



Entwicklerhandbuch

Amazon Simple Storage Service



API-Version 2006-03-01

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

Amazon Simple Storage Service: Entwicklerhandbuch

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

Die Marken und Handelsmarken von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, die geeignet ist, Kunden irrezuführen oder Amazon in irgendeiner Weise herabzusetzen oder zu diskreditieren. Alle anderen Handelsmarken, die nicht Eigentum von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise zu Amazon gehören oder nicht, mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist Amazon S3?	1
Empfehlungen für API-Aufrufe	2
Direkte REST-API-Aufrufe tätigen	2
Berechtigungen	3
Entwickeln mit Amazon S3	4
Senden von Anforderungen	4
Über Zugriffsschlüssel	5
Anforderungsendpunkte	7
Senden von Anforderungen über IPv6	7
Anfragen stellen mit dem AWS SDKs	18
Senden von Anforderungen unter Verwendung der REST-API	59
Mit dem AWS CLI	72
Erfahren Sie mehr über die AWS CLI	73
Entwickeln mit AWS SDKs	73
SDK-Programmierschnittstellen	74
Angabe der Signaturversion bei der Anforderungsauthentifizierung	75
Holen Sie sich Amazon S3-Anforderungs-IDs für Support	84
Verwendung der AWS CLI um Anfrage-IDs zu erhalten	85
Verwenden von Windows PowerShell zum Abrufen von Anforderungs-IDs	85
Verwenden AWS CloudTrail Datenereignisse zum Abrufen von Anforderungs-IDs	85
Abrufen der Anforderungs-IDs mithilfe der S3-Server-Zugriffsprotokollierung	86
Abrufen der Anforderungs-IDs mithilfe von HTTP	86
Abrufen der Anforderungs-IDs mithilfe eines Webbrowsers	86
Verwendung der AWS SDKs zum Abrufen von Anforderungs-IDs	87
Unterstützte S3-API-Operationen auf Objektebene für S3-Tabellen	92
Codebeispiele	97
Amazon S3	103
Grundlagen	109
Szenarien	806
Serverless-Beispiele	1264
Amazon S3 Control	1276
Grundlagen	1276
S3-Verzeichnis-Buckets	1357
Grundlagen	1358

Szenarien	1484
Anfragen authentifizieren (AWS Signatur (Version 4))	1487
Authentifizierungsmethoden	1488
Einführung in das Signieren von Anfragen	1489
Verwenden eines Autorisierungs-Headers	1490
-Übersicht	1491
Signaturberechnung: Nutzlast in einem einzigen Block übertragen	1496
Signaturberechnung: Nutzlast in mehreren Blöcken übertragen	1516
Signaturberechnung: Einschließlich nachstehender Header	1529
Verwenden von Abfrageparametern	1536
Berechnung einer Signatur	1539
Ein Beispiel	1544
Beispiel 2	1546
Beispiele: Signaturberechnungen	1546
Beispiele für die Berechnung von Signaturen mit Java	1547
Beispiele für die Berechnung von Signaturen mit C#	1549
Authentifizieren von HTTP-POST-Anfragen	1550
Berechnung einer Signatur	1552
Amazon S3 Signature Version 4: Spezifische Richtlinien Schlüssel für die Authentifizierung	1554
Beispiele für Bucket-Richtlinien, bei denen Bedingungsschlüssel im Zusammenhang mit Signature Version 4 verwendet werden	1557
Browser-Based Uploads mit POST	1561
POST Object	1562
Description	1562
Versionsverwaltung	1563
Anforderungen	1564
Beispiele	1583
Verwandte Ressourcen	1584
POST Object-Wiederherstellung	1585
Description	1585
Archive mit Select Requests abfragen	1585
Archive wiederherstellen	1587
Anforderungen	1589
Antworten	1605
Beispiele	1606
Weitere Infos	1609

Browser-Based Uploads mit HTTP POST	1609
Eine Signatur berechnen	1611
Erstellen von HTML-Formularen	1612
HTML-Formulardeklaration	1613
HTML-Formularfelder	1614
POST-Richtlinie	1621
Ablauf	1622
Übereinstimmung von Bedingungen	1622
Bedingungen	1624
Escape-Zeichen	1629
Beispiel für einen POST-Upload	1630
Hochladen einer Datei auf Amazon S3 mithilfe von HTTP POST	1630
Browser-based lädt mit Amplify hoch AWS	1633
Verwendung der AWS JavaScript Bibliothek erweitern, um Dateien auf Amazon S3 hochzuladen	1633
Weitere Infos	1634
Allgemeine Anforderungsheader	1635
Allgemeine Antwortheader	1640
Fehlermeldungen	1645
REST-Fehlerantworten	1646
Liste der Fehlercodes	1648
Liste der SELECT-Fehlercodes für den Objekteinhalt	1678
Liste der Fehlercodes Replication-related	1692
Liste der Tagging-related Fehlercodes	1695
Liste der Amazon S3-Fehlercodes im Zusammenhang mit Anmerkungen	1696
Liste der Amazon S3-Fehlercodes auf Outposts	1700
Liste der Amazon S3 Storage Lens-Fehlercodes	1702
Liste der Amazon S3 Object Lambda-Fehlercodes	1713
Liste der asynchronen Amazon S3-Fehlercodes	1718
Liste der Amazon S3 Access Grants-Fehlercodes	1720
Liste der FSx-related Amazon-Fehlercodes	1723
Liste der Amazon S3-Tabellen-Fehlercodes	1724
Bewährte Methoden für Amazon-S3-Fehler	1725
Versuchen Sie es erneut InternalErrors	1726
Optimieren Sie die Anwendung für wiederholte Fehler SlowDown	1726
Isolieren von Fehlern	1726

AWS Glossar	1728
Ressourcen	1729
Anhang	1731
Anhang: Antwort SelectObjectContent	1732
Description	1732
Antworten	1732
Verwandte Ressourcen	1742
Anhang: OPTIONS-Objekt	1744
Description	1744
Anforderungen	1744
Antworten	1745
Beispiele	1747
Verwandte Ressourcen	1747
Anhang: SOAP-API	1748
Operationen im Dienst (SOAP-API)	1748
Operationen mit Buckets (SOAP-API)	1750
Operationen an Objekten (SOAP-API)	1764
Authentifizieren von SOAP-Anforderungen	1789
Festlegen der Zugriffsrichtlinie mit SOAP	1791
Allgemeine Elemente	1793
Antworten auf SOAP-Fehler	1793
Anhang: Anfragen authentifizieren (AWS Signatur (Version 2))	1795
Authentifizieren von Anfragen mithilfe der REST-API (AWS Signatur (Version 2))	1796
Signieren und Authentifizieren von REST-Anfragen (AWS Signatur (Version 2))	1799
Browser-based lädt mit POST hoch (AWS Signaturversion 2)	1815
Anhang: APIs für die Lifecycle-Konfiguration (veraltet)	1839
PUT-Bucket-Lebenszyklus (veraltet)	1841
BUCKET-Lebenszyklus ABRUFEN (veraltet)	1860
Dokumentverlauf	1872
.....	mdccclxxiii

Was ist Amazon S3?

