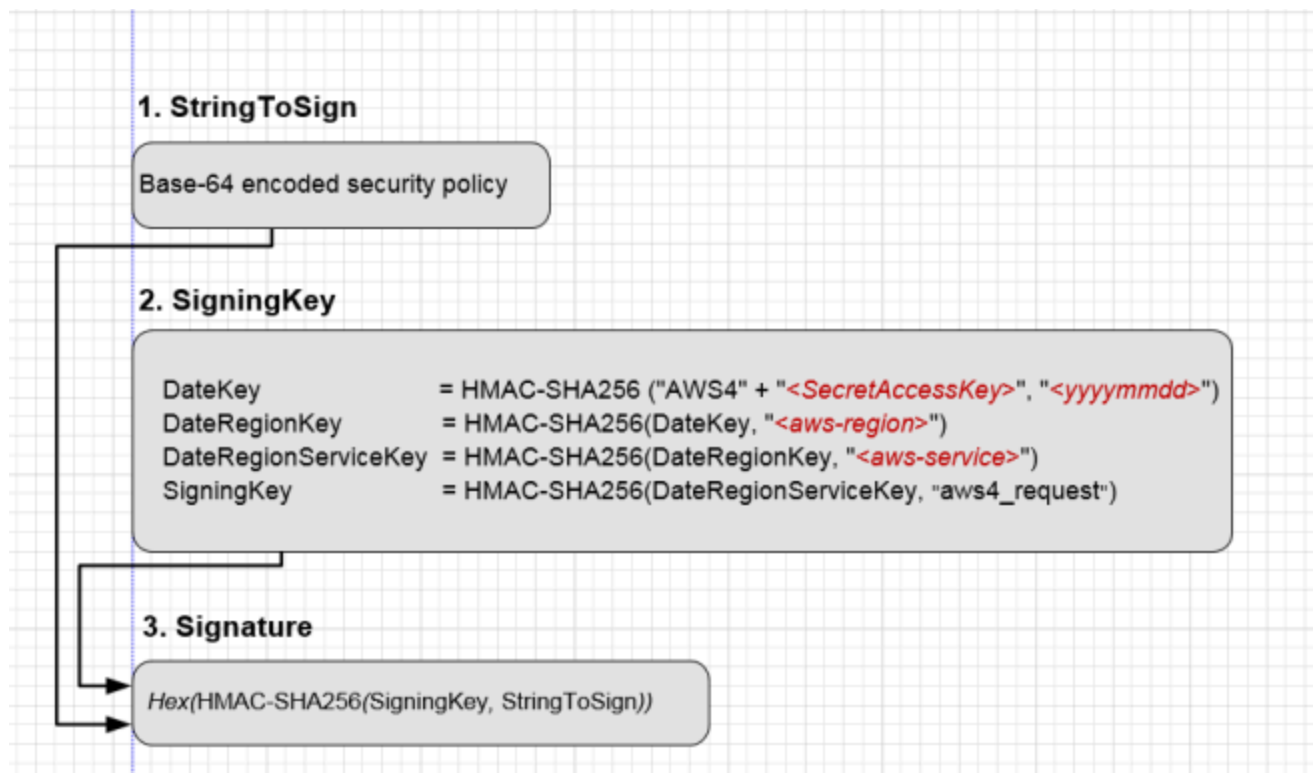
Nome do elemento	Descrição
x-amz-algorithm	O algoritmo que você usou para calcular a assinatura. Para AWS Signature Version 4, o valor é AWS4-HMAC-SHA256 .
x-amz-credential	Além do ID da chave de acesso, isso fornece as informações do escopo que você usou para calcular a chave para o cálculo da assinatura. É uma string da seguinte forma: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Por exemplo, <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .</pre>

Nome do elemento	Descrição
x-amz-date	O valor de data especificado na string com formato ISO8601. Por exemplo, "20130728T000000Z". A data deve ser a mesma usada na criação da chave de assinatura para o cálculo da assinatura.

3. Para o cálculo da assinatura, a política de POST é a string a ser assinada.

Cálculo de uma assinatura

O diagrama a seguir ilustra o processo de cálculo da assinatura.



Para calcular uma assinatura

1. Crie uma política usando a codificação UTF-8.
2. Converta a política com codificação UTF-8 em Base64. O resultado é a string a ser assinada.

3. Crie a assinatura como um hash HMAC-SHA256 da string a ser assinada. Você fornecerá a chave de assinatura como chave para a função hash.
4. Codifique a assinatura usando a codificação hexadecimal.

Para obter mais informações sobre a criação de formulários HTML, políticas de segurança e um exemplo, consulte estes subtópicos:

- [Criação de um formulário HTML \(usando o AWS Signature Version 4\)](#)
- [Política de POST](#)
- [Exemplo: upload baseado no navegador usando HTTP POST \(usando o AWS Signature Version 4\)](#)

Chaves de política específicas de autenticação do Amazon S3 Signature Version 4.

A tabela a seguir mostra as chaves de política relacionadas à autenticação do Amazon S3 Signature Version 4 que podem estar nas políticas do Amazon S3. Em uma política de bucket, você pode adicionar essas condições para aplicar um comportamento específico quando as solicitações forem autenticadas usando o Signature Version 4. Para obter exemplos de políticas, consulte [Exemplos de política de bucket que usam chaves de condição relacionadas ao Signature Version 4](#).

Chaves aplicáveis	Descrição
<code>s3:signatureversion</code>	Identifica a versão do AWS Signature à qual você deseja oferecer suporte para solicitações autenticadas. Para solicitações autenticadas, o Amazon S3 é compatível com o Signature Version 4 e o Signature Version 2. Você pode adicionar essa condição à sua política de bucket para exigir uma versão de assinatura específica. Valores válidos: "AWS" identifica o Signature Version 2

Chaves aplicáveis	Descrição
	"AWS4-HMAC-SHA256" identifica o Signature Version 4
s3:authType	O Amazon S3 é compatível com vários métodos de autenticação (consulte Autenticação de solicitações (AWS Signature Version 4)). Você pode usar essa chave de condição opcional para restringir as solicitações recebidas a usar um método específico de autenticação. Por exemplo, é possível permitir que somente o cabeçalho HTTP Authorization seja usado na autenticação de solicitação. Valores válidos: REST-HEADER REST-QUERY-STRING POST

Chaves aplicáveis	Descrição
s3:signatureAge	A duração em milissegundos da validade de uma assinatura em uma solicitação autenticada. Essa condição funciona para: • URLs pré-assinados: onde a condição mais restritiva vence. Para obter mais informações, consulte Trabalho com URLs pré-assinados. • POST pré-assinado: faça upload de arquivos diretamente para o S3 usando POST pré-assinado. Para obter mais informações, consulte Política de POST do Amazon S3. No Signature Version 2, esse valor é sempre definido como 0. No Signature Version 4, a chave de assinatura é válida por até 7 dias. Portanto, as assinaturas também são válidas por até 7 dias. Você pode usar essa condição para limitar ainda mais a idade da assinatura. Para obter mais informações, consulte Introdução à assinatura de solicitações. Exemplo de valor: : 100

Chaves aplicáveis	Descrição
s3:x-amz-content-sha256	É possível usar essa chave de condição para proibir conteúdo não assinado em seu bucket. Ao usar o Signature Version 4, para solicitações que usem o cabeçalho <code>Authorization</code>, você adiciona o cabeçalho <code>x-amz-content-sha256</code> no cálculo da assinatura e, em seguida, define seu valor para a carga útil do hash. Essa chave de condição não é compatível com o cabeçalho <code>x-amz-content-sha256</code> como parâmetro de string de consulta. Você pode usar essa chave de condição em sua política de bucket para negar qualquer upload cujas cargas úteis não estejam assinadas. Por exemplo, você pode negar uploads que usem o cabeçalho <code>Authorization</code> para autenticar solicitações, mas não assinem a carga útil. Para obter mais informações, consulte Cálculos de assinatura para o cabeçalho de autorização: transferência de carga útil em uma única parte (AWS Signature Version 4). Valor válido: UNSIGNED-PAYLOAD

Exemplos de política de bucket que usam chaves de condição relacionadas ao Signature Version 4

A política de bucket a seguir vai negar qualquer solicitação de URL pré-assinado do Amazon S3 em objetos em `examplebucket` se a assinatura tiver mais de dez minutos de idade.

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
        }
      }
    }
  ]
}
```

A política de bucket a seguir permite somente solicitações que usem o cabeçalho Authorization para autenticação de solicitação. Toda solicitação de POST ou URL pré-assinado será negada.

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"
        }
      }
    }
  ]
}
```

```

    }
  }
]
}

```

A política de bucket a seguir nega solicitações que usam URLs pré-assinados para autenticação de solicitações:

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"
        }
      }
    }
  ]
}

```

A política de bucket a seguir nega qualquer upload com cargas úteis não assinadas:

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"
    }
  }
}
```

Uploads baseados no navegador usando POST (AWS Signature Version 4)

Esta seção explica como fazer upload de arquivos diretamente no Amazon S3 por meio de um navegador usando solicitações HTTP POST. Ela também contém informações sobre como usar a biblioteca JavaScript do AWS Amplify para uploads de arquivos baseados no navegador no Amazon S3.

Tópicos

- [Objeto POST](#)
- [POST restauração de objeto](#)
- [Uploads baseados no navegador usando HTTP POST](#)
- [Cálculo de uma assinatura](#)
- [Criação de um formulário HTML \(usando o AWS Signature Version 4\)](#)
- [Política de POST](#)
- [Exemplo: upload baseado no navegador usando HTTP POST \(usando o AWS Signature Version 4\)](#)
- [Uploads baseados no navegador para o Amazon S3 usando a biblioteca do AWS Amplify](#)

Objeto POST

Descrição

A operação POST adiciona um objeto a um bucket especificado usando formulários HTML. POST é uma forma alternativa de PUT que permite uploads baseados no navegador como forma de colocar objetos em buckets. Os parâmetros transmitidos para PUT por meio de cabeçalhos HTTP são passados como campos de formulário para POST no corpo da mensagem com codificação multipart/form-data. Para adicionar um objeto a um bucket, é preciso ter acesso de WRITE no bucket. O Amazon S3 nunca armazena objetos parciais. Se você receber uma resposta bem-sucedida, pode ter certeza de que o objeto inteiro foi armazenado.

O Amazon S3 é um sistema distribuído. A menos que você tenha habilitado o versionamento para um bucket, se o Amazon S3 receber diversas solicitações de gravação para o mesmo objeto simultaneamente, apenas a última versão gravada do objeto será armazenada.

Para garantir que os dados não sejam corrompidos ao atravessar a rede, use o campo de formulário Content-MD5. Quando você usa esse campo, o Amazon S3 verifica o objeto em relação ao valor MD5 fornecido. Se eles não corresponderem, o Amazon S3 retornará um erro. Além disso, você pode calcular o valor MD5 ao publicar um objeto no Amazon S3 e comparar o ETag retornado com o valor calculado de MD5. O ETag reflete as alterações apenas no conteúdo de um objeto, não nos metadados.

Note

Para configurar seu aplicativo para enviar os cabeçalhos da solicitação antes de enviar o corpo da solicitação, use o código de status HTTP 100 (Continue). Para operações POST, o uso desse código de status ajuda a evitar o envio do corpo da mensagem se a mensagem for rejeitada com base nos cabeçalhos (por exemplo, devido a uma falha de autenticação ou um redirecionamento). Para obter mais informações sobre o código de status HTTP 100 (Continue), acesse a Seção 8.2.3 de <http://www.ietf.org/rfc/rfc2616.txt>.

O Amazon S3 criptografa automaticamente todos os novos objetos que são carregados em um bucket do S3. A configuração de criptografia de um objeto carregado depende da configuração de criptografia padrão do bucket de destino. Por padrão, todos os buckets têm uma configuração de criptografia padrão que usa criptografia do lado do servidor com chaves gerenciadas pelo Amazon S3 (SSE-S3).

Se o bucket de destino tiver uma configuração de criptografia padrão que usa criptografia do lado do servidor com uma chave do AWS Key Management Service (AWS KMS) (SSE-KMS), uma criptografia de camada dupla do lado do servidor com uma chave do AWS KMS (DSSE-KMS) ou uma chave de criptografia fornecida pelo cliente (SSE-C), o Amazon S3 usará a chave do KMS correspondente ou uma chave fornecida pelo cliente para criptografar o objeto carregado. Ao carregar um objeto, se você quiser alterar a configuração de criptografia do objeto carregado, poderá especificar o tipo de criptografia no lado do servidor. Você pode configurar SSE-S3, SSE-KMS, DSSE-KMS ou SSE-C. Para obter mais informações, consulte [Proteção de dados usando criptografia no lado do servidor](#) no Guia do usuário do Amazon Simple Storage Service.

Important

Ao criar sua solicitação, o campo `file` deve ser o último campo no formulário.

Versionamento

Se o controle de versão para um bucket for habilitado, o POST vai gerar automaticamente um ID de versão exclusivo para o objeto que está sendo adicionado. O Amazon S3 retorna esse ID na resposta usando o cabeçalho de resposta `x-amz-version-id`.

Se você suspender o controle de versão de um bucket, o Amazon S3 sempre usará `null` como o ID de versão do objeto armazenado em um bucket.

Para obter mais informações sobre como retornar o estado de versionamento de um bucket, consulte [GetBucketVersioning](#).

O Amazon S3 é um sistema distribuído. Se você habilitar o versionamento para um bucket e o Amazon S3 receber diversas solicitações de gravação para o mesmo objeto simultaneamente, todas as versões do objeto serão armazenadas.

Para ver exemplos de solicitações que usam versionamento, consulte [Solicitação de amostra](#).

Solicitações

Sintaxe

```
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--
```

Parâmetros da solicitação

Essa implementação da operação não usa parâmetros de solicitação.

Campos do formulário

Esta operação pode usar os seguintes campos de formulário.

Name (Nome)	Descrição	Obrigatório
AWSSecretKeyId	O ID da chave de acesso da AWS do proprietário do bucket que concede acesso a um usuário anônimo para uma solicitação que satisfaz o conjunto de restrições na política. Tipo: string Padrão: nenhum Restrições: será obrigatório se um documento de política estiver incluso na solicitação.	Condicional
acl	A lista de controle de acesso (ACL) do Amazon S3 especificada. Se a ACL especificada não for válida, um erro será gerado. Para obter mais informações sobre ACLs, consulte Visão geral da lista de	Não

Name (Nome)	Descrição	Obrigatório
	controle de acesso (ACL) no Guia do usuário do Amazon Simple Storage Service. Tipo: string Padrão: 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	Os cabeçalhos específicos para REST. Para obter mais informações, consulte PutObject. Tipo: string Padrão: nenhum	Não
file	Conteúdo de arquivo ou texto. Esse conteúdo deve ser o último campo no formulário. Não carregue mais de um arquivo por vez. Tipo: conteúdo de arquivo ou texto Padrão: nenhum	Sim

Name (Nome)	Descrição	Obrigatório
key	O nome da chave carregada. Para usar o nome de arquivo fornecido pelo usuário, use a variável <code>\${filename}</code> . Por exemplo, se uma usuária chamada Mary fizer upload do arquivo <code>example.jpg</code> e você especificar <code>/user/mary/\${filename}</code> , o nome da chave será <code>/user/mary/example.jpg</code> . Para obter mais informações, consulte Chave e metadados de objeto no Guia do usuário do Amazon Simple Storage Service. Tipo: string Padrão: nenhum	Sim
policy	Política de segurança que descreve o que é permitido na solicitação. As solicitações sem uma política de segurança são consideradas anônimas e funcionam apenas em buckets com gravação pública. Para obter mais informações, consulte Formulários HTML e Exemplos de upload no Guia do usuário do Amazon Simple Storage Service. Tipo: string Padrão: nenhum Restrições: uma política de segurança será necessária se o bucket não puder ser gravado publicamente.	Condicional

Name (Nome)	Descrição	Obrigatório
<code>success_action_redirect</code> , <code>redirect</code>	O URL para o qual o cliente é redirecionado após um upload bem-sucedido. Se <code>success_action_redirect</code> não for especificado, o Amazon S3 retornará o tipo de documento vazio especificado no campo <code>success_action_status</code>. Se o Amazon S3 não conseguir interpretar o URL, o campo será ignorado. Se houver falha no upload, o Amazon S3 exibirá um erro e não redirecionará o usuário para um URL. Tipo: string Padrão: nenhum  Note O nome do campo <code>redirect</code> está obsoleto e o suporte para <code>redirect</code> será removido no futuro.	Não

Name (Nome)	Descrição	Obrigatório
success_action_status	Se você não especificar <code>success_action_redirect</code>, o código de status será retornado ao cliente quando o upload for bem-sucedido. Esse campo aceita os valores 200, 201 ou 204 (o padrão). Se o valor estiver definido como 200 ou 204, o Amazon S3 retornará um documento vazio com um código de status 200 ou 204. Se o valor estiver definido como 201, o Amazon S3 retornará um documento XML com um código de status 201. Se o valor não estiver definido ou se estiver definido como um valor inválido, o Amazon S3 retornará um documento vazio com um código de status 204. Tipo: string Padrão: nenhum	Não

Name (Nome)	Descrição	Obrigatório
tagging	O conjunto de tags especificado a ser adicionado ao objeto. Para adicionar tags, use o seguinte esquema de codificação. <pre><Tagging> <TagSet> <Tag> <Key>TagName</Key> <Value>TagValue</Value> </Tag> ... </TagSet> </Tagging></pre>	Não
	Para obter mais informações, consulte Marcação de objetos no Guia do usuário do Amazon Simple Storage Service. Tipo: string Padrão: nenhum	

Name (Nome)	Descrição	Obrigatório
<code>x-amz-storage-class</code>	A classe de armazenamento a ser usada para armazenar o objeto. Se você não especificar uma classe, o Amazon S3 usará a classe de armazenamento padrão, STANDARD. O Amazon S3 oferece suporte a outras classes de armazenamento. Para obter mais informações, consulte Classes de armazenamento no Guia do usuário do Amazon Simple Storage Service. Tipo: string Padrão: STANDARD Valores válidos: REDUCED_REDUNDANCY EXPRESS_ONEZONE DEEP_ARCHIVE GLACIER GLACIER_IR INTELLIGENT_TIERING ONEZONE_IA STANDARD STANDARD_IA	Não
<code>x-amz-meta-*</code>	Os cabeçalhos que começam com esse prefixo são metadados definidos pelo usuário. Cada um é armazenado e retornado como um conjunto de pares de chave-valor. O Amazon S3 não valida nem interpreta metadados definidos pelo usuário. Para obter mais informações, consulte PutObject. Tipo: string Padrão: nenhum	Não

Name (Nome)	Descrição	Obrigatório
x-amz-security-token	O token de segurança do Amazon DevPay. Cada solicitação que usa o Amazon DevPay requer dois campos do formulário x-amz-security-token : um para o token de produto e outro para o token de usuário. Tipo: string Padrão: nenhum	Não
x-amz-signature	(AWS Signature Version 4) O hash HMAC-SHA256 da política de segurança. Tipo: string Padrão: nenhum	Condicional

Name (Nome)	Descrição	Obrigatório
x-amz-website-redirect-location	Se o bucket estiver configurado como um site, esse campo vai redirecionar solicitações desse objeto para outro objeto no mesmo bucket ou para um URL externo. O Amazon S3 armazena o valor desse cabeçalho nos metadados de objeto. Para obter informações sobre metadados de objeto, consulte Chave e metadados de objeto no Guia do usuário do Amazon Simple Storage Service. No exemplo a seguir, o cabeçalho da solicitação define o redirecionamento para um objeto (anotherPage.html) no mesmo bucket: <pre>x-amz-website-redirect-location: /anotherPage.html</pre> No exemplo a seguir, o cabeçalho da solicitação define o redirecionamento do objeto para outro site: <pre>x-amz-website-redirect-location: http://www.example.com/</pre> Para obter mais informações sobre hospedagem de sites no Amazon S3, consulte Hospedagem de sites no Amazon S3 e Como configurar redirecionamentos de páginas de sites no Guia do usuário do Amazon Simple Storage Service. Tipo: string Padrão: nenhum Restrições: o valor deve ter o prefixo /, http:// ou https://. O tamanho do valor é limitado a 2 KB.	Não

Campos adicionais do formulário de solicitação de soma de verificação

Ao fazer upload de um objeto, você pode especificar várias somas de verificação que deseja usar para verificar a integridade dos seus dados. Você pode especificar um algoritmo de soma de verificação adicional a ser usado pelo Amazon S3. Para obter mais informações sobre valores adicionais de soma de verificação, consulte [Verificação da integridade do objeto](#) no Guia do usuário do Amazon Simple Storage Service.

Name (Nome)	Descrição	Obrigatório
x-amz-checksum-algorithm	Indica o algoritmo usado para criar a soma de verificação do objeto. Se um valor for especificado, você deverá incluir o cabeçalho de soma de verificação correspondente. Caso contrário, sua solicitação gerará um erro 400. Os possíveis valores incluem: CRC32, CRC32C, SHA1 e SHA256.	Não
x-amz-checksum-crc32	Especifica a soma de verificação CRC32 de 32 bits com codificação base64 do objeto. Esse parâmetro será necessário quando o valor de x-amz-checksum-algorithm for CRC32.	Condicional
x-amz-checksum-crc32c	Especifica a soma de verificação CRC32C de 32 bits com codificação base64 do objeto. Esse parâmetro será necessário quando o valor de x-amz-checksum-algorithm for CRC32C.	Condicional
x-amz-checksum-sha1	Especifica o resumo SHA-1 de 160 bits com codificação base64 do objeto. Esse parâmetro será necessário quando o valor de x-amz-checksum-algorithm for SHA1.	Condicional
x-amz-checksum-sha256	Especifica o resumo SHA-256 de 256 bits com codificação base64 do objeto.	Condicional

Name (Nome)	Descrição	Obrigatório

Esse parâmetro será necessário quando o valor de `x-amz-checksum-algorithm` for SHA256.

Campos do formulário de solicitação específicos para criptografia no lado do servidor

A criptografia no lado do servidor é a criptografia de dados em repouso. O Amazon S3 criptografa os dados ao gravá-los em discos nos data centers da AWS e os descriptografa quando você os acessa. Ao fazer upload de um objeto, você pode especificar o tipo de criptografia no lado do servidor que deseja que o Amazon S3 use para criptografar o objeto.

Há quatro tipos de criptografia no lado do servidor:

- Criptografia do lado do servidor com chaves gerenciadas pelo Amazon S3 (SSE-S3): desde maio de 2022, todos os buckets do Amazon S3 têm a criptografia configurada por padrão. A opção padrão para a criptografia do lado do servidor é com SSE-S3. Cada objeto é criptografado com uma chave exclusiva. Como uma proteção adicional, a SSE-S3 criptografa a própria chave com uma chave-raiz que alterna regularmente. A SSE-S3 usa uma das cifras de bloco mais fortes disponíveis, o padrão de criptografia avançada de 256 bits (AES-256), para criptografar os dados.
- Criptografia do lado do servidor com chaves do AWS KMS (SSE-KMS): SSE-KMS é fornecido por meio de uma integração do serviço AWS KMS com o Amazon S3. Com o AWS KMS, você tem mais controle sobre suas chaves. Por exemplo, você pode visualizar chaves separadas, editar políticas de controle e seguir as chaves no AWS CloudTrail. Além disso, é possível criar e gerenciar chaves gerenciadas pelo cliente ou usar Chaves gerenciadas pela AWS que são exclusivas para você, para o seu serviço e para a sua região.
- Criptografia de camada dupla do lado do servidor com chaves do AWS KMS (DSSE-KMS): a criptografia de camada dupla do lado do servidor com AWS KMS keys (DSSE-KMS) é semelhante à SSE-KMS, mas aplica duas camadas individuais de criptografia no nível do objeto em vez apenas uma.
- Criptografia do lado do servidor com chaves fornecidas pelo cliente (SSE-C): com SSE-C, você gerencia as chaves de criptografia e o Amazon S3 gerencia a criptografia, ao gravar em discos, e a descriptografia, quando você acessa os objetos.

Note

Se você tiver a criptografia do lado do servidor com chaves fornecidas pelo cliente (SSE-C) bloqueada para seu bucket de uso geral, receberá um erro HTTP 403 Acesso negado ao especificar os cabeçalhos de solicitação de SSE-C ao gravar novos dados no bucket. Para obter mais informações, consulte [Bloqueio ou desbloqueio de SSE-C para um bucket de uso geral](#).

Para obter mais informações, consulte [Como proteger dados usando criptografia do lado do servidor](#) no Guia do usuário do Amazon Simple Storage Service.

Dependendo do tipo de criptografia do lado do servidor que você deseja usar, especifique os campos de formulário a seguir.

- Usar SSE-S3, SSE-KMS ou DSSE-KMS: se você quiser usar esses tipos de criptografia do lado do servidor, especifique os campos de formulário a seguir na solicitação.

Name (Nome)	Descrição	Obrigatório
x-amz-server-side-encryption	Especifica o algoritmo de criptografia do lado do servidor a ser usado quando o Amazon S3 cria um objeto. Para usar SSE-S3, especifique AES256. Para usar SSE-KMS, especifique <code>aws:kms</code>. Para usar DSSE-KMS, especifique <code>aws:kms:dsse</code>. Tipo: string Valor válido: <code>aws:kms</code>, <code>AES256</code>, <code>aws:kms:dsse</code>	Sim
x-amz-server-side-encryption-aws-kms-key-id	Se o cabeçalho <code>x-amz-server-side-encryption</code> tiver um valor válido de <code>aws:kms</code> ou <code>aws:kms:dsse</code>, esse cabeçalho vai especificar o ID da AWS KMS key que foi usada para criptografar o objeto. Tipo: string	Sim, se o valor de <code>x-amz-server-side-</code>

Name (Nome)	Descrição	Obrigatório
		encryption for aws:kms ou aws:kms:dsse
x-amz-server-side-encryption-context	Se x-amz-server-side-encryption tiver um valor válido de aws:kms ou aws:kms:dsse , esse cabeçalho vai especificar o contexto de criptografia do objeto. O valor deste cabeçalho é uma string UTF-8 com codificação base64 que contém pares de chave-valor formatados em JSON para o contexto de criptografia. Tipo: string	Não
x-amz-server-side-encryption-bucket-key-enabled	Se x-amz-server-side-encryption tiver um valor válido de aws:kms ou aws:kms:dsse , esse cabeçalho vai especificar se o Amazon S3 deve usar uma chave de bucket do S3 com SSE-KMS ou DSSE-KMS. Definir esse cabeçalho como true faz com que o Amazon S3 use uma chave de bucket do S3 para criptografia de objetos com SSE-KMS ou DSSE-KMS. Tipo: booliano	Não

 Note

Se você especificar x-amz-server-side-encryption:aws:kms ou x-amz-server-side-encryption:aws:kms:dsse, mas não fornecer x-amz-server-side-encryption-aws-kms-key-id, o Amazon S3 usará Chave gerenciada pela AWS (aws/S3) para proteger os dados.

- Usar SSE-C: se você quiser gerenciar suas próprias chaves de criptografia, deverá fornecer todos os seguintes campos de formulário na solicitação.

 Note

Se você usar SSE-C, o valor de ETag que o Amazon S3 retorna na resposta não será o MD5 do objeto.

Name (Nome)	Descrição	Obrigatório
x-amz-server-side-encryption-customer-algorithm	Especifica o algoritmo a ser usado ao criptografar o objeto. Tipo: string Padrão: nenhum Valor válido: AES256 Restrições: deve ser acompanhado por campos <code>x-amz-server-side-encryption-customer-key</code> e <code>x-amz-server-side-encryption-customer-key-MD5</code> válidos.	Sim

Name (Nome)	Descrição	Obrigatório
<code>x-amz-server-side-encryption-customer-key</code>	Especifica a chave de criptografia com codificação base64 fornecida pelo cliente para o Amazon S3 usar na criptografia de dados. Esse valor é usado para armazenar o objeto e, em seguida, é descartado. A Amazon não armazena a chave de criptografia. A chave deve ser apropriada para uso com o algoritmo especificado no cabeçalho <code>x-amz-server-side-encryption-customer-algorithm</code>. Tipo: string Padrão: nenhum Restrições: deve ser acompanhado por campos <code>x-amz-server-side-encryption-customer-algorithm</code> e <code>x-amz-server-side-encryption-customer-key-MD5</code> válidos.	Sim
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Especifica o resumo MD5 com codificação base64 de 128 bits da chave de criptografia de acordo com RFC 1321. O Amazon S3 usa esse cabeçalho para fazer uma verificação de integridade de mensagens e conferir se a chave de criptografia foi transmitida sem erros. Tipo: string Padrão: nenhum Restrições: deve ser acompanhado por campos <code>x-amz-server-side-encryption-customer-algorithm</code> e <code>x-amz-server-side-encryption-customer-key</code> válidos.	Sim

Respostas

Cabeçalhos de resposta

Essa implementação da operação pode incluir os seguintes cabeçalhos de resposta, além dos cabeçalhos que são comuns a todas as respostas. Para obter mais informações, consulte [Cabeçalhos de resposta comuns](#).

[Cabeçalhos de resposta comuns](#).

Name (Nome)	Descrição
<code>x-amz-checksum-crc32</code>	A soma de verificação CRC32 de 32 bits com codificação base64 do objeto. Tipo: string
<code>x-amz-checksum-crc32c</code>	A soma de verificação CRC32C de 32 bits com codificação base64 do objeto. Tipo: string
<code>x-amz-checksum-sha1</code>	O resumo SHA-1 de 160 bits com codificação base64 do objeto. Tipo: string
<code>x-amz-checksum-sha256</code>	O resumo SHA-256 de 256 bits com codificação base64 do objeto. Tipo: string
<code>x-amz-expiration</code>	Se uma ação <code>Expiration</code> for configurada para o objeto como parte da configuração do ciclo de vida do bucket, o Amazon S3 retornará esse cabeçalho. O valor do cabeçalho inclui um componente <code>expiry-date</code> e um componente <code>rule-id</code> codificado em URL. Para buckets com versão habilitada, esse cabeçalho se aplica somente às versões atuais. O Amazon S3 não fornece um cabeçalho para indicar quando uma versão não atual está qualificada para exclusão

Name (Nome)	Descrição
	permanente. Para obter mais informações, consulte PutBucketLifecycleConfiguration. Tipo: string
<code>success_action_redirect, redirect</code>	O URL para o qual o cliente é redirecionado após um upload bem-sucedido. Tipo: string Ancestral: PostResponse
<code>x-amz-server-side-encryption</code>	O algoritmo de criptografia no lado do servidor que foi usado para armazenar esse objeto no Amazon S3 (por exemplo, <code>AES256</code>, <code>aws:kms</code>, <code>aws:kms:dsse</code>). Tipo: string
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	Se o cabeçalho <code>x-amz-server-side-encryption</code> tiver um valor válido de <code>aws:kms</code>, esse cabeçalho vai especificar o ID da chave do KMS que foi usada para criptografar o objeto. Tipo: string
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	Se <code>x-amz-server-side-encryption</code> tiver um valor válido de <code>aws:kms</code>, esse cabeçalho vai indicar se o objeto está criptografado com SSE-KMS usando uma chave de bucket do S3. Se esse cabeçalho estiver definido como <code>true</code>, o objeto vai usar uma chave de bucket do S3 com SSE-KMS. Tipo: booliano

Name (Nome)	Descrição
<code>x-amz-server-side-encryption-customer-algorithm</code>	Se SSE-C tiver sido solicitado, a resposta incluirá esse cabeçalho, que confirma o algoritmo de criptografia usado. Tipo: string Valores válidos: AES256
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Se SSE-C tiver sido solicitado, a resposta incluirá esse cabeçalho para verificar a integridade da mensagem de ida e volta da chave de criptografia fornecida pelo cliente. Tipo: string
<code>x-amz-version-id</code>	Versão do objeto. Tipo: string

Elementos de resposta

Name (Nome)	Descrição
Bucket	O nome do bucket em que o objeto foi armazenado. Tipo: string Antepassado: PostResponse
ETag	O ETag (identificador de entidade) é um hash MD5 do objeto que você pode usar para fazer operações GET condicionais usando a tag de solicitação If-Modified com a operação de solicitação GET. O ETag reflete alterações apenas no conteúdo de um objeto, não nos metadados. Tipo: string

Name (Nome)	Descrição
	Antepassado: PostResponse
Key	O nome da chave de objeto. Tipo: string Antepassado: PostResponse
Location	O URI do objeto. Tipo: string Antepassado: PostResponse

Erros especiais

Essa implementação da operação não retorna erros especiais. Para obter informações gerais sobre erros do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Exemplos

Exemplo de solicitação

```
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

Resposta de exemplo com versionamento suspenso

Veja a seguir uma resposta de exemplo quando o versionamento do bucket está suspenso:

```
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
```

Nessa resposta, o ID da versão é null.

Resposta de exemplo com versionamento habilitado

Veja a seguir uma resposta de exemplo quando o versionamento do 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
```

Recursos relacionados

- [CopyObject](#)
- [Objeto POST](#)
- [GetObject](#)

POST restauração de objeto

Descrição

Essa operação executa os seguintes tipos de solicitações:

- `select`: executa uma consulta de seleção em um objeto arquivado
- `restore an archive`: restaura um objeto arquivado

Para usar essa operação, você deve ter permissões para executar as ações `s3:RestoreObject` e `s3:GetObject`. O proprietário do bucket tem essa permissão por padrão e pode conceder essa permissão a outras pessoas. Para obter mais informações sobre as permissões, consulte [Permissões relacionadas a operações de subrecursos de bucket](#) e [Gerenciamento de permissões de acesso aos recursos do Amazon S3](#) no Guia do usuário do Amazon Simple Storage Service.

Consulta em arquivos com solicitações de seleção

Use um tipo de solicitação de seleção para executar consultas SQL em objetos arquivados. Os objetos arquivados que estão sendo consultados pela solicitação de seleção devem ser formatados como arquivos de valores separados por vírgulas (CSV) descompactados. É possível executar consultas e análises personalizadas dos dados arquivados sem precisar restaurá-los para um nível mais avançado do Amazon S3. Para uma visão geral sobre solicitações de seleção, confira [Consulta de objetos arquivados](#) no Guia do usuário do Amazon Simple Storage Service.

Ao fazer uma solicitação de seleção:

- Defina um local para a saída da consulta de seleção. Deve ser um bucket do Amazon S3 na mesma Região da AWS que o bucket que contém o objeto de arquivo que está sendo consultado. A Conta da AWS que inicia o trabalho deve ter permissões para gravação no bucket do S3. Você pode especificar a classe de armazenamento e a criptografia dos objetos de saída armazenados no bucket. Para obter mais informações sobre saída, confira [Consulta de objetos arquivados](#) no Guia do usuário do Amazon Simple Storage Service.

Para obter mais informações sobre a estrutura S3 no corpo da solicitação, consulte:

- [PutObject](#)
- [Gerenciamento de acesso com ACLs](#) no Guia do usuário do Amazon Simple Storage Service

- [Proteção de dados usando criptografia do lado do servidor](#) no Guia do usuário do Amazon Simple Storage Service
- Defina a expressão SQL para o tipo de restauração SELECT da sua consulta na estrutura `SelectParameters` do corpo da solicitação. É possível usar expressões como os exemplos a seguir.
- A expressão a seguir retorna todos os registros do objeto especificado.

```
SELECT * FROM Object
```

- Supondo que você não está usando nenhum cabeçalho para dados armazenados no objeto, é possível especificar colunas com cabeçalhos posicionais.

```
SELECT s._1, s._2 FROM Object s WHERE s._3 > 100
```

- Se houver cabeçalhos e você definir `fileHeaderInfo` na estrutura CSV no corpo da solicitação para `USE`, especifique cabeçalhos na consulta. (Se você definir o campo `fileHeaderInfo` para `IGNORE`, a primeira linha será ignorado na consulta.) Você não pode combinar posições ordinais com nomes de coluna de cabeçalho.

```
SELECT s.Id, s.FirstName, s.SSN FROM S3Object s
```

Para obter mais informações sobre o uso do SQL com a restauração do Amazon Glacier Select, consulte [Referência de SQL para o Amazon S3 Select e o Amazon Glacier Select](#) no Guia do usuário do Amazon Simple Storage Service.

Ao fazer uma solicitação de seleção, você também pode fazer o seguinte:

- Para agilizar as consultas, especifique o nível `Expedited`. Para obter mais informações sobre níveis, consulte "Restauração de arquivos", mais adiante neste tópico.
- Especifique detalhes sobre o formato de serialização de dados do objeto de entrada que está sendo consultado e da serialização dos resultados de consulta codificados por CSV.

Confira a seguir outros fatos importantes sobre o recurso de seleção:

- Os resultados de saída são novos objetos do Amazon S3. Diferente das recuperações de arquivo, eles são armazenados até serem excluídos de modo explícito, seja manualmente ou por meio de uma política de ciclo de vida.

- Você pode emitir mais de uma solicitação de seleção no mesmo objeto do Amazon S3. O Amazon S3 não desduplica solicitações; portanto, evite emitir solicitações duplicadas.
- O Amazon S3 aceita uma solicitação de seleção mesmo que o objeto já tenha sido restaurado. Uma solicitação de seleção não retorna uma resposta de erro 409.

Restauração de arquivos

Objetos nas classes de armazenamento GLACIER e DEEP_ARCHIVE são arquivados. Para acessar um objeto arquivado, você deve iniciar uma solicitação de restauração primeiro. Isso restaura uma cópia temporária do objeto arquivado. Na solicitação de restauração, especifique o número de dias que você deseja que a cópia restaurada exista. Após o período especificado, o Amazon S3 exclui a cópia temporária, mas o objeto permanece arquivado na classe de armazenamento GLACIER ou DEEP_ARCHIVE da qual o objeto foi restaurado.

Para restaurar uma versão específica do objeto, você pode fornecer um ID de versão. Se você não fornecer um ID de versão, o Amazon S3 restaurará a versão atual.

O tempo necessário para concluir as tarefas de restauração depende da classe de armazenamento da qual o objeto está sendo restaurado e do nível de acesso aos dados que você especificar.

Ao restaurar um objeto arquivado (ou usar uma solicitação de seleção), você pode especificar uma das seguintes opções de nível de acesso aos dados no elemento `Tier` do corpo da solicitação:

- **Expedited** - as recuperações expressas permitem acessar rapidamente os dados armazenados na classe de armazenamento GLACIER quando forem necessárias solicitações urgentes ocasionais para um subconjunto de arquivos. Salvo para os maiores objetos arquivados (250 MB+), os dados acessados usando recuperações expressas ficam normalmente disponíveis dentro de 1 a 5 minutos. A capacidade provisionada garante que sua capacidade de recuperação para recuperações expressas esteja disponível quando você precisar dela. As recuperações expressas e a capacidade provisionada não estão disponíveis para a classe de armazenamento DEEP_ARCHIVE.
- **Standard** - As recuperações padrão permitem que você acesse alguns de seus objetos arquivados em algumas horas. Esta é a opção padrão para solicitações de recuperação do GLACIER e DEEP_ARCHIVE que não especificam a opção de recuperação. As recuperações padrão geralmente são concluídas em 3 a 5 horas a partir da classe de armazenamento GLACIER e em 12 horas a partir da classe de armazenamento DEEP_ARCHIVE.

- **Bulk:** as recuperações em massa são a opção de recuperação de menor custo do Amazon Glacier, permitindo recuperar grandes quantidades de dados, até mesmo petabytes, com um custo baixo em um dia. As recuperações em massa geralmente são concluídas em 5 a 12 horas a partir da classe de armazenamento GLACIER e em 48 horas a partir da classe de armazenamento DEEP_ARCHIVE.

Para obter mais informações sobre opções de recuperação de arquivo e capacidade provisionada para acesso aos dados Expedited, consulte [Restauração de objetos arquivados](#) no Guia do desenvolvedor do Amazon Simple Storage Service.

Você pode usar a atualização rápida de restauração do Amazon S3 para alterar a velocidade de restauração para uma mais rápida enquanto o processo estiver em andamento. Atualize a velocidade de uma restauração em andamento emitindo outra solicitação de restauração para o mesmo objeto, definindo um novo elemento de solicitação Tier. Ao emitir uma solicitação para atualizar o nível de restauração, você deve escolher um nível mais rápido do que o nível de restauração do andamento. Você não deve alterar outros parâmetros, como o elemento de solicitação Days. Para obter mais informações, consulte [Atualização da velocidade de uma restauração em andamento](#) no Guia do usuário do Amazon Simple Storage Service.

Para obter o status da restauração do objeto, você pode enviar uma solicitação HEAD. As operações retornam o cabeçalho x-amz-restore, que fornece informações sobre o status da restauração, na resposta. Você pode usar as notificações de eventos do Amazon S3 para receber notificações quando uma restauração for iniciada ou concluída. Para obter mais informações, consulte [Configuração de notificações de eventos do Amazon S3](#) no Guia do usuário do Amazon Simple Storage Service.

Depois de restaurar um objeto arquivado, você pode atualizar o período de restauração reemitindo a solicitação com um novo período. O Amazon S3 atualiza o período de restauração em relação à hora atual e cobra somente pela solicitação. Não há cobrança de transferência de dados. Você não pode atualizar o período de restauração quando o Amazon S3 está processando ativamente sua solicitação de restauração atual para o objeto.