Amazon Simple Storage Service (Amazon S3) ist ein Objektspeicherservice, der branchenführende Skalierbarkeit, Datenverfügbarkeit, Sicherheit und Leistung bietet. Kunden aller Größen und Branchen können Amazon S3 für die Speicherung und den Schutz beliebiger Datenmengen für eine Reihe von Anwendungsfällen verwenden, wie Data Lakes, Websites, mobile Anwendungen, Backup und Wiederherstellung, Archivierung, Unternehmensanwendungen, IoT-Geräte und Big-Data-Analysen. Amazon S3 bietet Verwaltungsfunktionen, mit denen Sie den Zugriff auf Ihre Daten optimieren, organisieren und konfigurieren können, um Ihre spezifischen geschäftlichen, organisatorischen und Compliance-Anforderungen zu erfüllen.

Dieses Handbuch bietet entwicklerorientierte Informationen zur Verwendung der Amazon S3-REST-API zum Erstellen von Anwendungen, die Daten in der Cloud speichern und abrufen.

Sie können einen beliebigen Toolkit einsetzen, der HTTP unterstützt, um die REST-API verwenden zu können. Sie können sogar einen Browser verwenden, um Objekte zu laden, wenn diese anonym lesbar sind.

Die REST-API verwendet die HTTP-Standardheader und -Statuscodes, sodass sich Standard-Browser und -Toolkits wie erwartet verhalten. In einigen Bereichen haben wir HTTP um Funktionen erweitert (z. B. haben wir Header hinzugefügt, um die Zugriffskontrolle zu unterstützen). In diesen Fällen haben wir die neuen Funktionen so hinzugefügt, dass sie dem Stil der standardmäßigen HTTP-Nutzung entsprechen.

Note

Die Unterstützung für SOAP über HTTP ist veraltet, aber immer noch über HTTPS verfügbar. Neue Amazon S3-Funktionen werden für SOAP jedoch nicht unterstützt. Wir empfehlen, dass Sie entweder diese REST-API oder die AWS SDKs unter dem folgenden Link verwenden:

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

Dieses Entwicklerhandbuch beinhaltet:

- [Allgemeine Anforderungsheader](#) und [Allgemeine Antwortheader](#)
- [Fehlermeldungen](#)
- [Browser-Based Uploads mit POST \(AWS Signatur \(Version 4\)\)](#)

 Important

Lesen Sie die folgenden Informationen zu Authentifizierung und Zugriffskontrolle, bevor Sie zu bestimmten API-Themen übergehen.

Anfragen an Amazon S3 können authentifiziert oder anonym sein. Für den authentifizierten Zugriff sind Anmeldeinformationen erforderlich, mit denen Sie Ihre Anfragen authentifizieren AWS können.

Empfehlungen für API-Aufrufe

REST-API-Aufrufe direkt aus Ihrem Code heraus zu tätigen, kann umständlich sein. Sie müssen den erforderlichen Code schreiben, um eine gültige Signatur zur Authentifizierung Ihrer Anfragen zu berechnen. Wir empfehlen stattdessen die folgenden Alternativen:

- Verwenden Sie die [AWS SDKs](#), um Ihre Anfragen zu senden.

Sehen Sie sich auch den [Beispielcode und die Bibliotheken](#) an.

Wenn Sie die SDKs verwenden, müssen Sie keinen Code schreiben, um eine Signatur für die Anforderungsauthentifizierung zu berechnen, da die SDK-Clients Ihre Anfragen mithilfe der von Ihnen bereitgestellten Zugriffsschlüssel authentifizieren. Sofern Sie keinen guten Grund haben, dies nicht zu tun, sollten Sie immer die AWS SDKs verwenden.

- Verwenden Sie die AWS CLI, um Amazon S3-API-Aufrufe zu tätigen. Informationen zur Einrichtung der AWS CLI und Beispiele für Amazon S3-Befehle finden Sie in den folgenden Themen:

[Richten Sie die AWS CLI](#) im Amazon Simple Storage Service-Benutzerhandbuch ein.

[Verwenden von Amazon S3 mit der AWS Befehlszeilenschnittstelle](#) im AWS Command Line Interface Benutzerhandbuch.

Direkte REST-API-Aufrufe tätigen

 Note

Der PUT-Anforderungs-Header ist auf eine Größe von 8 KB begrenzt. Innerhalb des PUT-Anforderungsheaders sind die systemdefinierten Metadaten auf eine Größe von 2 KB

begrenzt. Die Größe systemdefinierter Metadaten wird gemessen, indem die Summe der Anzahl der Byte in der US-ASCII Codierung jedes Schlüssels und Werts berechnet wird.

Wenn Sie Ihre eigenen REST-API-Aufrufe tätigen möchten, anstatt eine der oben genannten Alternativen zu verwenden, sollten Sie einige Dinge beachten.

- Um direkte REST-API-Aufrufe von Ihrem Code aus zu tätigen, erstellen Sie eine Signatur mit gültigen Anmeldeinformationen und fügen Sie die Signatur in Ihre Anfrage ein. Informationen zu verschiedenen Authentifizierungsmethoden und Signaturberechnungen finden Sie unter [Anfragen authentifizieren \(AWS Signatur \(Version 4\)\)](#).
- Die REST-API verwendet Standard-HTTP-Header und Statuscodes, sodass Standardbrowser und Toolkits erwartungsgemäß funktionieren. In einigen Bereichen haben wir HTTP um Funktionen erweitert (zum Beispiel haben wir Header hinzugefügt, um die Zugriffskontrolle zu unterstützen). In diesen Fällen haben wir die neuen Funktionen so hinzugefügt, dass sie dem Stil der standardmäßigen HTTP-Nutzung entsprechen. Weitere Informationen zum Senden von Anfragen finden Sie unter [Senden von Anforderungen](#).

Berechtigungen

Sie können über gültige Anmeldeinformationen verfügen, um Ihre Anfragen zu authentifizieren. Sie können jedoch keine Amazon S3-Ressourcen erstellen oder darauf zugreifen, sofern Sie nicht über S3-Berechtigungen des Kontoinhabers oder Bucket-Inhabers verfügen. Diese Berechtigungen werden in der Regel durch eine AWS Identity and Access Management (IAM-) [Richtlinie gewährt](#), z. B. eine Bucket-Richtlinie. Sie müssen beispielsweise über Berechtigungen verfügen, um einen S3-Bucket zu erstellen oder ein Objekt in einem Bucket abzurufen. Eine vollständige Liste der S3-Berechtigungen finden Sie unter [Aktionen, Ressourcen und Bedingungsschlüssel für Amazon S3](#).

Weitere Informationen zu den Berechtigungen für S3-API-Operationen nach S3-Ressourcentypen finden Sie unter [Erforderliche Berechtigungen für Amazon S3-API-Operationen](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Wenn Sie die Root-Benutzeranmeldedaten von Ihnen verwenden AWS-Konto, verfügen Sie über alle Berechtigungen. Die Verwendung von Root-Benutzeranmeldedaten wird jedoch nicht empfohlen. Stattdessen empfehlen wir, dass Sie AWS Identity and Access Management (IAM-) Rollen in Ihrem Konto erstellen und Benutzerberechtigungen verwalten. Weitere Informationen finden Sie unter [Zugriffsverwaltung](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Entwickeln mit Amazon S3

Dieser Abschnitt enthält entwicklerbezogene Themen zur Verwendung von Amazon S3. Weitere Informationen finden Sie in den nachstehenden Themen.

Themen

- [Senden von Anforderungen](#)
- [Entwickeln mit Amazon S3 unter Verwendung der AWS CLI](#)
- [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#)
- [Abrufen von Amazon S3-Anforderungs-IDs für AWS Support](#)
- [Unterstützte Amazon S3-API-Operationen auf Objektebene für S3-Tabellen](#)

Senden von Anforderungen

Amazon S3 ist ein REST-Service. Sie können Anfragen an Amazon S3 senden, indem Sie die REST-API oder die AWS SDK-Wrapper-Bibliotheken (siehe [Beispielcode und Bibliotheken](#)) verwenden, die die zugrunde liegende Amazon S3-REST-API umschließen und so Ihre Programmieraufgaben vereinfachen.

Jede Interaktion mit Amazon S3 erfolgt entweder authentifiziert oder anonym. Die Authentifizierung ist ein Vorgang, bei dem die Identität des Auftraggebers, der auf ein Amazon Web Services (AWS) Produkt zugreifen möchte, überprüft wird. Authentifizierte Anforderungen müssen einen Signaturwert enthalten, der den Sender der Anforderung authentifiziert. Der Signaturwert wird teilweise aus den Zugriffsschlüsseln des Anforderers generiert (AWS Zugriffsschlüssel-ID und geheimer Zugriffsschlüssel). Weitere Informationen zum Abrufen von Zugriffsschlüsseln finden Sie unter [Wie erhalte ich Sicherheitsanmeldeinformationen?](#) in der Allgemeine AWS-Referenz.

Wenn Sie das AWS SDK verwenden, berechnen die Bibliotheken die Signatur aus den von Ihnen angegebenen Schlüsseln. Wenn Sie in Ihrer Anwendung jedoch direkte REST-API-Aufrufe senden, müssen Sie den Code für die Berechnung der Signatur schreiben und diesen der Anforderung hinzufügen.

Themen

- [Über Zugriffsschlüssel](#)
- [Anforderungsendpunkte](#)

- [Stellen von Anforderungen an Amazon S3 über IPv6](#)
- [Anfragen stellen mit dem AWS SDKs](#)
- [Senden von Anforderungen unter Verwendung der REST-API](#)

Über Zugriffsschlüssel

In den folgenden Abschnitten werden die Arten der Zugriffsschlüssel vorgestellt, die Sie verwenden können, um authentifizierte Anfragen zu senden.

AWS-Konto auf Schlüssel zugreifen

Die Kontozugriffsschlüssel bieten vollen Zugriff auf die AWS Ressourcen, die dem Konto gehören. Im Folgenden finden Sie Beispiele für Zugriffsschlüssel:

- Zugriffsschlüssel-ID (eine alphanumerische Zeichenfolge mit 20 Zeichen). Beispiel:
AKIAIOSFODNN7EXAMPLE
- Geheimer Zugriffsschlüssel (eine Zeichenfolge mit 40 Zeichen). Beispiel: wJalrXUtnFEMI/
K7MDENG/bPxRfiCYEXAMPLEKEY

Die Zugriffsschlüssel-ID identifiziert einen eindeutig AWS-Konto. Sie können diese Zugriffsschlüssel verwenden, um authentifizierte Anforderungen an Amazon S3 zu senden.

IAM-Benutzer-Zugriffsschlüssel

Sie können eine AWS-Konto für Ihr Unternehmen erstellen. Es kann jedoch sein, dass mehrere Mitarbeiter in der Organisation Zugriff auf die AWS Ressourcen Ihrer Organisation benötigen. Die gemeinsame Nutzung Ihrer AWS-Konto Zugangsschlüssel reduziert die Sicherheit, und es ist möglicherweise nicht praktikabel, AWS-Konten für jeden Mitarbeiter einen eigenen Schlüssel zu erstellen. Darüber hinaus können Sie Ressourcen wie Buckets und Objekte nicht einfach freigeben, da sie im Besitz verschiedener Konten sind. Um Ressourcen freizugeben, müssen Sie Berechtigungen gewähren. Dies bedeutet zusätzlichen Aufwand.