Se o bucket tiver uma configuração de ciclo de vida com uma regra que inclui uma ação de expiração, a expiração do objeto substituirá a vida útil especificada em uma solicitação de restauração. Por exemplo, se você restaurar uma cópia do objeto por 10 dias, mas o objeto estiver programado para expirar em 3 dias, o Amazon S3 excluirá o objeto em 3 dias. Para obter mais informações sobre a configuração do ciclo de vida, consulte [PutBucketLifecycleConfiguration](#) e [Gerenciamento do ciclo de vida do objeto](#) no Guia do usuário do Amazon Simple Storage Service.

Solicitações

Sintaxe

```
POST /ObjectName?restore&versionId=VersionID HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Autenticação de solicitações \(AWS Signature Version 4\))
Content-MD5: MD5

request body
```

Note

A sintaxe mostra alguns cabeçalhos da solicitação. Para obter uma lista completa, consulte “Cabeçalhos de solicitação”, mais adiante neste tópico.

Parâmetros da solicitação

Essa implementação da operação não usa parâmetros de solicitação.

Cabeçalhos de solicitação

Name (Nome)	Descrição	Obrigatório
Content-MD5	O resumo MD5 de 128 bits com codificação base64 dos dados. Você deve usar esse cabeçalho como uma verificação de integridade da mensagem para verificar se o corpo da solicitação não foi corrompido em trânsito. Para obter mais informações, consulte a RFC 1864. Tipo: string Padrão: nenhum	Sim

Elementos da solicitação

Veja a seguir um exemplo em XML de um corpo de solicitação para restaurar um arquivo.

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Bulk</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

A tabela a seguir explica o XML para restauração de arquivo no corpo da solicitação.

Name (Nome)	Descrição	Obrigatório
RestoreRequest	Contêiner para informações de restauração. Tipo: Container	Sim
Days	Vida útil da cópia restaurada (ativa). O número mínimo de dias em que você pode restaurar um objeto do Amazon Glacier é 1. Depois que a cópia do objeto atinge a vida útil especificada, o Amazon S3 a remove do bucket. Se você estiver restaurando um arquivo, esse elemento será necessário. Não use esse elemento com um tipo de solicitação SELECT. Tipo: número inteiro positivo Antepassados: RestoreRequest	Sim, se estiver restaurando um arquivo
GlacierJobParameters	Contêiner para parâmetros de trabalho da classe de armazenamento do S3 Glacier. Não use esse elemento com um tipo de solicitação SELECT. Tipo: Container	Não

Name (Nome)	Descrição	Obrigatório
	Antepassados: RestoreRequest	
Tier	O nível de acesso aos dados a ser usado ao restaurar o arquivo. Standard é o padrão. Tipo: Enum Valores válidos: Expedited Standard Bulk Antepassados: GlacierJobParameters	Não

O XML a seguir é o corpo da solicitação para uma consulta de seleção em um objeto arquivado:

```
<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>
</RestoreRequest>
```

```

</SelectParameters>
<OutputLocation>
  <S3>
    <BucketName>Name of bucket</BucketName>
    <Prefix>Key prefix</Prefix>
    <CannedACL>Canned ACL string</CannedACL>
    <AccessControlList>
      <Grantee>
        <Type>Grantee Type</Type>
      </Grantee>
      <ID>Grantee identifier</ID>
      <URI>Grantee URI</URI>
      <Permission>Granted permission</Permission>
      <DisplayName>Display Name</DisplayName>
      <EmailAddress>email</EmailAddress>
    </AccessControlList>
    <Encryption>
      <EncryptionType>Encryption type</EncryptionType>
    </Encryption>
    <KMSKeyId>KMS Key ID</KMSKeyId>
    <KMSContext>Base64-encoded JSON<KMSContext>
    </KMSContext>
    <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>

```

As tabelas a seguir explicam o XML para um tipo de restauração SELECT no corpo da solicitação.

Name (Nome)	Descrição	Obrigatório
RestoreRequest	Contêiner para informações de restauração. Tipo: Container	Sim
Tier	O nível de acesso aos dados a ser usado ao restaurar o arquivo. Standard é o padrão. Tipo: Enum Valores válidos: Expedited Standard Bulk Antepassados: RestoreRequest	Não
Description	A descrição opcional da solicitação. Tipo: string Antepassados: RestoreRequest	Não
SelectParameters	Descreve os parâmetros da solicitação de trabalho de seleção. Tipo: Container Antepassados: RestoreRequest	Sim, se o tipo de solicitação for SELECT
OutputLocation	Descreve o local que recebe os resultados da solicitação de restauração de seleção. Tipo: contêiner para Amazon S3 Antepassados: RestoreRequest	Sim, se o tipo de solicitação for SELECT

O elemento de contêiner **SelectParameters** possui os seguintes elementos.

Name (Nome)	Descrição	Obrigatório
Expression	A expressão SQL. Por exemplo: - A expressão SQL a seguir recupera a primeira coluna dos dados do objeto armazenado no formato CSV: <pre>SELECT s._1 FROM Object s</pre> - A expressão SQL a seguir retorna tudo do objeto: <pre>SELECT * FROM Object</pre> Tipo: string Antepassados: <code>SelectParameters</code>	Sim
ExpressionType	Identifica o tipo de expressão. Tipo: string Valores válidos: SQL Antepassados: <code>SelectParameters</code>	Sim
InputSerialization	Descreve o formato de serialização do objeto. Tipo: contêiner para CSV Antepassados: <code>SelectParameters</code>	Sim
OutputSerialization	Descreve como os resultados do trabalho de seleção são serializados. Tipo: contêiner para CSV	Sim

Name (Nome)	Descrição	Obrigatório
	Antepassados: <code>SelectParameters</code>	

O elemento de contêiner CSV no elemento **InputSerialization** contém os seguintes elementos.

Name (Nome)	Descrição	Obrigatório
<code>RecordDelimiter</code>	Um único caractere usado para separar registros individuais na entrada. Em vez do valor padrão, você pode especificar um delimitador arbitrário. Tipo: <code>string</code> Padrão: <code>\n</code> Antepassados: <code>CSV</code>	Não
<code>FieldDelimiter</code>	Um único caractere usado para separar campos individuais em um registro. Você pode especificar um delimitador arbitrário. Tipo: <code>string</code> Padrão: <code>,</code> Antepassados: <code>CSV</code>	Não
<code>QuoteCharacter</code>	Um único caractere usado como escape quando o delimitador de campo faz parte do valor. Considere este exemplo em um arquivo CSV: <code>"a, b"</code> Colocar os valores entre aspas faz com que esse valor seja um campo único. Se você não usar as aspas, a vírgula será	Não

Name (Nome)	Descrição	Obrigatório
	um delimitador de campo (o que a torna dois valores de campo separados, a e b). Tipo: string Padrão: " Antepassados: CSV	
QuoteEscapeCharacter	Um único caractere usado para fazer o escape do caractere de aspas em um valor que já recebeu escape. Por exemplo, o valor "" a , b "" é analisado como " a , b ". Tipo: string Padrão: " Antepassados: CSV	Não

Name (Nome)	Descrição	Obrigatório
FileHeaderInfo	Descreve a primeira linha nos dados de entrada. É um dos valores de ENUM. • NONE: a primeira linha não é um cabeçalho. • IGNORE: a primeira linha é um cabeçalho, mas você não pode usar os valores do cabeçalho para indicar a coluna em uma expressão. Você pode usar a posição da coluna (como <code>_1</code>, <code>_2</code> etc.) para indicar a coluna (<code>SELECT s._1 FROM OBJECT s</code>). • Use: a primeira linha é um cabeçalho e você pode usar o valor do cabeçalho para identificar uma coluna em uma expressão (<code>SELECT "name" FROM OBJECT</code>). Tipo: Enum Valores válidos: NONE USE IGNORE Antepassados: CSV	Não
Comments	Um único caractere usado para indicar que uma linha deve ser ignorada quando o caractere está presente no início da linha. Você pode especificar qualquer caractere para indicar uma linha de comentário. Tipo: string Antepassados: CSV	Não

O elemento de contêiner CSV (nos elementos **OutputSerialization**) contém os seguintes elementos.

Name (Nome)	Descrição	Obrigatório
QuoteFields	Indica se devem ser usadas aspas nos campos de saída. • ALWAYS: sempre use aspas nos campos de saída. • ASNEEDED: use aspas nos campos de saída quando necessário. Tipo: Enum Valores válidos: ALWAYS ASNEEDED Padrão: AsNeeded Antepassados: CSV	Não
RecordDelimiter	Um único caractere usado para separar registros individuais na saída. Em vez do valor padrão, você pode especificar um delimitador arbitrário. Tipo: string Padrão: \n Antepassados: CSV	Não
FieldDelimiter	Um único caractere usado para separar campos individuais em um registro. Você pode especificar um delimitador arbitrário. Tipo: string Padrão: ,	Não

Name (Nome)	Descrição	Obrigatório
	Antepassados: CSV	
QuoteCharacter	Um único caractere usado como escape quando o delimitador de campo faz parte do valor. Por exemplo, se o valor for a, b, o Amazon S3 colocará esse valor de campo entre aspas, do seguinte modo: " a , b " . Tipo: string Padrão: " Antepassados: CSV	Não
QuoteEscapeCharacter	Um único caractere usado para fazer o escape do caractere de aspas em um valor que já recebeu escape. Por exemplo, se o valor for " a , b " , o Amazon S3 colocará o valor entre aspas, do seguinte modo: "" " a , b "" " . Tipo: string Antepassados: CSV	Não

O elemento de contêiner do S3 (no elemento **OutputLocation**) contém os seguintes elementos.

Name (Nome)	Descrição	Obrigatório
AccessControlList	Uma lista de concessões que controlam o acesso aos resultados intermediários. Tipo: contêiner para concessão Antepassados: S3	Não
BucketName	O nome do bucket do S3 no qual os resultados da restauração de seleção são armazenados. O bucket deve estar na	Sim

Name (Nome)	Descrição	Obrigatório
	mesma Região da AWS do bucket que contém o objeto de arquivo de entrada. Tipo: string Antepassados: S3	
CannedACL	A lista de controle de acesso (ACL) pré-configurada a ser aplicada aos resultados da restauração de seleção. Tipo: string 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> Antepassados: S3	Não
Encryption	Contém informações de criptografia dos resultados armazenados. Tipo: contêiner para criptografia Antepassados: S3	Não
Prefix	O prefixo que precede os resultados da restauração de seleção. O tamanho máximo do prefixo é 512 bytes. Tipo: string Antepassados: S3	Sim

Name (Nome)	Descrição	Obrigatório
StorageClass	A classe de armazenamento usada para armazenar os resultados da restauração de seleção. Tipo: string Valores válidos: STANDARD REDUCED_REDUNDANCY STANDARD_IA ONEZONE_IA Antepassados: S3	Não
Tagging	Contêiner para informações de tags. Tipo: estrutura Tag Antepassados: S3	Não
UserMetadata	Contém uma lista de metadados a ser armazenada com os resultados da restauração de seleção. Tipo: estrutura MetadataEntry Antepassados: S3	Não

O elemento do contêiner Grantee (no elemento **AccessControlList**) contém os seguintes elementos.

Name (Nome)	Descrição	Obrigatório
DisplayName	O nome da tela do favorecido. Tipo: string Antepassados: Grantee	Não
EmailAddress	O endereço de e-mail do favorecido.	Não

Name (Nome)	Descrição	Obrigatório
ID	Tipo: string	Não
	Antepassados: Grantee	
	O ID de usuário canônico do favorecido.	
Type	Tipo: string	Não
	Antepassados: Grantee	
	O tipo do favorecido.	
URI	Tipo: string	Não
	Antepassados: Grantee	
	O URI do grupo do favorecido.	
Permission	Tipo: string	Não
	Antepassados: Grantee	
	Permissão concedida.	

O elemento de contêiner **Encryption** (no S3) contém os seguintes elementos.

Name (Nome)	Descrição	Obrigatório
EncryptionType	O algoritmo de criptografia no lado do servidor usado para armazenar os resultados do trabalho. O padrão é sem criptografia.	Não

Name (Nome)	Descrição	Obrigatório
	Tipo: string Valores válidos aws : kms AES256 Antepassados: Encryption	
KMSContext	Opcional. Se o tipo de criptografia for aws : kms, você poderá usar esse valor para especificar o contexto da criptografia nos resultados da restauração de seleção. Tipo: string Antepassados: criptografia	Não
KMSKeyId	O ID da chave do AWS Key Management Service (AWS KMS) a ser usado na criptografia do objeto. Tipo: string Antepassados: Encryption	Não

O elemento de contêiner **TagSet** (no elemento **Tagging**) contém o seguinte elemento.

Name (Nome)	Descrição	Obrigatório
Tag	Contém tags. Tipo: Container Antepassados: TagSet	Não

O elemento de contêiner Tag (no elemento **TagSet**) contém os seguintes elementos.

Name (Nome)	Descrição	Obrigatório
Key		Não

Name (Nome)	Descrição	Obrigatório
	Nome da tag. Tipo: string Antepassados: Tag	
Value	Valor da tag. Tipo: string Antepassados: Tag	Não

O elemento de contêiner **MetadataEntry** (no elemento **UserMetadata**) contém os seguintes elementos de par de chave-valor a serem armazenados com um objeto.

Name (Nome)	Descrição	Obrigatório
MetadataKey	A chave de metadados. Tipo: string Antepassados:	Não
MetadataEntry	O valor dos metadados. Tipo: string Antepassados:	Não

Respostas

Uma operação bem-sucedida retorna o código de status 200 OK ou 202 Accepted.

- Se a cópia do objeto não tiver sido restaurada anteriormente, o Amazon S3 retornará 202 Accepted na resposta.

- Se a cópia do objeto tiver sido restaurada anteriormente, o Amazon S3 retornará 200 OK na resposta.

Cabeçalhos de resposta

Essa implementação da operação usa somente os cabeçalhos de resposta que são comuns à maioria das respostas. Para obter mais informações, consulte [Cabeçalhos de resposta comuns](#).

Elementos de resposta

Essa operação não retorna elementos de resposta.

Erros especiais

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
RestoreAlreadyInProgress	A restauração do objeto já está em andamento. (Esse erro não se aplica às solicitações do tipo SELECT.)	409 Conflito	Cliente
GlacierExpeditedRetrievalNotAvailable	As recuperações rápidas da classe de armazenamento do S3 Glacier não estão disponíveis no momento. Tente novamente mais tarde. (Retornado se não houver capacidade suficiente para processar a solicitação Expedited. Esse erro se aplica somente às recuperações Expedited e não às recuperações Standard ou Bulk.)	503	N/D

Exemplos

Restaurar um objeto por dois dias usando a opção de recuperação rápida

A solicitação de restauração a seguir restaura uma cópia do objeto `photo1.jpg` do Amazon Glacier por um período de dois dias usando a opção de recuperação rápida.

```
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>
```

Se o `examplebucket` não tiver uma cópia restaurada do objeto, o Amazon S3 retornará a seguinte resposta `202 Accepted`.

```
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
```

Se uma cópia do objeto já tiver sido restaurada, o Amazon S3 retornará uma resposta `200 OK` e atualizará somente o tempo de expiração da cópia restaurada.

Consultar um arquivo com uma solicitação SELECT

Este é um exemplo de resposta de restauração de seleção.

```
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>
```

O Amazon S3 retorna a seguinte resposta 202 Accepted.

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/cHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZ1zYQvPiBlZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
x-amz-restore-output-path: js-test-s3/qE8nk5M0XIj-LuZE2HXNw6empQm3znLkH1MWInRYPs-
0r12W0uj6LyYm-neTvm1-btz3wbBxfMhPykd3jk1-lvZE7w42/
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

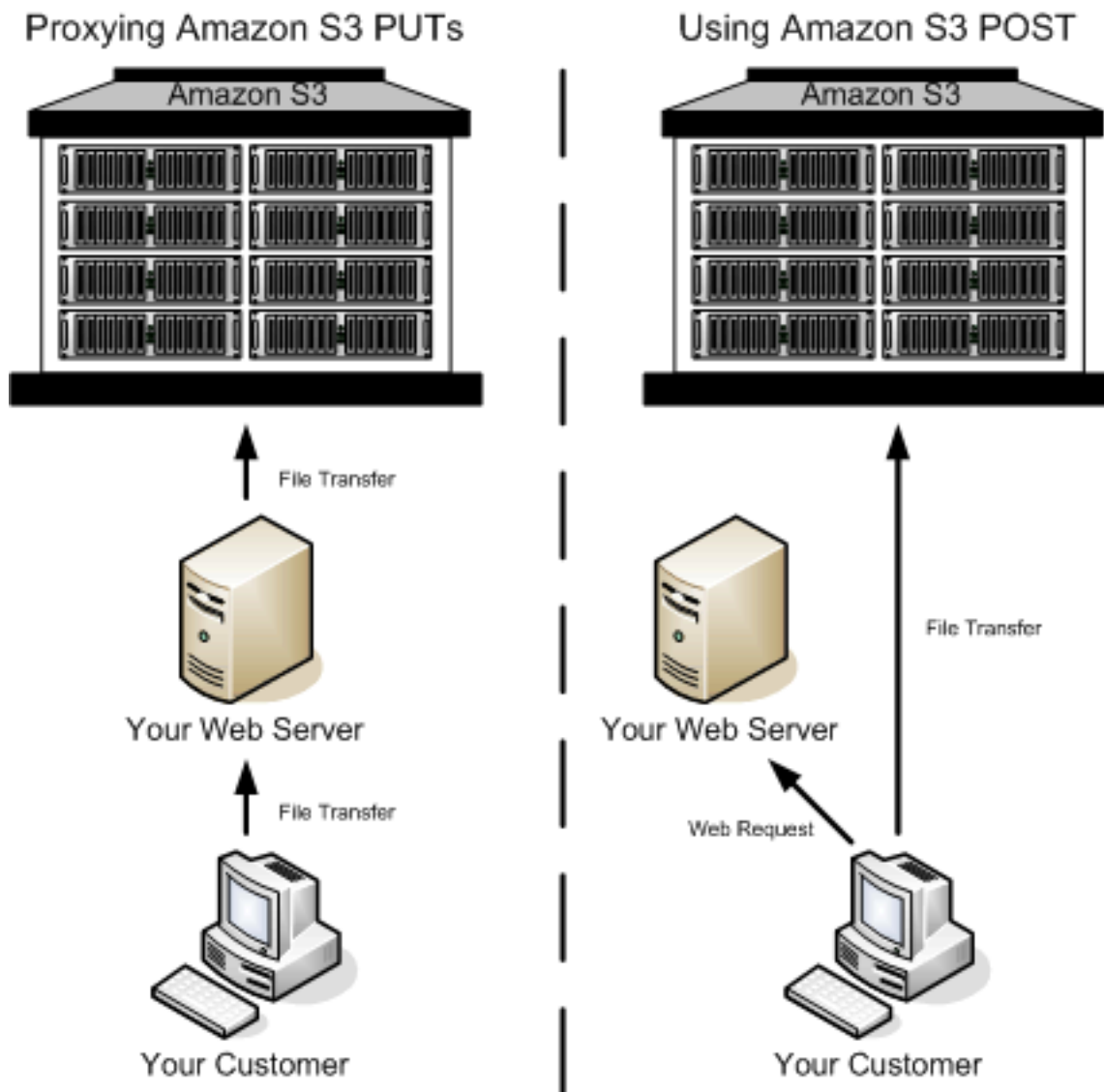
Mais informações

- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)
- [Referência de SQL para Seleção do Amazon S3 e Seleção do S3 Glacier](#) no Guia do desenvolvedor do Amazon Simple Storage Service

Uploads baseados no navegador usando HTTP POST

O Amazon S3 é compatível com solicitações HTTP POST para que os usuários possam fazer upload de conteúdo diretamente no Amazon S3. Ao usar POST, os usuários finais podem autenticar solicitações sem precisar transmitir dados por meio de um nó intermediário seguro que protege suas credenciais. Assim, o HTTP POST tem o potencial de reduzir a latência.

A figura a seguir mostra um upload do Amazon S3 usando uma solicitação POST.



1. O usuário acessa sua página em um navegador da Web.
2. A página da Web tem um formulário HTML que contém toda as informações necessárias para que o usuário faça upload de conteúdo no Amazon S3.
3. O usuário faz upload do conteúdo no Amazon S3 por meio do navegador da Web.

O processo para enviar solicitações POST baseadas no navegador é o seguinte:

1. Crie uma política de segurança especificando condições que restrinjam o que você deseja permitir na solicitação, como o nome do bucket em que os objetos podem ser carregados e os prefixos do nome da chave que você deseja permitir para o objeto que está sendo criado.

2. Crie uma assinatura com base na política. Para solicitações autenticadas, o formulário deve incluir uma assinatura válida e a política.
3. Crie um formulário HTML que os usuários possam acessar para fazer upload de objetos no bucket do Amazon S3.

A seção a seguir descreve como criar uma assinatura para autenticar uma solicitação. Para obter informações sobre como criar formulários e políticas de segurança, consulte [Criação de um formulário HTML \(usando o AWS Signature Version 4\)](#).

Cálculo de uma assinatura

Para solicitações autenticadas, o formulário HTML deve incluir campos para uma política de segurança e uma assinatura.

- Uma política de segurança (consulte [Política de POST](#)) controla o que é permitido na solicitação.
- A política de segurança é a `StringToSign` (consulte [Introdução à assinatura de solicitações](#)) no cálculo da assinatura.



Para calcular uma assinatura

1. Crie uma política usando a codificação UTF-8.
2. Converta os bytes da política com codificação UTF-8 em base64. O resultado é a `StringToSign`.
3. Crie uma chave de assinatura.
4. Use a chave de assinatura para assinar a `StringToSign` usando o algoritmo de assinatura HMAC-SHA256.

Para obter mais informações sobre a criação de formulários HTML, políticas de segurança e um exemplo, consulte:

- [Criação de um formulário HTML \(usando o AWS Signature Version 4\)](#)
- [Política de POST](#)
- [Exemplo: upload baseado no navegador usando HTTP POST \(usando o AWS Signature Version 4\)](#)

Criação de um formulário HTML (usando o AWS Signature Version 4)

Tópicos

- [Declaração de formulário HTML](#)
- [Campos do formulário HTML](#)

Para permitir que os usuários façam upload de conteúdo no Amazon S3 usando os navegadores (solicitações HTTP POST), use formulários HTML. Os formulários HTML são formados por uma declaração de formulário e campos de formulário. A declaração de formulário contém informações de alto nível sobre a solicitação. Os campos do formulário contêm informações detalhadas da solicitação.

Esta seção descreve como criar formulários HTML. Para obter um exemplo prático de upload baseado no navegador usando HTTP POST e cálculos de assinatura relacionados para autenticação

de solicitações, consulte [Exemplo: upload baseado no navegador usando HTTP POST \(usando o AWS Signature Version 4\)](#).

O formulário e a política devem ser codificados em UTF-8. Aplique a codificação UTF-8 no formulário especificando `charset=UTF-8` no atributo `content`. Confira a seguir um exemplo de codificação UTF-8 no cabeçalho HTML.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

Confira a seguir um exemplo de codificação UTF-8 em um cabeçalho de solicitação.

```
Content-Type: text/html; charset=UTF-8
```

Note

Os dados e os limites do formulário (excluindo o conteúdo do arquivo) não podem exceder 20 KB.

Declaração de formulário HTML

A declaração do formulário HTML tem os três atributos a seguir:

- **action:** o URL que processa a solicitação, que deve ser definido como o URL do bucket. Por exemplo, se o nome do seu bucket for `examplebucket`, o URL será `http://examplebucket.s3.amazonaws.com/`.

Note

O nome chave é especificado em um campo do formulário.

- **method:** o método deve ser POST.

- `enctype`: o tipo de compartimento (`enctype`) deve ser definido como `multipart/form-data` para uploads de arquivos e de áreas de texto. Para obter mais informações sobre `enctype`, consulte [RFC 1867](#).

Essa é uma declaração de formulário para o bucket `examplebucket`.

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

Campos do formulário HTML

A tabela a seguir descreve uma lista de campos que podem ser usados em um formulário. Entre outros campos, há um campo de assinatura que você pode usar para autenticar solicitações. Há campos para você especificar o algoritmo de cálculo da assinatura (`x-amz-algorithm`), o escopo da credencial (`x-amz-credential`) que você usou para gerar a chave de assinatura e a data (`x-amz-date`) usada para calcular a assinatura. O Amazon S3 usa essas informações para recriar a assinatura. Se as assinaturas corresponderem, o Amazon S3 processará a solicitação.

Note

A variável `${filename}` é substituída automaticamente pelo nome do arquivo fornecido pelo usuário e é reconhecida por todos os campos do formulário. Se o navegador ou o cliente fornecer um caminho completo ou parcial para o arquivo, apenas o texto que vem depois da última barra (/) ou barra invertida (\) será usado (por exemplo, `C:\Program Files\directory1\file.txt` será interpretado como `file.txt`). Se nenhum arquivo ou nome de arquivo for fornecido, a variável será substituída por uma string vazia.

Se você não fornecer os elementos necessários para solicitações autenticadas, como o elemento `policy`, a solicitação será considerada anônima e só será bem-sucedida se você tiver configurado o bucket para leitura e gravação públicas.

Nome do elemento	Descrição	Obrigatório
<code>acl</code>	Uma lista de controle de acesso (ACL) do Amazon S3. Se uma ACL inválida for especific	Não

Nome do elemento	Descrição	Obrigatório
	ada, o Amazon S3 negará a solicitação. Consulte mais informações sobre ACLs em Uso de ACLs do Amazon S3. Tipo: string Padrão: 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	Cabeçalhos específicos para REST. Para obter mais informações, consulte PutObject .	Não
Content-Type		
Content-Disposition		
Content-Encoding		
Expires		
key	O nome da chave do objeto carregado. Para usar o nome de arquivo fornecido pelo usuário, use a variável <code>\${filename}</code>. Por exemplo, se você fizer upload de um arquivo <code>photo1.jpg</code> e especificar <code>/user/user1/\${filename}</code> como nome da chave, o arquivo será armazenado como <code>/user/user1/photo1.jpg</code>. Para obter mais informações, consulte Chave e metadados de objeto no Guia do usuário do Amazon Simple Storage Service.	Sim

Nome do elemento	Descrição	Obrigatório
<code>policy</code>	Política de segurança com codificação base64 que descreve o que é permitido na solicitação. Para solicitações autenticadas, é necessária uma política. As solicitações sem uma política de segurança são consideradas anônimas e terão sucesso apenas em buckets com gravação pública.	Obrigatório para solicitações autenticadas
<code>success_action_redirect</code>	O URL para o qual o cliente é redirecionado após um upload bem-sucedido. Se <code>success_action_redirect</code> não for especificado ou o Amazon S3 não conseguir interpretar o URL, o Amazon S3 retornará o tipo de documento vazio especificado no campo <code>success_action_status</code>. Se houver falha no upload, o Amazon S3 retornará um erro e não redirecionará o usuário para outro URL.	Não

Nome do elemento	Descrição	Obrigatório
success_action_status	O código de status retornado ao cliente após o upload bem-sucedido se <code>success_action_redirect</code> não for especificado. Os valores válidos são 200, 201 ou 204 (padrão). Se o valor estiver definido como 200 ou 204, o Amazon S3 retornará um documento vazio com o código de status especificado. Se o valor estiver definido como 201, o Amazon S3 retornará um documento XML com um código de status 201. Para obter informações sobre o conteúdo do documento XML, consulte Objeto POST. Se o valor não estiver definido ou for inválido, o Amazon S3 retornará um documento vazio com um código de status 204.  Note Algumas versões do Adobe Flash Player não lidam muito bem com respostas HTTP com um corpo vazio. Para oferecer suporte a uploads por meio do Adobe Flash, recomendamos definir <code>success_action_status</code> como 201.	Não

Nome do elemento	Descrição	Obrigatório
x-amz-algorithm	O algoritmo de assinatura usado para autenticar a solicitação. Para AWS Signature Version 4, o valor é AWS4-HMAC-SHA256 . Este campo é necessário se um documento de política estiver incluso na solicitação.	Obrigatório para solicitações autenticadas
x-amz-credential	Além do ID da chave de acesso, esse campo também fornece informações de escopo que identificam a região e o serviço para os quais a assinatura é válida. Deve ser o mesmo escopo usado no cálculo da chave de assinatura para o cálculo da assinatura. É uma string da seguinte forma: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Por exemplo: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Para o Amazon S3, a string aws-service é s3. Para conferir a lista de strings aws-region do Amazon S3, consulte Regiões e endpoints na Referência geral da AWS. Será obrigatório se um documento de política estiver incluso na solicitação.	Obrigatório para solicitações autenticadas

Nome do elemento	Descrição	Obrigatório
x-amz-date	É o valor de data no formato ISO8601. Por exemplo, 20130728T000000Z . É a mesma data que você usou para criar a chave de assinatura (por exemplo, 20130728). Também deve ser o mesmo valor fornecido na política (x-amz-date) que você assinou. Será obrigatório se um documento de política estiver incluso na solicitação.	Obrigatório para solicitações autênticas das
x-amz-security-token	Um token de segurança usado pelo Amazon DevPay e por credenciais de sessão Se a solicitação estiver usando o Amazon DevPay, serão necessários dois campos do formulário x-amz-security-token : um para o token de produto e outro para o token de usuário. Para obter mais informações, consulte Uso do DevPay no Guia do usuário do Amazon Simple Storage Service. Se a solicitação estiver usando credenciais de sessão, será necessário um formulário x-amz-security-token . Para obter mais informações, consulte Solicitação de credenciais de segurança temporárias no Manual do usuário do IAM.	Não
x-amz-signature	(AWS Signature Version 4) O hash HMAC-SHA256 da política de segurança. Este campo é necessário se um documento de política estiver incluso na solicitação.	Obrigatório para solicitações autênticas das

Nome do elemento	Descrição	Obrigatório
x-amz-meta-*	Os nomes de campo que começam com esse prefixo são metadados definidos pelo usuário. Cada um é armazenado e retornado como um conjunto de pares de chave-valor. O Amazon S3 não valida nem interpreta metadados definidos pelo usuário. Para obter mais informações, consulte PutObject .	Não
x-amz-*	Consulte Objeto POST (Objeto POST para outros cabeçalhos x-amz-*).	Não
file	Conteúdo de arquivo ou texto. O arquivo ou conteúdo deve ser o último campo no formulário. Não carregue mais de um arquivo por vez.	Sim

Itens condicionais são necessários para solicitações autenticadas e são opcionais para solicitações anônimas.

Agora que você já sabe como criar formulários, vai aprender a criar uma política de segurança que possa ser assinada. Para obter mais informações, consulte [Política de POST](#).

Política de POST

Tópicos

- [Expiração](#)
- [Correspondência de condição](#)
- [Condições](#)
- [Caracteres de escape](#)

A política necessária para fazer solicitações autenticadas usando HTTP POST é um documento com codificação UTF-8 e base64 gravado em JavaScript Object Notation (JSON) que especifica as condições que a solicitação deve satisfazer. Dependendo de como você estrutura o documento de política, é possível controlar a granularidade do acesso por upload, por usuário, para todos os uploads ou de acordo com outras configurações que atendam às suas necessidades.

Esta seção descreve a política de POST. Para conferir exemplos de cálculos de assinatura usando a política de POST, consulte [Exemplo: upload baseado no navegador usando HTTP POST \(usando o AWS Signature Version 4\)](#).

Note

Embora o documento de política seja opcional, recomendamos que você use um para controlar o que é permitido na solicitação. Se você tornar o bucket publicamente gravável, não terá nenhum controle sobre quais usuários podem gravar no seu bucket.

Veja a seguir um exemplo de um documento de política de POST.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

A política de POST sempre contém os elementos `expiration` e `conditions`. O exemplo de política usa dois tipos de correspondência de condições (correspondência exata e correspondência inicial). As seções a seguir descrevem esses elementos.

Expiração

O elemento `expiration` especifica a data e a hora de expiração da política de POST no formato de data GMT ISO8601. Por exemplo, `2013-08-01T12:00:00.000Z` especifica que a política de POST não é válida após a meia-noite GMT de 1º de agosto de 2013.

Correspondência de condição

Veja a seguir uma tabela que descreve os tipos de correspondência de condições que você pode usar para especificar as condições da política de POST (descritas na próxima seção). Embora seja necessário especificar pelo menos uma condição para cada campo especificado no formulário, é possível criar critérios de correspondência mais complexos especificando várias condições para um campo.

Tipo de correspondência de condição	Descrição
Correspondências exatas	O valor do campo do formulário deve corresponder ao valor especificado. Este exemplo indica que a ACL deve ser definida como pública para leitura: <pre>{"acl": "public-read" }</pre> Este exemplo é uma forma alternativa para indicar que a ACL deve ser definida como pública para leitura: <pre>["eq", "\$acl", "public-read"]</pre>
Inicia com	O valor deve começar com o valor especificado. Este exemplo indica que a chave do objeto deve iniciar com user/user1: <pre>["starts-with", "\$key", "user/user1/"]</pre>
Tipos de conteúdo correspondentes em uma lista separada por vírgulas	Os valores de tipos de conteúdo para uma condição <code>starts-with</code> que inclui vírgulas são interpretados como listas. Cada valor na lista deve atender à condição para que toda a condição seja aprovada. Por exemplo, dada a seguinte condição: <pre>["starts-with", "\$Content-Type", "image/"]</pre>

Tipo de correspondência de condição	Descrição
	O valor a seguir passaria pela condição: <pre>"image/jpg,image/png,image/gif"</pre> O valor a seguir não passaria pela condição: <pre>["image/jpg,text/plain"]</pre>  Note Os elementos de dados diferentes de <code>Content-Type</code> são tratados como strings, independentemente da presença de vírgulas.
Corresponder qualquer conteúdo	Para configurar a política de POST para permitir qualquer conteúdo em um campo de formulário, use <code>starts-with</code> com um valor vazio (<code>""</code>). Este exemplo permite qualquer valor para <code>success_action_redirect</code> : <pre>["starts-with", "\$success_action_redirect", ""]</pre>
Especificar intervalos	Para os campos de formulário que aceitam intervalos, separe os limites superior e inferior com uma vírgula. Este exemplo permite um tamanho de arquivo entre 1 e 10 MiB: <pre>["content-length-range", 1048576, 10485760]</pre>

As condições específicas permitidas em uma política de POST estão descritas em [Condições](#).

Condições

As `conditions` em uma política de POST são uma matriz de objetos, sendo que cada um é usado para validar a solicitação. Você pode usar essas condições para restringir o que é permitido na solicitação. Por exemplo, as condições de política anteriores exigem o seguinte:

- A solicitação deve especificar o nome do bucket `johnsmith`.
- O nome da chave do objeto deve ter o prefixo `user/eric`.
- A ACL do objeto deve ser definida como `public-read`.

Cada campo especificado em um formulário (exceto `x-amz-signature`, `file`, `policy` e nomes de campos com o prefixo `x-ignore-`) deve aparecer na lista de condições.

Note

Todas as variáveis dentro do formulário são expandidas antes da validação da política de POST. Portanto, qualquer correspondência de condição deve ser realizada nos campos de formulário expandidos. Suponha que você queira restringir o nome da chave do objeto a um prefixo específico (`user/user1`). Nesse caso, você define o campo do formulário da chave como `user/user1/${filename}`. A política de POST deve ser `[ "starts-with", "$key", "user/user1/" ]` (não insira `[ "starts-with", "$key", "user/user1/${filename}" ]`). Para obter mais informações, consulte [Correspondência de condição](#).

As condições do documento de política são descritas na tabela a seguir.

Nome do elemento	Descrição
<code>acl</code>	Especifica o valor da ACL que deve ser usado no envio do formulário. Essa condição oferece suporte à correspondência exata e ao tipo de correspondência de condição <code>starts-with</code> discutido na seção a seguir.
<code>bucket</code>	

Nome do elemento	Descrição
	Especifica o nome do bucket aceitável. Essa condição oferece suporte ao tipo de correspondência de condição de correspondência exata.
<code>content-length-range</code>	Os tamanhos mínimo e máximo permitidos para o conteúdo carregado. Essa condição oferece suporte ao tipo de correspondência de condição <code>content-length-range</code> .
<code>Cache-Control</code> <code>Content-Type</code> <code>Content-Disposition</code> <code>Content-Encoding</code> <code>Expires</code>	Cabeçalhos específicos para REST. Para obter mais informações, consulte Objeto POST. Essa condição oferece suporte à correspondência exata e ao tipo de correspondência de condição <code>starts-with</code> .
<code>key</code>	O nome de chave aceitável ou um prefixo do objeto carregado. Essa condição oferece suporte à correspondência exata e ao tipo de correspondência de condição <code>starts-with</code> .
<code>success_action_redirect</code> <code>redirect</code>	O URL para o qual o cliente é redirecionado após um upload bem-sucedido. Essa condição oferece suporte à correspondência exata e ao tipo de correspondência de condição <code>starts-with</code> .

Nome do elemento	Descrição
<code>success_action_status</code>	O código de status retornado ao cliente após o upload bem-sucedido se <code>success_action_redirect</code> não for especificado. Essa condição oferece suporte à correspondência exata.
<code>x-amz-algorithm</code>	O algoritmo de assinatura que deve ser usado durante o cálculo da assinatura. Para AWS Signature Version 4, o valor é <code>AWS4-HMAC-SHA256</code>. Essa condição oferece suporte à correspondência exata.
<code>x-amz-credential</code>	As credenciais usadas para calcular a assinatura. Oferece informações de ID de chave de acesso e escopo, identificando a região e o serviço para os quais a assinatura é válida. Deve ser o mesmo escopo usado no cálculo da chave de assinatura para o cálculo da assinatura. É uma string da seguinte forma: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Por exemplo: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Para o Amazon S3, a string <code>aws-service</code> é <code>s3</code>. Para conferir a lista de strings <code>aws-region</code> do Amazon S3, consulte Regiões e endpoints na Referência geral da AWS. Este campo será necessário se um documento de política de POST estiver incluído na solicitação. Essa condição oferece suporte à correspondência exata.

Nome do elemento	Descrição
x-amz-date	O valor de data especificado na string com formato ISO8601. Por exemplo, 20130728T000000Z . A data deve ser a mesma usada na criação da chave de assinatura para o cálculo da assinatura. Este campo será necessário se um documento de política de POST estiver incluído na solicitação. Essa condição oferece suporte à correspondência exata.
x-amz-security-token	Token de segurança do Amazon DevPay. Cada solicitação que usa o Amazon DevPay requer dois campos do formulário x-amz-security-token : um para o token de produto e outro para o token de usuário. Consequentemente, os valores devem ser separados por vírgulas. Por exemplo, se o token de usuário for eW91dHViZQ== e o token de produto for b0hnNVNKWVJIQTA= , defina a entrada da política de POST como: { "x-amz-security-token": "eW91dHViZQ==,b0hnNVNKWVJIQTA=" } . Para obter mais informações sobre o Amazon DevPay, consulte Uso do DevPay no Guia do usuário do Amazon Simple Storage Service.
x-amz-meta-*	Metadados especificados pelo usuário. Essa condição oferece suporte à correspondência exata e ao tipo de correspondência de condição starts-with .

Nome do elemento	Descrição
x-amz-*	Consulte Objeto POST (Objeto POST para outros cabeçalhos x-amz-*). Essa condição oferece suporte à correspondência exata.

Note

Se o seu kit de ferramentas adicionar mais campos de formulário (por exemplo, o Flash adiciona filename), será necessário adicioná-los ao documento de política de POST. Se essa funcionalidade puder ser controlada, adicione o prefixo x-ignore- ao campo para que o Amazon S3 ignore o recurso e para que futuras versões não sejam afetadas.

Caracteres de escape

Os caracteres de escape dentro de um documento de política de POST são descritos na tabela a seguir.

Sequência de escape	Descrição
\\	Barra invertida
\\\$	Símbolo do dólar
\\b	Apagar
\\f	Feed do formulário
\\n	Nova linha

Sequência de escape	Descrição
\r	Carriage return
\t	Guia horizontal
\v	Guia vertical
\uxxxx	Todos os caracteres do Unicode

Agora que você tem familiaridade com formulários e políticas e entende como a assinatura funciona, experimente um exemplo de upload de POST. É necessário gravar o código para calcular a assinatura. O exemplo fornece um formulário de amostra e uma política de POST que você pode usar para testar seus cálculos de assinatura. Para obter mais informações, consulte [Exemplo: upload baseado no navegador usando HTTP POST \(usando o AWS Signature Version 4\)](#).

Exemplo: upload baseado no navegador usando HTTP POST (usando o AWS Signature Version 4)

Esta seção mostra um exemplo do uso de uma solicitação HTTP POST para fazer upload de conteúdo diretamente no Amazon S3.

Para obter mais informações sobre o Signature Version 4, consulte [Processo de assinatura do Signature Version 4](#).

Upload de um arquivo no Amazon S3 usando HTTP POST

Este exemplo fornece uma política de POST de exemplo e um formulário que você pode usar para fazer upload de um arquivo. Este tópico usa a política de exemplo e as credenciais fictícias para mostrar o fluxo de trabalho e a assinatura e o hash da política resultantes. Você pode usar esses dados como conjunto de testes para verificar seu código de cálculo de assinatura.

O exemplo usa as credenciais de exemplo a seguir nos cálculos da assinatura. Você pode usar essas credenciais para verificar o código de cálculo da assinatura. No entanto, você deve substituí-las por suas próprias credenciais ao enviar solicitações para a AWS.