In solchen Szenarien können Sie AWS Identity and Access Management (IAM) verwenden, um Benutzer unter Ihrem Namen AWS-Konto mit eigenen Zugriffsschlüsseln zu erstellen und IAM-Benutzerrichtlinien anzuhängen, die diesen Benutzern die entsprechenden Ressourcenzugriffsberechtigungen gewähren. Um diese Benutzer besser verwalten zu können, ermöglicht IAM Ihnen das Erstellen von Benutzergruppen und das Gewähren von Berechtigungen auf Gruppenebene, die für alle Benutzer in der betreffenden Gruppe gelten.

Diese Benutzer werden als IAM-Benutzer bezeichnet, die Sie innerhalb von Ihnen erstellen und verwalten. AWS Das übergeordnete Konto steuert die Möglichkeiten der Benutzer, auf zuzugreifen AWS. Alle von einem IAM-Benutzer erstellten Ressourcen werden über das übergeordnete AWS-Konto gesteuert und bezahlt. Diese IAM-Benutzer können mittels ihrer eigenen Sicherheitsanmeldeinformationen authentifizierte Anfragen an Amazon S3 senden. Weitere Informationen zum Erstellen und Verwalten von Benutzern unter Ihrem Konto AWS-Konto finden Sie auf der [AWS Identity and Access Management Produktdetailseite](#).

Temporäre Sicherheitsanmeldeinformationen

Mit IAM können Sie nicht nur IAM-Benutzer mit ihren eigenen Zugriffsschlüsseln erstellen, sondern auch jedem IAM-Benutzer temporäre Sicherheitsanmeldeinformationen (temporäre Zugriffsschlüssel und ein Sicherheitstoken) gewähren, damit dieser auf Ihre AWS Dienste und Ressourcen zugreifen kann. Sie können Benutzer auch außerhalb von in Ihrem System verwaltete AWS. Diese Benutzer werden als verbundene Benutzer bezeichnet. Darüber hinaus können Benutzer Anwendungen sein, die Sie für den Zugriff auf Ihre Ressourcen erstellen. AWS

IAM stellt die AWS -Security-Token-Service API bereit, mit der Sie temporäre Sicherheitsanmeldeinformationen anfordern können. Sie können entweder die AWS STS-API oder das AWS SDK verwenden, um diese Anmeldeinformationen anzufordern. Die API gibt temporäre Sicherheitsanmeldeinformationen (Zugriffsschlüssel-ID und geheimen Zugriffsschlüssel) und ein Sicherheitstoken zurück. Diese Anmeldeinformationen sind nur für die Dauer gültig, die Sie bei der Anforderung angeben. Sie verwenden die Zugriffsschlüssel-ID und den geheimen Schlüssel auf dieselbe Weise, wie Sie diese beim Senden von Anfragen über Ihr AWS-Konto oder IAM-Benutzerzugriffsschlüssel verwenden. Zusätzlich muss jede Anfrage, die Sie an Amazon S3 senden, das Token enthalten.

Ein IAM-Benutzer kann diese temporären Sicherheitsanmeldeinformationen für die eigene Verwendung anfordern oder sie an verbundene Benutzer oder Anwendungen vergeben. Wenn Sie temporäre Sicherheitsanmeldeinformationen für verbundene Benutzer anfordern, müssen Sie einen Benutzernamen und eine IAM-Richtlinie bereitstellen, in der die Berechtigungen definiert sind, die Sie mit diesen temporären Sicherheitsanmeldeinformationen verknüpfen möchten. Ein verbundener Benutzer kann nicht mehr Berechtigungen als der übergeordnete IAM-Benutzer erhalten, der die temporären Anmeldeinformationen angefordert hat.

Sie können diese temporären Sicherheitsanmeldeinformationen beim Senden von Anfragen an Amazon S3 verwenden. Die API-Bibliotheken berechnen anhand dieser Anmeldeinformationen den notwendigen Signaturwert, um Ihre Anforderung zu authentifizieren. Wenn Sie beim Senden von Anfragen abgelaufene Anmeldeinformationen verwenden, lehnt Amazon S3 die Anfrage ab.

Informationen zum Signieren von Anfragen mittels temporärer Sicherheitsanmeldeinformationen in Ihren REST-API-Anfragen finden Sie unter [Signieren und Authentifizieren von REST-Anfragen \(AWS Signatur \(Version 2\)\)](#). Hinweise zum Senden von Anfragen mithilfe von AWS SDKs finden Sie unter [Anfragen stellen mit dem AWS SDKs](#).

Weitere Informationen zur Unterstützung temporärer Anmeldeinformationen in IAM finden Sie unter [Temporäre Sicherheitsanmeldeinformationen](#) im IAM-Benutzerhandbuch.

Um die Sicherheit zu verbessern, können Sie für den Zugriff auf Ihre Amazon-S3-Ressourcen eine Multi-Factor Authentication (MFA) anfordern, indem Sie eine Bucket-Richtlinie konfigurieren. Weitere Informationen finden Sie unter [Beispiel für Bucket-Richtlinien: MFA erforderlich](#). Nachdem Sie für den Zugriff auf Ihre Amazon-S3-Ressourcen eine MFA angefordert haben, können Sie auf diese Ressourcen nur durch die Bereitstellung temporärer Anmeldeinformationen zugreifen, die mittels eines MFA-Schlüssels erstellt wurden. Weitere Informationen finden Sie auf der [Detailseite zur AWS Multi-Factor Authentifizierung](#) und [zur Konfiguration des MFA-Protected API-Zugriffs](#) im IAM-Benutzerhandbuch.

Anforderungsendpunkte

Sie senden REST-Anfragen an den vordefinierten Endpunkt des Service. Eine Liste aller AWS Dienste und der entsprechenden Endpunkte finden Sie unter [Regionen und Endpunkte](#) im Allgemeinen AWS-Referenz

Stellen von Anforderungen an Amazon S3 über IPv6

Amazon Simple Storage Service (Amazon S3) unterstützt zusätzlich zum IPv4-Protokoll die Möglichkeit, mit dem Internet Protocol Version 6 (IPv6) auf S3-Buckets zuzugreifen. Amazon-S3-Dual-Stack-Endpunkte unterstützen Anforderungen an S3-Buckets über IPv6 und IPv4. Für den Zugriff auf Amazon S3 über IPv6 fallen keine zusätzlichen Gebühren an. Weitere Informationen zu Preisen finden Sie unter [Amazon-S3-Preise](#).

Note

Die folgenden Informationen beziehen sich auf öffentliche Amazon S3-IPv6-Endpunkte. Informationen zu AWS Virtual Private Cloud (VPC) -Endpunkten finden Sie unter für Amazon S3. [AWS PrivateLink](#)

Themen

- [Erste Schritte für Anforderungen über IPv6](#)
- [Verwenden von IPv6-Adressen in IAM-Richtlinien](#)
- [Testen der IP-Adresskompatibilität](#)
- [Verwenden von Amazon-S3-Dual-Stack-Endpunkten](#)

Erste Schritte für Anforderungen über IPv6

Um eine Anforderung für einen S3-Bucket über IPv6 zu erstellen, brauchen Sie einen Dual-Stack-Endpunkt. Der nächste Abschnitt beschreibt Anfragen über IPv6 unter Verwendung von Dual-Stack-Endpunkten.

Nachfolgend sind einige Dinge beschrieben, die Sie wissen sollten, bevor Sie versuchen, über IPv6 auf einen Bucket zuzugreifen.

- Der Client und das Netzwerk, die auf den Bucket zugreifen, müssen für IPv6 aktiviert sein.
- Für den IPv6-Zugriff werden Anforderungen im virtuellen Hosting- und im Pfad-Stil unterstützt. Weitere Informationen finden Sie unter [Amazon-S3-Dual-Stack-Endpunkte](#).
- Wenn Sie die Quell-IP-Adresse in Ihren AWS Identity and Access Management (IAM-) Benutzer- oder Bucket-Richtlinien verwenden, müssen Sie die Richtlinien so aktualisieren, dass sie IPv6-Adressbereiche enthalten. Weitere Informationen finden Sie unter [Verwenden von IPv6-Adressen in IAM-Richtlinien](#).
- Bei Verwendung von IPv6 geben die Serverzugriff-Protokolldateien IP-Adressen in einem IPv6-Format aus. Sie müssen vorhandene Tools, Skripts und Software aktualisieren, mit denen Sie Amazon-S3-Protokolldateien analysieren, sodass sie die mit IPv6 formatierte Remote IP-Adressen analysieren können. Weitere Informationen finden Sie unter [Protokollieren von Anfragen mit Serverzugriffsprotokollierung](#).

Note

Wenn Sie Probleme mit IPv6-Adressen in Protokolldateien haben, wenden Sie sich an den [AWS Support](#).

Anforderungen über IPv6 unter Verwendung von Dual-Stack-Endpunkten

Sie stellen Anforderungen Amazon-S3-API-Aufrufen über IPv6, indem Sie Dual-Stack-Endpunkte verwenden. Die Amazon-S3-API-Vorgänge funktionieren genauso, egal ob Sie über IPv6 oder über IPv4 auf Amazon S3 zugreifen. Die Leistung sollte ebenfalls dieselbe bleiben.

Wenn Sie die REST-API verwenden, greifen Sie direkt auf einen Dual-Stack-Endpunkt zu. Weitere Informationen finden Sie unter [Dual-stack Endpunkte](#).

Wenn Sie AWS Command Line Interface (AWS CLI) und AWS SDKs verwenden, können Sie einen Parameter oder ein Flag verwenden, um zu einem Dual-Stack-Endpunkt zu wechseln. Sie können den Dual-Stack-Endpunkt auch direkt als Override des Amazon-S3-Endpunkts in der Konfigurationsdatei angeben.

Sie können einen Dual-Stack-Endpunkt verwenden, um über IPv6 auf einen Bucket zuzugreifen. Dazu können Sie Folgendes verwenden:

- Das, siehe AWS CLI. [Verwenden Sie Dual-Stack-Endpunkte von AWS CLI](#)
- Die AWS SDKs, siehe [Verwenden Sie Dual-Stack-Endpunkte von AWS SDKs](#).
- Die REST-API, siehe [Senden von Anforderungen an Dual-Stack-Endpunkte unter Verwendung der REST-API](#).

Funktionen, die über IPv6 nicht zur Verfügung stehen

Die folgende Funktion wird derzeit beim Zugriff auf einen S3-Bucket über IPv6 nicht unterstützt: Statisches Website-Hosting von einem S3-Bucket aus.

Verwenden von IPv6-Adressen in IAM-Richtlinien

Bevor Sie versuchen, mit IPv6 auf einen Bucket zuzugreifen, müssen Sie sicherstellen, dass alle IAM-Benutzer- oder S3-Bucket-Richtlinien, die für die IP-Adressfilterung verwendet werden, aktualisiert werden, um den IPv6-Adressbereich zu berücksichtigen. Richtlinien für die IP-Adressfilterung, die nicht für die Verarbeitung von IPv6-Adressen aktualisiert werden, führen womöglich dazu, dass Clients den Zugriff auf den Bucket fälschlicherweise verlieren oder erhalten, wenn sie beginnen, IPv6 zu verwenden. Weitere Informationen über die Verwaltung von Zugriffsberechtigungen mit IAM finden Sie unter [Identity and Access Management für Amazon S3](#).

IAM-Richtlinien, die IP-Adressen filtern, verwenden [Bedingungsoperatoren für IP-Adressen](#). Die folgende Bucket-Richtlinie identifiziert den Bereich 54.240.143.* als Bereich zulässiger IPv4-Adressen durch Verwendung von Bedingungsoperatoren für IP-Adressen. Alle IP-Adressen außerhalb dieses

Bereichs erhalten keinen Zugriff auf den Bucket (examplebucket). Alle IPv6-Adressen liegen außerhalb des zulässigen Bereichs, deshalb verhindert diese Richtlinie, dass IPv6-Adressen auf zugreife 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"}
      }
    }
  ]
}
```

Sie können das Condition-Element der Bucket-Richtlinie ändern, um die IPv4- (54.240.143.0/24) und IPv6- (2001:DB8:1234:5678::/64) Adressbereiche zuzulassen, wie im folgenden Beispiel gezeigt. Sie können denselben Typ Condition-Block verwenden, wie im Beispiel gezeigt, um Ihre IAM-Benutzer- und Bucket-Richtlinien zu aktualisieren.

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

Bevor Sie IPv6 verwenden, müssen Sie alle relevanten IAM-Benutzer- und Bucket-Richtlinien aktualisieren, die IP-Adressfilterung verwenden. Vermeiden Sie die Verwendung von IP-Adressfiltern in Bucket-Richtlinien.

Sie können Ihre IAM-Benutzerrichtlinien in der IAM-Konsole unter überprüfen. <https://console.aws.amazon.com/iam/> Weitere Informationen zu IAM finden Sie im [IAM-Benutzerhandbuch](#). Informationen zum Bearbeiten von S3-Bucket-Richtlinien finden Sie unter [Hinzufügen einer Bucket-Richtlinie](#).

Testen der IP-Adresskompatibilität

Wenn Sie Linux/Unix oder Mac OS X verwenden, können Sie testen, ob Sie über IPv6 auf einen Dual-Stack-Endpunkt zugreifen können, indem Sie den `curl` Befehl verwenden, wie im folgenden Beispiel gezeigt:

Example

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

Sie erhalten Informationen wie im folgenden Beispiel gezeigt zurück. Wenn Sie über IPv6 verbunden sind, ist die verbundene IP-Adresse eine IPv6-Adresse.