Parâmetro	Valor
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

Política e formulário de exemplo

A política de POST a seguir é compatível com uploads no Amazon S3 com condições 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" }
]
```

Ela define as seguintes condições na solicitação:

- O upload deve ocorrer antes do meio-dia UTC em 30 de dezembro de 2015.
- O conteúdo só pode ser enviado para `sigv4examplebucket`. O bucket deve estar na região que você especificou no escopo da credencial (parâmetro do formulário `x-amz-credential`), porque a assinatura fornecida é válida somente dentro desse escopo.
- Você pode fornecer qualquer nome de chave que comece com `user/user1`. Por exemplo, `user/user1/MyPhoto.jpg`.
- A ACL deve ser definida como `public-read`.
- Se o upload for bem-sucedido, o navegador do usuário será redirecionado para `http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html`.
- O objeto deve ser um arquivo de imagem.
- A tag `x-amz-meta-uuid` deve ser definida como `14365123651274`.
- A `x-amz-meta-tag` pode conter qualquer valor.

Veja a seguir uma versão codificada em Base64 dessa política de POST. Você usa esse valor como a `StringToSign` no cálculo da assinatura.

```
eyJhZlZxhwaXJhdGlvbiI6IClYMDU1LTUyLTUwVDEyOjAwOjAwLjAwMFoiLA0KICAiY29uZGl0aW9ucyI6IFsNCiAgICB7ImJ
```

Quando você copia/cola a política anterior, ela deve ter retornos de carro e novas linhas para que o hash computado corresponda a esse valor (ou seja, texto ASCII, com terminadores de linha CRLF).

Usando credenciais de exemplo para criar uma assinatura, o valor da assinatura é o seguinte (no cálculo da assinatura, a data é igual a `x-amz-date` na política (20151229)):

```
8afdbf4008c03f22c2cd3cdb72e4afbb1f6a588f3255ac628749a66d7f09699e
```

O formulário de exemplo a seguir especifica a política de POST anterior e é compatível com uma solicitação POST para o `sigv4examplebucket`. Copie/cole o conteúdo em um editor de texto e salve-o como `exampleform.html`. Em seguida, você pode fazer upload de arquivos de imagem para o bucket específico usando o `exampleform.html`. Sua solicitação será bem-sucedida se a assinatura fornecida corresponder à assinatura calculada pelo Amazon S3.

Note

Você deve atualizar o nome do bucket, as datas, a credencial, a política e a assinatura com valores válidos para que o upload seja realizado no 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>
```

Os parâmetros de postagem não diferenciam maiúsculas de minúsculas. Por exemplo, é possível especificar `x-amz-signature` ou `X-Amz-Signature`.

Uploads baseados no navegador para o Amazon S3 usando a biblioteca do AWS Amplify

Esta seção descreve como fazer upload de arquivos no Amazon S3 usando a biblioteca JavaScript do AWS Amplify.

Para obter informações sobre como configurar a biblioteca do AWS Amplify, consulte [Instalação e configuração do AWS Amplify](#).

Uso da biblioteca JavaScript do AWS Amplify para fazer upload de arquivos no Amazon S3

O módulo `Storage` da biblioteca do AWS Amplify oferece um mecanismo simples de upload baseado no navegador para gerenciar o conteúdo do usuário no armazenamento público ou privado do Amazon S3.

Example: Configuração manual do AWS Amplify

O exemplo a seguir mostra a configuração manual para usar o módulo de armazenamento do AWS Amplify. A implementação padrão do módulo de armazenamento usa o 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: Colocar dados no Amazon S3

O exemplo a seguir mostra como put dados públicos no Amazon S3.

```
Storage.put('test.txt', 'Hello')  
  .then (result => console.log(result))  
  .catch(err => console.log(err));
```

O exemplo a seguir mostra como put dados privados no Amazon S3.

```
Storage.put('test.txt', 'Private Content', {  
  level: 'private',  
  contentType: 'text/plain'  
})  
  .then (result => console.log(result))  
  .catch(err => console.log(err));
```

Para obter mais informações sobre como usar o módulo de armazenamento do AWS Amplify, consulte [Armazenamento do AWS Amplify](#).

Mais informações

[Introdução ao AWS Amplify](#)

Cabeçalhos de solicitação comuns

A tabela a seguir descreve os cabeçalhos que podem ser usados por diversos tipos de solicitações REST do Amazon S3.

Nome do cabeçalho	Descrição
Authorization	As informações necessárias para solicitar a autenticação. Para obter mais informações, acesse Cabeçalho de autenticação no Guia do desenvolvedor do Amazon Simple Storage Service. Para solicitações anônimas, esse cabeçalho não é obrigatório.
Access-Control-Request-Method	Uma lista de métodos HTTP que é enviada como uma solicitação CORS preliminar. Se a avaliação de CORS preliminar for bem-sucedida, os métodos especificados poderão ser usados na solicitação CORS seguinte.
Content-Length	Comprimento da mensagem (sem cabeçalhos) de acordo com a RFC 2616. Esse cabeçalho é necessário para PUTs e operações que carregam XML, como registros em log e ACLs.
Content-Type	O tipo de conteúdo do recurso, caso a solicitação tenha conteúdo no corpo. Exemplo: text/plain
Content-MD5	O resumo MD5 de 128 bits com codificação base64 da mensagem (sem os cabeçalhos) de acordo com a RFC 1864. Esse cabeçalho pode ser usado como uma verificação de integridade da mensagem para conferir se os dados são os mesmos dados que foram enviados originalmente. Embora seja opcional, recomendamos usar o mecanismo Content-MD5 como uma verificação de integridade completa. Para obter mais informações sobre a autenticação de solicitações REST, acesse Autenticação REST no Guia do desenvolvedor do Amazon Simple Storage Service.
Date	A data que pode ser usada para criar a assinatura contida no cabeçalho Authorization . Se o cabeçalho Date precisar

Nome do cabeçalho	Descrição
	ser usado na assinatura, ele deverá ser especificado no formato básico ISO 8601. Neste caso, o cabeçalho <code>x-amz-date</code> não é necessário. Observe que quando <code>x-amz-date</code> está presente, ele sempre substitui o valor do cabeçalho <code>Date</code>. Se o cabeçalho <code>Date</code> não for usado na assinatura, ele poderá ser um dos formatos de data completa especificados por RFC 2616, seção 3.3. Por exemplo, a data/hora <code>Wed, 01 Mar 2006 12:00:00 GMT</code> é um cabeçalho de data/hora válido para usar com o Amazon S3. Se você estiver usando o cabeçalho <code>Date</code> para assinatura, ele deverá estar no formato <code>YYYYMMDD'T'HHMMSS'Z'</code> básico ISO 8601. Se <code>Date</code> for especificado, mas não estiver em formato básico ISO 8601, você também deverá incluir o cabeçalho <code>x-amz-date</code>. Se <code>Date</code> for especificado em formato básico ISO 8601, isso será suficiente para assinar solicitações e você não precisará do cabeçalho <code>x-amz-date</code>. Para obter mais informações, consulte Tratamento de datas no Signature versão 4 no Glossário da Amazon Web Services.
Expect	Quando seu aplicativo usa 100-continue, ele não envia o corpo da solicitação até receber uma confirmação. Se a mensagem for rejeitada com base nos cabeçalhos, o corpo da mensagem não será enviado. Esse cabeçalho só poderá ser usado se você estiver enviando um corpo. Valores válidos: 100-continue

Nome do cabeçalho	Descrição
Host	Para solicitações em estilo de caminho, o valor é <code>s3.amazonaws.com</code>. Para solicitações em estilo virtual, o valor é <code>BucketName.s3.amazonaws.com</code>. Para mais informações, acesse Hospedagem virtual no Guia do usuário do Amazon Simple Storage Service. Esse cabeçalho é necessário para HTTP 1.1 (a maioria dos kits de ferramentas adiciona esse cabeçalho automaticamente); opcional para solicitações HTTP/1.0.
Origin	Um endpoint que especifica o nome do servidor do solicitante inicial.
x-amz-content-sha256	Ao usar a versão 4 de assinatura para autenticar a solicitação, este cabeçalho fornece um hash da carga útil da solicitação. Para obter mais informações, consulte Cálculos de assinatura a para o cabeçalho de autorização: transferência de carga útil em uma única parte (AWS Signature Version 4). Ao carregar o objeto em partes, você define o valor como <code>STREAMING-AWS4-HMAC-SHA256-PAYLOAD</code> para indicar que a assinatura cobre somente cabeçalhos e que não há carga útil. Para obter mais informações, consulte Cálculos de assinatura a para o cabeçalho de autorização: transferência de carga útil em vários fragmentos (upload fragmentado) (AWS Signature Version 4).

Nome do cabeçalho	Descrição
x-amz-date	A data usada para criar a assinatura no cabeçalho de autorização. O formato deve ser ISO 8601 básico no formato <code>YYYYMMDD'T'HHMMSS'Z'</code> . Por exemplo, a data/hora <code>20170210T120000Z</code> é um <code>x-amz-date</code> válido a ser usado com o Amazon S3. <code>x-amz-date</code> é opcional para todas as solicitações; ele pode ser usado para substituir a data usada em solicitações de assinatura. Se o cabeçalho <code>Date</code> for especificado no formato básico ISO 8601, <code>x-amz-date</code> não será necessário. Quando <code>x-amz-date</code> está presente, ele sempre substitui o valor do cabeçalho <code>Date</code>. Para obter mais informações, consulte Tratamento de datas no Signature versão 4 no Glossário da Amazon Web Services.

Nome do cabeçalho	Descrição
x-amz-security-token	Esse cabeçalho pode ser usado nos seguintes cenários: • Para fornecer tokens de segurança para operações do Amazon DevPay – Cada solicitação que usa o Amazon DevPay requer dois cabeçalhos <code>x-amz-security-token</code> : um para o token de produto e outro para o token de usuário. Quando o Amazon S3 recebe uma solicitação autenticada, ele compara a assinatura computada com a assinatura fornecida. Cabeçalhos de vários valores formatados incorretamente que são usados para calcular uma assinatura podem causar problemas de autenticação. • Para fornecer um token de segurança ao usar credenciais de segurança temporárias - Ao fazer solicitações usando credenciais de segurança temporárias obtidas do IAM, você deve fornecer um token de segurança usando esse cabeçalho. Para obter mais informações sobre credenciais de segurança temporárias, consulte Como fazer solicitações. Esse cabeçalho é necessário para solicitações que usam o Amazon DevPay e solicitações assinadas usando credenciais de segurança temporárias.

Cabeçalhos de resposta comuns

A tabela a seguir descreve os cabeçalhos de resposta comuns à maioria das respostas do Amazon S3.

Name (Nome)	Descrição
Access-Control-Allow-Credentials	Um booleano que determina se o servidor permite que as solicitações do CORS contendo credenciais. Se o cabeçalho da solicitação <code>Access-Control-Allow-Origin</code> estiver definido como <code>'*'</code>, o cabeçalho da resposta <code>Access-Control-Allow-Credentials</code> será omitido; caso contrário, será definido como <code>true</code> quando a avaliação do CORS for bem-sucedida. Tipo: booleano Padrão: nenhum
Access-Control-Allow-Headers	Uma lista de cabeçalhos HTTP permitidos para suas solicitações do CORS. O cabeçalho da resposta <code>Access-Control-Allow-Headers</code> é retornado para avaliações bem-sucedidas do CORS e especifica explicitamente todos os <code>Access-Control-Request-Headers</code> permitidos. Tipo: string Padrão: nenhum
Access-Control-Allow-Methods	Uma lista que especifica quais métodos HTTP são permitidos. O Amazon S3 só permitirá solicitações do CORS de métodos de CORS permitidos quando a avaliação do CORS for bem-sucedida. Tipo: string Padrão: nenhum
Access-Control-Allow-Origin	A localização da origem permitida. O Amazon S3 só enviará o cabeçalho da resposta <code>Access-Control-Allow-Origin</code> quando a avaliação do CORS for bem-sucedida. Se a origem da solicitação corresponder a <code>'*'</code> nas

Name (Nome)	Descrição
	origens permitidas da configuração do CORS, o ' * ' será retornado nesse cabeçalho de resposta em vez da origem original. Tipo: string Padrão: nenhum
Access-Control-Expose-Headers	Uma lista que permite que um servidor identifique um cabeçalho de resposta que expõe o acesso aos aplicativos quando a avaliação do CORS é bem-sucedida. Tipo: string Padrão: nenhum
Access-Control-Max-Age	O tempo em segundos que seu navegador pode armazenar em cache a resposta para uma solicitação preliminar do CORS conforme identificado pelo recurso, pelo método HTTP e pela origem. O cabeçalho da resposta Access-Control-Max-Age só é retornado quando a avaliação do CORS é bem-sucedida. Tipo: inteiro Padrão: nenhum
Vary	Uma lista que indica em quais cabeçalhos de solicitação o resultado da avaliação do CORS varia. O cabeçalho da resposta Vary só é retornado quando a avaliação do CORS é bem-sucedida. Tipo: string Padrão: nenhum
Content-Length	O comprimento em bytes do corpo da resposta. Tipo: string Padrão: nenhum

Name (Nome)	Descrição
Content-Type	O tipo MIME do conteúdo. Por exemplo, Content-Type: text/html; charset=utf-8 . Tipo: string Padrão: nenhum
Connection	Um valor que especifica se a conexão com o servidor está aberta ou fechada. Tipo: Enum Valores válidos: open close Padrão: nenhum
Date	A data e a hora em que o Amazon S3 respondeu; por exemplo, Quarta, 1º de março de 2006 12:00:00 GMT. Tipo: string Padrão: nenhum

Name (Nome)	Descrição
ETag	Um ETag (identificador de entidade) representa uma versão específica do objeto. O ETag reflete as alterações apenas no conteúdo de um objeto, não em seus metadados. O ETag pode ou não ser um digest MD5 do objeto de dados. Tudo depende de como o objeto foi criado e de como está criptografado. Funciona assim: • Objetos criados por meio do Console de gerenciamento da AWS, pelo objeto PUT, POST Object, ou pela operação Copy: • Os objetos que são texto simples ou são criptografados no lado do servidor com chaves gerenciadas pelo Amazon S3 (SSE-S3) têm ETags que são um resumo MD5 dos dados. • Os objetos criptografados no lado do servidor com chaves fornecidas pelo cliente (SSE-C) ou chaves do AWS Key Management Service (AWS KMS) (SSE-KMS) têm ETags que não são um resumo MD5 dos dados do objeto. • Os objetos criados pela operação Multipart Upload ou Upload Part Copy têm ETags que não são resumos MD5, independentemente do método de criptografia. Tipo: string
Server	O nome do servidor que criou a resposta. Tipo: string Padrão: AmazonS3
x-amz-delete-marker	Um valor que especifica se o objeto retornado foi (true) ou não (false) um marcador de exclusão. Tipo: booleano Valores válidos: true false Padrão: Falso

Name (Nome)	Descrição
x-amz-id-2	Um token especial usado junto com o cabeçalho x-amz-request-id para ajudar a AWS a solucionar problemas. Para obter informações sobre o Suporte ao uso desses IDs de solicitação, consulte Solução de problemas do Amazon S3. Tipo: string Padrão: nenhum
x-amz-request-id	Um valor criado pelo Amazon S3 que identifica com exclusividade a solicitação. Esse valor é usado junto com o cabeçalho x-amz-id-2 para ajudar a AWS a solucionar problemas. Para obter informações sobre o Suporte ao uso desses IDs de solicitação, consulte Solução de problemas do Amazon S3. Tipo: string Padrão: nenhum
x-amz-server-side-encryption	O algoritmo de criptografia no lado do servidor usado para armazenar esse objeto no Amazon S3 (por exemplo, AES256, aws:kms). Valores válidos: AES256 aws:kms
x-amz-version-id	A versão do objeto. Quando você habilita o versionamento, o Amazon S3 gera um número aleatório para objetos adicionados a um bucket. O valor tem codificação UTF-8 e está pronto para URL. Quando você PUT um objeto em um bucket em que o versionamento foi suspenso, o ID da versão é sempre null. Tipo: string Valores válidos: null qualquer string com codificação UTF-8 pronta para URL Padrão: nulo

Respostas de erro

Esta seção oferece informações de referência sobre erros do Amazon S3.

Note

- Em geral, os proprietários de bucket do S3 são cobrados pelas solicitações com respostas HTTP 200 OK bem-sucedidas e respostas de erro 4XX do cliente HTTP. Os proprietários de bucket não são cobrados pelas respostas de erro 5XX do servidor HTTP, como erros HTTP 503 Slow Down. Para obter mais informações sobre códigos de erro do S3 em códigos de status HTTP 3XX e 4XX que não são cobrados, consulte [Faturamento por respostas de erro do Amazon S3](#) no Guia do usuário do Amazon S3. Consulte mais informações sobre as cobranças de um bucket de pagamento pelo solicitante em [Como funciona o pagamento pelo solicitante](#) no Guia do usuário do Amazon S3.
- O suporte a SOAP via HTTP está obsoleto, mas o SOAP continua disponível via HTTPS. Os novos recursos do Amazon S3 não são compatíveis com SOAP. Em vez de SOAP, recomendamos usar a API REST ou os AWS SDKs.

Tópicos

- [Respostas de erro de REST](#)
- [Lista de códigos de erro](#)
- [Lista de códigos de erro de conteúdo do objeto SELECT](#)
- [Lista de códigos de erro relacionados à replicação](#)
- [Lista de códigos de erro relacionados à marcação](#)
- [Lista de códigos de erro relacionados à anotação do Amazon S3](#)
- [Lista de códigos de erro do Amazon S3 em Outposts](#)
- [Lista de códigos de erro da Lente de Armazenamento do Amazon S3](#)
- [Lista de códigos de erro do Amazon S3 Object Lambda](#)
- [Lista de códigos de erro assíncronos do Amazon S3](#)
- [Lista de códigos de erro da Concessão de Acesso do Amazon S3](#)
- [Lista de códigos de erro relacionados ao Amazon FSx](#)
- [Lista de códigos de erro das Tabelas do Amazon S3](#)

- [Melhores práticas com relação a erros do Amazon S3](#)

Respostas de erro de REST

Quando ocorre um erro, as informações do cabeçalho contêm o seguinte:

- Content-Type: application/xml
- O código de status HTTP 3xx, 4xx ou 5xx apropriado