```
* 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
```

Wenn Sie Microsoft 7 oder 10 verwenden, können Sie testen, ob Sie über IPv6 oder IPv4 auf einen Dual-Stack-Endpunkt zugreifen können, indem Sie den `ping` Befehl wie im folgenden Beispiel gezeigt verwenden.

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

Verwenden von Amazon-S3-Dual-Stack-Endpunkten

Amazon S3-Dual-Stack-Endpunkte unterstützen Anforderungen an S3-Buckets über IPv6 und IPv4. In diesem Abschnitt wird die Verwendung von Dual-Stack-Endpunkten beschrieben.

Themen

- [Amazon-S3-Dual-Stack-Endpunkte](#)
- [Verwenden Sie Dual-Stack-Endpunkte von AWS CLI](#)
- [Verwenden Sie Dual-Stack-Endpunkte von AWS SDKs](#)
- [Verwenden von Dual-Stack-Endpunkte von der REST-API](#)

Amazon-S3-Dual-Stack-Endpunkte

Wenn Sie eine Anforderung an einen Dual-Stack-Endpunkt richten, wird die Bucket-URL in eine IPv6- oder eine IPv4-Adresse aufgelöst. Weitere Informationen zum Zugriff auf einen Bucket über IPv6 finden Sie unter [Stellen von Anforderungen an Amazon S3 über IPv6](#).

Wenn Sie die REST-API verwenden, können Sie direkt auf einen Amazon-S3-Endpunkt zugreifen, indem Sie den Endpunktnamen (URI) verwenden. Über einen Dual-Stack-Endpunkt können Sie auf einen S3-Bucket zugreifen, indem Sie einen Virtual-Hosted-Style- oder Path-Style-Endpunktnamen verwenden. Amazon S3 unterstützt nur regionale Dual-Stack-Endpunktnamen, d. h. Sie müssen die Region als Teil des Namens angeben.

Verwenden Sie die folgenden Namenskonventionen für Endpunktnamen im Virtual-Hosted-Style und im Path-Style:

- Dual-Stack-Endpunkte im Virtual-Hosted-Style:

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

- Path-style Dual-Stack-Endpunkt:

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

Weitere Informationen zum Namensstil für Endpunkte finden Sie unter [Auf einen Amazon S3-Bucket zugreifen und ihn auflisten](#). Eine Liste der Amazon-S3-Endpunkte finden Sie unter [Regionen und Endpunkte](#) in der Allgemeine AWS-Referenz.

Important

Für Dual-Stack-Endpunkte kann eine Transfer Acceleration verwendet werden. Weitere Informationen finden Sie unter [Erste Schritte mit Amazon S3 Transfer Acceleration](#).

 Note

Die beiden Arten von Virtual Private Cloud (VPC) -Endpunkten, die auf Amazon S3 zugreifen (Schnittstellen-VPC-Endpunkte und Gateway-VPC-Endpunkte), bieten jetzt Dual-Stack-Unterstützung. Weitere Informationen zu VPC-Endpunkten für Amazon S3 finden Sie unter für Amazon S3. [AWS PrivateLink](#)

Wenn Sie AWS Command Line Interface (AWS CLI) und AWS SDKs verwenden, können Sie einen Parameter oder ein Flag verwenden, um zu einem Dual-Stack-Endpunkt zu wechseln. Sie können den Dual-Stack-Endpunkt auch direkt als Override des Amazon-S3-Endpunkts in der Konfigurationsdatei angeben. In den folgenden Abschnitten wird beschrieben, wie Dual-Stack-Endpunkte aus den und den AWS CLI SDKs verwendet werden. AWS

Verwenden Sie Dual-Stack-Endpunkte von AWS CLI

Dieser Abschnitt enthält Beispiele für AWS CLI Befehle, mit denen Anfragen an einen Dual-Stack-Endpunkt gestellt werden. Anweisungen zur Einrichtung von finden Sie AWS CLI unter [Entwickeln mit Amazon S3 unter Verwendung der AWS CLI](#).

Sie setzen den Konfigurationswert `true` in einem Profil in Ihrer AWS Config Datei `use_dualstack_endpoint` auf, um alle Amazon S3-Anfragen, die über die `s3api` AWS CLI Befehle `s3` und gestellt werden, an den Dual-Stack-Endpunkt für die angegebene Region weiterzuleiten. Sie geben die Region in der Konfigurationsdatei oder in einem Befehl mit der Option `--region` an.

Wenn Sie Dual-Stack-Endpunkte mit dem verwenden AWS CLI, werden sowohl als auch die `path virtual` Adressierungsstile unterstützt. Der Adressierungsstil, der in der Konfigurationsdatei festgelegt wird, steuert, ob der Bucketname im Hostnamen enthalten oder Teil der URL ist. Standardmäßig versucht die CLI, den virtuellen Stil zu verwenden, wann immer das möglich ist, verwendet aber auch den Pfadstil, wenn das notwendig ist. Weitere Informationen finden Sie unter [AWS CLI -Amazon-S3-Konfiguration](#).

Sie können auch Konfigurationsänderungen über einen Befehl vornehmen, wie im folgenden Beispiel gezeigt, das im Standardprofil `use_dualstack_endpoint` auf `true` und `addressing_style` auf `virtual` setzt.

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

Wenn Sie einen Dual-Stack-Endpoint nur für bestimmte AWS CLI Befehle (nicht für alle Befehle) verwenden möchten, können Sie eine der folgenden Methoden verwenden:

- Sie können den Dual-Stack-Endpoint pro Befehl verwenden, indem Sie den Parameter `--endpoint-url` auf `https://s3.dualstack.aws-region.amazonaws.com` oder `http://s3.dualstack.aws-region.amazonaws.com` für jeden s3- oder s3api-Befehl setzen.

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

- Sie können separate Profile in Ihrer AWS Config Datei einrichten. Legen Sie beispielsweise ein Profil an, das `use_dualstack_endpoint` auf `true` setzt, und ein Profil, das `use_dualstack_endpoint` nicht setzt. Wenn Sie einen Befehl ausführen, geben Sie an, welches Profil Sie verwenden wollen, abhängig davon, ob Sie den Dual-Stack-Endpoint verwenden wollen oder nicht.

Note

Wenn AWS CLI Sie den verwenden, können Sie die Übertragungsbeschleunigung derzeit nicht mit Dual-Stack-Endpunkten verwenden. Die Unterstützung für das AWS CLI wird jedoch bald verfügbar sein. Weitere Informationen finden Sie unter [S3 Transfer Acceleration aktivieren und verwenden](#).

Verwenden Sie Dual-Stack-Endpunkte von AWS SDKs

Dieser Abschnitt enthält Beispiele für den Zugriff auf einen Dual-Stack-Endpoint mithilfe der SDKs.
AWS

AWS SDK für Java Beispiel für Dual-Stack-Endpunkte

Das folgende Beispiel veranschaulicht, wie Sie beim Erstellen eines Amazon-S3-Clients mit dem AWS SDK für Java Dual-Stack-Endpunkte aktivieren.

Anweisungen zum Erstellen und Testen eines funktionierenden Java-Beispiels finden Sie unter [Erste Schritte](#) im AWS SDK für Java Entwicklerhandbuch.