O corpo da resposta também contém informações sobre o erro. A resposta de erro de exemplo a seguir mostra a estrutura dos elementos comuns a todas as respostas de erro 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>
```

A tabela a seguir explica os elementos de resposta de erro de REST.

Name (Nome)	Descrição
Code	O código de erro é uma sequência que identifica exclusivamente uma condição de erro. O objetivo desse código é ser lido e compreendido pelos programas que detectam e tratam erros por tipo. Para obter mais informações, consulte Lista de códigos de erro. Tipo: string Antepassado: Error
Error	Contêiner para todos os elementos de erro. Tipo: Container Ancestral: Nenhum

Name (Nome)	Descrição
Message	A mensagem de erro contém uma descrição genérica da condição do erro em inglês. Ela é destinada ao público humano. Programas simples exibem a mensagem diretamente ao usuário final se encontrarem uma condição de erro que não conhecem ou não tratam. Programas sofisticados com tratamento de erro mais exaustivo e internacionalização própria são mais prováveis de ignorar a mensagem de erro. Tipo: string Antepassado: Error
RequestId	ID da solicitação associada ao erro. Tipo: string Antepassado: Error
Resource	O bucket ou objeto envolvido no erro. Tipo: string Antepassado: Error

Muitas respostas de erro contêm dados estruturados adicionais para serem lidos e compreendidos pelo desenvolvedor que diagnostica erros de programação. Por exemplo, se enviar um cabeçalho Content-MD5 com uma solicitação PUT de REST que não corresponde ao resumo calculado no servidor, você receberá um erro `BadDigest`. A resposta do erro também inclui como elementos de detalhes o resumo que o servidor calculou, e o resumo que você informou o servidor para esperar. Durante o desenvolvimento, você pode usar essas informações para diagnosticar o erro. Em produção, um programa bem-comportado pode incluir essas informações em seu log de erros.

Para obter informações sobre elementos gerais de resposta, acesse [Respostas de erro](#).

Lista de códigos de erro

A tabela a seguir lista os códigos de erro do Amazon S3.

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
AccessControlListNotSupported	O bucket não permite ACLs.	400 Bad Request	Cliente
AccessDenied	Acesso negado	403 Forbidden	Cliente
AccessPointAlreadyOwnedByYou	Já existe um ponto de acesso com um nome idêntico em sua conta.	409 Conflict	Cliente
AccountProblem	Há um problema em sua Conta da AWS que impede a conclusão bem-sucedida da operação. Se precisar de assistência adicional, acesse Fale conosco .	403 Forbidden	Cliente
AllAccessDisabled	Todo o acesso a este recurso do Amazon S3 foi desabilitado. Se precisar de assistência adicional, acesse Fale conosco .	403 Forbidden	Cliente
AmbiguousGrantByEmailAddress	O endereço de e-mail que você forneceu está associado a mais de uma conta.	400 Bad Request	Cliente
AuthorizationHeaderMalformed	O cabeçalho de autorização que você forneceu não é válido.	400 Bad Request	N/D

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
AuthorizationQueryParametersError	Os parâmetros da consulta de autorização que você forneceu não são válidos.	400 Bad Request	N/D
BadDigest	O valor de Content-MD5 ou da soma de verificação que você especificou não corresponde ao que o servidor recebeu.	400 Bad Request	Cliente
BucketAlreadyExists	O nome do bucket solicitado não está disponível. O namespace do bucket é compartilhado por todos os usuários do sistema. Especifique um nome diferente e tente novamente.	409 Conflict	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
BucketAlreadyOwnedByYou	O bucket que você tentou criar já existe e você é o proprietário. O Amazon S3 retorna esse erro em todas as Regiões da AWS, exceto na região Leste dos EUA (Norte da Virgínia) (us-east-1). Para compatibilidade legada, se você recriar um bucket existente que já possui em us-east-1, o Amazon S3 retornará 200 OK e redefinirá as listas de controle de acesso (ACLs) do bucket. Para o Amazon S3 em Outposts, o bucket que você tentou criar já existe em seu Outpost e você é o proprietário.	409 Conflict (em todas as regiões, exceto us-east-1)	Cliente
BucketHasAccessPointsAttached	O bucket que você tentou excluir tem pontos de acesso anexados. Exclua os pontos de acesso antes de excluir o bucket.	400 Bad Request	Cliente
BucketNotEmpty	O bucket que você tentou excluir não está vazio.	409 Conflict	Cliente
ClientTokenConflict	Seu token de idempotência de ponto de acesso multirregional já foi usado para uma solicitação diferente.	409 Conflict	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
ConnectionClosedByRequester	Exibido para o chamador original quando um erro é encontrado ao ler o corpo de WriteGetObjectResponse.	400 Bad Request	Cliente
ConditionalRequestConflict	Ocorreu uma operação conflitante. Você poderá repetir a solicitação se estiver usando PutObject . Se estiver usando o upload fragmentado, você deverá iniciar outra solicitação CreateMultipartUpload e recarregar cada parte.	409 Conflict	Cliente
CredentialsNotSupported	Essa solicitação não oferece suporte a credenciais.	400 Bad Request	Cliente
CrossLocationLoggingProhibited	O registro em log entre regiões não é permitido. Os buckets em uma Região da AWS não podem registrar informações em um bucket em outra região.	403 Forbidden	Cliente
DeviceNotActiveError	O dispositivo não está ativo no momento.	400 Bad Request	Cliente
EndpointNotFound	Direcione as solicitações ao endpoint correto.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
EntityTooSmall	O upload proposto é menor do que o tamanho mínimo permitido do objeto.	400 Bad Request	Cliente
EntityTooLarge	O upload proposto excede o tamanho máximo permitido do objeto. Para obter mais informações, consulte Endpoints e cotas do Amazon Simple Storage Service na Referência geral da AWS.	400 Bad Request	Cliente
	O download proposto excede o tamanho máximo permitido.	405 Method Not Allowed	Cliente
ExpiredToken	O token fornecido expirou.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
<code>IllegalLocationConstraintException</code>	Esse erro poderá ocorrer pelos seguintes motivos: • Você está tentando acessar um bucket de uma região diferente daquela em que o bucket existe. • Você está tentando criar um bucket com uma restrição de localização que corresponde a uma região diferente do endpoint regional para o qual a solicitação foi enviada.	400 Bad Request	Cliente
<code>IllegalVersioningConfigurationException</code>	A configuração de versionamento especificada na solicitação não é válida.	400 Bad Request	Cliente
<code>IncompleteBody</code>	Você não forneceu o número de bytes especificado pelo cabeçalho HTTP Content-Length.	400 Bad Request	Cliente
<code>IncorrectEndpoint</code>	O bucket especificado existe em outra região. Direcione as solicitações ao endpoint correto.	400 Bad Request	Cliente
<code>IncorrectNumberOfFilesInPostRequest</code>	O POST exige exatamente um upload de arquivo por solicitação.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
<code>InlineDataTooLarge</code>	Os dados em linha excedem o tamanho máximo permitido.	400 Bad Request	Cliente
<code>InternalError</code>	Ocorreu um erro interno. Tente novamente.	500 Internal Server Error	Servidor
<code>InvalidAccessKeyId</code>	O ID de chave de acesso da AWS fornecido não existe em nossos registros.	403 Forbidden	Cliente
<code>InvalidAccessPoint</code>	O nome ou a conta do ponto de acesso especificado não é válido.	400 Bad Request	Cliente
<code>InvalidAccessPointAliasError</code>	O nome do alias do ponto de acesso especificado não é válido.	400 Bad Request	Cliente
<code>InvalidAddressingHeader</code>	Você deve especificar o perfil Anônimo.	N/D	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidArgument	Esse erro poderá ocorrer pelos seguintes motivos: • Uma solicitação ListBuckets é feita a um endpoint regional diferente da região especificada no parâmetro <code>bucket-region</code>. • O argumento especificado não é válido. • A solicitação não tinha o cabeçalho exigido. • O argumento especificado estava incompleto ou no formato errado. • O argumento especificado deve ter um comprimento maior ou igual a três.	400 Bad Request	Cliente
InvalidBucketAclWithObjectOwnership	O bucket não pode ter ACLs definidas com a configuração <code>BucketOwnerEnforced</code> de <code>ObjectOwnership</code> .	400 Bad Request	Cliente
InvalidBucketName	O bucket especificado não é válido.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidBucketOwnerAWSAccountID	O valor do parâmetro esperado do proprietário do bucket deve ser um ID de Conta da AWS.	400 Bad Request	Cliente
InvalidBucketState	A solicitação não é válida para o estado atual do bucket.	409 Conflict	Cliente
InvalidDigest	O valor de Content-MD5 ou da soma de verificação que você especificou não é válido.	400 Bad Request	Cliente
InvalidEncryptionAlgorithmError	A solicitação de criptografia que você especificou não é válida. O valor válido é AES256.	400 Bad Request	Cliente
InvalidHostHeader	Os cabeçalhos do host fornecidos na solicitação usaram o endereçamento de estilo incorreto.	400 Bad Request	Cliente
InvalidHttpMethod	A solicitação é feita usando um método HTTP inesperado.	400 Bad Request	Cliente
InvalidLocationConstraint	A restrição de localização (Região) especificada não é válida. Para obter mais informações sobre como selecionar uma região para seus buckets, consulte Visão geral dos buckets .	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidObjectState	A operação não é válida para o estado atual do objeto.	403 Forbidden	Cliente
InvalidPart	Uma ou mais das partes especificadas não foram encontradas. A parte pode não ter sido carregada ou o ETag (identificador de entidade) especificado pode não ter correspondido ao ETag da parte.	400 Bad Request	Cliente
InvalidPartOrder	A lista de partes não estava em ordem crescente. A lista de partes deve ser especificada em ordem por número da parte.	400 Bad Request	Cliente
InvalidPayer	Todo acesso a esse objeto foi desabilitado. Se precisar de assistência adicional, acesse Fale conosco .	403 Forbidden	Cliente
InvalidPolicyDocument	O conteúdo do formulário não atende às condições especificadas no documento de política.	400 Bad Request	Cliente
InvalidRange	O intervalo solicitado não pode ser satisfeito.	416 Intervalo solicitado não satisfatório	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidRegion	Você tentou criar um ponto de acesso multirregional em uma região pela qual não optou.	403 proibido	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidRequest	Esse erro poderá ocorrer pelos seguintes motivos: • Uma solicitação ListBuckets não paginada é feita a partir de uma conta que tem uma cota de bucket de uso geral aprovada superior a 10.000. Você deve fazer solicitações paginadas para listar os buckets em uma conta com mais de 10.000 buckets. • A solicitação está usando a versão incorreta da assinatura. Use o AWS4-HMAC-SHA256 (Signature versão 4). • Um ponto de acesso só pode ser criado para um bucket existente. • O ponto de acesso não está em um estado em que possa ser excluído. • Um ponto de acesso só pode ser listado para um bucket existente. • O próximo token não é válido. •	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
	Pelo menos uma ação deve ser especificada em uma regra de ciclo de vida. • Pelo menos uma regra de ciclo de vida deve ser especificada. • O número de regras de ciclo de vida não deve exceder o limite permitido de 1.000 regras. • O intervalo do parâmetro <code>MaxResults</code> não é válido. • As solicitações SOAP devem ser feitas por meio de uma conexão HTTPS. • A Aceleração de Transferências do Amazon S3 não é compatível com buckets com nomes que não estejam em conformidade com o DNS. • A Aceleração de Transferências do Amazon S3 não é compatível com buckets com ponto (.) no nome. •		

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
	O endpoint da Aceleração de Transferências do Amazon S3 só permite solicitações no estilo virtual. • A Aceleração de Transferências do Amazon S3 não está configurada neste bucket. • A Aceleração de Transferências do Amazon S3 está desabilitada neste bucket. • A Aceleração de Transferências do Amazon S3 não é compatível com este bucket. Para obter ajuda, entre em contato com Suporte. • A Aceleração de Transferências do Amazon S3 não pode ser habilitada neste bucket. Para obter ajuda, entre em contato com Suporte. • Valores conflitantes fornecidos em cabeçalhos HTTP e parâmetros de consulta. •		

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
	Valores conflitantes fornecidos nos cabeçalhos HTTP e nos campos do formulário POST. Solicitação CopyObject feita em objetos maiores que 5 GB.		
InvalidSessionException	Retornado se a sessão não existir mais porque ultrapassou o limite de tempo ou expirou.	400 Bad Request	Cliente
InvalidSignature	A assinatura da solicitação que o servidor calculou não corresponde à assinatura que você forneceu. Verifique a chave de acesso secreta da AWS e o método de assinatura. Para ter mais informações, consulte Assinar e autenticar as solicitações REST .	400 Bad Request	Cliente
InvalidSecurity	As credenciais de segurança fornecidas não são válidas.	403 Forbidden	Cliente
InvalidSOAPRequest	O corpo da solicitação SOAP não é válido.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
<code>InvalidStorageClass</code>	A classe de armazenamento que você especificou não é válida.	400 Bad Request	Cliente
<code>InvalidTargetBucketForLogging</code>	O bucket de destino para registro em log não existe, não é de sua propriedade ou não tem as concessões apropriadas para o grupo de entrega de logs.	400 Bad Request	Cliente
<code>InvalidToken</code>	O token fornecido foi malformado ou é inválido por algum outro motivo.	400 Bad Request	Cliente
<code>InvalidURI</code>	Não foi possível analisar o URI especificado.	400 Bad Request	Cliente
<code>KeyTooLongError</code>	A chave é muito longa.	400 Bad Request	Cliente
<code>KMS.DisabledException</code>	A solicitação foi recusada porque a chave do KMS especificada não está habilitada.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
KMS.InvalidKeyUsageException	A solicitação foi recusada por um dos seguintes motivos: • O valor KeyUsage da chave do KMS é incompatível com a operação de API. • O algoritmo de criptografia ou de assinatura especificado para a operação é incompatível com o tipo de material na chave do KMS (KeySpec). Para criptografia, descriptografia, nova criptografia e geração de chaves de dados, o KeyUsage deve ser ENCRYPT_DECRYPT. Para assinar e verificar mensagens, o KeyUsage deve ser SIGN_VERIFY. Para gerar e verificar códigos de autenticação de mensagens (MACs), o KeyUsage deve ser GENERATE_VERIFY_MAC. Para gerar segredos do contrato de chave, o KeyUsage deve ser KEY_AGREEMENT. Para encontrar o KeyUsage de uma chave do KMS, use a operação DescribeKey. Para encontrar os algoritmos de criptografia ou de assinatura	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
	compatíveis com uma chave do KMS específica, use a operação DescribeKey.		

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
KMS.KMSInvalidStateException	A solicitação foi recusada porque o estado do recurso especificado não é válido para essa solicitação. Essa exceção tem um dos seguintes significados: • O estado da chave do KMS não é compatível com a operação. Para encontrar o estado da chave, use a operação <code>DescribeKey</code>. Para ter mais informações sobre quais estados de chave são compatíveis com cada operação do KMS, consulte Key states of AWS KMS keys no Guia do desenvolvedor do AWS Key Management Service. • Em relação a operações criptográficas em chaves do KMS em armazenamentos de chaves personalizado, essa exceção representa uma falha geral com muitas causas possíveis. Para identificar a causa, leia a mensagem de erro que acompanha a exceção.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
<code>KMS.NotFoundException</code>	A solicitação foi recusada porque não foi possível encontrar a entidade ou o recurso especificado.	400 Bad Request	Cliente
<code>MalformedACLError</code>	O ACL fornecido não foi bem formado ou não foi validado com relação a nosso esquema publicado.	400 Bad Request	Cliente
<code>MalformedPOSTRequest</code>	O corpo da solicitação POST não tem dados de formulários/de várias partes bem formados.	400 Bad Request	Cliente
<code>MalformedXML</code>	O XML fornecido não foi bem formado ou não foi validado com relação a nosso esquema publicado.	400 Bad Request	Cliente
<code>MaxMessageLengthExceeded</code>	Sua solicitação era muito grande.	400 Bad Request	Cliente
<code>MaxPostPreDataLengthExceededError</code>	Os campos de solicitação POST anteriores ao arquivo de upload eram muito grandes.	400 Bad Request	Cliente
<code>MetadataTooLarge</code>	Os cabeçalhos de metadados excedem o tamanho máximo permitido de metadados.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
MethodNotAllowed	O método especificado não é permitido neste recurso.	405 Method Not Allowed	Cliente
MissingAttachment	Um anexo SOAP era esperado, mas nenhum foi encontrado.	400 Bad Request	Cliente
MissingAuthenticationToken	A solicitação não foi assinada.	403 proibido	Cliente
MissingContentLength	Você deve fornecer o cabeçalho HTTP de tamanho do conteúdo.	411 Length Required	Cliente
MissingRequestBodyError	Você enviou um documento XML vazio como uma solicitação.	400 Bad Request	Cliente
MissingSecurityElement	A solicitação SOAP 1.1 não tem um elemento de segurança.	400 Bad Request	Cliente
MissingSecurityHeader	Falta um cabeçalho obrigatório na solicitação.	400 Bad Request	Cliente
NoLoggingStatusForKey	Não existe um sub-recurso de status de registro em log para uma chave.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
NoSuchAsyncRequest	A solicitação especificada não foi encontrada.	404 Not Found	Cliente
NoSuchBucket	O bucket especificado não existe.	404 Not Found	Cliente
NoSuchBucketPolicy	O bucket especificado não tem uma política de bucket.	404 Not Found	Cliente
NoSuchCORSConfiguration	O bucket especificado não tem uma configuração de CORS.	404 Not Found	Cliente
NoSuchKey	A chave especificada não existe.	404 Not Found	Cliente
NoSuchLifecycleConfiguration	A configuração de ciclo de vida especificada não existe.	404 Not Found	Cliente
NoSuchMultiRegionAccessPoint	O ponto de acesso multirregional especificado não existe.	404 Not Found	Cliente
NoSuchObjectLockConfiguration	O objeto especificado não tem uma configuração ObjectLock.	404 Not Found	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
NoSuchWebsiteConfiguration	O bucket especificado não tem uma configuração de site.	404 Not Found	Cliente
NoSuchTagSet	A tag especificada não existe.	404 Not Found	Cliente
NoSuchUpload	O carregamento fracionado especificado não existe. O ID do carregamento pode não ser válido ou o carregamento fracionado pode ter sido anulado ou concluído.	404 Not Found	Cliente
NoSuchVersion	O ID de versão especificado na solicitação não corresponde a uma versão existente.	404 Not Found	Cliente
NotDeviceOwnerError	O dispositivo que gerou o token não pertence ao usuário autenticado.	400 Bad Request	Cliente
NotImplemented	Um cabeçalho que você forneceu implica uma funcionalidade que não foi implementada.	501 Not Implemented	Servidor
NotModified	O recurso não foi alterado.	304 Not Modified	Servidor

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
NoTransformationDefined	Nenhuma transformação encontrada para esse ponto de acesso do Object Lambda.	404 Not Found	Cliente
NotSignedUp	Sua conta não está cadastrada no serviço do Amazon S3. É necessário se cadastrar para poder usar o Amazon S3. Você pode se cadastrar no seguinte URL: https://aws.amazon.com/s3	403 Forbidden	Cliente
ObjectLockConfigurationNotFoundError	A configuração do Bloqueio de Objetos não existe para esse bucket.	404 Not Found	Cliente
OwnershipControlsNotFoundError	Os controles de propriedade do bucket não foram encontrados.	404 Not Found	Cliente
OperationAborted	Uma operação condicional conflitante está em andamento neste recurso. Tente novamente.	409 Conflict	Cliente
PermanentRedirect	O bucket que você está tentando acessar deve ser endereçado usando o endpoint especificado. Envie todas as solicitações futuras para esse endpoint.	301 Moved Permanently	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
PermanentRedirectControlError	A operação de API que você está tentando acessar deve ser endereçada usando o endpoint especificado. Envie todas as solicitações futuras para esse endpoint.	301 Moved Permanently	Cliente
PreconditionFailed	Pelo menos uma das pré-condições que você especificou não foi atendida.	412 Precondition Failed	Cliente
Redirect	Redirecionamento temporário. Será feito um redirecionamento ao bucket enquanto o servidor Sistema de Nomes de Domínio (DNS) está sendo atualizado.	307 Temporary Redirect	Cliente
RequestHeaderSectionTooLarge	O cabeçalho da solicitação e os parâmetros de consulta usados para fazer a solicitação excedem o tamanho máximo permitido.	400 Bad Request	Cliente
RequestIsNotMultipartContent	Uma solicitação POST de bucket deve ser do tipo dados de formulário/de várias partes.	412 Precondition Failed	Cliente
RequestTimeout	Sua conexão de soquete com o servidor não foi lida ou gravada dentro do período de tempo limite.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
RequestTimeTooSkewed	A diferença entre a hora da solicitação e a hora do servidor é muito grande.	403 Forbidden	Cliente
RequestTorrentOfBucketError	Não é permitido solicitar o arquivo torrent de um bucket.	400 Bad Request	Cliente
ResponseInterrupted	Exibido para o chamador original quando um erro é encontrado ao ler o corpo de WriteGetObjectResponse.	400 Bad Request	Cliente
RestoreAlreadyInProgress	A restauração do objeto já está em andamento.	409 Conflict	Cliente
ServerSideEncryptionConfigurationNotFoundError	A configuração de criptografia do lado do servidor não foi encontrada.	400 Bad Request	Cliente
ServiceUnavailable	O serviço não consegue lidar com a solicitação.	503 Service Unavailable	Servidor

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
SignatureDoesNotMatch	A assinatura da solicitação que o servidor calculou não corresponde à assinatura que você forneceu. Verifique a chave de acesso secreta da AWS e o método de assinatura. Consulte mais informações em Assinar e autenticar as solicitações REST e em Autenticar solicitações SOAP .	403 Forbidden	Cliente
SlowDown	Diminua a taxa de solicitação.	503 Slow Down	Servidor
503 SlowDown	Espere e tente novamente	503 Slow Down	Servidor
TagPolicyException	A política de marcação não permite o valor especificado para a chave de tag a seguir.	400 Bad Request	Cliente
TemporaryRedirect	Será feito um redirecionamento ao bucket enquanto o servidor Sistema de Nomes de Domínio (DNS) está sendo atualizado.	307 Temporary Redirect	Cliente
TokenCodeInvalidError	O número de série e/ou o código de token que você forneceu não são válidos.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
TokenRefreshRequired	O token fornecido deve ser atualizado.	400 Bad Request	Cliente
TooManyAccessPoints	Você tentou criar mais pontos de acesso do que o permitido para uma conta. Para obter mais informações, consulte Endpoints e cotas do Amazon Simple Storage Service na Referência geral da AWS.	400 Bad Request	Cliente
TooManyBuckets	Você tentou criar mais buckets do que o permitido para uma conta. Para obter mais informações, consulte Endpoints e cotas do Amazon Simple Storage Service na Referência geral da AWS.	400 Bad Request	Cliente
TooManyMultiRegionAccessPointregionsError	Você tentou criar um ponto de acesso multirregional com mais regiões do que o permitido para uma conta. Para obter mais informações, consulte Endpoints e cotas do Amazon Simple Storage Service na Referência geral da AWS.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
TooManyMultiRegionAccessPoints	Você tentou criar mais pontos de acesso multirregionais do que o permitido para uma conta. Para obter mais informações, consulte Endpoints e cotas do Amazon Simple Storage Service na Referência geral da AWS.	400 Bad Request	Cliente
UnauthorizedAccessError	Aplicável somente nas regiões da China. Retornado quando uma solicitação é feita para um bucket que não tem uma licença ICP. Consulte mais informações em ICP Recordal .	403 Forbidden	Cliente
UnexpectedContent	Essa solicitação contém um conteúdo incompatível.	400 Bad Request	Cliente
UnexpectedIPError	Aplicável somente nas regiões da China. Essa solicitação foi recusada porque o IP era inesperado.	403 Forbidden	Cliente
UnsupportedArgument	A solicitação continha um argumento incompatível.	400 Bad Request	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
UnsupportedSignature	A solicitação fornecida é assinada com uma versão não compatível do Security Token Service (STS) ou a versão da assinatura não é aceita.	400 Bad Request	Cliente
UnresolvableGrantByEmailAddress	O endereço de e-mail que você forneceu não corresponde a nenhuma conta registrada.	400 Bad Request	Cliente
UserKeyMustBeSpecified	A solicitação POST do bucket deve conter o nome do campo especificado. Se ele estiver especificado, confira a ordem dos campos.	400 Bad Request	Cliente
NoSuchAccessPoint	O ponto de acesso especificado não existe.	404 Not Found	Cliente
InvalidTag	Sua solicitação contém uma entrada de tag que não é válida. Por exemplo, sua solicitação pode conter chaves duplicadas, chaves ou valores muito longos ou tags do sistema.	400 Bad Request	Cliente
MalformedPolicy	Sua política contém uma entidade principal que não é válida.	400 Bad Request	Cliente

Lista de códigos de erro de conteúdo do objeto SELECT

Important

O Amazon S3 Select não está mais disponível para novos clientes. Os clientes existentes do Amazon S3 Select podem continuar usando o recurso normalmente. [Saiba mais](#)

A tabela a seguir contém erros especiais que o `SELECT Object Content` pode retornar. Para obter informações gerais sobre erros do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
AmbiguousFieldName	O nome do campo correspon de a vários campos no arquivo. Verifique a expressão SQL e o arquivo e tente novamente.	400	Cliente
Busy	O serviço não está disponível. Tente novamente mais tarde.	503	Cliente
CastFailed	Uma tentativa de converter de um tipo de dados em outro usando CAST falhou na expressão SQL.	400	Cliente
ColumnTooLong	O comprimento de uma coluna no resultado é maior que <code>maxCharsPerColumn</code> de 1 MB.	400	Cliente
CSVEscapingRecordDelimiter	Um delimitador de registro entre aspas foi encontrado no arquivo. Para permitir delimitadores de registros entre aspas, defina <code>AllowQuotedRecordDelimiter</code> como 'TRUE'.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
<code>CSVParsingError</code>	Ocorreu um erro ao analisar o arquivo CSV. Verifique o arquivo e tente novamente.	400	Cliente
<code>CSVUnescapedQuote</code>	Uma aspa sem escape foi encontrada durante a análise do arquivo CSV. Para permitir delimitadores de registros entre aspas, defina <code>AllowQuotedRecordDelimiter</code> como <code>'TRUE'</code> .	400	Cliente
<code>EmptyRequestBody</code>	O corpo da solicitação não pode estar vazio.	400	Cliente
<code>EvaluatorBindingDoesNotExist</code>	O nome da coluna ou o caminho fornecido não existe na expressão SQL.	400	Cliente
<code>EvaluatorInvalidArguments</code>	Há um número incorreto de argumentos na chamada da função na expressão SQL.	400	Cliente
<code>EvaluatorInvalidTimestampFormatPattern</code>	A string de formato de carimbo de data/hora na expressão SQL não é válida.	400	Cliente
<code>EvaluatorInvalidTimestampFormatPatternSymbol</code>	O padrão de formato de carimbo de data/hora contém um símbolo na expressão SQL que não é válido.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
EvaluatorInvalidTimestampFormatPatternSymbolForParsing	O padrão de formato de carimbo de data/hora contém um símbolo de formato válido que não pode ser aplicado à análise de carimbo de data/hora na expressão SQL.	400	Cliente
EvaluatorInvalidTimestampFormatPatternToken	O padrão de formato de carimbo de data/hora contém um token na expressão SQL que não é válido.	400	Cliente
EvaluatorLikePatternInvalidEscapeSequence	Um argumento fornecido à expressão LIKE não era válido.	400	Cliente
EvaluatorNegativeLimit	LIMIT não pode ser negativo.	400	Cliente
EvaluatorTimestampFormatPatternDuplicateFields	O padrão de formato de carimbo de data/hora contém vários especificadores de formato que representam o campo de carimbo de data/hora na expressão SQL.	400	Cliente
EvaluatorTimestampFormatPatternHourClockAmPmMismatch	O padrão de formato de carimbo de data/hora contém um símbolo de hora no formato de 12 horas, mas não contém um campo AM/PM, ou contém um especificador de formato de 24 horas e contém um campo AM/PM na expressão SQL.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
EvaluatorUnterminatedTimestampFormatPatternToken	O padrão de formato de carimbo de data/hora contém um token não terminado na expressão SQL.	400	Cliente
ExpressionTooLong	A expressão SQL é muito longa. O tamanho máximo em bytes de uma expressão SQL é 256 KB.	400	Cliente
ExternalEvalException	A consulta não pode ser avaliada. Verifique o arquivo e tente novamente.	400	Cliente
IllegalSqlFunctionArgument	Um argumento ilegal foi usado na função SQL.	400	Cliente
IncorrectSqlFunctionArgumentType	Um tipo de argumento incorreto foi especificado em uma chamada de função na expressão SQL.	400	Cliente
IntegerOverflow	Ocorreu um estouro ou subfluxo de números inteiros na expressão SQL.	400	Cliente
InternalError	Ocorreu um erro interno.	500	Cliente
InvalidCast	Uma tentativa de converter de um tipo de dados em outro usando CAST falhou na expressão SQL.	400	Cliente
InvalidColumnIndex	O índice da coluna na expressão SQL não é válido.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidCompressionFormat	O arquivo não está em um formato de compactação compatível. Somente GZIP e BZIP2 são compatíveis.	400	Cliente
InvalidDataSource	O tipo de fonte de dados não é válido. Somente CSV, JSON e Parquet são compatíveis.	400	Cliente
InvalidDataType	A expressão SQL contém um tipo de dados que não é válido.	400	Cliente
InvalidExpressionType	O valor ExpressionType não é válido. Somente expressões SQL são compatíveis.	400	Cliente
InvalidFileHeaderInfo	O valor FileHeaderInfo não é válido. Somente o NONE, o USE, e o IGNORE são compatíveis.	400	Cliente
InvalidJsonType	O valor JsonType não é válido. Somente DOCUMENT e LINES são compatíveis.	400	Cliente
InvalidKeyPath	O caminho da chave na expressão SQL não é válido.	400	Cliente
InvalidQuoteFields	O valor QuoteFields não é válido. Somente ALWAYS e ASNEEDED são compatíveis.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
<code>InvalidRequestParameter</code>	O valor de um parâmetro no elemento <code>SelectRequest</code> não é válido. Verifique a documentação da API do serviço e tente novamente.	400	Cliente
<code>InvalidScanRange</code>	O intervalo de varredura fornecido não é válido.	400	Cliente
<code>InvalidTableAlias</code>	A expressão SQL contém um alias de tabela que não é válido.	400	Cliente
<code>InvalidTextEncoding</code>	O tipo de codificação não é válido. Só há suporte para a codificação UTF-8.	400	Cliente
<code>JSONParsingError</code>	Ocorreu um erro ao analisar o arquivo JSON. Verifique o arquivo e tente novamente.	400	Cliente
<code>LexerInvalidChar</code>	A expressão SQL contém um caractere que não é válido.	400	Cliente
<code>LexerInvalidIONLiteral</code>	A expressão SQL contém um operador que não é válido.	400	Cliente
<code>LexerInvalidLiteral</code>	A expressão SQL contém um operador que não é válido.	400	Cliente
<code>LexerInvalidOperator</code>	A expressão SQL contém um literal que não é válido.	400	Cliente
<code>LikeInvalidInputs</code>	O argumento fornecido à cláusula LIKE na expressão SQL não é válido.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
MalformedXML	O XML fornecido não foi bem formado ou não foi validado com relação a nosso esquema publicado. Verifique a documentação do serviço e tente novamente.	400	Cliente
MaxOperatorsExceeded	Falha ao analisar a expressão SQL. Tente reduzir a complexidade. Por exemplo, reduza o número de operadores usados.	400	Cliente
MethodNotAllowed	O método especificado não é permitido neste recurso.	405 Method Not Allowed	Cliente
MissingRequiredParameter	A entidade SelectRequest não possui um parâmetro obrigatório. Verifique a documentação do serviço e tente novamente.	400	Cliente
MultipleDataSourcesUnsupported	Não há suporte para várias fontes de dados.	400	Cliente
NumberFormatError	Ocorreu um erro ao analisar um número. Esse erro pode ser causado por falta ou excesso de fluxo de números inteiros.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
ObjectSerializationConflict	InputSerialization especifica mais de um formato (CSV, JSON ou Parquet) ou OutputSerialization especifica mais de um formato (CSV ou JSON). Para InputSerialization e OutputSerialization, você pode especificar somente um formato para cada um.	400	Cliente
OverMaxColumn	O número de colunas no resultado é maior que o número máximo permitido de colunas.	400	Cliente
OverMaxParquetBlockSize	O arquivo Parquet está acima do tamanho máximo do grupo de linhas.	400	Cliente
OverMaxRecordSize	O tamanho de um registro na entrada ou no resultado é maior que o limite de maxCharsPerRecord de 1 MB.	400	Cliente
ParquetParsingError	Ocorreu um erro ao analisar o arquivo Parquet. Verifique o arquivo e tente novamente.	400	Cliente
ParquetUnsupportedCompressionCodec	O codec de compactação Parquet especificado não é compatível.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
ParseAsteriskIsNotAloneInSelectList	Outras expressões não são permitidas na lista SELECT quando * é usado sem notação de pontos na expressão SQL.	400	Cliente
ParseCannotMixSqbandWildcardInSelectList	Não é possível misturar [] e * na mesma expressão em uma lista SELECT na expressão SQL.	400	Cliente
ParseCastArity	A expressão SQL CAST tem aridade incorreta.	400	Cliente
ParseEmptySelect	A expressão SQL contém uma cláusula SELECT vazia.	400	Cliente
ParseExpected2TokenTypes	O token esperado na expressão SQL não foi encontrado.	400	Cliente
ParseExpectedArgumentDelimiter	O delimitador de argumentos esperado na expressão SQL não foi encontrado.	400	Cliente
ParseExpectedDatePart	A parte da data esperada na expressão SQL não foi encontrada.	400	Cliente
ParseExpectedExpression	A expressão SQL esperada não foi encontrada.	400	Cliente
ParseExpectedIdentifierForAlias	O identificador esperado para o alias na expressão SQL não foi encontrado.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
ParseExpectedIdentForAt	O identificador esperado para o nome AT na expressão SQL não foi encontrado.	400	Cliente
ParseExpectedIdentForGroupName	GROUP não tem suporte na expressão SQL.	400	Cliente
ParseExpectedKeyword	A palavra-chave esperada na expressão SQL não foi encontrada.	400	Cliente
ParseExpectedLeftParenAfterCast	O parêntese esquerdo esperado depois de CAST na expressão SQL não foi encontrado.	400	Cliente
ParseExpectedLeftParenBuiltinFunctionCall	O parêntese esquerdo esperado na expressão SQL não foi encontrado.	400	Cliente
ParseExpectedLeftParenValueConstructor	O parêntese esquerdo esperado na expressão SQL não foi encontrado.	400	Cliente
ParseExpectedMember	A expressão SQL contém um uso incompatível de MEMBER.	400	Cliente
ParseExpectedNumber	O número esperado na expressão SQL não foi encontrado.	400	Cliente
ParseExpectedRightParenBuiltinFunctionCall	O caractere de parêntese direito esperado na expressão SQL não foi encontrado.	400	Cliente
ParseExpectedTokenType	O token esperado na expressão SQL não foi encontrado.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
ParseExpectedTypeName	O nome do tipo esperado na expressão SQL não foi encontrado.	400	Cliente
ParseExpectedWhenClause	A cláusula WHEN esperada na expressão SQL não foi encontrada. CASE não é compatível.	400	Cliente
ParseInvalidContextForWildcardInSelectList	O uso de * na lista SELECT na expressão SQL não é válido.	400	Cliente
ParseInvalidPathComponent	A expressão SQL contém um componente de caminho que não é válido.	400	Cliente
ParseInvalidTypeParameter	A expressão SQL contém um valor de parâmetro que não é válido.	400	Cliente
ParseMalformedJoin	JOIN não tem suporte na expressão SQL.	400	Cliente
ParseMissingIdentifierAt	O identificador esperado depois do símbolo @ na expressão SQL não foi encontrado.	400	Cliente
ParseNonUnaryAggregateFunctionCall	Somente um argumento é permitido para funções agregadas na expressão SQL.	400	Cliente
ParseSelectMissingFrom	A expressão SQL contém um FROM ausente depois da lista SELECT.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
ParseUnexpectedKeyword	A expressão SQL contém uma palavra-chave inesperada.	400	Cliente
ParseUnexpectedOperator	A expressão SQL contém um operador inesperado.	400	Cliente
ParseUnexpectedTerm	A expressão SQL contém um termo inesperado.	400	Cliente
ParseUnexpectedToken	A expressão SQL contém um token inesperado.	400	Cliente
ParseUnknownOperator	A expressão SQL contém um operador que não é válido.	400	Cliente
ParseUnsupportedAlias	A expressão SQL contém um uso incompatível de ALIAS.	400	Cliente
ParseUnsupportedCallWithStar	Somente COUNT com (*) como parâmetro é permitido na expressão SQL.	400	Cliente
ParseUnsupportedCase	A expressão SQL contém um uso incompatível de CASE.	400	Cliente
ParseUnsupportedCaseClause	A expressão SQL contém um uso incompatível de CASE.	400	Cliente
ParseUnsupportedLiteralsGroupBy	A expressão SQL contém um uso incompatível de GROUP BY .	400	Cliente
ParseUnsupportedSelect	A expressão SQL contém um uso incompatível de SELECT.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
ParseUnsupportedSyntax	A expressão SQL contém uma sintaxe incompatível.	400	Cliente
ParseUnsupportedToken	A expressão SQL contém um token incompatível.	400	Cliente
TruncatedInput	Falha na descompactação do objeto. Verifique se o objeto está compactado corretamente usando o formato especificado na solicitação.	400	Cliente
UnauthorizedAccess	Você não tem autorização para executar esta operação.	401	Cliente
UnrecognizedFormatException	Encontramos um tipo de registro que não é válido.	400	Cliente
UnsupportedFunction	Encontramos uma função SQL incompatível.	400	Cliente
UnsupportedParquetType	O tipo de Parquet especificado é incompatível.	400	Cliente
UnsupportedRangeHeader	Não há suporte para um cabeçalho de intervalo para essa operação.	400	Cliente
UnsupportedScanRangeInput	Não há suporte para consultas de intervalo de varredura nesse tipo de objeto.	400	Cliente
UnsupportedSqlOperation	Encontramos uma operação SQL incompatível.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
UnsupportedSqlStructure	Encontramos uma estrutura SQL incompatível. Verifique a Referência SQL.	400	Cliente
UnsupportedStorageClass	Encontramos uma classe de armazenamento que é incompatível. Somente as classes de armazenamento STANDARD, STANDARD_IA e ONEZONE_IA são compatíveis.	400	Cliente
UnsupportedSyntax	Encontramos uma sintaxe que não é válida.	400	Cliente
UnsupportedTypeForQuerying	Sua consulta contém um tipo incompatível para comparação (por exemplo, verificar se um tipo de coluna Parquet INT96 é maior que 0).	400	Cliente
ValueParseFailure	Ocorreu uma falha na análise do carimbo de data/hora na expressão SQL.	400	Cliente

Lista de códigos de erro relacionados à replicação

A tabela a seguir contém erros especiais que a operação `Replication` pode retornar. Para obter informações gerais sobre erros do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidArgument	Esse erro poderá ocorrer pelos seguintes motivos: • O elemento <Account> está vazio. Ele deve conter um ID de conta válido. • A Conta da AWS especificada no elemento <Account> deve corresponder ao proprietário do bucket de destino. • ReplicationTime-Status deve conter um valor. • ReplicationTime-ReplicationTimeValue deve conter um valor. • O valor Replication-ReplicationTimeValue-Minutes deve ser 15. • ReplicationMetrics deve conter um Status. • ReplicationMetrics deve conter um EventThreshold . • O valor EventThreshold-ReplicationTimeValue-Minutes deve ser 15.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
	• Rule ID não deve conter caracteres não ASCII.		
InvalidRequest	Esse erro poderá ocorrer pelos seguintes motivos: • O <Owner> em <AccessControlTranslation> tem um valor e, portanto, o elemento <Account> deve ser especificado. • O elemento <Account> está vazio. Ele deve conter um ID de conta válido. • O destination de replicação deve conter ReplicationTime e Metrics, ou nenhum deles. • ReplicationTime e ReplicationMetrics devem ter o mesmo status. • O Controle de Tempo de Replicação do S3 (S3 RTC) não é compatível com esta Região da AWS.	400	Cliente

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
ReplicationConfigurationNotFoundErro	Não há configuração de replicação para esse bucket.	404 Not Found (404 Não encontrado)	Cliente

Lista de códigos de erro relacionados à marcação

A tabela a seguir contém erros especiais que as operações `TagResource`, `UntagResource` e `ListTagsForResource` podem retornar para grupos da Lente de Armazenamento. Para obter informações gerais sobre erros gerais do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidRequest	A Região da AWS no ARN do recurso não corresponde à região especificada nessa solicitação. A conta da AWS no ARN do recurso não corresponde ao ID da conta especificada nessa solicitação. A partição da AWS no resourceArn é inválido.	400 Bad Request	Não compatível
InvalidTag	Essa solicitação contém uma chave ou um valor de tag que não é válido. Os caracteres válidos incluem o seguinte: [a-zA-Z+ -= . _ : /] . As chaves de tag podem conter até 128 caracteres. Os valores de tag podem conter	400 Bad Request	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
	até 256 caracteres. Há chaves de tag duplicadas em sua solicitação. As chaves de tag definidas pelo usuário não podem começar com <code>aws:</code> .		
NoSuchResource	O recurso especificado não existe.	404 Not Found	Não compatível
TagPolicyException	A política de marcação não permite o valor especificado para a chave de tag a seguir.	400 Bad Request	Não compatível
TooManyTags	O número de tags excede o limite de 50 tags.	400 Bad Request	Não compatível

Lista de códigos de erro relacionados à anotação do Amazon S3

A tabela a seguir contém erros especiais que as operações de anotação (`PutObjectAnnotation`, `GetObjectAnnotation`, `ListObjectAnnotations`, `DeleteObjectAnnotation`, `UpdateBucketMetadataAnnotationTableConfiguration`) e o comportamento `CopyObject` relacionado à anotação podem retornar. Para obter informações gerais sobre erros do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
AccessDenied	Acesso negado. O chamador não tem permissão para realizar a operação de anotação solicitada ou o objeto está protegido pelo	403 Forbidden	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
	Bloqueio de Objetos no modo COMPLIANCE.		
AnnotationLimitExceeded	A contagem de anotações do objeto excede o limite (1000).	400 Bad Request	Não compatível
AnnotationNameTooLong	O nome da anotação é muito longo. Os nomes de anotação podem ter até 512 bytes de comprimento.	400 Bad Request	Não compatível
AnnotationRequestConflict	Uma operação conflitante está em andamento nesta anotação. Isso pode ocorrer quando uma anotação de origem é excluída durante uma cópia entre regiões.	409 Conflict	Não compatível
EntityTooSmall	O upload proposto é menor do que o tamanho mínimo permitido. A carga útil da anotação deve ter pelo menos 1 byte.	400 Bad Request	Não compatível
InvalidAnnotationName	O nome da anotação contém caracteres inválidos ou começa com um prefixo reservado (aws ou s3, sem distinção entre maiúsculas e minúsculas). Os nomes das anotações podem conter somente letras Unicode, números, sublinhado (_), ponto (.) e hífen (-).	400 Bad Request	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidArgument	A diretiva de anotação não é válida. Os valores válidos são COPY e EXCLUDE. Para ListObjectAnnotations , max-annotation-results deve estar entre 0 e 1000.	400 Bad Request	Não compatível
InvalidPrefix	O prefixo de anotação que você forneceu não é válido.	400 Bad Request	Não compatível
InvalidRequest	Esse erro pode ocorrer pelos seguintes motivos: o objeto é criptografado com SSE-C, que não oferece suporte a anotações. O objeto de origem tem anotações que não podem ser copiadas para o destino porque o destino está em uma região sem suporte. A solicitação de configuração da tabela de anotação não é válida (por exemplo, desabilitar a tabela enquanto especifica a criptografia, habilitar uma tabela que já está habilitada, atualizar uma tabela no status CREATING ou enviar uma solicitação vazia).	400 Bad Request	Não compatível
MethodNotAllowed	O método especificado não é permitido neste recurso. Por exemplo, esse erro ocorre quando a solicitação tem como alvo um marcador de exclusão.	405 Method Not Allowed	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
NoSuchAnnotation	A anotação especificada não existe.	404 Not Found	Não compatível
NoSuchKey	A chave especificada não existe.	404 Not Found	Não compatível
NoSuchVersion	O ID de versão especificado na solicitação não corresponde a uma versão existente.	404 Not Found	Não compatível
NotImplemented	As anotações não são compatíveis com buckets de diretório do Amazon S3.	501 Not Implemented	Não compatível
OperationAborted	Uma operação condicional conflitante está em andamento neste recurso. Isso pode ocorrer durante modificações simultâneas ou quando uma contagem de anotações de origem ou um ETag muda durante uma cópia entre regiões. Tente novamente.	409 Conflict	Não compatível
PreconditionFailed	Pelo menos uma das pré-condições que você especificou não foi atendida. O ETag x-amz-object-if-match não corresponde ao ETag do objeto atual.	412 Precondition Failed	Não compatível
UnsupportedMediaType	A carga útil da solicitação contém um tipo de mídia incompatível. A carga útil da anotação deve ser texto válido com codificação UTF-8.	415 Unsupported Media Type	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
UserAnnotationName MustBeSpecified	Você deve especificar um nome de anotação de usuário.	400 Bad Request	Não compatível

Lista de códigos de erro do Amazon S3 em Outposts

A tabela a seguir contém erros especiais que uma operação do Amazon S3 em Outposts pode retornar. Para obter informações gerais sobre erros do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
BadRequest	O bucket está em um estado de transição devido a uma tentativa anterior de exclusão. Tente novamente mais tarde.	400 Bad Request	Não compatível
InvalidRequest	Esse erro poderá ocorrer pelos seguintes motivos: - A configuração da Amazon VPC é obrigatória. - O acesso público não é permitido nos pontos de acesso do S3 em Outposts.	400 Bad Request	Cliente
InvalidOutpostState	A solicitação não é válida para o estado atual do Outpost.	409 Conflito	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidRequest	O ponto de acesso não está em um estado em que possa ser excluído.	400 Bad Request	Não compatível
NoSuchOutpost	O Outpost especificado não existe.	404 Not Found (404 Não encontrado)	Não compatível
UnsupportedOperation	A ação especificada não era compatível.	404 Not Found (404 Não encontrado)	Não compatível
InsufficientCapacity	Capacidade insuficiente.	507 Armazenamento insuficiente	Não compatível

Lista de códigos de erro da Lente de Armazenamento do Amazon S3

A tabela a seguir contém erros especiais que as operações da Lente de Armazenamento do Amazon S3 podem retornar. Para obter informações gerais sobre erros gerais do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
AccessDenied	Esta região não é compatível como região de origem da Lente de Armazenamento do S3.	403 Forbidden	Não compatível
AccountNotAuthorized	Esta conta não está autorizada a usar o AWS Organizations. Use a conta gerencial ou a conta de administrador delegado.	403 Forbidden	Não compatível
ActivityMetricsMustEnabled	As métricas de atividade devem estar habilitadas.	400 Bad Request	Não compatível
AWSOrganizationsNotInUseException	Esta conta não faz parte da sua organização.	403 Forbidden	Não compatível
DefaultConfigurationDeleteForbidden	A configuração padrão não pode ser excluída.	403 Forbidden	Não compatível
DuplicateStorageLensGroupARN	Há duas ou mais entradas do mesmo ARN do grupo da Lente de Armazenamento nessa configuração.	400 Bad Request	Não compatível
EmptyExcludeContainer	Esse erro ocorre pelos seguintes motivos: - O contêiner de exclusão não pode estar vazio. - O contêiner de exclusão não pode ter zero buckets.	400 Bad Request	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
	O contêiner de exclusão não pode ter zero regiões.		
EmptyExcludeElement	Você deve especificar um grupo da Lente de Armazenamento com o elemento Exclude.	400 Bad Request	Não compatível
EmptyIncludeContainer	Esse erro ocorre pelos seguintes motivos: - O contêiner de inclusão não pode estar vazio. - O contêiner de inclusão não pode ter zero buckets. - O contêiner de inclusão não pode ter zero regiões.	400 Bad Request	Não compatível
InvalidAWSOrgArn	Há um ARN do AWS Organizations incorreto na configuração.	400 Bad Request	Não compatível
EmptyIncludeElement	Você deve especificar um grupo da Lente de Armazenamento com o elemento Include.	400 Bad Request	Não compatível
InvalidBucketFilter	As configurações no nível organizacional não oferecem suporte a filtros de bucket.	400 Bad Request	Não compatível
InvalidConfigId	O ID de configuração não é válido.	400 Bad Request	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidDestination	O ARN do bucket do S3 está incorreto.	400 Bad Request	Não compatível
InvalidEncryptionMethod	Apenas um método de criptografia pode ser especificado.	400 Bad Request	Não compatível
InvalidFilterForDefaultConfiguration	A configuração padrão não deve incluir nenhum filtro.	400 Bad Request	Não compatível
InvalidIncludeExcludeContainers	Você pode especificar um contêiner Include ou um contêiner Exclude em uma configuração. Não é possível especificar ambos em uma configuração.	400 Bad Request	Não compatível
InvalidIncludeExcludeElements	Somente um elemento Include ou Exclude é permitido. Pelo menos um elemento Include ou Exclude deve estar presente.	400 Bad Request	Não compatível
InvalidKMSEncryptionKeyId	O ARN do ID da chave do KMS não é válido.	400 Bad Request	Não compatível
InvalidMaximumPrefixDepth	MaxDepth deve estar dentro do intervalo [1,10].	400 Bad Request	Não compatível
InvalidMinimumStorageBytesPercentage	MinStorageBytesPercentage deve estar dentro do intervalo [1.00,100.00].	400 Bad Request	Não compatível
InvalidOrganizationARN	O ARN do AWS Organizations na configuração não é válido.	400 Bad Request	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
<code>InvalidOrganizationForDefaultConfiguration</code>	A configuração padrão não oferece suporte a métricas no nível da organização.	400 Bad Request	Não compatível
<code>InvalidRegionForDefaultConfiguration</code>	A região especificada não é compatível com a configuração padrão.	400 Bad Request	Não compatível
<code>InvalidRegionName</code>	O nome da região não é válido.	400 Bad Request	Não compatível
<code>InvalidStorageLensArn</code>	O ARN da Lente de Armazenamento do S3 não é necessário na entrada.	400 Bad Request	Não compatível
<code>InvalidStorageLensGroupARN</code>	O ARN desse grupo da Lente de Armazenamento não é válido ou somente grupos da Lente de Armazenamento em sua conta são permitidos. Além disso, você deve seguir a estrutura de ARN do grupo da Lente de Armazenamento: <code>arn::s3::storage-lens-group/</code> e respeitar o limite de 64 caracteres. Os nomes de grupo da Lente de Armazenamento também podem conter somente os seguintes caracteres: a-z, A-Z, 0-9, hifens (-) e sublinhados (_).	400 Bad Request	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
MissingAccountLevelActivityMetrics	As métricas de atividade devem ser habilitadas no nível da conta quando as métricas de atividade são habilitadas no nível do bucket.	400 Bad Request	Não compatível
MissingBucketLevelActivityMetrics	As métricas de atividade devem ser habilitadas no nível do bucket quando as métricas de atividade são habilitadas no nível da conta.	400 Bad Request	Não compatível
MissingEncryptionMethod	O método de criptografia não pode ficar em branco. Especifique SSE-KMS ou SSE-S3.	400 Bad Request	Não compatível
MissingPrefixLevelStorageMetrics	As métricas de armazenamento no nível do prefixo são obrigatórias quando o nível do prefixo está habilitado.	400 Bad Request	Não compatível
OrganizationAccessDenied	Esta conta não está autorizada a adicionar o AWS Organizations.	403 proibido	Não compatível
OrgConfigurationNotSupported	A região especificada não é compatível com o AWS Organizations na configuração.	403 proibido	Não compatível
ServiceNotEnabledForOrg	O perfil vinculado ao serviço da Lente de Armazenamento do S3 não está habilitado para a organização.	403 proibido	Não compatível
StorageMetricsMustEnabled	As métricas de armazenamento no nível do prefixo devem estar habilitadas.	400 Bad Request	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
TooManyBuckets	O contêiner de buckets não pode ter mais de 50 buckets.	400 Bad Request	Não compatível
TooManyRegions	O contêiner de regiões não pode ter mais de 50 regiões.	400 Bad Request	Não compatível
TooManyStorageLens Groups	Não é possível anexar mais de 50 grupos ao painel da Lente de Armazenamento.	400 Bad Request	Não compatível

A tabela a seguir contém erros especiais que as operações de grupo da Lente de Armazenamento do S3 podem retornar. Para obter informações gerais sobre erros gerais do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
AccessDenied	Você não tem permissão para realizar ações de grupo da Lente de Armazenamento. Esta região não é compatível como região de origem de grupos da Lente de Armazenamento do S3.	403 Forbidden	Não compatível
ConfigurationAlreadyExists	A configuração especificada já existe.	409 Conflict	Não compatível
DuplicateElement	As tags devem ser exclusivas. O operador lógico And inclui chaves de tag duplicadas. O operador lógico Or inclui tags duplicadas. O	400 Bad Request	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
	operador lógico inclui prefixos ou sufixos duplicados.		
InvalidAge	DaysLessThan e DaysGreaterThanOrEqual devem ser números positivos.	400 Bad Request	Não compatível
InvalidFilter	Um filtro deve incluir um dos seguintes elementos: And, Or, MatchAnyTag, MatchAnyPrefix, MatchAnySuffix, MatchObjectAge, MatchObjectSize.	400 Bad Request	Não compatível
InvalidLogicalOperator	Pelo menos dois subelementos devem estar presentes nos operadores lógicos And ou Or.	400 Bad Request	Não compatível
InvalidMatchAnyPrefix	O parâmetro MatchAnyPrefix não pode ser vazio.	400 Bad Request	Não compatível
InvalidMatchAnySuffix	O parâmetro MatchAnySuffix não pode ser vazio.	400 Bad Request	Não compatível
InvalidMatchAnyTag	O parâmetro MatchAnyTag não pode ser vazio.	400 Bad Request	Não compatível
InvalidMatchObjectAge	O parâmetro MatchObjectAge não pode ser vazio.	400 Bad Request	Não compatível
InvalidMatchObjectSize	O parâmetro MatchObjectSize não pode ser vazio.	400 Bad Request	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidName	O parâmetro de nome do grupo da Lente de Armazenamento deve ter entre 1 e 64 caracteres. O parâmetro Name do grupo da Lente de Armazenamento deve usar o padrão <code>^[a-zA-Z0-9\-_]+\$</code> .	400 Bad Request	Não compatível
InvalidNumericCombination	Essa combinação de idade ou tamanho do objeto não é válida.	400 Bad Request	Não compatível
InvalidPrefix	O tamanho máximo de um prefixo é 1.024 caracteres. A string de prefixo não pode ser vazia.	400 Bad Request	Não compatível
InvalidSize	BytesLessThan e BytesGreaterThan devem ser números positivos. O tamanho máximo do objeto não pode exceder 50 TB. O tamanho mínimo do objeto não pode ser maior ou igual a 50 TB.	400 Bad Request	Não compatível
InvalidSuffix	O tamanho máximo de um sufixo é 1.024 caracteres. A string de sufixo não pode ser vazia.	400 Bad Request	Não compatível

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
InvalidTag	A chave de tag do objeto não pode exceder 128 caracteres. A string da chave de tag do objeto não pode ser nula ou vazia. O tamanho máximo de um valor de tag é 256 caracteres. A chave de tag do objeto contém caracteres que não são válidos. A chave de tag do objeto deve conter somente a-z, A-Z, 0-9, espaços e os seguintes caracteres: ^(_.: / = + \ - @] *) \$.	400 Bad Request	Não compatível
MismatchedName	O nome especificado na solicitação não corresponde ao nome do grupo da Lente de Armazenamento.	400 Bad Request	Não compatível
TooManyConfigurations	Você tentou criar mais configurações de grupo da Lente de Armazenamento do que as 50 permitidas.	400 Bad Request	Não compatível
TooManyElements	O Element excede o número máximo de elementos permitidos em um operador lógico. Somente 10 prefixos, sufixos ou tags são permitidos.	400 Bad Request	Não compatível

Lista de códigos de erro do Amazon S3 Object Lambda

A tabela a seguir contém erros especiais que o S3 Object Lambda pode retornar. Para obter informações sobre erros gerais do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

As respostas de erro recebidas dos pontos de acesso de suporte durante solicitações não GetObject são enviadas ao chamador inalteradas.

Código de erro	Descrição	Código de status HTTP	
LambdaInvalidResponse	Exibido para o chamador original quando WriteGetObjectResponse responde com ValidationError ao AWS Lambda. Leia a mensagem ValidationError para obter mais detalhes. Nem todos os casos de ValidationError resultam em um erro LambdaInvalidResponse.	400 Bad Request	
LambdaInvocationFailed	A invocação da função do Lambda falhou. Os chamadores podem receber o erro a seguir quando o S3 Object Lambda não consegue invocar com êxito a função do Lambda configurada. A mensagem de erro pode conter detalhes sobre um eventual erro exibido pelo serviço AWS Lambda ao invocar a função (por exemplo, código de status, código de erro,	400 Bad Request	

Código de erro	Descrição	Código de status HTTP	
	mensagem de erro e ID da solicitação).		
LambdaNotFound	A função do AWS Lambda não foi encontrada. A função do Lambda configurada (ou a versão ou o alias) não foi encontrada quando invocada. A configuração do ponto de acesso do S3 Object Lambda deve apontar para o ARN correto da função do Lambda. A mensagem de erro pode conter detalhes sobre um eventual erro exibido pelo serviço AWS Lambda ao invocar a função (por exemplo, código de status, código de erro, mensagem de erro e ID da solicitação).	404 Not Found	

Código de erro	Descrição	Código de status HTTP	
LambdaPermissionError	O chamador não está autorizado a invocar a função do Lambda. O chamador deve ter permissão para invocar a função do Lambda. Confira as políticas anexadas ao chamador e se elas têm permissão para usar <code>lambda:Invoke</code> para a função configurada. A mensagem de erro pode conter detalhes sobre um eventual erro exibido pelo serviço AWS Lambda ao invocar a função (por exemplo, código de status, código de erro, mensagem de erro e ID da solicitação).	403 Forbidden	
LambdaResponseNotReceived	A função do Lambda foi encerrada sem chamar <code>WriteGetObjectResponse</code>. Os dados de resposta <code>GetObject</code> são fornecidos pela função do Lambda chamando a operação da API <code>WriteGetObjectResponse</code>. Os logs do Amazon CloudWatch para a função podem fornecer mais informações sobre por que a função não chamou essa operação de API, apesar de sair normalmente.	500 Internal Service Error	

Código de erro	Descrição	Código de status HTTP	
LambdaRuntimeError	A função do Lambda falhou durante a execução. Um erro explícito foi recebido da função do Lambda. Para obter detalhes sobre a falha, verifique os logs do AWS CloudFormation.	500 Internal Service Error	
LambdaTimeout	A função do Lambda não respondeu no tempo permitido. A função do Lambda não conseguiu concluir a chamada para WriteGetObjectResponse em 60 segundos.	500 Internal Service Error	

Código de erro	Descrição	Código de status HTTP	
SlowDown	Reduza sua taxa de solicitações para operações que envolvam o AWS Lambda. A invocação da função foi limitada pelo AWS Lambda, talvez porque tenha atingido o limite de simultaneidade configurado. Para obter mais informações, consulte Gerenciamento de simultaneidade para uma função do Lambda no Guia do desenvolvedor do AWS Lambda. A mensagem de erro pode conter detalhes sobre um eventual erro exibido pelo serviço AWS Lambda ao invocar a função (por exemplo, código de status, código de erro, mensagem de erro e ID da solicitação).	503 Slow Down	
ValidationError	Os erros de validação podem ser exibidos pela operação da API <code>WriteGetObjectResponse</code> e podem ocorrer por vários motivos. Leia a mensagem de erro para ter mais detalhes.	400 Bad Request	

Lista de códigos de erro assíncronos do Amazon S3

A tabela a seguir contém erros especiais que as solicitações assíncronas podem retornar. Para obter informações gerais sobre erros do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Esses erros são retornados quando você consulta sobre o estado de uma solicitação assíncrona, como usando `DescribeMultiRegionAccessPointOperation`. Como essas solicitações são assíncronas, todos esses erros têm um código de status de 200 OK.

Código de erro	Descrição	Código de status HTTP	
<code>AccessDenied</code>	Acesso negado.	200 OK	
<code>InternalErrors</code>	Ocorreu um erro interno do servidor.	200 OK	
<code>MalformedPolicy</code>	A sintaxe da política especificada não é válida.	200 OK	
<code>MultiRegionAccessPointAlreadyOwnedByYou</code>	Você já tem um ponto de acesso multirregional com o mesmo nome.	200 OK	
<code>MultiRegionAccessPointModifiedByAnotherRequest</code>	A ação falhou porque outra solicitação está modificando o recurso especificado. Tente reenviar a solicitação após a conclusão da solicitação anterior.	200 OK	
<code>MultiRegionAccessPointNotReady</code>	O ponto de acesso multirregional especificado não está pronto para ser atualizado.	200 OK	
<code>MultiRegionAccessPointSameBucketRegion</code>	Os buckets usados para criar um ponto de acesso multirregional não podem estar na mesma região.	200 OK	
<code>MultiRegionAccessPointUnsupportedRegion</code>	Um dos buckets fornecidos para criar o ponto de acesso multirregional está em uma região sem suporte.	200 OK	
<code>NoSuchBucket</code>	O bucket especificado não existe.	200 OK	

Código de erro	Descrição	Código de status HTTP	
NoSuchMultiRegionAccessPoint	O ponto de acesso multirregional especificado não existe.	200 OK	

Lista de códigos de erro da Concessão de Acesso do Amazon S3

A tabela a seguir contém erros especiais que as solicitações da Concessão de Acesso do S3 podem retornar. Para obter informações gerais sobre erros do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Código de erro	Descrição	Código de status HTTP	
AccessGrantAlreadyExists	A concessão de acesso especificada já existe	409	
AccessGrantsInstanceAlreadyExists	A instância da Concessão de Acesso já existe	409	
AccessGrantsInstanceNotEmptyError	Limpe os locais antes de excluir a instância de concessões de acesso	400	
AccessGrantsInstanceNotExistsError	A instância da Concessão de Acesso não existe	404	
AccessGrantsInstanceResourcePolicyNotExists	A política de recursos da instância da Concessão de Acesso não existe	404	
AccessGrantsLocationAlreadyExistsError	O local de concessão de acesso especificado já existe	409	
AccessGrantsLocationNotEmptyError	Limpe as concessões de acesso antes de excluir a localização delas	400	

Código de erro	Descrição	Código de status HTTP	
AccessGrantsLocationsQuotaExceededError	A cota de locais de concessões de acesso foi excedida. Cota de locais da Concessão de Acesso: <i><valor></i> . Acesse o S3 se for necessário um aumento.	409	
AccessGrantsQuotaExceededError	A cota de concessões de acesso foi excedida. Cota de concessões de acesso: <i><valor></i> . Acesse o S3 se for necessário um aumento.	409	
InvalidTag	Há chaves de tag duplicadas em sua solicitação. Remova as chaves de tag duplicadas e tente novamente.	400	
InvalidAccessGrant	A concessão de acesso especificada é inválida	400	
InvalidAccessGrantLocation	O local especificado de concessões de acesso é inválido	400	
InvalidIamRole	O perfil do IAM especificado é inválido	400	
InvalidIdentityCenterInstance	A instância do centro de identidade especificada é inválida	400	
InvalidResourcePolicy	A política de recursos especificada é inválida	400	
InvalidResourcePolicy	A política de recursos especificada é inválida	400	

Código de erro	Descrição	Código de status HTTP	
InvalidTag	Essa solicitação contém uma chave ou um valor de tag que não é válido. Os caracteres válidos incluem o seguinte: [a-zA-Z+-. _:/]. As chaves de tag podem conter até 128 caracteres. Os valores de tag podem conter até 256 caracteres.	400	
NoSuchAccessGrantError	A concessão de acesso especificada não existe	404	
NoSuchAccessGrantsLocationError	O local de concessões de acesso especificado não existe	404	
AccessDenied	Você não tem permissões <i><permissão solicitada></i> para o prefixo do S3 solicitado: <i><destino solicitado></i>	403 Forbidden	
StsNotAuthorizedError	Ocorreu um erro (StsNotAuthorizedError) ao chamar a operação GetDataAccess: usuário: access-grants.s3.amazonaws.com não tem permissão para realizar: sts:AssumeRole no recurso: <i><ARN do perfil do IAM></i>	403	
StsPackedPolicyTooLargeError	Ocorreu um erro (StsPackedPolicyTooLargeError) ao chamar a operação GetDataAccess: token serializado muito grande para a sessão	400	

Código de erro	Descrição	Código de status HTTP	
StsValidationError	A mensagem de erro varia de acordo com o erro de validação.	400	
InvalidTags	As chaves de tag não podem começar com o prefixo AWS reservado para tags do sistema."	400	
TooManyTags	O número de tags excede o limite de 50 tags. Remova algumas tags e tente novamente.	400	

Lista de códigos de erro relacionados ao Amazon FSx

A tabela a seguir contém erros especiais que as solicitações feitas ao Amazon FSx por meio de um ponto de acesso do Amazon S3 podem retornar. Para obter informações gerais sobre erros do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Código de erro	Descrição	Código de status HTTP	
AccessDenied	Esse erro poderá ocorrer pelos seguintes motivos: • O chamador não está autorizado a realizar a operação do S3 especificada. • O chamador não está autorizado a acessar o recurso do Amazon FSx para a chave especificada.	403 Forbidden	

Código de erro	Descrição	Código de status HTTP	
InsufficientCapacity	A capacidade máxima de armazenamento do sistema de arquivos foi atingida.	507 Insufficient Capacity	
InvalidKey	A chave especificada é inválida.	400 Bad Request	
SlowDown	Diminua a taxa de solicitação.	503 Slow Down	

Lista de códigos de erro das Tabelas do Amazon S3

A tabela a seguir contém erros especiais que as operações das Tabelas do Amazon S3 podem retornar. Para obter informações gerais sobre erros do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Código de erro	Descrição	Código de status HTTP
AccessDeniedException	A ação não pode ser executada porque você não tem a permissão necessária.	403 Forbidden
BadRequestException	A solicitação é inválida ou malformada.	400 Bad Request
ConflictException	A solicitação falhou porque há um conflito com uma gravação anterior. Você poderá repetir a solicitação.	409 Conflict
ForbiddenException	O chamador não está autorizado a fazer a solicitação.	403 Forbidden
InternalServerErrorException	A atualização falhou devido a um erro interno do servidor.	500 Internal

Código de erro	Descrição	Código de status HTTP
		Server Error
NotFoundException	A solicitação foi recusada porque não foi possível encontrar o recurso especificado.	404 Not Found
TooManyRequestsException	O limite de número de solicitações por segundo foi excedido.	429 Too Many Requests

Melhores práticas com relação a erros do Amazon S3

Muitas respostas de erro contêm dados estruturados adicionais para serem lidos e compreendidos pelo desenvolvedor que diagnostica erros de programação. Por exemplo, se você enviar um cabeçalho Content-MD5 com uma solicitação PUT de REST que não corresponde ao resumo calculado no servidor, você receberá um erro BadDigest. A resposta do erro também inclui como elementos de detalhes o resumo que calculamos, e o resumo que você nos informou para esperar. Durante o desenvolvimento, você pode usar essas informações para diagnosticar o erro. Em produção, um programa bem-comportado pode incluir essas informações em seu log de erros.

Ao desenvolver uma aplicação para uso com o Amazon S3, é importante tratar os erros do Amazon S3 de maneira adequada. Esta seção descreve os problemas a serem considerados ao desenvolver um aplicativo.

Tentar InternalErrors novamente

Os erros internos são erros que ocorrem no ambiente do Amazon S3.

Solicitações que recebem uma resposta de InternalError podem não ter sido processadas. Por exemplo, se uma solicitação PUT retornar um InternalError, um GET subsequente poderá recuperar o valor antigo ou o valor atualizado.

Se o Amazon S3 retornar uma resposta de InternalError, tente a solicitação novamente.

Ajustar o aplicativo para erros repetidos de SlowDown

Como com qualquer sistema distribuído, o S3 tem mecanismos de proteção que detectam consumo excessivo intencional ou involuntário de recursos e reagem adequadamente. Os erros de SlowDown podem ocorrer quando uma alta taxa de solicitações aciona um desses mecanismos. Reduzir a taxa de solicitações reduzirá ou eliminará erros desse tipo. De modo geral, a maioria dos usuários não experimentará esses erros regularmente. No entanto, se você quiser obter mais informações ou estiver enfrentando erros de desaceleração em grande quantidade ou inesperados, publique no nosso [fórum de desenvolvedores do Amazon S3](#) ou cadastre-se no Suporte <https://aws.amazon.com/premiumsupport/>.

Erros isolados

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

O Amazon S3 fornece um conjunto de códigos de erro que são usados pelas APIs SOAP e REST. A API SOAP retorna códigos de erro padrão do Amazon S3. A API REST é desenvolvida para parecer um servidor HTTP padrão e interagir com clientes HTTP existentes (por exemplo, navegadores, bibliotecas de cliente HTTP, proxies, caches etc.) Para garantir que os clientes HTTP tratem erros corretamente, mapeamos cada erro do Amazon S3 para um código de status HTTP.

Os códigos de status HTTP são menos expressivos que os códigos de erro do Amazon S3 e contêm menos informações sobre o erro. Por exemplo, os erros NoSuchKey e NoSuchBucket do Amazon S3 são mapeados para o código de status HTTP 404 Not Found.

Embora os códigos de status HTTP contenham menos informações sobre o erro, os clientes que entendem HTTP, mas não a API do Amazon S3, geralmente tratam o erro corretamente.

Portanto, para lidar com erros ou para relatar erros do Amazon S3 para os usuários finais, use o código de erro do Amazon S3 em vez do código de status HTTP, uma vez que ele contém a maioria das informações sobre o erro. Além disso, ao depurar o aplicativo, você também deve consultar o elemento <Details> legível pelo usuário da resposta de erro XML.

AWSGlossário da

Para obter a terminologia mais recente da AWS, consulte o [glossário da AWS](#) na Referência do Glossário da AWS.

Recursos do Amazon S3

A tabela a seguir lista os recursos relacionados que serão úteis à medida que você utilizar este serviço.

Recurso	Descrição
Guia do usuário do Amazon Simple Storage Service	O Guia de introdução fornece um tutorial rápido do serviço com base em um caso de uso simples.
Guia do usuário do Amazon Simple Storage Service	O Guia do desenvolvedor descreve como realizar tarefas usando as operações do Amazon S3.
Perguntas frequentes e técnicas sobre o Amazon S3	As perguntas frequentes abordam as 20 dúvidas mais comuns entre os desenvolvedores deste produto.
Notas de versão do Amazon S3	As notas de release fornecem uma visão geral de alto nível da versão atual. Elas observam especificamente os novos recursos, correções e problemas conhecidos.
Ferramentas para a Amazon Web Services	Uma central de ponto de partida para localizar documentação, exemplos de código, notas de versão e outras informações para ajudar você a desenvolver aplicativos inovadores com ferramentas e SDKs da AWS.
Console de gerenciamento da AWS	O console permite que você realize a maioria das funções do Amazon S3 sem programação.
Fóruns de discussão da	Um fórum comunitário para que os desenvolvedores discutam questões técnicas relacionadas à Amazon Web Services.
Suporte Centro	A página inicial para obter suporte técnico da AWS, incluindo o acesso aos nossos Fóruns de desenvolvedores, Perguntas frequentes técnicas, Página de status do serviço e Premium Support.

Recurso	Descrição
Suporte	A principal página da Web para obter informações sobre o Suporte, um canal de suporte de resposta rápida e com atendimento individual, para ajudar você a desenvolver e executar aplicativos nos serviços de infraestrutura da AWS.
Informações sobre produtos do Amazon S	A primeira página da web para obter informações sobre o Amazon S3.
Entre em contato conosco	Um ponto central de contato para consultas relativas a faturamento da AWS, conta, eventos, abuso etc.
Condições de uso	Informações detalhadas sobre o uso de direitos autorais e marca registrada na Amazon.com e outros tópicos.

Apêndice

Tópicos

- [Apêndice: resposta SelectObjectContent](#)
- [Apêndice: objeto OPTIONS](#)
- [Apêndice: API SOAP](#)
- [Apêndice: autenticação de solicitações \(AWS Signature Version 2\)](#)
- [Apêndice: APIs de configuração do ciclo de vida \(obsoletas\)](#)

Apêndice: resposta SelectObjectContent

Descrição

A operação do Amazon S3 Select filtra o conteúdo de um objeto do Amazon S3 com base em uma declaração de linguagem de consulta estruturada (SQL) simples. Como o tamanho da resposta dessa operação é desconhecido, o Amazon S3 Select transmite a resposta como uma série de mensagens e inclui um cabeçalho `Transfer-Encoding` com **chunked** como valor na resposta.

Para obter mais informações sobre o Amazon S3 Select, consulte [Seleção de conteúdo de objetos](#) no Guia do usuário do Amazon Simple Storage Service.

Para obter mais informações sobre o uso do SQL com o Amazon S3 Select, consulte [Referência de SQL para o Amazon S3 Select e o Amazon Glacier Select](#) no Guia do usuário do Amazon Simple Storage Service.

Respostas

Uma operação bem-sucedida do Amazon S3 Select retorna o código de status `200 OK`.

Cabeçalhos de resposta

Essa implementação da operação usa somente os cabeçalhos de resposta que são comuns à maioria das respostas. Para obter mais informações, consulte [Cabeçalhos de resposta comuns](#).

Corpo da resposta

Como o tamanho da resposta do Amazon S3 Select é desconhecido, o Amazon S3 transmite a resposta como uma série de mensagens e inclui um cabeçalho `Transfer-Encoding` com **chunked** como valor na resposta. O exemplo a seguir mostra o formato de resposta no nível superior:

```
<Message 1>
<Message 2>
<Message 3>
.....
<Message n>
```

Cada mensagem consiste em duas seções: o prelúdio e os dados. A seção de prelúdio consiste em 1) tamanho total de bytes da mensagem e 2) tamanho combinado de todos os cabeçalhos. A seção de dados consiste em 1) cabeçalhos e 2) uma carga útil.

Cada seção termina com uma soma de verificação (CRC) inteira big-endian de 4 bytes. O Amazon S3 Select usa CRC32 (geralmente chamado de GZIP CRC32) para calcular os dois CRCs. Para obter mais informações sobre CRC32, consulte [Especificação do formato de arquivo GZIP versão 4.3](#).

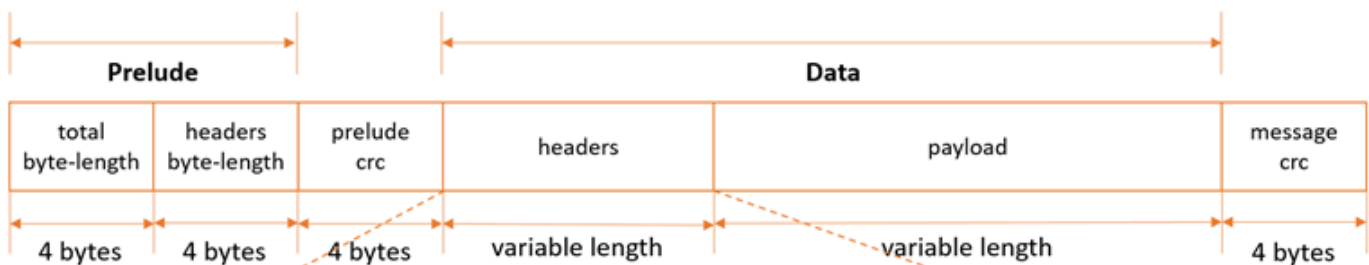
O total de sobrecarga de mensagem, incluindo o prelúdio e ambas as somas de verificação, é de 16 bytes.

Note

Todos os valores de integer nas mensagens estão na ordem de bytes da rede ou na ordem big-endian.

O diagrama a seguir mostra os componentes que formam uma mensagem e um cabeçalho. Há vários cabeçalhos por mensagem.

Message:

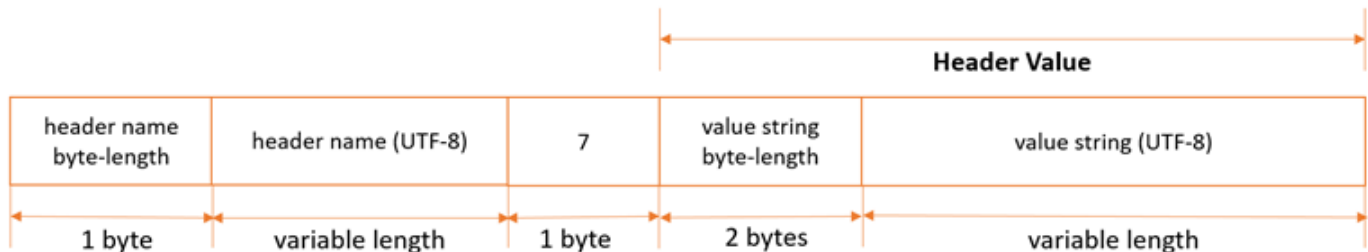


Headers (multiple headers per message):



Note

Para o Amazon S3 Select, o tipo de valor do cabeçalho é sempre 7 (type=String). Para esse tipo, o valor do cabeçalho consiste em dois componentes: um comprimento inteiro big-endian de 2 bytes e uma string UTF-8 com esse tamanho de byte. O diagrama a seguir mostra os componentes que formam os cabeçalhos do Amazon S3 Select.

Amazon S3 Select Headers (type=String):

Cálculos de tamanho em bytes da carga útil (esses dois cálculos são 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 mensagem contém os seguintes componentes:

- Prelúdio: sempre com tamanho fixo de 8 bytes (dois campos de 4 bytes cada):
 - Primeiros quatro bytes: tamanho total em bytes: tamanho em bytes do inteiro big-endian da mensagem inteira (incluindo o próprio campo de tamanho total de 4 bytes).
 - Segundos quatro bytes: tamanho em bytes dos cabeçalhos: tamanho em bytes do inteiro big-endian da parte de cabeçalhos da mensagem (excluindo o próprio campo de tamanho dos cabeçalhos).
- CRC do prelúdio: soma de verificação do inteiro big-endian (CRC) de 4 bytes da parte do prelúdio da mensagem (excluindo a própria CRC). O prelúdio tem uma CRC separada da CRC da mensagem (veja abaixo) para garantir que informações corrompidas de tamanho em bytes possam ser detectadas imediatamente, sem causar comportamentos de buffer patológicos.

- **Cabeçalhos:** conjunto de metadados que anotam a mensagem, como o tipo de mensagem, o formato da carga útil, e assim por diante. As mensagens podem ter vários cabeçalhos, portanto, essa parte da mensagem pode ter tamanhos em bytes diferentes, dependendo do tipo de mensagem. Cabeçalhos são pares de chave-valor, onde a chave e o valor são strings UTF-8. Os cabeçalhos podem aparecer em qualquer ordem na parte dos cabeçalhos da mensagem, e qualquer tipo de cabeçalho pode aparecer apenas uma vez.

Para o Amazon S3 Select, confira a seguir uma lista de nomes de cabeçalho e o conjunto de valores válidos, dependendo do tipo de mensagem.

- **Cabeçalho MessageType:**
 - HeaderName => ":message-type"
 - HeaderValues válidos => "error", "event"
- **Cabeçalho EventType:**
 - HeaderName => ":event-type"
 - HeaderValues válidos => "Records", "Cont", "Progress", "Stats", "End"
- **Cabeçalho ErrorCode:**
 - HeaderName => ":error-code"
 - HeaderValues válidos => Código de erro da tabela na seção [Lista de códigos de erro de conteúdo do objeto SELECT](#).
- **Cabeçalho ErrorMessage:**
 - HeaderName => ":error-message"
 - HeaderValues válidos => Mensagem de erro retornada pelo serviço para ajudar a diagnosticar erros no nível da solicitação.
- **Carga útil:** pode ser qualquer coisa.
- **CRC da mensagem:** soma de verificação do inteiro big-endian (CRC) de 4 bytes do início da mensagem até o início da soma de verificação (ou seja, tudo na mensagem, exceto a própria CRC da mensagem).

Cada cabeçalho contém os seguintes componentes. Pode haver vários cabeçalhos por mensagem.

- **Tamanho em bytes do nome do cabeçalho:** tamanho em bytes do nome do cabeçalho.
- **Nome do cabeçalho:** o nome do cabeçalho que indica o tipo de cabeçalho. Valores válidos: ":message-type" ":event-type" ":error-code" ":error-message"

- Tipo de valor de cabeçalho: enumeração que indica o tipo de valor de cabeçalho. Para o Amazon S3 Select, esse valor é sempre 7.
- Tamanho em bytes da string de valor: (para o Amazon S3 Select) tamanho em bytes da string de valor do cabeçalho.
- String de valor do cabeçalho: (para o Amazon S3 Select) valor da string do cabeçalho. Os valores válidos para esse campo variam de acordo com o tipo do cabeçalho. Consulte as seções abaixo para ver os valores válidos para cada tipo de cabeçalho e cada tipo de mensagem.

Para o Amazon S3 Select, as respostas podem ser mensagens dos seguintes tipos:

- Mensagem de registros: pode conter um único registro, registros parciais ou vários registros. Dependendo do tamanho do resultado, uma resposta pode conter uma ou mais dessas mensagens.
- Mensagem de continuação: o Amazon S3 envia periodicamente essa mensagem para manter a conexão TCP aberta. Essas mensagens aparecem aleatoriamente nas respostas. O cliente deve detectar o tipo de mensagem e processar conforme necessário.
- Mensagem de andamento: o Amazon S3 envia essa mensagem periodicamente, se solicitada. Ela contém informações sobre o andamento de uma consulta que foi iniciada, mas ainda não foi concluída.
- Mensagem de estatísticas: o Amazon S3 envia essa mensagem no final da solicitação. Ela contém estatísticas sobre a consulta.
- Mensagem final: indica que a solicitação foi concluída e que nenhuma outra mensagem será enviada. Você não deve presumir que a solicitação foi concluída até o cliente receber uma mensagem End.
- Mensagem RequestLevelError: o Amazon S3 enviará essa mensagem se a solicitação falhar por algum motivo. Ela contém o código de erro e a mensagem de erro da falha. Se o Amazon S3 enviar uma mensagem RequestLevelError, ele não enviará uma mensagem End.

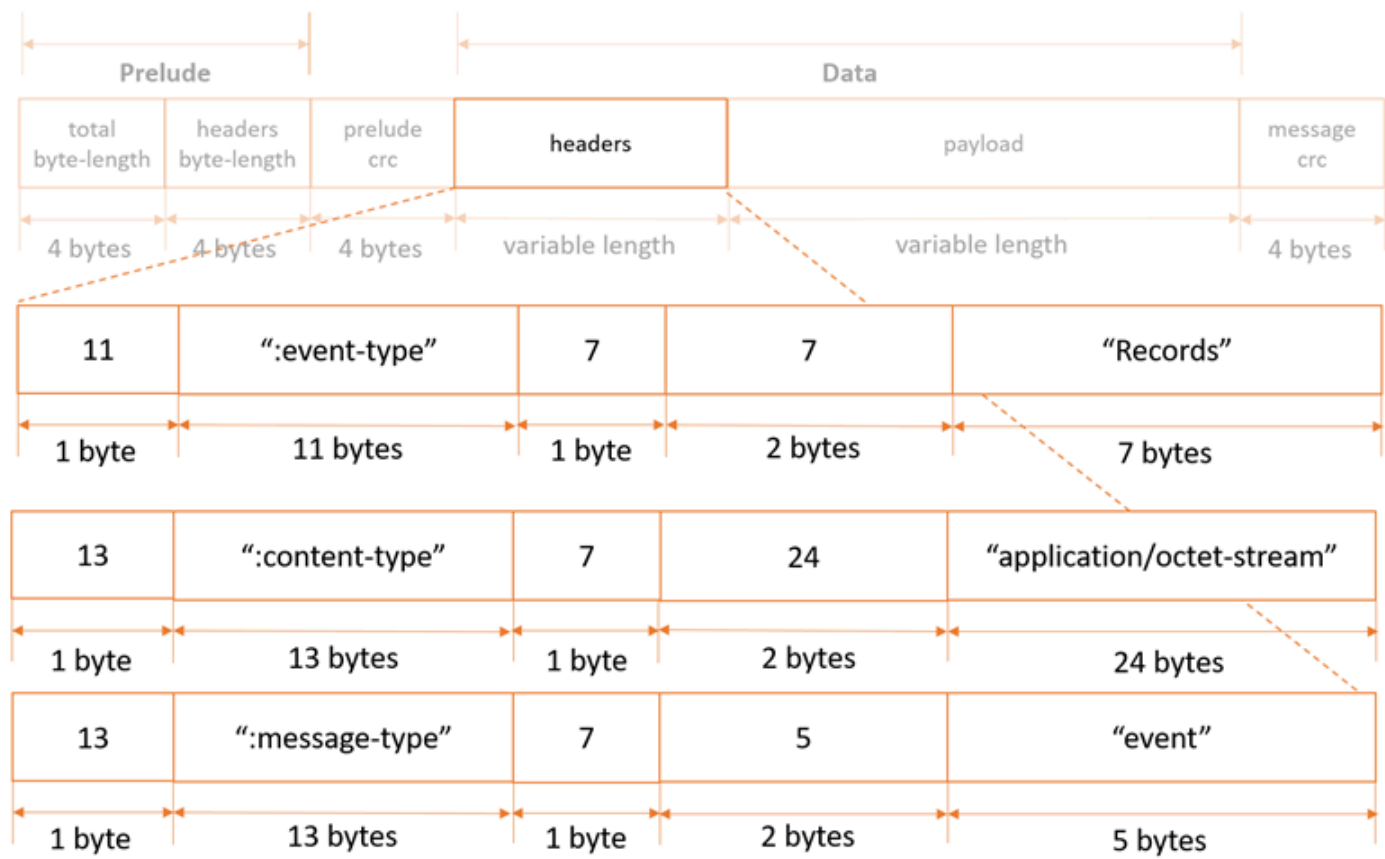
As seções a seguir explicam a estrutura de cada tipo de mensagem com mais detalhes.

Para obter exemplos de código e testes de unidade que usam esse protocolo, consulte [AWS C Event Stream](#) no site do GitHub.

Mensagem de registros

Especificação do cabeçalho

As mensagens de registros contêm três cabeçalhos, da seguinte forma:



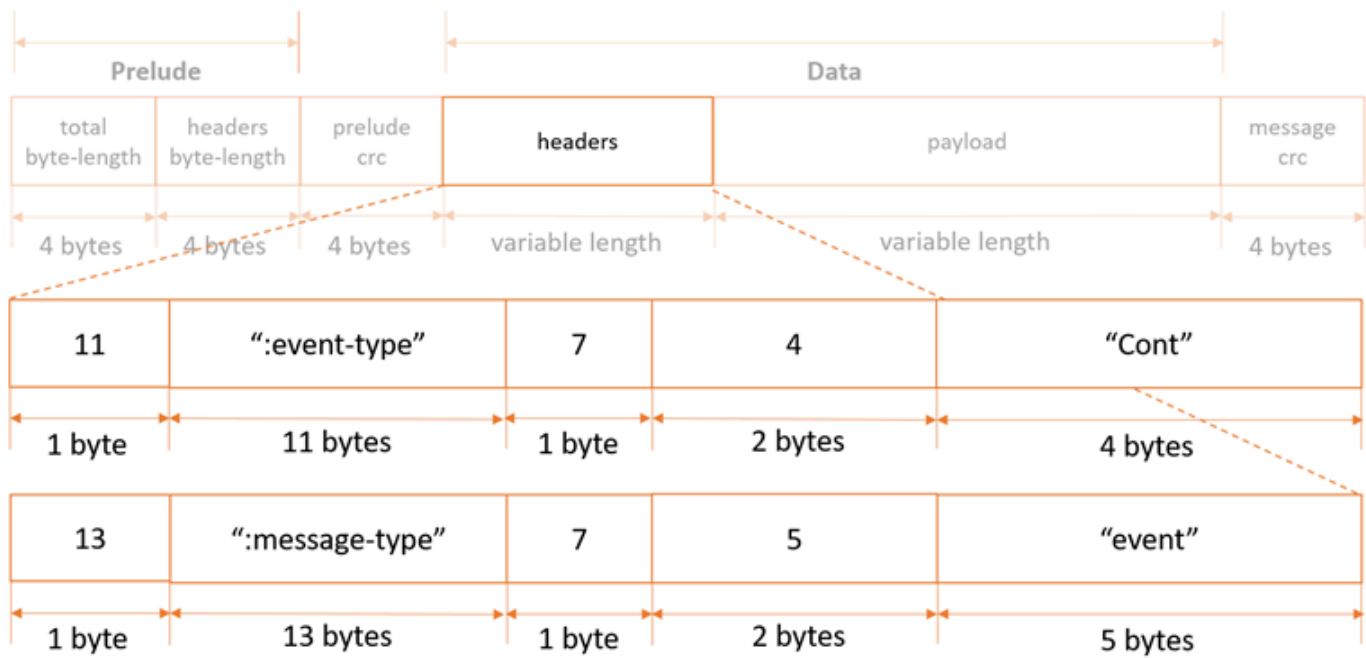
Especificação da carga útil

As cargas úteis de mensagens de registros podem conter um único registro, registros parciais ou vários registros.

Mensagem de continuação

Especificação do cabeçalho

As mensagens de continuação contêm dois cabeçalhos, da seguinte forma:



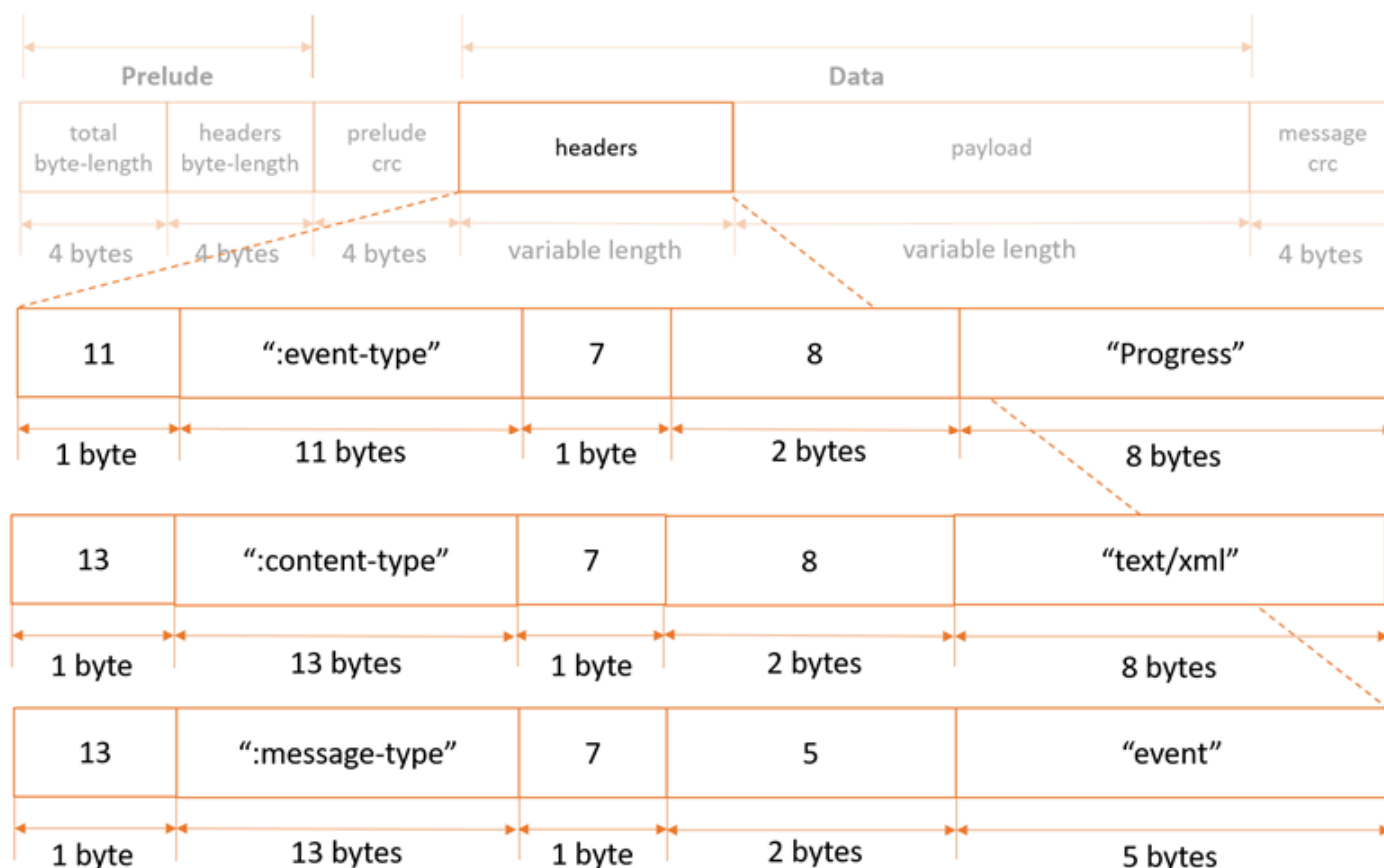
Especificação da carga útil

As mensagens de continuação não têm carga útil.

Mensagem de andamento

Especificação do cabeçalho

As mensagens de andamento contêm três cabeçalhos, da seguinte forma:



Especificação da carga útil

A carga útil da mensagem de andamento é um documento XML com informações sobre o andamento de uma solicitação.

- BytesScanned => Número de bytes que foram processados antes de serem descompactados (se o arquivo estiver compactado).
- BytesProcessed => Número de bytes que foram processados após serem descompactados (se o arquivo estiver compactado).
- BytesReturned => Número atual de bytes de dados de carga útil de registros retornados pelo Amazon S3.

Para arquivos não compactados, BytesScanned e BytesProcessed são iguais.

Exemplo:

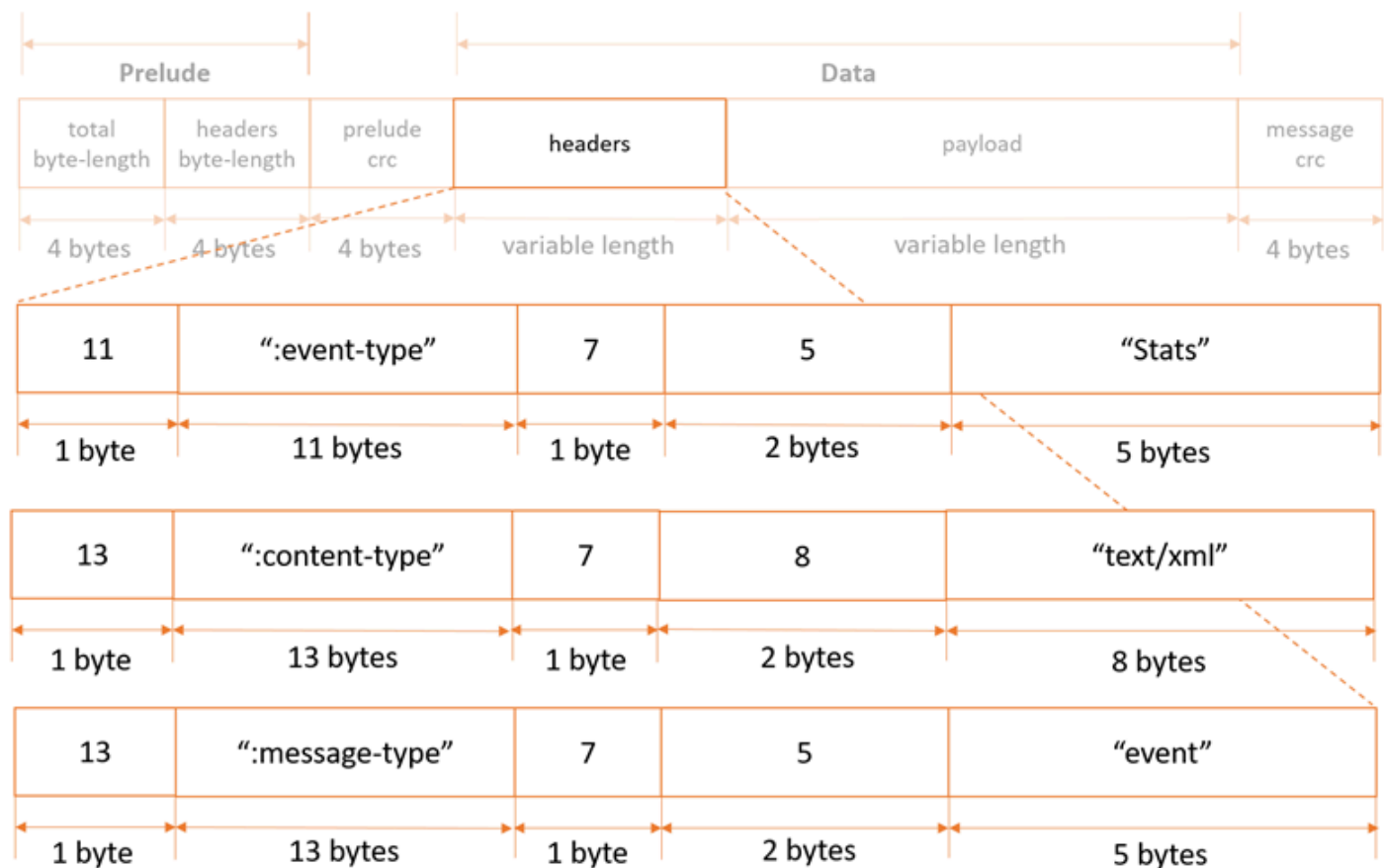
```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<Progress>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Progress>
```

Mensagem de estatísticas

Especificação do cabeçalho

As mensagens de estatísticas contêm três cabeçalhos, da seguinte forma:



Especificação da carga útil

A carga útil da mensagem de estatísticas é um documento XML com informações sobre as estatísticas de uma solicitação quando o processamento é concluído.

- **BytesScanned** => Número de bytes que foram processados antes de serem descompactados (se o arquivo estiver compactado).

- BytesProcessed => Número de bytes que foram processados após serem descompactados (se o arquivo estiver compactado).
- BytesReturned => Número total de bytes de dados de carga útil de registros retornados pelo Amazon S3.

Para arquivos não compactados, BytesScanned e BytesProcessed são iguais.

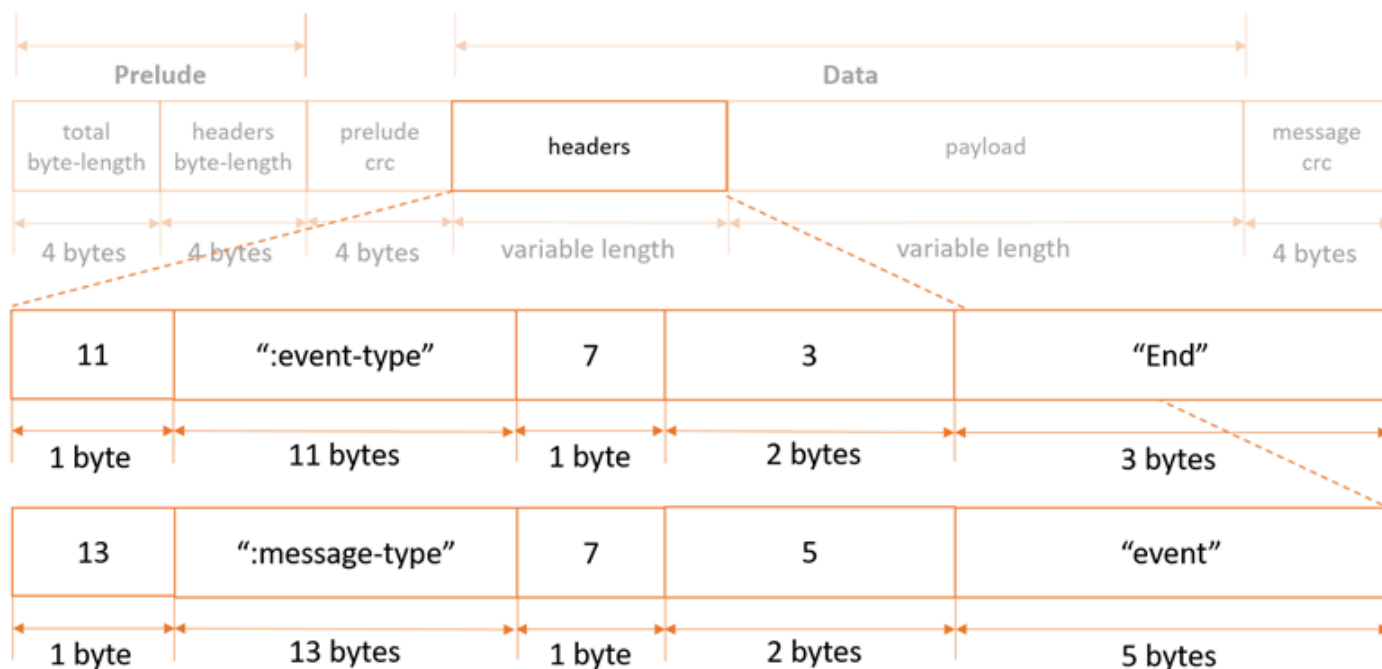
Exemplo:

```
<?xml version="1.0" encoding="UTF-8"?>
<Stats>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Stats>
```

Mensagem final

Especificação do cabeçalho

As mensagens finais contêm dois cabeçalhos, da seguinte forma:



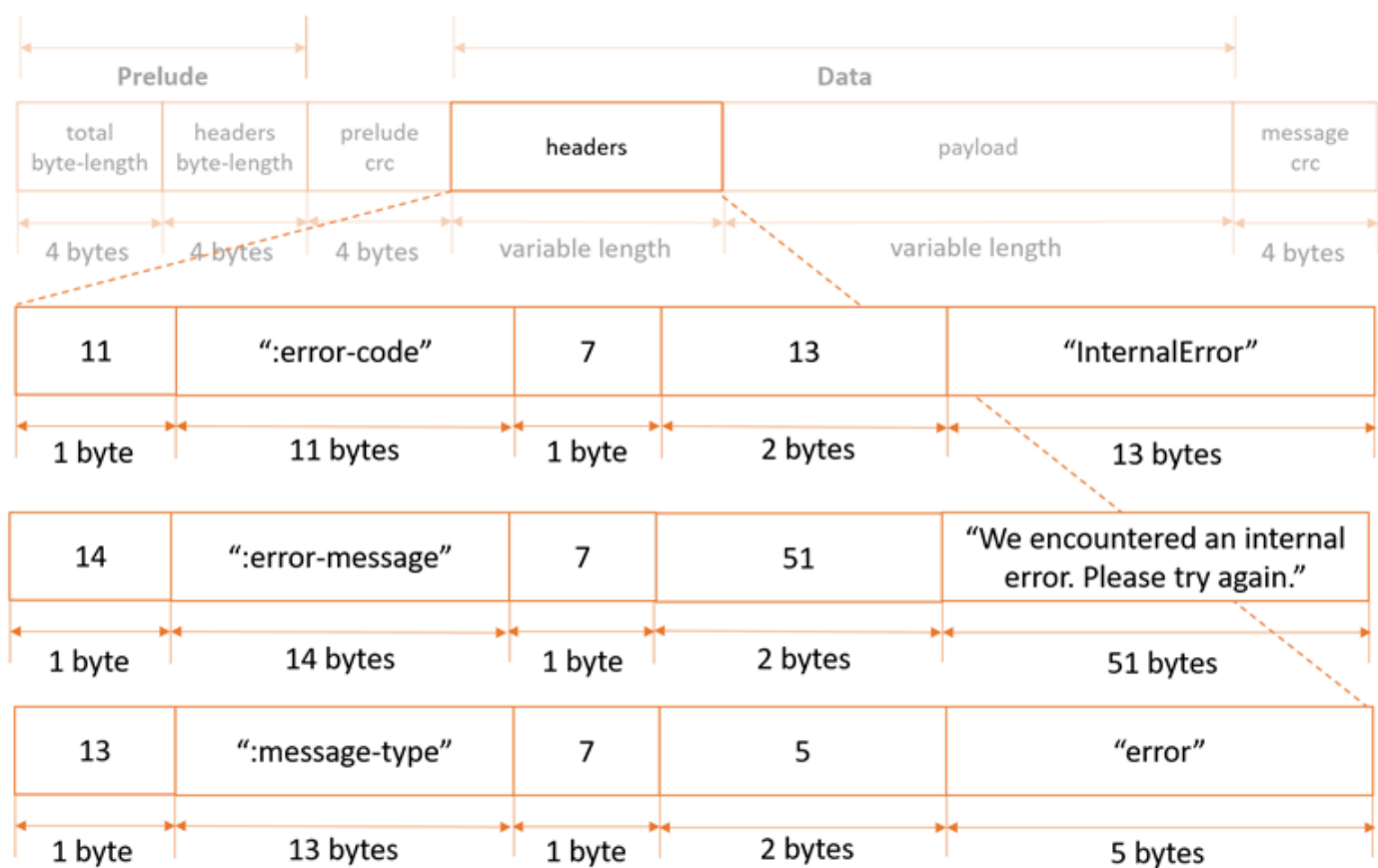
Especificação da carga útil

As mensagens finais não têm carga útil.

Mensagem de erro no nível da solicitação

Especificação do cabeçalho

As mensagens de erro no nível da solicitação contêm três cabeçalhos, da seguinte forma:



Para obter uma lista de possíveis códigos de erro e mensagens de erro, consulte a [Lista de códigos de erro de conteúdo do objeto SELECT](#).

Especificação da carga útil

As mensagens de erro no nível da solicitação não têm carga útil.

Recursos relacionados

- [SelectObjectContent](#)

- [GetObject](#)
- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)

Apêndice: objeto OPTIONS

Descrição

Um navegador pode enviar essa solicitação de simulação para o Amazon S3 para determinar se ele pode enviar uma solicitação real com a origem, o método HTTP e os cabeçalhos específicos.

O Amazon S3 oferece suporte ao compartilhamento de recursos de origem cruzada (CORS), permitindo que você adicione um sub-recurso `cors` em um bucket. Quando um navegador envia essa solicitação de simulação, o Amazon S3 responde avaliando as regras definidas na configuração de `cors`.

Se `cors` não estiver habilitado no bucket, o Amazon S3 retornará uma resposta 403 Forbidden.

Para obter mais informações sobre o CORS, consulte [Como habilitar o compartilhamento de recursos de origem cruzada](#) no Guia do usuário do Amazon Simple Storage Service.

Solicitações

Sintaxe

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

Parâmetros da solicitação

Essa operação não introduz nenhum parâmetro de solicitação específico, mas pode conter parâmetros de solicitação exigidos pela solicitação real.

Cabeçalhos de solicitação

Name (Nome)	Descrição	Obrigatório
Origin	Identifica a origem da solicitação de origem cruzada para o Amazon S3. Por exemplo, <code>http://www.exemplo.com</code> .	Sim

Name (Nome)	Descrição	Obrigatório
	Tipo: string Padrão: nenhum	
Access-Control-Request-Method	Identifica qual método HTTP será usado na solicitação real. Tipo: string Padrão: nenhum	Sim
Access-Control-Request-Headers	Uma lista delimitada por vírgulas de cabeçalhos HTTP que serão enviados na solicitação real. Por exemplo, para colocar um objeto com criptografia do lado do servidor, essa solicitação de simulação determinará se é possível incluir o cabeçalho <code>x-amz-server-side-encryption</code> na solicitação. Tipo: string Padrão: nenhum	Não

Elementos da solicitação

Essa implementação da operação não usa elementos de solicitação.

Respostas

Cabeçalhos de resposta

Cabeçalho	Descrição
Access-Control-Allow-Origin	

Cabeçalho	Descrição
	A origem que você enviou em sua solicitação. Se a origem na solicitação não for permitida, o Amazon S3 não incluirá esse cabeçalho na resposta. Tipo: string
Access-Control-Max-Age	Por quanto tempo, em segundos, os resultados da solicitação de simulação podem ser armazenados em cache. Tipo: string
Access-Control-Allow-Methods	O método HTTP que foi enviado na solicitação original. Se o método na solicitação não for permitido, o Amazon S3 não incluirá esse cabeçalho na resposta. Tipo: string
Access-Control-Allow-Headers	Uma lista delimitada por vírgulas de cabeçalhos HTTP que o navegador pode enviar na solicitação real. Se algum dos cabeçalhos solicitados não for permitido, o Amazon S3 não incluirá esse cabeçalho na resposta e a resposta não conterá nenhum dos cabeçalhos com o prefixo <code>Access-Control</code>. Tipo: string
Access-Control-Expose-Headers	Uma lista delimitada por vírgulas de cabeçalhos HTTP. Esse cabeçalho fornece ao cliente JavaScript acesso a esses cabeçalhos na resposta à solicitação real. Tipo: string

Elementos de resposta

Essa implementação da operação não retorna elementos de resposta.

Exemplos

Exemplo: enviar uma solicitação de simulação OPTIONS para um bucket habilitado para **cors**

Um navegador pode enviar essa solicitação de simulação para o Amazon S3 para determinar se ele pode enviar a solicitação PUT real da origem `http://www.exemplo.com` para o bucket do Amazon S3 chamado `examplebucket`.

Exemplo de solicitação

```
OPTIONS /exampleobject HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

Resposta da amostra

```
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
```

Recursos relacionados

- [GetBucketCors](#)
- [DeleteBucketCors](#)
- [PutBucketCors](#)

Apêndice: API SOAP

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

Esta seção descreve a API SOAP com relação às operações de serviço, bucket e objeto. As solicitações SOAP, autenticadas e anônimas, devem ser enviadas para o Amazon S3 usando SSL. O Amazon S3 retorna um erro quando você envia uma solicitação SOAP via HTTP.

O WSDL mais recente do Amazon S3 está disponível em docs.aws.amazon.com/2006-03-01/AmazonS3.wSDL.

Tópicos

- [Operações no serviço \(API SOAP\)](#)
- [Operações em buckets \(API SOAP\)](#)
- [Operações em objetos \(API SOAP\)](#)
- [Autenticar solicitações SOAP](#)
- [Configurar políticas de acesso padrão com SOAP](#)
- [Elementos comuns](#)
- [Respostas de erro de SOAP](#)

Operações no serviço (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

Esta seção descreve as operações que você pode realizar no serviço do Amazon S3.

Tópicos

- [ListAllMyBuckets \(API SOAP\)](#)

ListAllMyBuckets (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `ListAllMyBuckets` retorna uma lista de todos os buckets de propriedade do remetente da solicitação.

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>bcaf1fffd86f41161ca5fb16fd081034f</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes</Name>
      <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
    </Bucket>
    <Bucket>
      <Name>samples</Name>
    </Bucket>
  </Buckets>
</ListAllMyBucketsResult>
```

```
<CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
</Bucket>
</Buckets>
</ListAllMyBucketsResult>
```

Corpo da resposta

- **Owner :**

Isso fornece informações que o Amazon S3 usa para representar sua identidade para fins de autenticação e controle de acesso. O ID é um identificador exclusivo e permanente do desenvolvedor que fez a solicitação. DisplayName é um nome legível por humanos que representa o desenvolvedor que fez a solicitação. Não é exclusivo e pode mudar com o tempo. Recomendamos que você associe seu DisplayName ao nome do fórum.

- **Name :**

O nome de um bucket. Se um de seus buckets tiver sido excluído recentemente, o nome do bucket excluído ainda poderá estar presente nessa lista por um tempo.

- **CreationDate :**

A hora em que o bucket foi criado.

Controle de acesso

Você deve se autenticar com um ID válido de chave de acesso da AWS. Solicitações anônimas nunca podem listar buckets, e você só pode listar buckets dos quais seja o proprietário.

Operações em buckets (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

Esta seção descreve as operações que você pode realizar em buckets do Amazon S3.

Tópicos

- [CreateBucket \(API SOAP\)](#)
- [DeleteBucket \(API SOAP\)](#)
- [ListBucket \(API SOAP\)](#)
- [GetBucketAccessControlPolicy \(API SOAP\)](#)
- [SetBucketAccessControlPolicy \(API SOAP\)](#)
- [GetBucketLoggingStatus \(API SOAP\)](#)
- [SetBucketLoggingStatus \(API SOAP\)](#)

CreateBucket (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `CreateBucket` cria um bucket. Nem toda string é um nome de bucket aceitável. Para obter informações sobre restrições de nomenclatura de buckets, consulte [Como trabalhar com buckets do Amazon S3](#).

Note

Para determinar se existe um nome de bucket, use `ListBucket` e defina `MaxKeys` como 0. Uma resposta `NoSuchBucket` indica que o bucket está disponível, uma resposta `AccessDenied` indica que outra pessoa é proprietária do bucket e uma resposta `Success` indica que você é o proprietário do bucket ou tem permissão para acessá-lo.

Example Crie um bucket chamado “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**: O nome do bucket que você está tentando criar.
- **AccessControlList**: A lista de controle de acesso do novo bucket. O elemento é opcional. Se não for fornecido, o bucket será criado com uma política de acesso que concede acesso **FULL_CONTROL** ao solicitante.

Controle de acesso

Você deve se autenticar com um ID válido de chave de acesso da AWS. Solicitações anônimas nunca podem criar buckets.

Recursos relacionados

- [ListBucket \(API SOAP\)](#)

DeleteBucket (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação DeleteBucket exclui um bucket. Todos os objetos no bucket devem ser excluídos antes que o bucket em si possa ser excluído.

Example

Este exemplo exclui o 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**: O nome do bucket que você deseja excluir.

Controle de acesso

Somente o proprietário do bucket tem permissão para excluí-lo, independentemente da política de controle de acesso no bucket.

ListBucket (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `ListBucket` retorna informações sobre alguns dos itens no bucket.

Para uma introdução geral à operação de lista, consulte [Listagem de chaves de objeto](#).

Solicitações

Este exemplo lista até 1000 chaves no bucket “quotes” que têm o prefixo “notes”.

Sintaxe

```
<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>
```

Parâmetros

Name (Nome)	Descrição	Obrigatório
<code>prefix</code>	Limita a resposta às chaves que começam com o prefixo indicado. Você pode usar prefixos para separar um bucket em diferentes conjuntos de chaves de modo semelhante à forma como um sistema de arquivos usa pastas. Tipo: string Padrão: nenhum	Não
<code>marker</code>	Indica onde começar a listar no bucket. A lista incluirá apenas as chaves que ocorrem lexicograficamente após o marcador. Isso é útil para paginação: para obter a próxima página de resultados, use a última chave da página atual como marcador. Tipo: string	Não

Name (Nome)	Descrição	Obrigatório
	Padrão: nenhum	
max-keys	O número máximo de chaves que você quer ver no corpo da resposta. O servidor pode retornar menos do que essa quantidade de chaves, mas não retornará mais. Tipo: string Padrão: nenhum	Não
delimiter	Faz com que as chaves com a mesma string entre o prefixo e a primeira ocorrência do delimitador sejam agrupadas em um único elemento de resultado na coleção CommonPrefixes. Essas chaves agrupadas não são retornadas em nenhum outro lugar na resposta. Tipo: string Padrão: nenhum	Não

Resposta de êxito

Essa resposta presume que o bucket contenha as seguintes chaves:

```
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
```

Sintaxe

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>backups</Name>
  <Prefix>notes</Prefix>
  <MaxKeys>1000</MaxKeys>
  <Delimiter></Delimiter>
  <IsTruncated>false</IsTruncated>
```

```
<Contents>
  <Key>notes/todos.txt</Key>
  <LastModified>2006-01-01T12:00:00.000Z</LastModified>
  <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
  <Size>5126</Size>
  <StorageClass>STANDARD</StorageClass>
  <Owner>
    <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <StorageClass>STANDARD</StorageClass>
</Contents>
<CommonPrefixes>
  <Prefix>notes/2005-05-23/</Prefix>
</CommonPrefixes>
<CommonPrefixes>
  <Prefix>notes/2005-05-28/</Prefix>
</CommonPrefixes>
</ListBucketResult>
```

Como você pode ver, muitos campos na resposta refletem os parâmetros da solicitação. `IsTruncated`, `Contents` e `CommonPrefixes` são os únicos elementos de resposta que podem conter novas informações.

Elementos de resposta

Name (Nome)	Descrição
Contents	Metadados sobre cada objeto retornado. Tipo: metadados XML Antepassado: ListBucketResult
CommonPrefixes	Uma resposta poderá conter <code>CommonPrefixes</code> somente se você especificar um <code>delimiter</code> . Ao fazer isso, <code>CommonPrefixes</code> contém todas as chaves (se houver) entre <code>Prefix</code> e a próxima ocorrência da string especificada pelo <code>delimiter</code> . Na verdade, <code>CommonPrefixes</code> lista as chaves que agem como subdiretórios no diretório especificado pelo <code>Prefix</code> . Por exemplo, se <code>prefix</code> for <code>notes/</code> e <code>delimiter</code> for uma barra (/), em <code>notes/summer/july</code> , o prefixo comum será <code>notes/summer/</code> .

Name (Nome)	Descrição
	Tipo: string Antepassado: ListBucketResult
Delimiter	Faz com que as chaves com a mesma string entre o prefixo e a primeira ocorrência do delimitador sejam agrupadas em um único elemento de resultado na coleção CommonPrefixes. Essas chaves agrupadas não são retornadas em nenhum outro lugar na resposta. Tipo: string Antepassado: ListBucketResult
IsTruncated	Especifica se todos os resultados foram retornados (verdadeiro) ou não (falso). Todos os resultados podem não ser retornados se o número de resultados exceder o especificado por MaxKeys. Tipo: string Antepassado: booleano
Marker	Indica onde começar a listar no bucket. Tipo: string Antepassado: ListBucketResult
MaxKeys	O número máximo de chaves retornadas no corpo da resposta. Tipo: string Antepassado: ListBucketResult
Name	Nome do bucket. Tipo: string Antepassado: ListBucketResult

Name (Nome)	Descrição
Prefix	Chaves que começam com o prefixo indicado. Tipo: string Antepassado: ListBucketResult

Corpo da resposta

Para obter informações sobre a resposta da lista, consulte [Listagem de resposta de chaves](#).

Controle de acesso

Para listar as chaves de um bucket, você precisa ter acesso READ ao bucket.

GetBucketAccessControlPolicy (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `GetBucketAccessControlPolicy` busca a política de controle de acesso de um bucket.

Example

Este exemplo recupera a política de controle de acesso para o 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>
```

Corpo da resposta

A resposta contém a política de controle de acesso do bucket. Para obter uma explicação sobre essa resposta, consulte [Política de acesso SOAP](#).

Controle de acesso

Você deve ter direitos READ_ACP sobre o bucket para recuperar a política de controle de acesso de um bucket.

SetBucketAccessControlPolicy (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `SetBucketAccessControlPolicy` define a política de controle de acesso para um bucket existente. Se for bem-sucedida, a política de controle de acesso anterior do bucket será totalmente substituída pela política de controle de acesso especificada.

Example

Dê ao usuário especificado (geralmente o proprietário) acesso `FULL_CONTROL` ao 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>
```

Controle de acesso

Você deve ter direitos `WRITE_ACP` sobre o bucket para definir a política de controle de acesso de um bucket.

GetBucketLoggingStatus (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `GetBucketLoggingStatus` recupera o status de registro em log de um bucket existente.

Para uma introdução geral a esse recurso, consulte [Logs do 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>
```

Controle de acesso

Apenas o proprietário de um bucket tem permissão para invocar essa operação.

SetBucketLoggingStatus (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação SetBucketLoggingStatus atualiza o status de registro em log de um bucket existente.

Para uma introdução geral a esse recurso, consulte [Logs do servidor](#).

Example

Esse exemplo de solicitação habilita o registro em log de acesso ao servidor para o bucket 'mybucket' e configura os logs para serem entregues a 'mylogs' com o prefixo '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>
```

Controle de acesso

Apenas o proprietário de um bucket tem permissão para invocar essa operação.

Operações em objetos (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

Esta seção descreve as operações que você pode realizar em objetos do Amazon S3.

Tópicos

- [PutObjectInline \(API SOAP\)](#)
- [PutObject \(API SOAP\)](#)
- [CopyObject \(API SOAP\)](#)
- [GetObject \(API SOAP\)](#)
- [GetObjectExtended \(API SOAP\)](#)
- [DeleteObject \(API SOAP\)](#)
- [GetObjectAccessControlPolicy \(API SOAP\)](#)
- [SetObjectAccessControlPolicy \(API SOAP\)](#)

PutObjectInline (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `PutObjectInline` adiciona um objeto a um bucket. Os dados do objeto são fornecidos no corpo da mensagem SOAP.

Se um objeto já existir em um bucket, o novo objeto o substituirá porque o Amazon S3 armazena a última solicitação de gravação. No entanto, o Amazon S3 é um sistema distribuído. Se o Amazon

S3 receber várias solicitações de gravação para o mesmo objeto quase simultaneamente, todos os objetos poderão ser armazenados, mesmo que apenas um vença no final. O Amazon S3 não fornece bloqueio de objetos; se você precisar disso, incorpore-o à camada de aplicativo.

Para garantir que um objeto não seja corrompido na rede, calcule o MD5 de um objeto, realize PUT no Amazon S3 e compare o Etag retornado com o valor calculado de MD5.

PutObjectInline não é adequado para uso com objetos grandes. O sistema limita essa operação ao trabalho com objetos de 1 MB ou menos. PutObjectInline falhará com o código de status `InlineDataTooLargeError` se o parâmetro `Data` codificar um objeto com mais de 1 MB. Para fazer upload de objetos grandes, considere usar a API PutObject não integrada ou a API REST.

Example

Este exemplo grava alguns textos e metadados no objeto "Nelson" no bucket "quotes", concede ao usuário (geralmente o proprietário) acesso `FULL_CONTROL` ao objeto e torna o objeto legível por partes anônimas.

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>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2006-01-01T12:00:00.000Z</lastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```

Elementos

- **Bucket**: O bucket ao qual adicionar o objeto.
- **Key**: A chave a ser atribuída ao objeto.
- **Metadata**: Você pode fornecer pares de metadados de nome e valor no elemento de metadados. Eles serão armazenados com o objeto.
- **Data**: A forma com codificação base 64 dos dados.
- **ContentLength**: O comprimento dos dados em bytes.
- **AccessControlList**: Uma lista de controle de acesso para o recurso. O elemento é opcional. Se for omitido, o solicitante terá acesso FULL_CONTROL ao objeto. Se o objeto já existir, a política de controle de acesso preexistente será substituída.

Respostas

- **ETag**: ETag (identificador de entidade) é um hash MD5 do objeto que você pode usar para fazer buscas condicionais do objeto usando `GetObjectExtended`. O ETag reflete apenas as alterações no conteúdo de um objeto, não nos metadados.
- **LastModified**: O carimbo de data/hora do Amazon S3 para o objeto salvo.

Controle de acesso

Você deve ter acesso **WRITE** ao bucket para colocar objetos no bucket.

Recursos relacionados

- [PutObject \(API SOAP\)](#)
- [CopyObject \(API SOAP\)](#)

PutObject (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `PutObject` adiciona um objeto a um bucket. Os dados do objeto são anexados como um anexo DIME.

Para garantir que um objeto não seja corrompido na rede, calcule o MD5 de um objeto, realize **PUT** no Amazon S3 e compare o Etag retornado com o valor calculado de MD5.

Se um objeto já existir em um bucket, o novo objeto o substituirá porque o Amazon S3 armazena a última solicitação de gravação. No entanto, o Amazon S3 é um sistema distribuído. Se o Amazon S3 receber várias solicitações de gravação para o mesmo objeto quase simultaneamente, todos os objetos poderão ser armazenados, mesmo que apenas um vença no final. O Amazon S3 não fornece bloqueio de objetos; se você precisar disso, incorpore-o à camada de aplicativo.

Example

Este exemplo coloca alguns dados e metadados no objeto “Nelson” do bucket “quotes”, concede ao usuário (geralmente o proprietário) acesso FULL_CONTROL ao objeto e torna o objeto legível por partes anônimas. Neste exemplo, o anexo real não é mostrado.

Sample Request

```
<PutObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2007-05-11T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObject>
```

Sample Response

```
<PutObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
```

```
<PutObjectResponse>
  <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  <LastModified>2006-03-01T12:00:00.183Z</LastModified>
</PutObjectResponse>
</PutObjectResponse>
```

Elementos

- **Bucket**: O bucket ao qual adicionar o objeto.
- **Key**: A chave a ser atribuída ao objeto.
- **Metadata**: Você pode fornecer pares de metadados de nome e valor no elemento de metadados. Eles serão armazenados com o objeto.
- **ContentLength**: O comprimento dos dados em bytes.
- **AccessControlList**: Uma lista de controle de acesso para o recurso. O elemento é opcional. Se for omitido, o solicitante terá acesso FULL_CONTROL ao objeto. Se o objeto já existir, a política de controle de acesso preexistente será substituída.

Respostas

- **ETag**: ETag (identificador de entidade) é um hash MD5 do objeto que você pode usar para fazer buscas condicionais do objeto usando GetObjectExtended. O ETag reflete apenas as alterações no conteúdo de um objeto, não nos metadados.
- **LastModified**: O carimbo de data/hora do Amazon S3 para o objeto salvo.

Controle de acesso

Para colocar objetos em um bucket, você deve ter acesso WRITE ao bucket.

Recursos relacionados

- [CopyObject \(API SOAP\)](#)

CopyObject (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

Descrição

A operação `CopyObject` cria uma cópia de um objeto quando você especifica a chave e o bucket de um objeto de origem e a chave e o bucket de um destino.

Ao copiar um objeto, você pode preservar todos os metadados (padrão) ou especificar novos metadados. No entanto, a ACL não é preservada e está definida como `private` para o usuário que faz a solicitação. Para substituir a configuração de ACL padrão, especifique uma nova ACL ao gerar uma solicitação de cópia. Para obter mais informações, consulte [Usar ACLs](#).

Todas as solicitações de cópia devem ser autenticadas. Além disso, você deve ter acesso de leitura ao objeto de origem e acesso de gravação ao bucket de destino. Para obter mais informações, consulte [Uso de acesso de autenticação](#).

Para copiar um objeto somente sob determinadas condições, como se o Etag corresponde ou se o objeto foi modificado antes ou depois de uma data especificada, use os parâmetros de solicitação `CopySourceIfUnmodifiedSince`, `CopyIfUnmodifiedSince`, `CopySourceIfMatch` ou `CopySourceIfNoneMatch`.

Note

Talvez seja necessário configurar o tempo limite do soquete da pilha SOAP para copiar objetos grandes.

Sintaxe da solicitação

```
<CopyObject xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <SourceBucket>source_bucket</SourceBucket>
  <SourceObject>source_object</SourceObject>
```



```
<DestinationBucket>destination_bucket</DestinationBucket>
<DestinationObject>destination_object</DestinationObject>
<MetadataDirective>{REPLACE | COPY}</MetadataDirective>
<Metadata>
  <Name>metadata_name</Name>
  <Value>metadata_value</Value>
</Metadata>
...
<AccessControlList>
  <Grant>
    <Grantee xsi:type="user_type">
      <ID>user_id</ID>
      <DisplayName>display_name</DisplayName>
    </Grantee>
    <Permission>permission</Permission>
  </Grant>
  ...
</AccessControlList>
<CopySourceIfMatch>etag</CopySourceIfMatch>
<CopySourceIfNoneMatch>etag</CopySourceIfNoneMatch>
<CopySourceIfModifiedSince>date_time</CopySourceIfModifiedSince>
<CopySourceIfUnmodifiedSince>date_time</CopySourceIfUnmodifiedSince>
<AWSAccessKeyId>AWSAccessKeyId</AWSAccessKeyId>
<Timestamp>TimeStamp</Timestamp>
<Signature>Signature</Signature>
</CopyObject>
```

Parâmetros da solicitação

Name (Nome)	Descrição	Obrigatório
SourceBucket	O nome do bucket de origem.	Sim
	Tipo: string	
	Padrão: nenhum	
	Restrições: um bucket de origem válido.	
SourceKey	O nome da chave do objeto de origem.	Sim
	Tipo: string	

Name (Nome)	Descrição	Obrigatório
	Padrão: nenhum Restrições: a chave de um objeto de origem válido ao qual você tem acesso READ.	
DestinationBucket	O nome do bucket do de destino. Tipo: string Padrão: nenhum Restrições: você deve ter acesso WRITE ao bucket de destino.	Sim
DestinationKey	A chave do objeto de destino. Tipo: string Padrão: nenhum Restrições: você deve ter acesso WRITE ao bucket de destino.	Sim

Name (Nome)	Descrição	Obrigatório
MetadataDirective	Especifica se os metadados são copiados do objeto de origem ou substituídos pelos metadados fornecidos na solicitação. Tipo: string Padrão: COPY Valores válidos: COPY REPLACE Restrições: valores diferentes de COPY ou REPLACE resultarão em um erro imediato. Você não pode copiar um objeto para si mesmo, a menos que o cabeçalho MetadataDirective seja especificado e seu valor definido como REPLACE.	Não
Metadata	Especifica os pares de nome-valor dos metadados a serem definidos para o objeto. Se MetadataDirective estiver definido como COPY, todos os metadados serão ignorados. Tipo: string Padrão: nenhum Restrições: nenhuma.	Não
AccessControlList	Concede acesso aos usuários por endereços de e-mail ou ID de usuário canônico. Tipo: string Padrão: nenhum Restrições: nenhuma	Não

Name (Nome)	Descrição	Obrigatório
CopySourceIfMatch	Copiará o objeto se o ETag (identificador de entidade) corresponder à tag especificada; caso contrário, retornará um PreconditionFailed. Tipo: string Padrão: nenhum Restrições: nenhuma. Se o Etag não corresponder, o objeto não será copiado.	Não
CopySourceIfNoneMatch	Copiará o objeto se o ETag (identificador de entidade) for diferente do Etag especificado; caso contrário, retornará um erro. Tipo: string Padrão: nenhum Restrições: nenhuma.	Não
CopySourceIfUnmodifiedSince	Copiará o objeto se ele não tiver sido modificado desde o horário especificado; caso contrário, retornará um PreconditionFailed. Tipo: dateTime Padrão: nenhum	Não

Name (Nome)	Descrição	Obrigatório
CopySourceIfModifiedSince	Copiará o objeto se ele tiver sido modificado desde o horário especificado; caso contrário, retornará um erro. Tipo: dateTime Padrão: nenhum	Não

Sintaxe da resposta

```
<CopyObjectResponse xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">  
  <CopyObjectResponse>  
    <ETag>"etag"</ETag>  
    <LastModified>timestamp</LastModified>  
  </CopyObjectResponse>  
</CopyObjectResponse>
```

Elementos de resposta

Confira a seguir uma lista de elementos de resposta.

Note

A API SOAP não retorna espaços em branco extras. O espaço em branco extra só é retornado pela API REST.

Name (Nome)	Descrição
Etag	Retorna o Etag do novo objeto. O ETag reflete apenas as alterações no conteúdo de um objeto, não nos metadados. Tipo: string Antepassado: CopyObjectResult

Name (Nome)	Descrição
LastModified	Retorna a data em que o objeto foi modificado pela última vez. Tipo: string Antepassado: CopyObjectResult

Para obter informações sobre elementos de resposta gerais, consulte [Uso de cabeçalhos de resposta de erro REST](#).

Erros especiais

Não há erros especiais para essa operação. Para obter informações sobre erros gerais do Amazon S3, consulte [Lista de códigos de erro](#).

Exemplos

Este exemplo copia o objeto flotsam do bucket pacific para o objeto jetsam do bucket atlantic, preservando seus metadados.

Exemplo de solicitação

```
<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>
```

Resposta da amostra

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
```

```
</CopyObjectResponse>
</CopyObjectResponse>
```

Este exemplo copia o objeto "tweedledee" do bucket wonderland para o objeto "tweedledum" do bucket wonderland, substituindo os metadados.

Exemplo de solicitação

```
<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>
```

Resposta da amostra

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Recursos relacionados

- [PutObject \(API SOAP\)](#)
- [PutObjectInline \(API SOAP\)](#)

GetObject (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `GetObject` retorna a versão atual de um objeto. Se você tentar executar `GetObject` em um objeto que tenha um marcador de exclusão como a versão atual, o S3 retornará um erro 404. Você não pode usar a API SOAP para recuperar uma versão especificada de um objeto. Para fazer isso, use a API REST. Para obter mais informações, consulte [Versionamento](#). Para obter mais opções, use a operação [GetObjectExtended \(API SOAP\)](#).

Note

Os nomes de chave de objeto com o valor “soap” não são compatíveis com [solicitações de estilo de hospedagem virtual](#). Quanto a valores de nome de chave de objeto em que se usa “soap”, deve-se usar, em vez disso, um [URL de estilo de caminho](#).

Example

Este exemplo obtém o objeto "Nelson" do 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>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
  </GetObjectResponse>
</GetObjectResponse>
```

Elementos

- **Bucket**: O bucket do qual recuperar o objeto.
- **Key**: A chave que identifica o objeto.
- **GetMetadata**: Os metadados serão retornados com o objeto se for verdadeiro.
- **GetData**: Os dados do objeto serão retornados se for verdadeiro.
- **InlineData**: Se for verdadeiro, os dados serão retornados, com codificação base 64, como parte do corpo SOAP da resposta. Se for falso, os dados serão retornados como um anexo SOAP. A opção `InlineData` não é adequada para uso com objetos grandes. O sistema limita essa operação a trabalhar com 1 MB de dados ou menos. Uma solicitação `GetObject` com o sinalizador `InlineData` definido falhará com o código de status `InlineDataTooLargeError` se o parâmetro `Data` resultante tiver codificado mais de 1 MB. Para baixar objetos grandes, considere chamar `GetObject` sem definir o sinalizador `InlineData` ou use a API REST.

Elementos retornados

- **Metadata**: Os metadados em pares de nome-valor armazenados com o objeto.

- **Data**: Se `InlineData` for verdadeiro na solicitação, o elemento conterá os dados do objeto codificados em base 64.
- **LastModified**: A hora em que o objeto foi armazenado no Amazon S3.
- **ETag**: O ETag (identificador de entidade) do objeto. Trata-se de um hash do objeto que pode ser usado para obter dados condicionais. O ETag reflete apenas as alterações no conteúdo de um objeto, não nos metadados.

Controle de acesso

Você poderá ler um objeto somente se tiver recebido acesso `READ` ao objeto.

Downloads de SOAP fragmentados e retomáveis

Para oferecer flexibilidade a operações `GET`, o Amazon S3 oferece suporte a downloads fragmentados e retomáveis.

Escolha entre as seguintes opções:

- Para downloads de objetos grandes, divida-os em partes menores. Para obter mais informações, consulte . [Operações GET de intervalo](#)
- Para operações `GET` que falham, você pode criar seu aplicativo para baixar o restante em vez do arquivo inteiro. Para obter mais informações, consulte . [Recuperação de erros REST GET](#)

Operações GET de intervalo

Para alguns clientes, você talvez queira dividir downloads grandes em downloads menores. Para dividir um `GET` em unidades menores, use `Range`.

Antes de dividir um `GET` em unidades menores, você deve determinar seu tamanho. Por exemplo, a solicitação a seguir obtém o tamanho do 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>
```

O Amazon S3 retorna a seguinte resposta.

```
<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>
```

Confira abaixo uma solicitação que baixa o primeiro megabyte do 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>
```

O Amazon S3 retorna o primeiro megabyte do arquivo e o Etag do arquivo.

```
<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 garantir que o arquivo não tenha sido alterado desde o download da parte anterior, especifique o elemento `IfMatch`. Embora o elemento `IfMatch` não seja obrigatório, ele é recomendado para conteúdo que provavelmente mudará.

Confira abaixo uma solicitação que obtém o restante do arquivo, usando o cabeçalho da solicitação `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>"828ef3fdfa96f00ad9f27c383fc9ac7f"</IfMatch>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</GetObject>

```

O Amazon S3 retorna a resposta a seguir e o restante do arquivo.

```

<GetObjectResponse xmlns="http://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>

```

```

    <Description>OK</Description>
  </Status>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>>Muntz</Value>
  </Metadata>
  <Data>--remainder of bigfile--</Data>
  <LastModified>2006-01-01T12:00:00.000Z</LastModified>
  <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
</GetObjectResponse>
</GetObjectResponse>

```

GetObject com versão

A solicitação a seguir retorna a versão especificada do objeto no 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>

```

Resposta da amostra

```

<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <GetObjectResponse>
    <Status>
      <Code>200</Code>
    </Status>
    <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>
```

Recuperação de erros REST GET

Se um objeto GET falhar, você poderá obter o restante do arquivo especificando o intervalo a ser baixado. Para fazer isso, você deve obter o tamanho do objeto usando `ListBucket` e executar um GET de intervalo no restante do arquivo. Para obter mais informações, consulte [GetObjectExtended \(API SOAP\)](#).

Recursos relacionados

[Operações em objetos \(API SOAP\)](#)

GetObjectExtended (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

`GetObjectExtended` é exatamente igual a [GetObject \(API SOAP\)](#), exceto pelo fato de oferecer suporte aos seguintes elementos adicionais que podem ser usados para realizar grande parte da mesma funcionalidade fornecida pelos cabeçalhos HTTP GET (acesse <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>).

`GetObjectExtended` oferece suporte aos seguintes elementos, além dos que são compatíveis com `GetObject`:

- `ByteRangeStart`, `ByteRangeEnd`: Esses elementos especificam que somente uma parte dos dados do objeto deve ser recuperada. Eles seguem o comportamento dos intervalos de bytes HTTP (acesse <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35>).

- **IfModifiedSince**: Retorne o objeto somente se o carimbo de data e hora do objeto for posterior ao carimbo de data e hora especificado. (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.25>)
- **IfUnmodifiedSince**: Retorne o objeto somente se o carimbo de data e hora do objeto for anterior ou igual ao carimbo de data e hora especificado. (acesse <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.28>)
- **IfMatch**: Retorne o objeto somente se o ETag corresponder às tags fornecidas. (acesse <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24>)
- **IfNoneMatch**: Retorne o objeto somente se o ETag não corresponder às tags fornecidas. (acesse <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26>)
- **ReturnCompleteObjectOnConditionFailure**: **ReturnCompleteObjectOnConditionFailure**: se for verdadeiro, e se a solicitação incluir um elemento de intervalo e um ou os dois elementos **IfUnmodifiedSince**/**IfMatch**, e a condição falhar, retorne o objeto inteiro em vez de uma falha. Isso habilita a funcionalidade **If-Range** (acesse <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27>).

DeleteObject (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `DeleteObject` remove o objeto especificado do Amazon S3. Depois de excluído, não há método para restaurar ou recuperar um objeto.

Note

Se você excluir um objeto que não existe, o Amazon S3 retornará um sucesso (não uma mensagem de erro).

Example

Este exemplo exclui o objeto "Nelson" do 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**: O bucket que contém o objeto.
- **Key**: A chave que identifica o objeto.

Controle de acesso

Você só pode excluir um objeto se tiver acesso **WRITE** ao bucket, independentemente de quem é o proprietário do objeto ou de quais direitos são concedidos a ele.

GetObjectAccessControlPolicy (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `GetObjectAccessControlPolicy` busca a política de controle de acesso de um objeto.

Example

Este exemplo recupera a política de controle de acesso para o objeto "Nelson" do 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>
```

Corpo da resposta

A resposta contém a política de controle de acesso do bucket. Para obter uma explicação sobre essa resposta, consulte a [Política de acesso SOAP](#).

Controle de acesso

Você deve ter direitos READ_ACP sobre o objeto para recuperar a política de controle de acesso de um objeto.

SetObjectAccessControlPolicy (API SOAP)

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

A operação `SetObjectAccessControlPolicy` define a política de controle de acesso para um objeto existente. Se for bem-sucedida, a política de controle de acesso anterior do objeto será totalmente substituída pela política de controle de acesso especificada.

Example

Este exemplo fornece ao usuário especificado (geralmente o proprietário) acesso `FULL_CONTROL` ao objeto "Nelson" do 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>
```

Chave

Controle de acesso

Você deve ter direitos WRITE_ACP sobre o objeto para definir a política de controle de acesso de um bucket.

Autenticar solicitações SOAP

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

Cada solicitação não anônima deve conter informações de autenticação para estabelecer a identidade do principal que faz a solicitação. Em SOAP, as informações de autenticação são colocadas nos seguintes elementos da solicitação SOAP:

- Seu ID da chave de acesso da AWS

Note

Ao fazer solicitações SOAP autenticadas, não há suporte para credenciais de segurança temporárias. Para obter mais informações sobre os tipos de credenciais, consulte [Fazer solicitações](#).

- **Timestamp:** Deve ser um dateTime (acesse <http://www.w3.org/TR/xmlschema-2/#dateTime>) o fuso horário universal coordenado (horário médio de Greenwich), como

2009-01-01T12:00:00.000Z. Haverá falha na autorização se esse time stamp tiver mais de 15 minutos de diferença do relógio nos servidores do Amazon S3.

- **Signature:** O resumo da RFC 2104 HMAC-SHA1 (acesse <http://www.ietf.org/rfc/rfc2104.txt>) da concatenação de "AmazonS3" + OPERAÇÃO + carimbo de data/hora, usando sua chave de acesso secreta da AWS como chave. Por exemplo, na solicitação de exemplo CreateBucket a seguir, o elemento de assinatura conteria o resumo HMAC-SHA1 do valor "AmazonS3CreateBucket2009-01-01T12:00:00.000Z":

Por exemplo, na solicitação de exemplo CreateBucket a seguir, o elemento de assinatura conteria o resumo HMAC-SHA1 do 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

As solicitações SOAP, autenticadas e anônimas, devem ser enviadas para o Amazon S3 usando SSL. O Amazon S3 retorna um erro quando você envia uma solicitação SOAP via HTTP.

Important

Devido a interpretações diferentes em relação a como a precisão de tempo extra deve ser aplicada, os usuários de .NET devem tomar cuidado para não enviar datas e horas excessivamente específicas ao Amazon S3. Isso pode ser realizado criando manualmente objetos DateTime com precisão de apenas milissegundos.

Configurar políticas de acesso padrão com SOAP

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

O controle de acesso pode ser definido no momento em que um bucket ou um objeto são gravados ao se incluir o elemento "AccessControlList" com a solicitação para CreateBucket, PutObjectInline ou PutObject. O elemento AccessControlList é descrito em [Identity and Access Management para Amazon S3](#). Se nenhuma lista de controle de acesso for especificada com essas operações, o recurso será criado com uma política de acesso padrão que dá ao solicitante acesso FULL_CONTROL (esse é o caso mesmo que a solicitação seja uma solicitação PutObjectInline ou PutObject para um objeto que já exista).

A seguir está uma solicitação que grava dados em um objeto, torna o objeto legível por administradores anônimos e dá ao usuário especificado direitos FULL_CONTROL sobre o bucket (a maioria dos desenvolvedores vai querer dar a si mesmo acesso FULL_CONTROL a seu próprio bucket).

Example

A seguir está uma solicitação que grava dados em um objeto e torna o objeto legível por administradores anônimos.

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
```

```

    <Grantee xsi:type="CanonicalUser">
      <ID>75cc57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
      <DisplayName>chriscustomer</DisplayName>
    </Grantee>
    <Permission>FULL_CONTROL</Permission>
  </Grant>
  <Grant>
    <Grantee xsi:type="Group">
      <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
    </Grantee>
    <Permission>READ</Permission>
  </Grant>
</AccessControlList>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2009-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>

```

Sample Response

```

<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>

```

A política de controle de acesso pode ser lida ou configurada para um bucket ou objeto existentes usando os métodos `GetBucketAccessControlPolicy`, `GetObjectAccessControlPolicy`, `SetBucketAccessControlPolicy` e `SetObjectAccessControlPolicy`. Para obter mais informações, consulte a explicação detalhada desses métodos.

Elementos comuns

Você pode incluir os seguintes elementos relacionados a autorização com qualquer solicitação SOAP:

- **AWSAccessKeyId**: O ID da chave de acesso da AWS do solicitante
- **Timestamp**: A hora atual do seu sistema
- **Signature**: A assinatura da solicitação

Respostas de erro de SOAP

 Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

Em SOAP, um resultado de erro será retornado ao cliente como uma falha de SOAP, com o código de resposta HTTP 500. Se você não receber uma falha de SOAP, sua solicitação terá sido bem-sucedida. O código de falha de SOAP do Amazon S3 é composto de um código de falha padrão de SOAP 1.1 ("servidor" ou "cliente") concatenado com o código de erro específico ao Amazon S3. Por exemplo: "Server.InternalError" ou "Client.NoSuchBucket". O elemento da sequência da falha de SOAP contém uma mensagem de erro genérica, legível pelo usuário em inglês. Finalmente, o elemento de detalhes da falha de SOAP contém informações diversas relevantes para o erro.

Por exemplo, se você tentar excluir o objeto "Fred", que não existe, o corpo da resposta de SOAP conterá uma falha de SOAP "NoSuchKey".

O exemplo a seguir mostra uma resposta de erro de SOAP.

```
<soapenv:Body>
  <soapenv:Fault>
    <Faultcode>soapenv:Client.NoSuchKey</Faultcode>
    <Faultstring>The specified key does not exist.</Faultstring>
    <Detail>
      <Key>Fred</Key>
    </Detail>
  </soapenv:Fault>
</soapenv:Body>
```

A tabela a seguir explica os elementos de resposta de erros de SOAP

Name (Nome)	Descrição
Detail	Contêiner para a chave envolvida no erro
	Tipo: Container

Name (Nome)	Descrição
	Antepassado: Body.Fault
Fault	Contêiner para informações de erro. Tipo: Container Antepassado: Body
Faultcode	O código de falha é uma string que identifica exclusivamente uma condição de erro. O objetivo desse código é ser lido e compreendido pelos programas que detectam e tratam erros por tipo. Para obter mais informações, consulte Lista de códigos de erro. Tipo: string Antepassado: Body.Fault
Faultstring	A string de falha contém uma descrição genérica da condição do erro em inglês. Ela é destinada ao público humano. Programas simples exibem a mensagem diretamente ao usuário final se encontrarem uma condição de erro que não conhecem ou não tratam. Programas sofisticados com tratamento de erro mais exaustivo e internacionalização própria são mais prováveis de ignorar a string de falha. Tipo: string Antepassado: Body.Fault
Key	Identifica a chave envolvida no erro Tipo: string Antepassado: Body.Fault

Apêndice: autenticação de solicitações (AWS Signature Version 2)

Important

Esta seção descreve como autenticar solicitações usando o AWS Signature Version 2.

O Signature versão 2 está sendo desativado (defasado), o Amazon S3 aceitará somente solicitações de API assinadas usando o Signature versão 4. Para obter mais informações, consulte [AWS Signature Version 2 desativado \(defasado\) para o Amazon S3](#)

O Signature Version 4 é compatível com todas as Regiões da AWS. Essa é a única versão compatível com novas regiões. Para obter mais informações, consulte [Autenticação de solicitações \(AWS Signature Version 4\)](#) na Referência de APIs do Amazon Simple Storage Service.

O Amazon S3 oferece a possibilidade de identificar qual versão da assinatura de API foi usada para assinar uma solicitação. É importante identificar se algum dos fluxos de trabalho estão utilizando as assinaturas do Signature versão 2 e atualizá-los para que usem o Signature versão 4 para evitar que seus negócios sejam impactados.

- Se você estiver usando os logs de eventos do CloudTrail (opção recomendada), consulte [Identificação de solicitações do Amazon S3 Signature Version 2 usando o CloudTrail](#) sobre como consultar e identificar essas solicitações.
- Se você estiver usando os logs de acesso ao servidor do Amazon S3, consulte [Uso dos logs de acesso ao servidor do Amazon S3 para identificar solicitações](#)

Tópicos

- [Autenticação de solicitações usando a API REST \(AWS Signature Version 2\)](#)
- [Assinatura e autenticação de solicitações REST \(AWS Signature Version 2\)](#)
- [Uploads baseados no navegador usando POST \(AWS Signature Version 2\)](#)

Autenticação de solicitações usando a API REST (AWS Signature Version 2)

Ao acessar o Amazon S3 usando REST, é necessário fornecer os seguintes itens na solicitação para que ela seja autenticada:

Elementos da solicitação

- ID da chave de acesso da AWS: cada solicitação deve conter o ID da chave de acesso da identidade que você estiver usando para enviar a solicitação.
- Assinatura: cada solicitação deve conter uma assinatura de solicitação válida. Caso contrário, a solicitação será rejeitada.

A assinatura de uma solicitação é calculada com a chave de acesso secreta, um segredo compartilhado conhecido apenas por você e pela AWS.

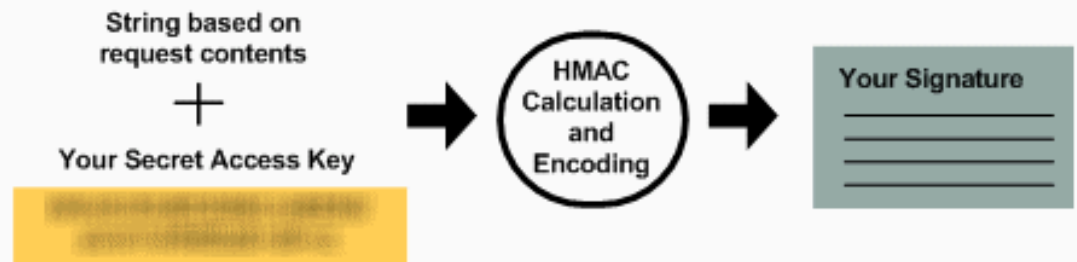
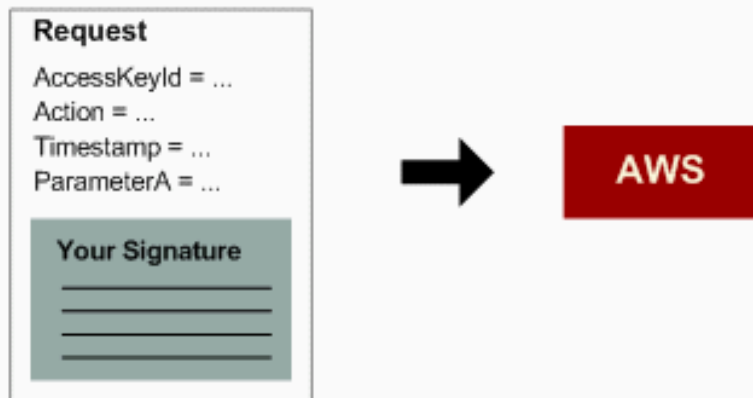
- Time stamp: cada solicitação deve conter a data e a hora de criação da solicitação, representada como uma string em UTC.
- Data: cada solicitação deve conter o time stamp da solicitação.

Dependendo da ação da API que você está usando, é possível fornecer uma data e hora de expiração para a solicitação em vez (ou além) do time stamp. Consulte o tópico de autenticação da respectiva ação para determinar o que é necessário.

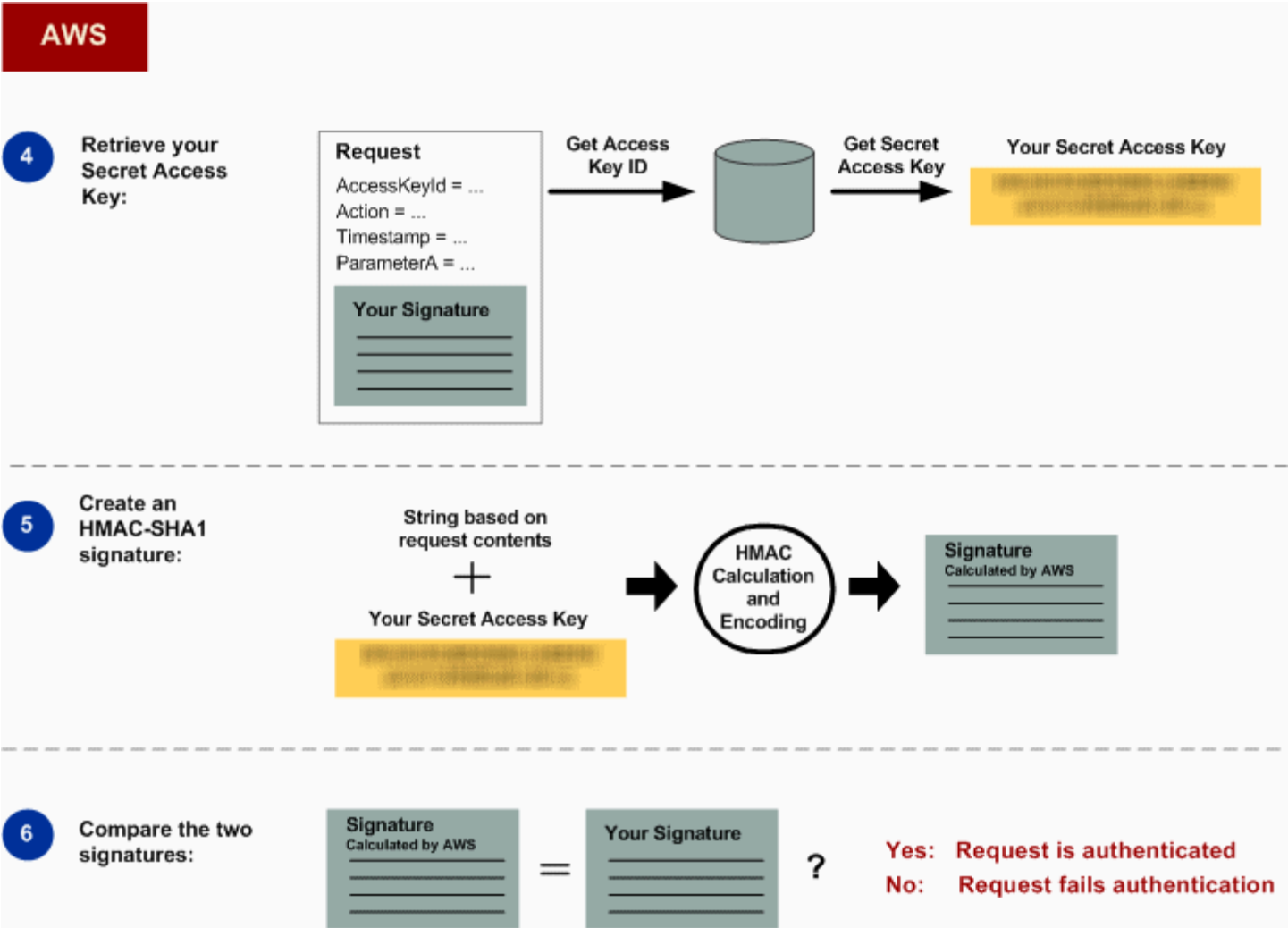
Veja a seguir as etapas gerais para autenticar solicitações ao Amazon S3. Pressupõe-se que você tem as credenciais de segurança necessárias, o ID de chave de acesso e a chave de acesso secreta.

You**1 Create a request:****Request**

AccessKeyId = ...
Action = ...
Timestamp = ...
ParameterA = ...

2 Create an HMAC-SHA1 signature:**3 Send the request and signature to AWS:**

- 1 Crie uma solicitação para a AWS.
- 2 Calcule a assinatura usando a chave de acesso secreta.
- 3 Envie a solicitação para o Amazon S3. Inclua o ID de chave de acesso e a assinatura na solicitação. O Amazon S3 executa as próximas três etapas.



Informações de autenticação detalhadas

Para obter informações detalhadas sobre a autenticação REST, consulte [Assinatura e autenticação de solicitações REST \(AWS Signature Version 2\)](#).

Assinatura e autenticação de solicitações REST (AWS Signature Version 2)

Tópicos

- [Uso de credenciais de segurança temporárias](#)
- [Cabeçalho de autenticação](#)
- [Canonização de solicitação para assinatura](#)
- [Criar o elemento CanonicalizedResource](#)
- [Criar o elemento CanonicalizedAmzHeaders](#)
- [Elementos StringToSign de cabeçalho HTTP posicionais versus nomeados](#)
- [Requisito de time stamp](#)
- [Exemplos de autenticação](#)
- [Problemas de assinatura de solicitação REST](#)
- [Alternativa de autenticação de solicitação por string de consulta](#)

Note

Este tópico explica solicitações de autenticação usando o Signature versão 2. O Amazon S3 agora é compatível com o mais recente Signature versão 4. Essa versão mais recente de assinatura é compatível com todas as regiões e qualquer nova região depois de 30 de janeiro de 2014 oferecerá suporte somente ao Signature versão 4. Para obter mais informações, acesse [Autenticação de solicitações \(AWS Signature versão 4\)](#) na Referência de APIs do Amazon Simple Storage Service.

Autenticação é o processo de provar sua identidade ao sistema. A identidade é um fator importante nas decisões de controle de acesso do Amazon S3. As solicitações são permitidas ou negadas em parte com base na identidade do solicitante. Por exemplo, o direito de criar buckets está reservado a desenvolvedores registrados e (por padrão) o direito de criar objetos em um bucket está reservado para o proprietário do bucket em questão. Como um desenvolvedor, você fará solicitações que

invocam esses privilégios e, portanto, precisará provar sua identidade ao sistema, autenticando suas solicitações. Esta seção explica como fazer isso.

 Note

O conteúdo nesta seção não se aplica a HTTP POST. Para obter mais informações, consulte [Uploads baseados no navegador usando POST \(AWS Signature Version 2\)](#).

A API REST do Amazon S3 usa um esquema HTTP personalizado com base em um HMAC de chave (código de autenticação de mensagem hash) para autenticação. Para autenticar uma solicitação, você primeiro concatena elementos selecionados da solicitação para formar uma string. Depois, você pode usar sua chave de acesso secreta da AWS para calcular o HMAC dessa string. Informalmente, chamamos desse processo de “assinar a solicitação” e chamamos o resultado do algoritmo do HMAC de assinatura, pois ele simula as propriedades de segurança de uma assinatura real. Finalmente, você adiciona esta assinatura como um parâmetro da solicitação usando a sintaxe descrita nesta seção.

Quando o sistema recebe uma solicitação autenticada, ele busca a chave de acesso secreta da AWS que você afirma ter e a usa da mesma forma para computar uma assinatura para a mensagem que recebeu. Então, ele compara a assinatura que calculou com a assinatura apresentada pelo solicitante. Se há correspondência entre as duas assinaturas, o sistema concluirá que o solicitante deve ter acesso à chave de acesso secreta da AWS e, portanto, age com a autoridade do principal para quem a chave foi emitida. Se as duas assinaturas não correspondem, a solicitação é abandonada e o sistema responde com uma mensagem de erro.

Example Solicitação REST autenticada do 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 credenciais de segurança temporárias

Se você assinar sua solicitação usando credenciais de segurança temporárias (consulte [Fazer solicitações](#)), você deverá incluir o token de segurança correspondente em sua solicitação, adicionando o cabeçalho `x-amz-security-token`.

Quando você obtém credenciais de segurança temporárias usando a API do AWS Security Token Service, a resposta inclui credenciais de segurança temporárias e um token de sessão. Forneça o valor do token de sessão no cabeçalho `x-amz-security-token` ao enviar solicitações ao Amazon S3. Para obter informações sobre a API do AWS Security Token Service fornecida pelo IAM, acesse [Ação](#) no Guia de referência da API do AWS Security Token Service.

Cabeçalho de autenticação

A API REST do Amazon S3 usa o cabeçalho padrão HTTP Authorization para passar informações de autenticação. (O nome do cabeçalho padrão é infeliz porque ele carrega informações de autenticação, não de autorização.) No esquema de autenticação do Amazon S3, o cabeçalho Autorização tem a seguinte forma:

```
Authorization: AWS AWSAccessKeyId:Signature
```

Um ID da chave de acesso da AWS e uma chave de acesso secreta da AWS são emitidos para os desenvolvedores quando eles se registram. Para autenticação de solicitação, o elemento `AWSAccessKeyId` identifica o ID de chave de acesso que foi usado para computar a assinatura e, indiretamente, o desenvolvedor que fez a solicitação.

O elemento `Signature` é o RFC 2104 HMAC-SHA1 dos elementos selecionados da solicitação e, portanto, a parte `Signature` do cabeçalho Autorização variará de uma solicitação para outra. Se a assinatura da solicitação calculada pelo sistema corresponder ao elemento `Signature` incluído na solicitação, o solicitante terá demonstrado a posse da chave de acesso secreta da AWS. A solicitação será então processada na identidade do desenvolvedor para quem a chave foi emitida e com a autoridade dele.

A seguir está uma pseudogramática que ilustra a criação do cabeçalho da solicitação Authorization. (No exemplo, `\n` significa o ponto do código Unicode U+000A, geralmente chamado de nova linha).

```

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>

```

O HMAC-SHA1 é um algoritmo definido pelo hash de chave [RFC 2104 para autenticação de mensagem](#). O algoritmo recebe como input duas strings de byte, uma chave e uma mensagem. Para a autenticação de solicitação do Amazon S3, use sua chave de acesso secreta da AWS (YourSecretAccessKey) como a chave e a codificação UTF-8 do elemento StringToSign como a mensagem. A saída de HMAC-SHA1 também é uma string de byte, chamada de resumo. O parâmetro de solicitação Signature é criado pela codificação Base64 desse resumo.

Canonização de solicitação para assinatura

Lembre-se de que quando o sistema recebe uma solicitação autenticada, ele compara a assinatura de solicitação computada com a assinatura fornecida na solicitação em StringToSign. Por esse motivo, é necessário computar a assinatura usando o mesmo método usado pelo Amazon S3. Nós chamamos o processo de colocar uma solicitação em um formulário estabelecido para assinatura de canonização.

Criar o elemento CanonicalizedResource

CanonicalizedResource representa o recurso do Amazon S3 visado pela solicitação. Crie-o para uma solicitação REST como se segue:

1. Inicie com uma string vazia ("").
2. Se a solicitação especificar um bucket usando o cabeçalho de host HTTP (estilo hosted virtual), adicione o nome do bucket precedido por uma "/" (por exemplo, "/bucketname"). Para

solicitações de estilo de caminho e solicitações que não seja endereçada a um bucket, não faça nada. Para obter mais informações sobre solicitações de estilo hospedadas virtualmente, consulte [Hospedagem virtual de buckets](#).

Para uma solicitação no estilo de hospedagem virtual "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg", o CanonicalizedResource é "/"awsexamplebucket1".

Para uma solicitação em estilo de caminho, "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg", o CanonicalizedResource é "".

3. Adicione a parte do caminho de um URI de solicitação HTTP decodificado, até a query string, mas sem incluí-la.

Para uma solicitação no estilo de hospedagem virtual "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg", o CanonicalizedResource é "/"awsexamplebucket1/photos/puppy.jpg".

Para uma solicitação no estilo de hospedagem virtual "https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg", o CanonicalizedResource é "/awsexamplebucket1/photos/puppy.jpg". Neste ponto, o CanonicalizedResource é o mesmo para a solicitação em estilo hosted virtual e em estilo de caminho.

Para uma solicitação que não seja endereçada a um bucket, como [GET Service](#), adicione "/".

4. Se a solicitação endereça um sub-recurso, como ?versioning, ?location, ?acl, ?lifecycle ou ?versionid, adicione o sub-recurso, seu valor, se houver um, e o ponto de interrogação. Observe que, em caso de vários sub-recursos, os sub-recursos devem ser classificados em ordem lexicográfica por nome de sub-recurso e ser separados por '&', por exemplo, ?acl&versionId=*value*.

Os sub-recursos que devem ser incluídos ao criar o elemento CanonicalizedResource são acl, lifecycle, location, logging, notification, partNumber, policy, requestPayment, uploadId, uploads, versionId, versioning, versions e website.

Se a solicitação especificar os parâmetros de query string que cancelam os valores de cabeçalho da resposta (consulte [Objeto GET](#)), adicione os parâmetros de query string e seus valores. Ao assinar, você não codifica esses valores; contudo, ao fazer a solicitação, você deve codificar esses valores de parâmetros. Os parâmetros de query string em uma solicitação GET incluem response-content-type, response-content-language, response-expires,

`response-cache-control`, `response-content-disposition` e `response-content-encoding`.

O parâmetro de query string `delete` deve ser incluído ao criar o `CanonicalizedResource` para uma solicitação de exclusão de vários objetos.

Os elementos do `CanonicalizedResource` que vêm da URI da solicitação HTTP devem ser assinados literalmente como aparecem na solicitação HTTP, incluindo metacaracteres de codificação de URL.

O `CanonicalizedResource` pode ser diferente da URI da solicitação HTTP. Em particular, se sua solicitação usa o cabeçalho HTTP `Host` para especificar um bucket, o bucket não aparece na URI da solicitação HTTP. Contudo, o `CanonicalizedResource` continua a incluir o bucket. Os parâmetros de query string podem também aparecer na URI da solicitação, mas não estão incluídos em `CanonicalizedResource`. Para obter mais informações, consulte [Hospedagem virtual de buckets](#).

Criar o elemento `CanonicalizedAmzHeaders`

Para criar a parte de `CanonicalizedAmzHeaders` de `StringToSign`, selecione todos os cabeçalhos de solicitações HTTP que comecem com `'x-amz-'` (usando uma comparação que não diferencie maiúsculas e minúsculas) e use o processo a seguir.

1. Converta cada nome de cabeçalho HTTP para minúsculas. Por exemplo, `'X-Amz-Date'` torna-se `'x-amz-date'`.
2. Classifique a coleção de cabeçalhos por ordem lexicográfica por nome de cabeçalho.
3. Combine campos do cabeçalho com o mesmo nome em um par “header-name:comma-separated-value-list” conforme descrito em RFC 2616, seção 4.2, sem qualquer espaço entre os valores. Por exemplo, os dois cabeçalhos de metadados `'x-amz-meta-username: fred'` e `'x-amz-meta-username: barney'` seriam combinados em único cabeçalho `'x-amz-meta-username: fred,barney'`.
4. “Desdobre” os cabeçalhos longos que abrangem várias linhas (como permitido em RFC 2616, seção 4.2) substituindo o espaço de dobramento (incluindo a nova linha) por um único espaço.
5. Remova todos os espaços ao redor dos dois pontos no cabeçalho. Por exemplo, o cabeçalho `'x-amz-meta-username: fred, barney'` se tornaria `'x-amz-meta-username:fred,barney'`.
6. Finalmente, adicione um caractere de nova linha (`U+000A`) para cada cabeçalho canonizado na lista resultante. Crie o elemento `CanonicalizedResource` concatenando todos os cabeçalhos dessa lista em uma única string.

Elementos StringToSign de cabeçalho HTTP posicionais versus nomeados

Os primeiros elementos de cabeçalho do `StringToSign` (`Content-Type`, `Date` e `Content-MD5`) são de natureza posicional. `StringToSign` não inclui os nomes desses cabeçalhos, somente seus valores da solicitação. Em contraste, os elementos `'x-amz-'` são nomeados. Os nomes de cabeçalho e os valores de cabeçalho aparecem em `StringToSign`.

Se um cabeçalho posicional chamado para a definição de `StringToSign` não estiver presente na sua solicitação (por exemplo, `Content-Type` ou `Content-MD5` são opcionais para solicitações `PUT` e sem sentido para solicitações `GET`), substitua a string vazia (`""`) para essa posição.

Requisito de time stamp

Um time stamp válido (usando o cabeçalho HTTP `Date` ou uma alternativa `x-amz-date`) é obrigatório para solicitações autenticadas. Além disso, o time stamp do cliente, incluído com uma solicitação autenticada, não deve exceder 15 minutos do tempo do sistema do Amazon S3 quando a solicitação é recebida. Caso contrário, haverá falha na solicitação com o código de erro `RequestTimeTooSkewed`. A intenção dessas restrições é limitar a possibilidade de que solicitações interceptadas possam ser reenviadas por um adversário. Para uma proteção mais forte contra espionagem, use o transporte HTTPS para solicitações autenticadas.

Note

A restrição de validação na data da solicitação se aplica somente a solicitações autenticadas que não usem a autenticação por query string. Para obter mais informações, consulte [Alternativa de autenticação de solicitação por string de consulta](#).

Algumas bibliotecas de clientes HTTP não expõem a capacidade para configurar o cabeçalho `Date` para uma solicitação. Se tiver problemas para incluir o valor do cabeçalho `"Data"` nos cabeçalhos canonizados, você pode configurar o time stamp para a solicitação usando um cabeçalho `'x-amz-date'`. O valor do cabeçalho `x-amz-date` deve estar em um dos formatos RFC 2616 (<http://www.ietf.org/rfc/rfc2616.txt>). Quando um cabeçalho `x-amz-date` está presente em uma solicitação, o sistema ignorará qualquer cabeçalho `Date` ao calcular a assinatura da solicitação. Portanto, se você incluir o cabeçalho `x-amz-date`, use a string vazia para o `Date` quando criar o `StringToSign`. Consulte a próxima seção para ver um exemplo.

Exemplos de autenticação

Os exemplos nesta seção usam as credenciais (não trabalho) na tabela a seguir.

Parâmetro	Valor
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecret AccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

No exemplo `StringToSigns`, o formato não é significativo e `\n` significa o ponto de código Unicode U+000A, chamado geralmente de nova linha. Além disso, os exemplos usam "+0000" para designar o fuso horário. Você pode usar "GMT" para designar o fuso horário, mas as assinaturas mostradas nos exemplos serão diferentes.

Objeto GET

Este exemplo obtém um objeto do bucket `awsexamplebucket1`.

Solicitação	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>

Observe que o `CanonicalizedResource` inclui o nome do bucket, mas a URI da solicitação HTTP não o inclui. (O bucket é especificado pelo cabeçalho de host.)

Note

O script Python a seguir calcula a assinatura precedente, usando os parâmetros fornecidos. É possível usar esse script para criar suas próprias assinaturas, substituindo as chaves e StringToSign conforme apropriado.

```
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()}")
```

PUT objeto

Este exemplo coloca um objeto no bucket awsexamplebucket1 bucket.

Solicitação	StringToSign
<pre>PUT /photos/puppy.jpg HTTP/1.1 Content-Type: image/jpeg Content-Length: 94328 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:15:45 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE:</pre>	<pre>PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Solicitação	StringToSign
<i>iqRzw+ileNPu1fhspnRs8n0jjIA=</i>	

Observe o cabeçalho Content-Type na solicitação e em StringToSign. Também observe que o Content-MD5 está em branco em StringToSign porque não está presente na solicitação.

Lista

Este exemplo lista o conteúdo do bucket `awsexamplebucket1`.

Solicitação	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>

Observe a barra final no CanonicalizedResource e a ausência de parâmetros de query string.

Fetch

Este exemplo busca o sub-recurso de política de controle de acesso para o bucket “`awsexamplebucket1`” bucket.

Solicitação	StringToSign
<pre>GET /?acl HTTP/1.1 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 19:44:46 +0000</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:44:46 +0000\n</pre>

Solicitação	StringToSign
<i>Authorization: AWS AKIAIOSFODNN7EXAMPLE:82ZHiFIjc+WbcwFKGUVeQspPn+0=</i>	/awsexamplebucket1/?acl

Observe como o parâmetro de query string do sub-recurso está incluído no CanonicalizedResource.

Excluir

Este exemplo exclui um objeto do bucket “awsexamplebucket1” usando o estilo de caminho e a alternativa de data.

Solicitação	StringToSign
<pre>DELETE /awsexamplebucket1/photos/puppy.jpg HTTP/1.1 User-Agent: dotnet Host: s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:20:27 +0000 x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE:XbyTlbQdu9Xw5o8P4iMwPktxQd8=</pre>	<pre>DELETE\n \n \n Tue, 27 Mar 2007 21:20:26 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Observe como usamos o método alternativo 'x-amz-date' de especificar a data (porque a biblioteca de cliente nos impediu de configurar a data, por exemplo). Nesse caso, o x-amz-date tem precedência sobre o cabeçalho Date. Portanto, entrada de data na assinatura deve conter o valor de cabeçalho x-amz-date.

Carregar

Este exemplo faz upload de um objeto para um bucket de estilo hosted virtual CNAME com metadados.

Solicitação	StringToSign
<pre>PUT /db-backup.dat.gz HTTP/1.1</pre>	<pre>PUT\n</pre>

Solicitação	StringToSign
<pre>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@example.com X-Amz-Meta-FileChecksum: 0x02661779 X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment; filename=database.dat Content-Encoding: gzip Content-Length: 5913339 Authorization: AWS AKIAIOSFODNN7EXAMPLE: jtBQa0Aq+DkULFI8qrpwIjGEx0E=</pre>	<pre>4gJE4saaMU4BqNR0kLY+lw==\n application/x-download\n Tue, 27 Mar 2007 21:06:08 +0000\n x-amz-acl:public-read\n x-amz-meta-checksumalgorithm:crc32\n x-amz-meta-filechecksum:0x02661779\n x-amz-meta-reviewedby:joe@example.com,jane@example.com\n /static.example.com/db-backup.dat.gz</pre>

Observe como os cabeçalhos 'x-amz-' são classificados, os espaços são removidos e os cabeçalhos são convertidos em minúsculas. Observe também que vários cabeçalhos com o mesmo nome foram unidos usando vírgulas para separar valores.

Observe como somente os cabeçalhos de entidade HTTP Content-Type e Content-MD5 aparecem em StringToSign. Os outros cabeçalhos de entidade Content-* não aparecem.

Mais uma vez, observe que o CanonicalizedResource inclui o nome do bucket, mas a URI da solicitação HTTP não o inclui. (O bucket é especificado pelo cabeçalho de host.)

Listar todos os meus buckets

Solicitação	StringToSign
<pre>GET / HTTP/1.1 Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:29:59 +0000</pre>	<pre>GET\n \n \n</pre>

Solicitação	StringToSign
<i>Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdERIC03wnaRNKh60qZehG9s=</i>	Wed, 28 Mar 2007 01:29:59 +0000\n /

Chaves Unicode

Solicitação	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

Os elementos em StringToSign que foram derivados da URI de solicitação são obtidos literalmente, incluindo a codificação do URL e a capitalização.

Problemas de assinatura de solicitação REST

Quando a autenticação de solicitação REST falha, o sistema responde à solicitação com um documento de erro em XML. As informações contidas neste documento de erro têm o objetivo de ajudar os desenvolvedores a diagnosticar o problema. Especificamente, o elemento StringToSign do documento de erro SignatureDoesNotMatch diz exatamente que canonização de solicitação o sistema está usando.

Alguns toolkits inserem silenciosamente os cabeçalhos que você não conhece antecipadamente, como a adição do cabeçalho Content-Type durante um PUT. Na maioria desses casos, o valor do cabeçalho inserido permanece constante, permitindo que você descubra os cabeçalhos que faltam, usando ferramentas como Ethereal ou o tcpmon.

Alternativa de autenticação de solicitação por string de consulta

Você pode autenticar determinados tipos de solicitações passando as informações necessárias como parâmetros de query string em vez de usar o cabeçalho HTTP Authorization. Isso é útil para habilitar o acesso de navegadores de terceiros a seus dados privados do Amazon S3 sem um proxy na solicitação. A ideia é criar uma solicitação "pré-assinada" e codificá-la como um URL que o navegador de um usuário final pode recuperar. Além disso, você pode limitar uma solicitação pré-assinada, especificando um tempo de expiração.

Para obter mais informações sobre como usar parâmetros de consulta para autenticar solicitações, consulte [Autenticação de solicitações: uso parâmetros de consulta \(AWS Signature Version 4\)](#) na Referência de APIs do Amazon Simple Storage Service. Para obter exemplos de uso de SDKs da AWS para gerar URLs pré-assinados, consulte [Compartilhamento de objetos com URLs pré-assinados](#).

Criar uma assinatura

Veja a seguir um exemplo de solicitação REST do Amazon S3 autenticada por query string.

```
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
```

O método de autenticação por query string não requer nenhum cabeçalho especial HTTP. Em vez disso, os elementos de autenticação exigidos são especificados como parâmetros de query string:

Nome de parâmetro de string de consulta	Valor de exemplo	Descrição
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE	Seu ID de chave de acesso da AWS. Especifica a chave de acesso secreta da AWS usada para assinar a solicitação e, indiretamente, a

Nome de parâmetro de string de consulta	Valor de exemplo	Descrição
		identidade do desenvolvedor que fez a solicitação.
Expires	1141889120	O tempo quando a assinatura vai expirar, especificado como o número de segundos desde o epoch (00:00:00 UTC em 1º de janeiro de 1970). Uma requisição recebida depois desse tempo (de acordo com o servidor) será rejeitada.
Signature	vjbyPxybdZaNmGa%2ByT272YEAiv4%3D	A codificação do URL da codificação Base64 do HMAC-SHA1 de StringToSign.

O método de autenticação de solicitação por query string difere ligeiramente do método comum, mas somente no formato do parâmetro da solicitação `Signature` e no elemento `StringToSign`. A seguir está a pseudogramática que ilustra o método de autenticação de solicitação por query string.

```
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` é o ID da chave de acesso secreta da AWS que a Amazon atribui a você ao se cadastrar para ser um desenvolvedor da Amazon Web Services. Observe como o `Signature` é codificado por URL para ser apropriado para a colocação na query string. Observe

também que no `StringToSign`, o elemento posicional `HTTP Date` foi substituído por `Expires`. O `CanonicalizedAmzHeaders` e o `CanonicalizedResource` são os mesmos.

 Note

No método de autenticação por query string, você não utiliza o cabeçalho `Date` nem o `x-amz-date request` para calcular o string para assinar.

Autenticação de solicitação por string de consulta

Solicitação	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>

Supomos que quando um navegador faz a solicitação `GET`, ele não fornece um cabeçalho `Content-MD5` ou `Content-Type`, nem define os cabeçalhos `x-amz-` e assim essas partes de `StringToSign` são deixadas em branco.

Usar codificação Base64

As assinaturas de solicitação HMAC devem ser codificadas em Base64. A codificação Base64 converte a assinatura em uma string simples ASCII que pode ser anexada à solicitação. Os caracteres que poderiam aparecer na string da assinatura como mais (+), barra (/) e igual (=) devem ser codificados se forem usados em uma URI. Por exemplo, se o código de autenticação inclui um sinal de mais (+), codifique-o como `%2B` na solicitação. Codifique uma barra como `%2F` e o sinal de igual como `%3D`.

Para obter exemplos de codificação Base64, consulte do Amazon 3 [Exemplos de autenticação](#).

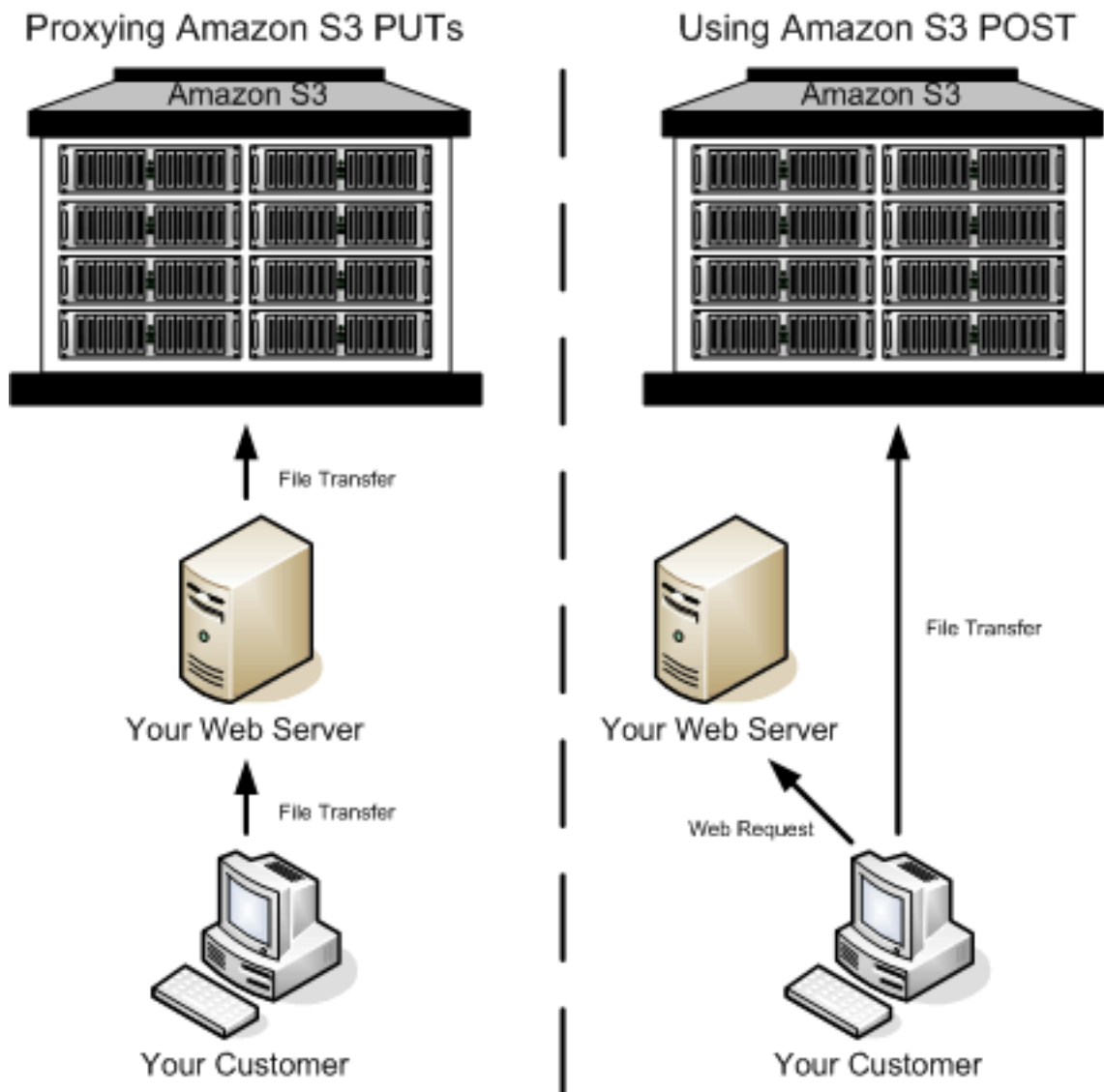
Uploads baseados no navegador usando POST (AWS Signature Version 2)

O Amazon S3 é compatível com POST, o que permite aos usuários fazer upload de conteúdo diretamente para o Amazon S3. POST foi projetada para simplificar uploads, reduzir a latência do upload e economizar seu dinheiro nas aplicações em que os usuários fazem upload de dados para armazenamento para o Amazon S3.

Note

A autenticação de solicitação discutida nesta seção é baseada no AWS Signature Version 2, um protocolo para autenticar solicitações de entrada da API para serviços da AWS. Agora, o Amazon S3 é compatível com o Signature Version 4, um protocolo para autenticação de solicitações de API de entrada para produtos da AWS, em todas as Regiões da AWS. Neste momento, as Regiões da AWS criadas antes de 30 de janeiro de 2014 continuarão oferecendo suporte ao protocolo anterior, Signature Version 2. Todas as regiões novas a partir de 30 de janeiro de 2014 oferecerão suporte apenas ao Signature versão 4. Portanto, todas as solicitações para essas regiões devem ser feitas com o Signature versão 4. Para obter mais informações, consulte [\(Autenticação de solicitações em uploads baseados em navegador usando POST \(AWS Signature Version 4\)\)](#) na Referência de APIs do Amazon Simple Storage Service.

A figura a seguir mostra um upload usando POST do Amazon S3.



Fazer upload usando POST

- 1 O usuário abre um navegador e acessa sua página da web.
- 2 A página da web tem um formulário HTTP que contém toda as informações necessárias para que o usuário faça upload de conteúdo no Amazon S3.
- 3 O usuário faz uploads do conteúdo diretamente para o Amazon S3.

Note

Não há suporte para autenticação por query string para POST.

Formulários HTML (AWS Signature Version 2)

Tópicos

- [Codificação do formulário HTML](#)
- [Declaração do formulário HTML](#)
- [Campos do formulário HTML](#)
- [Criação de política](#)
- [Criar uma assinatura](#)
- [Redirecionamento](#)

Ao se comunicar com o Amazon S3, você normalmente usa as APIs REST ou SOAP para executar as operações “put”, “get”, “delete”, entre outras. Com POST, os usuários fazem upload dos dados diretamente para o Amazon S3 por meios dos navegadores, que não são capazes de processar a API SOAP ou criar uma solicitação PUT de REST.

Note

As APIs SOAP para o Amazon S3 não estão disponíveis para novos clientes e chegam ao fim da vida útil (EOL) em 31 de agosto de 2025. Recomendamos usar a API REST ou os AWS SDKs.

Para permitir que os usuários façam upload de conteúdo para o Amazon S3 usando os navegadores, use formulários HTML. Os formulários HTML são formados por uma declaração de formulário e campos de formulário. A declaração de formulário contém informações de alto nível sobre a solicitação. Os campos de formulário contêm informações detalhadas sobre a solicitação, bem como a política usada para autenticá-la e garantir que ela satisfaça as condições especificadas.

Note

Os dados e os limites do formulário (excluindo o conteúdo do arquivo) não podem exceder 20 KB.

Esta seção explica como usar formulários HTML.

Codificação do formulário HTML

O formulário e a política devem ser codificados em UTF-8. Aplique a codificação UTF-8 no formulário especificando isso no cabeçalho HTML ou como um cabeçalho de solicitação.

Note

A declaração de formulário HTML não aceita parâmetros de autenticação por query string.

A seguir um exemplo de codificação UTF-8 no cabeçalho HTML:

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

A seguir um exemplo de codificação UTF-8 em um cabeçalho de solicitação:

```
Content-Type: text/html; charset=UTF-8
```

Declaração do formulário HTML

A declaração de formulário tem três componentes: a ação, o método e o tipo de compartimento. Se qualquer um desses valores for definido de maneira incorreta, a solicitação falhará.

A ação especifica o URL que processa a solicitação, que deve ser definido como o URL do bucket. Por exemplo, se o nome do seu bucket for `awsexamplebucket1` e a região for Oeste dos EUA (Norte da Califórnia), o URL será `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/`.

Note

O nome chave é especificado em um campo do formulário.

O método deve ser POST.

O tipo de compartimento (enctype) deve ser especificado e definido como multipart/form-data para uploads de arquivos e de áreas de texto. Para obter mais informações, acesse [RFC 1867](#).

Example

O exemplo a seguir é uma declaração de formulário para o bucket "awsexamplebucket1".

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

Campos do formulário HTML

A tabela a seguir descreve os campos que podem ser usados em um formulário HTML.

 Note

A variável `${filename}` é substituída automaticamente pelo nome do arquivo fornecido pelo usuário e é reconhecida por todos os campos do formulário. Se o navegador ou o cliente fornece um caminho completo ou parcial para o arquivo, apenas o texto que vem depois da última barra (/) ou barra invertida (\) será usado. Por exemplo, "C:\Program Files \directory1\file.txt" será interpretado como "file.txt". Se nenhum arquivo ou nome de arquivo for fornecido, a variável será substituída por uma string vazia.

Nome do campo	Descrição	Obrigatório
AWSAccessKeyId	O ID da chave de acesso da AWS do proprietário do bucket que concede acesso a um usuário anônimo para uma solicitação que satisfaz o conjunto de restrições na política. Este campo é necessário se a solicitação inclui um documento de política.	Condicional
acl	Uma lista de controle de acesso (ACL) do Amazon S3. Se uma lista de controle de	Não

Nome do campo	Descrição	Obrigatório
	acesso inválida for especificada, um erro será gerado. Tipo: string Padrão: 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	Cabeçalhos específicos para REST. Para obter mais informações, consulte Objeto PUT .	Não
key	O nome da chave carregada. Para usar o nome de arquivo fornecido pelo usuário, use a variável <code>\${filename}</code>. Por exemplo, se o usuário Betty carrega o arquivo lolcatz.jpg e você especifica <code>/user/betty/\${filename}</code>, o arquivo é armazenado como <code>/user/betty/lolcatz.jpg</code>. Para obter mais informações, consulte Trabalhar com metadados de objetos.	Sim
policy	Política de segurança que descreve o que é permitido na solicitação. As solicitações sem uma política de segurança são consideradas anônimas e terão sucesso apenas em buckets com gravação pública.	Não

Nome do campo	Descrição	Obrigatório
<code>success_action_redirect, redirect</code>	O URL para o qual o cliente é redirecionado após um upload bem-sucedido. O Amazon S3 anexa os valores de bucket, chave e etag como parâmetros de query string ao URL. Se <code>success_action_redirect</code> não for especificado, o Amazon S3 retornará o tipo de documento vazio especificado no campo <code>success_action_status</code>. Se o Amazon S3 não for capaz de interpretar o URL, o campo será ignorado. Se houver falha no upload, o Amazon S3 exibirá um erro e não redirecionará o usuário para um URL. Para obter mais informações, consulte Redirecionamento.  Note O nome do campo de redirecionamento está obsoleto e o suporte para ele será removido no futuro.	Não

Nome do campo	Descrição	Obrigatório
success_action_status	O código de status retornado ao cliente após o upload bem-sucedido se success_action_redirect não for especificado. Os valores válidos são 200, 201 ou 204 (padrão). Se o valor estiver definido como 200 ou 204, o Amazon S3 retornará um documento vazio com um código de status 200 ou 204. Se o valor estiver definido como 201, o Amazon S3 retornará um documento XML com um código de status 201. Para obter informações sobre o conteúdo do documento XML, consulte Objeto POST. Se o valor não estiver definido ou for um valor inválido, o Amazon S3 retornará um documento vazio com um código de status 204.	Não

 **Note**

Algumas versões do Adobe Flash Player não lidam muito bem com respostas HTTP com um corpo vazio. Para oferecer suporte a uploads por meio do Adobe Flash, recomendamos definir success_action_status como 201.

Nome do campo	Descrição	Obrigatório
signature	A assinatura HMAC criada com a chave de acesso secreta correspondente ao <code>AWSSecretAccessKey</code> fornecido. Este campo é necessário se um documento de política estiver incluso na solicitação. Para ter mais informações, consulte Gerenciamento de identidade e acesso para o Amazon S3.	Condicional
x-amz-security-token	Um token de segurança usado por credenciais de sessão Se a solicitação estiver usando o Amazon DevPay, serão necessários dois campos do formulário <code>x-amz-security-token</code> : um para o token de produto e outro para o token de usuário. Se a solicitação estiver usando credenciais de sessão, será necessário um formulário <code>x-amz-security-token</code> . Para obter mais informações, consulte Credenciais de segurança temporárias no Manual do usuário do IAM.	Não
Outros nomes de campos com o prefixo x-amz-meta-	Metadados especificados pelo usuário. O Amazon S3 não valida ou usa esses dados. Para obter mais informações, consulte Objeto PUT.	Não

Nome do campo	Descrição	Obrigatório
file	Conteúdo de arquivo ou texto. O arquivo ou conteúdo deve ser o último campo no formulário. Todos os campos abaixo deles serão ignorados. Não carregue mais de um arquivo por vez.	Sim

Criação de política

Tópicos

- [Expiração](#)
- [Condições](#)
- [Correspondência de condição](#)
- [Caracteres de escape](#)

A política é um documento JSON codificado em Base64 e UTF-8 que especifica as condições que a solicitação deve satisfazer, sendo usado para autenticar o conteúdo. Dependendo de como os documentos de política forem elaborados, eles podem ser usados por upload, por usuário, para todos os uploads ou de acordo com outros formatos que atendam as suas necessidades.

Note

Embora o documento de política seja opcional, o recomendamos fortemente em vez de tornar um bucket aberto ao público para gravação.

Veja a seguir um exemplo de um documento de política:

```
{ "expiration": "2007-12-01T12:00:00.000Z",  
  
  "conditions": [  
  
    {"acl": "public-read" },
```

```
{ "bucket": "awsexamplebucket1" },  
  
  [ "starts-with", "$key", "user/eric/" ],  
  
]  
  
}
```

O documento de política contém a expiração e as condições.

Expiração

O elemento expiração especifica a data de expiração da política no formato de data UTC ISO 8601. Por exemplo, "2007-12-01T12:00:00.000Z" especifica que a política não tem mais validade depois da meia-noite UTC do dia 1º de dezembro de 2007. A expiração é necessária em uma política.

Condições

As condições no documento de política validam o conteúdo do objeto carregado. Cada campo especificado no formulário (exceto `AWSSignatureVersion`, assinatura, arquivo, política e nomes de campos com o prefixo `x-ignore-`) deve estar incluso na lista de condições.

Note

Caso existam vários campos com o mesmo nome, os valores devem ser separados por vírgulas. Por exemplo, se existem dois campos chamados "x-amz-meta-tag", o primeiro tem o valor "Ninja" e o segundo tem o valor "Stallman", o documento de política seria definido como `Ninja,Stallman`.

Todas as variáveis dentro do formulário são expandidas antes da validação da política. Portanto, qualquer correspondência de condição deve ser realizada nos campos expandidos. Por exemplo, se o campo chave for definido como `user/betty/${filename}`, a política deve ser `[ "starts-with", "$key", "user/betty/" ]`. Não insira `[ "starts-with", "$key", "user/betty/${filename}" ]`. Para obter mais informações, consulte [Correspondência de condição](#).

A tabela a seguir descreve as condições do documento de política.

Nome do elemento	Descrição
acl	Especifica as condições que a ACL deve satisfazer. Oferece suporte à correspondência exata e a <code>starts-with</code>.
content-length-range	Especifica os tamanhos mínimo e máximo permitidos para o conteúdo carregado. Oferece suporte à correspondência por intervalo.
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	Cabeçalhos específicos para REST. Oferece suporte à correspondência exata e a <code>starts-with</code>.
chave	O nome da chave carregada. Oferece suporte à correspondência exata e a <code>starts-with</code>.
success_action_redirect, redirect	O URL para o qual o cliente é redirecionado após um upload bem-sucedido. Oferece suporte à correspondência exata e a <code>starts-with</code>.
success_action_status	O código de status retornado ao cliente após o upload bem-sucedido se <code>success_action_redirect</code> não for especificado. Oferece suporte à correspondência exata.
x-amz-security-token	Token de segurança do Amazon DevPay.

Nome do elemento	Descrição
	Cada solicitação que usa o Amazon DevPay requer dois campos do formulário <code>x-amz-security-token</code> : um para o token de produto e outro para o token de usuário. Consequentemente, os valores devem ser separados por vírgulas. Por exemplo, se o token de usuário é <code>eW91dHVIZQ==</code> e o token de produto for <code>b0hnNVNKWVJIQTA=</code> , defina a entrada da política para: <code>{ "x-amz-security-token": "eW91dHVIZQ==,b0hnNVNKWVJIQTA=" }</code> .
Outros nomes de campos com o prefixo <code>x-amz-meta-</code>	Metadados especificados pelo usuário. Oferece suporte à correspondência exata e a <code>starts-with</code> .

Note

Se o seu toolkit traz campos adicionais (por exemplo, o Flash adiciona nome do arquivo), é necessário adicioná-los ao documento de política. Se essa funcionalidade puder ser controlada, adicione o prefixo `x-ignore-` ao campo para que o Amazon S3 ignore o recurso e para que futuras versões não sejam afetadas.

Correspondência de condição

A tabela a seguir descreve os tipos de correspondência de condição. Embora seja necessário especificar uma condição para cada campo especificado no formulário, é possível criar critérios de correspondência mais complexos especificando várias condições para um campo.

Condição	Descrição
Correspon dências exatas	Correspondências exatas verificam se os campos correspondem a valores específicos. Este exemplo indica que a ACL deve ser definida como pública para leitura:

Condição	Descrição
	<pre>{"acl": "public-read" }</pre> Este exemplo é uma forma alternativa para indicar que a ACL deve ser definida como pública para leitura: <pre>["eq", "\$acl", "public-read"]</pre>
Inicia com	Se o valor deve iniciar com um certo valor, use starts-with. Este exemplo indica que a chave deve iniciar com user/betty: <pre>["starts-with", "\$key", "user/betty/"]</pre>
Corresponder qualquer conteúdo	Para configurar a política para permitir qualquer conteúdo em um campo, use starts-with com um valor vazio. Este exemplo permite qualquer success_action_redirect: <pre>["starts-with", "\$success_action_redirect", ""]</pre>
Especificar intervalos	Para os campos que aceitam intervalos, separe os limites superior e inferior do intervalo com uma vírgula. Este exemplo permite um tamanho de arquivo entre 1 e 10 megabytes: <pre>["content-length-range", 1048579, 10485760]</pre>

Caracteres de escape

A tabela a seguir descreve os caracteres de escape dentro de um documento de política.

Sequência de escape	Descrição
\\	Barra invertida
\\$	Sinal de dólar
\b	Apagar
\f	Feed do formulário
\n	Nova linha
\r	Carriage return
\t	Guia horizontal
\v	Guia vertical
\u ~~xxxx~~	Todos os caracteres do Unicode

Criar uma assinatura

Etapas	Descrição
1	Codifique a política usando UTF-8.
2	Codifique os bytes UTF-8 usando Base64.
3	Assine a política com a chave de acesso secreta usando HMAC SHA-1.

Etapa	Descrição
4	Codifique a assinatura SHA-1 usando Base64.

Para obter informações gerais sobre autenticação, consulte [Gerenciamento de identidade e acesso para o Amazon S3](#).

Redirecionamento

Esta seção descreve como manipular redirecionamentos.

Redirecionamento geral

Após a conclusão da solicitação POST, o usuário é redirecionado para o local especificado no campo `success_action_redirect`. Se o Amazon S3 não for capaz de interpretar o URL, o campo `success_action_redirect` será ignorado.

Se `success_action_redirect` não for especificado, o Amazon S3 retornará o tipo de documento vazio especificado no campo `success_action_status`.

Se houver falha na solicitação POST, o Amazon S3 exibirá um erro e não fará o redirecionamento.

Redirecionamento pré-upload

Se o bucket foi criado usando `<CreateBucketConfiguration>`, os usuários finais poderão exigir um redirecionamento. Se isso ocorrer, alguns navegadores podem manipular o redirecionamento de maneira incorreta. Isso é relativamente raro, mas é mais provável que ocorra logo após a criação do bucket.

Exemplos de uploads (AWS Signature Version 2)

Tópicos

- [Upload de arquivo](#)
- [Upload de área de texto](#)

Note

A autenticação de solicitação discutida nesta seção é baseada no AWS Signature Version 2, um protocolo para autenticar solicitações de entrada da API para serviços da AWS.

Agora, o Amazon S3 é compatível com o Signature Version 4, um protocolo para autenticação de solicitações de API de entrada para produtos da AWS, em todas as Regiões da AWS. Neste momento, as Regiões da AWS criadas antes de 30 de janeiro de 2014 continuarão oferecendo suporte ao protocolo anterior, Signature Version 2. Todas as regiões novas a partir de 30 de janeiro de 2014 oferecerão suporte apenas ao Signature versão 4. Portanto, todas as solicitações para essas regiões devem ser feitas com o Signature versão 4. Para obter mais informações, consulte [Exemplos: upload baseado em navegador usando HTTP POST \(usando o AWS Signature Version 4\)](#) na Referência de APIs do Amazon Simple Storage Service.

Upload de arquivo

Este exemplo mostra o processo completo para criação de uma política e um formulário que pode ser usado para carregar um arquivo anexo.

Criação de política e formulário

A política a seguir é compatível com uploads para o bucket `awsexamplebucket1` do Amazon S3.

```
{ "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 requer o seguinte:

- O upload deve ocorrer antes das 12:00 UTC em 1º de dezembro de 2007.
- O upload do conteúdo deve ser feito para o bucket `awsexamplebucket1`.
- A chave deve começar com `"user/eric/"`.
- A ACL está definida para leitura pública.

- O `success_action_redirect` está definido como `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html`.
- O objeto é um arquivo de imagem.
- A tag `x-amz-meta-uuid` deve ser definida como `14365123651274`.
- A tag `x-amz-meta-tag` pode conter qualquer valor.

Veja a seguir uma versão codificada em Base64 dessa política.

```
eyJhZlZxhwaXJhdGlvbiI6IClYMDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgcyJidWN
```

Crie uma assinatura usando suas credenciais. Por exemplo `0RavWzkygo6QX9caELEqKi9kDbU=` é a assinatura para o documento de política anterior.

O formulário a seguir comporta uma solicitação POST para o bucket `amzn-s3-demo-bucket` que usa essa 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>
```

Exemplo de solicitação

Essa solicitação pressupõe que a imagem carregada tem 117.108 bytes; os dados da imagem não estão incluídos.

```
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"

eyJhZXBwaXJhdGlvbiI6ICIyMDA3LTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgZidWN
--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--
```

Exemplo de resposta

```
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
```

Upload de área de texto

Tópicos

- [Criação de política e formulário](#)
- [Exemplo de solicitação](#)
- [Exemplo de resposta](#)

O exemplo a seguir mostra o processo completo para criação de uma política e um formulário para carregar uma área de texto. Fazer upload de uma área de texto é útil para o envio de conteúdo criado pelo usuário, como postagens de um blog.

Criação de política e formulário

A política a seguir é compatível com uploads de área de texto para o bucket `awsexamplebucket1` do Amazon S3.

```
{ "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 requer o seguinte:

- O upload deve ocorrer antes das 12:00 GMT em 1º de dezembro de 2007.
- O upload do conteúdo deve ser feito para o bucket `awsexamplebucket1`.
- A chave deve começar com `"user/eric/"`.
- A ACL está definida para leitura pública.
- O `success_action_redirect` está definido como `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html`.
- O objeto é texto HTML.
- A tag `x-amz-meta-uuid` deve ser definida como `14365123651274`.
- A tag `x-amz-meta-tag` pode conter qualquer valor.

Veja a seguir uma versão codificada em Base64 dessa política.

```
eyAiZXhwaXJhdGlvbiI6IClYMDA3LTEyLTAxVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXR0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiRrZXkiLCaidXNlciLAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzc19hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2pC5zMy5hbWF6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSI6ICJ0ZXh0L2h0CAGIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXRoIiwgIiR4LWFtei1tIsICIiXQogIF0KfQo=
```

Crie uma assinatura usando suas credenciais. Por exemplo, qA7FWXKq6VvU681I9KdveT1cWgF= é a assinatura para o documento de política anterior.

O formulário a seguir comporta uma solicitação POST para o bucket amzn-s3-demo-bucket que usa essa 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>
```

Exemplo de solicitação

Essa solicitação pressupõe que a imagem carregada tem 117.108 bytes; os dados da imagem não estão incluídos.

```
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"
eyJhZXBwaXJhdGlvbiI6IClYMDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgZjIdWm
--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--
```

Exemplo de resposta

```
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 com Adobe Flash (AWS Signature Version 2)

Esta seção descreve como usar o POST com o Adobe Flash.

Segurança do Adobe Flash Player

Por padrão, o modelo de segurança do Adobe Flash Player proíbe que os Adobe Flash Players realizem conexões de rede com servidores fora do domínio que serve o arquivo SWF.

Para substituir o padrão, é necessário carregar um arquivo `crossdomain.xml` de leitura pública no bucket que aceitará uploads POST. Veja a seguir um exemplo de arquivo `crossdomain.xml`.

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy SYSTEM
"http://www.macromedia.com/xml/dtds/cross-domain-policy.dtd">
<cross-domain-policy>
<allow-access-from domain="*" secure="false" />
</cross-domain-policy>
```

Note

Para obter mais informações sobre o modelo de segurança do Adobe Flash, acesse o site da Adobe.

Adicionar o arquivo `crossdomain.xml` ao bucket permite que qualquer Adobe Flash Player se conecte ao arquivo `crossdomain.xml` no bucket. No entanto, isso não concede acesso ao bucket do Amazon S3 em si.

Considerações do Adobe Flash

A API `FileReference` no Adobe Flash adiciona o campo de formulário `Filename` à solicitação POST. Ao compilar aplicações do Adobe Flash que fazem upload para o Amazon S3 usando a ação da API `FileReference`, inclua a seguinte condição na política:

```
['starts-with', '$Filename', '']
```

Algumas versões do Adobe Flash Player não lidam muito bem com respostas HTTP que têm um corpo vazio. Para configurar a POST para retornar uma resposta sem corpo vazio, defina `success_action_status` para 201. O Amazon S3 retornará um documento XML com um código de status 201. Para obter informações sobre o conteúdo do documento XML, consulte [Objeto POST](#). Para obter informações campos de formulário, consulte [Campos do formulário HTML](#).

Apêndice: APIs de configuração do ciclo de vida (obsoletas)

A configuração do ciclo de vida do bucket foi atualizada para oferecer suporte a filtros baseados em tags de objetos. Ou seja, agora você pode especificar uma regra que determina o prefixo do nome da chave, uma ou mais tags de objetos ou ambos para selecionar um subconjunto de objetos ao qual a regra se aplica. As APIs foram atualizadas conforme necessário. Os tópicos a seguir descrevem a versão anterior das operações de ciclo de vida do bucket PUT e GET para fins de compatibilidade com versões anteriores.

Tópicos

- [PUT Bucket lifecycle \(obsoleto\)](#)
- [GET Bucket lifecycle \(obsoleto\)](#)

PUT Bucket lifecycle (obsoleto)

Descrição

Important

Para obter uma versão atualizada dessa API, consulte [PutBucketLifecycleConfiguration](#). Essa versão está desatualizada. As configurações de ciclo de vida existentes continuarão funcionando. Para novas configurações de ciclo de vida, use a API atualizada.

Cria uma nova configuração do ciclo de vida para o bucket ou substitui uma configuração do ciclo de vida existente. Para obter informações sobre configuração de ciclo de vida, consulte [Gerenciamento do ciclo de vida de objetos](#) no Guia do usuário do Amazon Simple Storage Service.

Permissões

Por padrão, todos os recursos do Amazon S3, incluindo buckets, objetos e sub-recursos relacionados (por exemplo, configuração de ciclo de vida e de site), são privados. Somente o proprietário do recurso, a Conta da AWS que o criou, pode acessá-lo. O proprietário do recurso pode conceder permissões de acesso a outros criando uma política de acesso. Para essa operação, os usuários devem obter a permissão `s3:PutLifecycleConfiguration`.

Também é possível negar explicitamente as permissões. A negação explícita também substitui todas as outras permissões. Se você quiser impedir que usuários ou contas removam ou excluam objetos do seu bucket, negue a eles permissões para as seguintes ações:

- `s3:DeleteObject`
- `s3:DeleteObjectVersion`
- `s3:PutLifecycleConfiguration`

Para obter mais informações sobre permissões, consulte [Gerenciamento de permissões de acesso a seus recursos do Amazon S3](#) no Guia do desenvolvedor do Amazon Simple Storage Service.

Solicitações

Sintaxe

```
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 obter detalhes sobre strings de autorização, consulte [Autenticação de solicitações \(AWS Signature Version 4\)](#).

Parâmetros da solicitação

Essa implementação da operação não usa parâmetros de solicitação.

Cabeçalhos de solicitação

Name (Nome)	Descrição	Obrigatório
Content-MD5	O resumo MD5 de 128 bits com codificação base64 dos dados. Você deve usar esse cabeçalho como uma verificação de integridade da mensagem para verificar se o corpo da solicitação não foi corrompido em trânsito. Para obter mais informações, consulte a RFC 1864. Tipo: string Padrão: nenhum	Sim

Corpo da Solicitação

Na solicitação, você especifica a configuração de ciclo de vida no corpo da solicitação. Essa configuração é especificada como XML. Veja a seguir um exemplo de configuração de ciclo de vida. Ela especifica uma regra. O `Prefix` na regra identifica os objetos aos quais a regra se aplica. A

regra também especifica duas ações (Transition e Expiration). Cada ação especifica um cronograma em que o Amazon S3 deve realizar a ação. O Status indica se a regra está habilitada ou desabilitada.

```
<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>
```

Se o estado do seu bucket for de versionamento habilitado ou suspenso, você pode ter várias versões do mesmo objeto: uma versão atual e nenhuma ou mais versões desatualizadas. A configuração de ciclo de vida a seguir define as ações (NoncurrentVersionTransition, NoncurrentVersionExpiration) que são específicas para versões de objetos não atuais.

```
<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>
```

Você pode usar a API de multipart upload para fazer upload de objetos grandes em partes. Para obter mais informações sobre uploads fragmentados, consulte [Visão geral do upload fragmentado](#)

no Guia do desenvolvedor do Amazon Simple Storage Service. Com a configuração de ciclo de vida, você pode instruir o Amazon S3 a cancelar uploads fragmentados incompletos — identificados pelo prefixo do nome da chave especificado na regra — caso não sejam concluídos dentro de um número especificado de dias. Quando o Amazon S3 cancela um upload fragmentado, ele exclui todas as partes associadas ao upload. Isso garante que você não tenha uploads fragmentados incompletos com partes armazenadas no Amazon S3 e, portanto, você não precisa pagar por custos de armazenamento para elas. Veja a seguir um exemplo de configuração de ciclo de vida que especifica uma regra com a ação `AbortIncompleteMultipartUpload`. Essa ação pede que o Amazon S3 cancele uploads fragmentados incompletos sete dias após a inicialização.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>SomeKeyPrefix</Prefix>
    <Status>rule-status</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

A tabela a seguir descreve os elementos XML na configuração de ciclo de vida.

Name (Nome)	Descrição	Obrigatório
AbortIncompleteMultipartUpload	Contêiner para especificar quando um upload fragmentado incompleto se torna elegível para uma operação de interrupção. Secundário: DaysAfterInitiation Tipo: Container Antepassado: Rule	Sim, se nenhuma outra ação for especificada para a regra
Date	Data em que deseja que o Amazon S3 realize a ação. Para obter mais informações, consulte Regras de ciclo de vida: com base em uma	Sim, se Days e ExpiredOb

Name (Nome)	Descrição	Obrigatório
	data específica no Guia do usuário do Amazon Simple Storage Service. O valor de data deve estar em conformidade com o formato ISO 8601. A hora é sempre meia-noite (UTC). Tipo: string Antepassado: <code>Expiration</code> ou <code>Transition</code>	<code>jectDeleteMarker</code> estiverem ausentes
Days	Especifica o número de dias após a criação de objeto quando a ação de regra específica entra em vigor. Tipo: inteiro não negativo quando usado com <code>Transition</code>, inteiro positivo quando usado com <code>Expiration</code> Antepassado: <code>Expiration</code>, <code>Transition</code>	Sim, se <code>Date</code> e <code>ExpiredObjectDeleteMarker</code> estiverem ausentes
DaysAfterInitiation	Especifica o número de dias quando o upload fragmentado deve ser concluído depois de iniciado. Se não for concluído no número especificado de dias, ele se tornará elegível para uma operação de interrupção e o Amazon S3 cancelará o upload fragmentado incompleto. Tipo: número inteiro positivo Antepassado: <code>AbortIncompleteMultipartUpload</code>	Sim, se uma tag principal for especificada

Name (Nome)	Descrição	Obrigatório
Expiration	Essa ação especifica um período na vida útil de um objeto em que o Amazon S3 deve realizar a ação de expiração apropriada. A ação realizada pelo Amazon S3 depende de se o bucket está habilitado para versionamento. • Se o versionamento nunca tiver sido habilitado no bucket, o Amazon S3 excluirá permanentemente a única cópia do objeto. • Se o bucket tiver o versionamento habilitado (ou se o versionamento estiver suspenso), a ação se aplicará apenas à versão atual do objeto. Um bucket com versionamento habilitado pode ter várias versões do mesmo objeto: uma versão atual e nenhuma ou mais versões desatualizadas. Em vez de excluir a versão atual, o Amazon S3 a transforma em uma versão desatualizada, adicionando um marcador de exclusão como a nova versão atual.	Sim, se nenhuma outra ação estiver presente na Rule.

 **Important**

Se o estado de um bucket estiver com versionamento suspenso, o Amazon S3 criará um marcador de exclusão com ID de versão null. Se você tiver uma versão com ID null, o Amazon S3 substituirá essa versão.

Name (Nome)	Descrição	Obrigatório
 Note Para definir a expiração para objetos desatualizados, use a ação <code>NoncurrentVersionExpiration</code> . Tipo: Container Filhos: Days ou Date Antepassado: Rule		
ID	Identificador exclusivo da regra. O valor não pode ser maior que 255 caracteres. Tipo: string Antepassado: Rule	Não
LifecycleConfiguration	Contêiner para regras de ciclo de vida. Você pode adicionar até 1.000 regras. Tipo: Container Filhos: Rule Ancestral: Nenhum	Sim

Name (Nome)	Descrição	Obrigatório
ExpiredObjectDeleteMarker	Em um bucket com versões (versionamento habilitado ou suspenso no bucket), você pode adicionar esse elemento na configuração de ciclo de vida para fazer com que o Amazon S3 exclua os marcadores de exclusão de objetos expirados. Por exemplo, consulte Exemplo 8: remoção de marcadores de exclusão de objetos expirados no Guia do usuário do Amazon Simple Storage Service. Não o adicione a um bucket sem versão, pois esse tipo de bucket não pode incluir marcadores de exclusão. Tipo: string Valores válidos: true false (o valor false é permitido, mas não está funcionando, o que significa que o Amazon S3 não vai realizar nenhuma ação) Antepassado: Expiration	Sim, se Date e Days estiverem ausentes

Name (Nome)	Descrição	Obrigatório
NoncurrentDays	Especifica o número de dias que um objeto precisa estar desatualizado antes do Amazon S3 poder executar a ação associada. Para obter informações sobre os cálculos de dias desatualizados, consulte Como o Amazon S3 calcula quando um objeto se torna desatualizado no Guia do desenvolvedor do Amazon Simple Storage Service. Tipo: inteiro não negativo quando usado com NoncurrentVersionTransition , inteiro positivo quando usado com NoncurrentVersionExpiration Antepassado: NoncurrentVersionExpiration ou NoncurrentVersionTransition	Sim
NoncurrentVersionExpiration	Especifica quando expiram as versões de objetos não atuais. Após a expiração, o Amazon S3 exclui permanentemente as versões de objetos não atuais. Defina essa ação de configuração de ciclo de vida em um bucket que tem versionamento habilitado (ou suspenso) para pedir que o Amazon S3 exclua versões de objetos desatualizadas em um período específico na vida útil do objeto. Tipo: Container Filhos: NoncurrentDays Antepassado: Rule	Sim, se nenhuma outra ação estiver presente na Rule

Name (Nome)	Descrição	Obrigatório
NoncurrentVersionTransition	Contêiner para a regra de transição que descreve quando objetos desatualizados fazem a transição para STANDARD_IA , ONEZONE_IA ou a classe de armazenamento. Se o bucket estiver habilitado para versionamento (ou se o versionamento estiver suspenso) , você pode definir essa ação para pedir que o Amazon S3 realize a transição de versões de objetos desatualizadas em um período específico na vida útil do objeto. Tipo: Container Filhos: NoncurrentDays e StorageClass Antepassado: Rule	Sim, se nenhuma outra ação estiver presente na Rule
Prefix	Prefixo de chaves de objeto que identifica um ou mais objetos aos quais essa regra se aplica. Tipo: string Antepassado: Rule	Sim
Rule	Contêiner para uma regra de ciclo de vida. Uma configuração de ciclo de vida pode conter até 1.000 regras. Tipo: Container Antepassado: LifecycleConfiguration	Sim

Name (Nome)	Descrição	Obrigatório
Status	Se habilitado, o Amazon S3 vai executar a regra conforme programado. Se estiver desabilitado, o Amazon S3 vai ignorar a regra. Tipo: string Antepassado: Rule Valores válidos: Enabled, Disabled	Sim
StorageClass	Especifica a classe de armazenamento do Amazon S3 para a qual o objeto fará a transição. Tipo: string Antepassado: Transition e NoncurrentVersionTransition Valores válidos: STANDARD_IA ONEZONE_IA GLACIER	Sim Esse elemento só será necessário se você especificar um ou os dois antepassados.

Name (Nome)	Descrição	Obrigatório
Transition	Essa ação especifica um período na vida útil dos objetos em que o Amazon S3 deve fazer a transição deles para <code>STANDARD_IA</code> , <code>ONEZONE_IA</code> ou a classe de armazenamento. Quando essa ação está em vigor, o que o Amazon S3 faz depende se o bucket está habilitado para versionamento. • Se o versionamento nunca tiver sido habilitado no bucket, o Amazon S3 fará a transição da única cópia do objeto para a classe de armazenamento especificada. • Se o bucket estiver habilitado para versionamento (ou o versionamento estiver suspenso), o Amazon S3 fará a transição somente das versões atuais dos objetos identificados na regra.  Note Um bucket com versionamento habilitado pode ter várias versões de um objeto. Essa ação não tem efeito em versões de objetos desatualizadas. Para fazer a transição de objetos desatualizados, use a ação <code>NoncurrentVersionTransition</code> .	Sim, se nenhuma outra ação estiver presente na Rule

Name (Nome)	Descrição	Obrigatório
	Tipo: Container	
	Filhos: Days ou Date e StorageClass	
	Antepassado: Rule	

Respostas

Cabeçalhos de resposta

Essa implementação da operação usa somente os cabeçalhos de resposta que são comuns à maioria das respostas. Para obter mais informações, consulte [Cabeçalhos de resposta comuns](#).

Elementos de resposta

Essa implementação da operação não retorna elementos de resposta.

Erros especiais

Essa implementação da operação não retorna erros especiais. Para obter informações gerais sobre erros do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Exemplos

Exemplo 1: adicionar configuração de ciclo de vida a um bucket que não está habilitado para versionamento

A configuração de ciclo de vida a seguir especifica duas regras, cada uma com uma ação.

- A ação Transition faz com que o Amazon S3 realize a transição de objetos com o prefixo “documents/” para a classe de armazenamento 30 dias após a criação.
- A ação Expiration faz com que o Amazon S3 exclua objetos com o prefixo “logs/” 365 dias após a criação.

```
<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>

```

Veja a seguir um exemplo de solicitação PUT `/?lifecycle` que adiciona a configuração de ciclo de vida anterior ao 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: q6yJDlIkBaGGfb3QLY69A==
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>

```

```
</Expiration>
</Rule>
</LifecycleConfiguration>
```

Veja a seguir uma resposta de exemplo.

```
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
```

Exemplo 2: adicionar a configuração de ciclo de vida a um bucket com versionamento

A configuração de ciclo de vida a seguir especifica duas regras, cada uma com uma ação para o Amazon S3 realizar. Você especifica essas ações quando o bucket está habilitado para versionamento ou o versionamento está suspenso:

- A ação `NoncurrentVersionExpiration` instrui o Amazon S3 a expirar as versões desatualizadas de objetos com o prefixo “logs/” 100 dias após os objetos se tornarem desatualizados.
- A ação `NoncurrentVersionTransition` instrui o Amazon S3 a fazer a transição de versões desatualizadas de objetos com o prefixo “documents/” para a classe de armazenamento 30 dias após se tornarem desatualizadas.

```
<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>

```

Veja a seguir um exemplo de solicitação PUT `/?lifecycle` que adiciona a configuração de ciclo de vida anterior ao 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>

```

Veja a seguir uma resposta de exemplo.

```

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
```

Exemplos adicionais

Para obter mais exemplos de transição de objetos para classes de armazenamento, como STANDARD_IA ou ONEZONE_IA, consulte [Exemplos de configuração de ciclo de vida](#).

Recursos relacionados

- [GetBucketLifecycleConfiguration](#)
- [POST restauração de objeto](#)
- Por padrão, o proprietário do recurso — nesse caso, o proprietário do bucket, que é a Conta da AWS que criou o bucket — pode realizar qualquer uma das operações. O proprietário do recurso também pode conceder permissão a outras pessoas para realizar a operação. Para obter mais informações, consulte os tópicos a seguir no Guia do usuário do Amazon Simple Storage Service:
 - [Especificação de permissões em uma política](#)
 - [Managing Access Permissions to Your Amazon S3 Resources \(Gerenciar permissões de acesso aos recursos do Amazon S](#)

GET Bucket lifecycle (obsoleto)

Descrição

Important

Para obter uma versão atualizada dessa API, consulte [GetBucketLifecycleConfiguration](#). Se você tiver configurado um ciclo de vida de bucket usando o elemento `<filter>`, deverá ver uma versão atualizada desse tópico. Esse tópico é fornecido para compatibilidade com versões anteriores.

Retorna as informações da configuração de `lifecycle` definida no bucket. Para obter informações sobre configuração de ciclo de vida, consulte [Gerenciamento do ciclo de vida de objetos](#) no Guia do usuário do Amazon Simple Storage Service.

Para usar essa operação, você deve ter a permissão para executar a ação `s3:GetLifecycleConfiguration`. Por padrão, o proprietário do bucket tem essa permissão. Ele pode conceder essa permissão a outros. Para obter mais informações sobre permissões, consulte [Gerenciamento de permissões de acesso a seus recursos do Amazon S3](#) no Guia do desenvolvedor do Amazon Simple Storage Service.

Solicitações

Sintaxe

```
GET /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Autenticação de solicitações \(AWS Signature Version 4\))
```

Parâmetros da solicitação

Essa implementação da operação não usa parâmetros de solicitação.

Cabeçalhos de solicitação

Essa implementação da operação usa somente os cabeçalhos de solicitação que são comuns a todas as operações. Para obter mais informações, consulte [Cabeçalhos de solicitação comuns](#).

Elementos da solicitação

Essa implementação da operação não usa elementos de solicitação.

Respostas

Cabeçalhos de resposta

Essa implementação da operação usa somente os cabeçalhos de resposta que são comuns à maioria das respostas. Para obter mais informações, consulte [Cabeçalhos de resposta comuns](#).

Elementos de resposta

Essa implementação de GET retorna os seguintes elementos de resposta.

Name (Nome)	Descrição	Obrigatório
AbortIncompleteMultipartUpload	Contêiner para especificar quando um upload fragmentado incompleto se torna elegível para uma operação de interrupção. Secundário: DaysAfterInitiation Tipo: Container Antepassado: Rule	Sim, se nenhuma outra ação for especificada para a regra
Date	Data em que deseja que o Amazon S3 realize a ação. Para obter mais informações, consulte Regras de ciclo de vida: com base em uma data específica no Guia do usuário do Amazon Simple Storage Service. O valor de data deve estar em conformidade com o formato ISO 8601. A hora é sempre meia-noite (UTC). Tipo: string	Sim, se Days e ExpiredObjectDeleteMarker estiverem ausentes

Name (Nome)	Descrição	Obrigatório
Days	Antepassado: <code>Expiration</code> ou <code>Transition</code> Especifica o número de dias após a criação de objeto quando a ação de regra específica entra em vigor. O tempo de elegibilidade do objeto é calculado como a hora de criação + o número de dias com a hora resultante arredondada para a meia-noite UTC do dia seguinte. Tipo: inteiro não negativo quando usado com <code>Transition</code>, inteiro positivo quando usado com <code>Expiration</code>. Antepassado: <code>Transition</code> ou <code>Expiration</code>	Sim, se <code>Date</code> e <code>ExpiredObjectDeleteMarker</code> estiverem ausentes
DaysAfterInitiation	Especifica o número de dias quando o upload fragmentado deve ser concluído depois de iniciado. Se não for concluído no número especificado de dias, ele se tornará elegível para uma operação de interrupção e o Amazon S3 cancelará o upload fragmentado incompleto. Tipo: número inteiro positivo Antepassado: <code>AbortIncompleteMultipartUpload</code>	Sim, se <code>Date</code> estiver ausente

Name (Nome)	Descrição	Obrigatório
Expiration	Essa ação especifica um período na vida útil do objeto em que o Amazon S3 deve realizar a ação de expiração apropriada. A ação de expiração ocorre somente em objetos elegíveis de acordo com o período especificado no elemento filho Date ou Days. A ação realizada pelo Amazon S3 depende de se o bucket está habilitado para versionamento. • Se o versionamento nunca tiver sido habilitado no bucket, o Amazon S3 excluirá permanentemente a única cópia do objeto. • Caso contrário, se o bucket tiver o versionamento habilitado (ou o versionamento estiver suspenso), a ação se aplicará somente à versão atual do objeto. Os buckets com versionamento habilitado ou suspenso podem ter várias versões do mesmo objeto: uma versão atual e nenhuma ou mais versões desatualizadas. Em vez de excluir a versão atual, o Amazon S3 a transforma em uma versão desatualizada, adicionando um marcador de exclusão como a nova versão atual.	Sim, se a tag principal for especificada

 **Important**

Se algum bucket tiver o versionamento suspenso, o Amazon S3 criará um marcador de exclusão com ID de versão null. Se você tiver uma

Name (Nome)	Descrição	Obrigatório
	versão com ID null, o Amazon S3 substituirá essa versão.  Note Para definir a expiração para objetos desatualizados, você deve usar a ação <code>NoncurrentVersionExpiration</code>. Tipo: Container Filhos: Days ou Date Antepassado: Rule	
ID	Identificador exclusivo da regra. O valor não pode ser maior que 255 caracteres. Tipo: string Antepassado: Rule	Não
LifecycleConfiguration	Contêiner para regras de ciclo de vida. Você pode adicionar até 1.000 regras. Tipo: Container Filhos: Rule Ancestral: Nenhum	Sim

Name (Nome)	Descrição	Obrigatório
ExpiredObjectDeleteMarker	Em um bucket com versões (versionamento ativado ou suspenso no bucket), esse elemento indica se o Amazon S3 excluirá os marcadores de exclusão de objetos expirados no bucket. Por exemplo, acesse Exemplo 8: especificar a ação de expiração para remover marcadores de exclusão de objetos expirados no Guia do usuário do Amazon Simple Storage Service. Tipo: string Valores válidos: true false (o valor false é permitido, mas não está funcionando; o Amazon S3 não vai agir se o valor for false) Antepassado: Expiration	Sim, se Date e Days estiverem ausentes
NoncurrentDays	Especifica o número de dias que um objeto precisa estar desatualizado antes do Amazon S3 poder executar a ação associada. Para obter informações sobre o cálculo de dias com desatualização, consulte Regras de ciclo de vida com base no número de dias no Guia do usuário do Amazon Simple Storage Service. Tipo: inteiro não negativo quando usado com NoncurrentVersionTransition , inteiro positivo quando usado com NoncurrentVersionExpiration Antepassado: NoncurrentVersionExpiration ou NoncurrentVersionTransition	Sim, somente se o antepassa do estiver presente

Name (Nome)	Descrição	Obrigatório
NoncurrentVersionExpiration	Especifica quando expiram as versões de objetos não atuais. Após a expiração, o Amazon S3 exclui permanentemente as versões de objetos não atuais. Defina essa ação de configuração de ciclo de vida em um bucket que tenha versionamento habilitado (ou suspenso), para solicitar que o Amazon S3 exclua versões de objetos desatualizadas em um período específico na vida útil do objeto. Tipo: Container Filhos: NoncurrentDays Antepassado: Rule	Sim, se nenhuma outra ação estiver presente na Rule

Name (Nome)	Descrição	Obrigatório
NoncurrentVersionTransition	Contêiner para a regra de transição que descreve quando objetos desatualizados fazem a transição para STANDARD_IA , ONEZONE_IA ou a classe de armazenamento. Se o bucket estiver habilitado para versionamento (ou o versionamento estiver suspenso) , você poderá definir essa ação para solicitar que o Amazon S3 realize a transição de versões de objetos desatualizadas para a classe de armazenamento em um período específico na vida útil do objeto. Tipo: Container Filhos: NoncurrentDays e StorageClass Antepassado: Rule	Sim, se nenhuma outra ação estiver presente na Rule
Prefix	Prefixo de chave de objeto que identifica um ou mais objetos aos quais a regra se aplica. Tipo: string Antepassado: Rule	Sim
Rule	Contêiner para uma regra de ciclo de vida. Tipo: Container Antepassado: LifecycleConfiguration	Sim

Name (Nome)	Descrição	Obrigatório
Status	Se habilitado, o Amazon S3 vai executar a regra conforme programado. Se desabilitado, o Amazon S3 vai ignorar a regra. Tipo: string Antepassado: Rule Valores válidos: Enabled ou Disabled	Sim
StorageClass	Especifica a classe de armazenamento do Amazon S3 para a qual o objeto fará a transição. Tipo: string Antepassado: Transition e NoncurrentVersionTransition Valores válidos: STANDARD_IA ONEZONE_IA	Sim

Name (Nome)	Descrição	Obrigatório
Transition	Essa ação especifica um período na vida útil dos objetos em que o Amazon S3 deve fazer a transição deles para STANDARD_IA , ONEZONE_IA ou a classe de armazenamento. Quando essa ação está em vigor, o que o Amazon S3 faz depende se o bucket está habilitado para versionamento. • Se o versionamento nunca tiver sido habilitado no bucket, o Amazon S3 fará a transição da única cópia do objeto para a classe de armazenamento especificada. • Quando o bucket está habilitado para versionamento (ou o versionamento está suspenso), o Amazon S3 faz a transição somente das versões atuais dos objetos identificados na regra.  Note Um bucket com versionamento habilitado ou suspenso pode conter várias versões de um objeto. Essa ação não tem efeito em versões de objeto desatualizadas. Para fazer a transição de objetos desatualizados, use a ação NoncurrentVersionTransition .	Sim, se nenhuma outra ação estiver presente na Rule

Name (Nome)	Descrição	Obrigatório
	Tipo: Container	
	Filhos: Days ou Date e StorageClass	
	Antepassado: Rule	

Erros especiais

Código de erro	Descrição	Código de status HTTP	Prefixo do código de falha SOAP
NoSuchLifecycleConfiguration	A configuração de ciclo de vida não existe.	404 Not Found (404 Não encontrado)	Cliente

Para obter informações gerais sobre erros do Amazon S3 e uma lista de códigos de erro, consulte [Respostas de erro](#).

Exemplos

Exemplo 1: recuperar um sub-recurso do ciclo de vida

Esse exemplo é uma solicitação GET para recuperar o sub-recurso `lifecycle` do bucket especificado e um exemplo de resposta com a configuração de ciclo de vida retornada.

Exemplo de solicitação

```
GET /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Thu, 15 Nov 2012 00:17:21 GMT
Authorization: signatureValue
```

Resposta da amostra

```
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>
```

Recursos relacionados

- [PutBucketLifecycleConfiguration](#)
- [DeleteBucketLifecycle](#)

Histórico do documento