```
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();
        }
    }
}
```

Wenn Sie das AWS SDK für Java unter Windows verwenden, müssen Sie möglicherweise die folgende Eigenschaft der Java Virtual Machine (JVM) festlegen:

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

AWS .NET SDK-Beispiel für einen Dual-Stack-Endpunkt

Wenn Sie das AWS SDK für .NET verwenden, verwenden Sie die `AmazonS3Config` Klasse, um die Verwendung eines Dual-Stack-Endpunkts zu aktivieren, wie im folgenden Beispiel gezeigt.

```
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.USWest2;
        private static IAmazonS3 client;

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

        private static async Task ListingObjectsAsync()
        {
            try
            {
                var request = new ListObjectsV2Request
                {
                    BucketName = bucketName,
                    MaxKeys = 10
                };
                ListObjectsV2Response response;
                do
                {
                    response = await client.ListObjectsV2Async(request);

                    // Process the response.
                    foreach (S3Object entry in response.S3Objects)
                }
            }
        }
    }
}
```

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

Informationen zum Einrichten und Ausführen der Codebeispiele finden Sie unter [Getting Started with the AWS SDK for .NET](#) im AWS SDK for .NET Developer Guide.

Verwenden von Dual-Stack-Endpunkte von der REST-API

Weitere Informationen über Anfragen an Dual-Stack-Endpunkte über die REST-API finden Sie unter [Senden von Anforderungen an Dual-Stack-Endpunkte unter Verwendung der REST-API](#).

Anfragen stellen mit dem AWS SDKs

Themen

- [Anfragen stellen mit AWS-Konto oder IAM-Benutzeranmeldedaten](#)
- [Anfragen unter Verwendung temporärer Anmeldeinformationen für IAM-Benutzer erstellen](#)
- [Anforderungen unter Verwendung temporärer Anmeldeinformationen verbundener Benutzer](#)

Sie können authentifizierte Anfragen an Amazon S3 senden, indem Sie entweder das AWS SDK verwenden oder die REST-API-Aufrufe direkt in Ihrer Anwendung ausführen. Die AWS SDK-API verwendet die von Ihnen angegebenen Anmeldeinformationen, um die Signatur für die Authentifizierung zu berechnen. Bei direkter Verwendung der REST-API in Ihren Anwendungen müssen Sie den benötigten Code zur Berechnung der Signatur für die Authentifizierung Ihrer Anforderung schreiben. Eine Liste der verfügbaren AWS SDKs finden Sie unter [Beispielcode und Bibliotheken](#).

Anfragen stellen mit AWS-Konto oder IAM-Benutzeranmeldedaten

Sie können Ihre AWS-Konto oder IAM-Benutzersicherheitsanmeldeinformationen verwenden, um authentifizierte Anfragen an Amazon S3 zu senden. Dieser Abschnitt enthält Beispiele dafür, wie Sie authentifizierte Anfragen mithilfe von AWS SDK für Java, AWS SDK für .NET und senden können. AWS SDK für PHP Eine Liste der verfügbaren AWS SDKs finden Sie unter [Beispielcode und Bibliotheken](#).

Jedes dieser AWS SDKs verwendet eine Anbieterkette für SDK-specific Anmeldeinformationen, um Anmeldeinformationen zu finden und zu verwenden und Aktionen im Namen des Eigentümers der Anmeldeinformationen auszuführen. Allen diesen Anbieterketten für Anmeldeinformationen ist gemeinsam, dass sie alle nach Ihrer lokalen AWS Anmeldeinformationsdatei suchen.

Weitere Informationen finden Sie in den folgenden Themen:

Themen

- [Um eine lokale Datei zu erstellen AWS Datei mit Anmeldeinformationen](#)
- [Senden authentifizierter Anfragen mit dem AWS SDKs](#)
- [Zugehörige Ressourcen](#)

Um eine lokale Datei zu erstellen AWS Datei mit Anmeldeinformationen

Die einfachste Methode, Anmeldeinformationen für Ihre AWS SDKs zu konfigurieren, ist die Verwendung einer AWS Anmeldeinformationsdatei. Wenn Sie die AWS Command Line Interface (AWS CLI) verwenden, haben Sie möglicherweise bereits eine lokale AWS Anmeldeinformationsdatei konfiguriert. Andernfalls gehen Sie wie folgt vor, um eine Anmeldeinformationsdatei einzurichten:

1. Melden Sie sich bei der an AWS-Managementkonsole und öffnen Sie die IAM-Konsole unter <https://console.aws.amazon.com/iam/>.
2. Erstellen Sie einen neuen Benutzer mit Berechtigungen, die auf die Services und Aktionen beschränkt sind, auf die Ihr Code Zugriff hat. Weitere Informationen zum Erstellen eines neuen Benutzers finden Sie unter [Erstellen von IAM-Benutzern \(Konsole\)](#). Folgen Sie der Anleitung bis Schritt 8.
3. Wählen Sie Download .csv (CSV herunterladen) aus, um eine lokale Kopie Ihrer AWS - Anmeldeinformationen zu speichern.
4. Gehen Sie auf Ihrem Computer zu Ihrem Stammverzeichnis und erstellen Sie ein .aws Verzeichnis. Auf Unix-based Systemen wie Linux oder OS X befindet sich diese an der folgenden Stelle:

```
~/ .aws
```

Unter Windows befindet es sich an der folgenden Position:

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

5. Erstellen Sie in dem Verzeichnis .aws eine neue Datei namens `credentials`.
6. Öffnen Sie die .csv-Datei mit den Anmeldeinformationen aus, die Sie von der IAM-Konsole heruntergeladen haben, und kopieren Sie ihren Inhalt in die `credentials`-Datei mit dem folgenden Format:

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

7. Speichern Sie die `credentials`-Datei, und löschen Sie die .csv-Datei, die Sie in Schritt 3 heruntergeladen haben.

Ihre gemeinsame Anmeldeinformationsdatei ist jetzt auf Ihrem lokalen Computer konfiguriert und kann mit den AWS SDKs verwendet werden.

Senden authentifizierter Anfragen mit dem AWS SDKs

Verwenden Sie die AWS SDKs, um authentifizierte Anfragen zu senden. Weitere Informationen zum Senden authentifizierter Anfragen finden Sie unter [AWS - Sicherheitsanmeldeinformationen](#) oder [Authentifizierung von IAM Identity Center](#).

Java

Informationen zur Authentifizierung von Anfragen mit dem AWS SDK für Java finden Sie unter Using [shared config and credentials files to global configure AWS SDKs and Tools und Authentication and access using AWS SDKs and Tools](#) Reference Guide AWS .

.NET

So senden Sie authentifizierte Anfragen mit Ihren AWS-Konto oder IAM-Benutzeranmeldeinformationen:

- Erstellen Sie eine Instance der `AmazonS3Client`-Klasse.
- Führen Sie eine der `AmazonS3Client`-Methoden aus, um Anfragen an Amazon S3 zu senden. Der Client erstellt aus den von Ihnen angegebenen Anmeldeinformationen die erforderliche Signatur und nimmt sie in die Anforderung auf, die er an Amazon S3 sendet.

Weitere Informationen finden Sie unter Anfragen mit AWS Konto- oder IAM-Benutzeranmeldedaten [stellen >](#).

Note

- Sie können den `AmazonS3Client`-Client erstellen, ohne Sicherheitsanmeldeinformationen anzugeben. Anforderung, die unter Verwendung dieses Clients gesendet werden, sind anonym und haben keine Signatur. Amazon S3 gibt einen Fehler zurück, wenn Sie anonyme Anfragen für eine Ressource stellen, die nicht öffentlich verfügbar ist.
- Sie können ein AWS-Konto erstellen und die erforderlichen Benutzer erstellen. Sie können auch Anmeldeinformationen für diese Benutzer verwalten. Sie benötigen diese Anmeldeinformationen, um die Aufgabe im folgenden Beispiel auszuführen. Weitere

Informationen finden Sie unter [Configure AWS credentials \(Konfigurieren von AWS - Anmeldeinformationen\)](#) im SDK für .NET -Entwicklerhandbuch.

Anschließend können Sie Ihre Anwendung auch so konfigurieren, dass Profile und Anmeldeinformationen aktiv abgerufen werden, und diese Anmeldeinformationen dann explizit beim Erstellen eines AWS Service-Clients verwenden. Weitere Informationen finden Sie unter [Accessing credentials and profiles in an application \(Zugriff auf Anmeldeinformationen und Profile in einer Anwendung\)](#) im SDK für .NET -Entwicklerhandbuch.

Das folgende C#-Beispiel veranschaulicht, wie sie die vorhergehenden Aufgaben ausführen. Informationen zum Einrichten und Ausführen der Codebeispiele finden Sie unter [Getting Started with the AWS SDK for .NET](#) im AWS SDK for .NET-Entwicklerhandbuch.

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

In diesem Abschnitt wird erklärt, wie Sie eine Klasse aus Version 3 von verwenden, AWS SDK für PHP um authentifizierte Anfragen mit Ihren AWS-Konto oder IAM-Benutzeranmeldeinformationen zu senden. Weitere Informationen zur AWS SDK for Ruby API finden Sie unter [AWS SDK for Ruby — Version 2](#).

Das folgende PHP-Beispiel veranschaulicht, wie der Client eine Anfrage unter Verwendung Ihrer Sicherheitsanmeldeinformationen durchführt, um alle Buckets für Ihr Konto aufzulisten.

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

Sie können den `S3Client`-Client erstellen, ohne Sicherheitsanmeldeinformationen anzugeben. Anforderung, die unter Verwendung dieses Clients gesendet werden, sind anonym und haben keine Signatur. Amazon S3 gibt einen Fehler zurück, wenn Sie anonyme Anfragen für eine Ressource stellen, die nicht öffentlich verfügbar ist. Weitere Informationen finden Sie unter [Erstellen von anonymen Clients](#) in der [AWS SDK für PHP - Dokumentation](#).

Ruby

Bevor Sie Version 3 von verwenden können, AWS SDK für Ruby um Amazon S3 aufzurufen, müssen Sie die AWS Zugangsdaten festlegen, die das SDK verwendet, um Ihren Zugriff auf Ihre Buckets und Objekte zu verifizieren. Wenn Sie gemeinsame Anmeldeinformationen im AWS Anmeldeinformationsprofil auf Ihrem lokalen System eingerichtet haben, kann Version 3 des SDK für Ruby diese Anmeldeinformationen verwenden, ohne dass Sie sie in Ihrem Code deklarieren müssen. Weitere Informationen zum Einrichten gemeinsamer Anmeldeinformationen finden Sie unter Anfragen mithilfe von AWS Konto- oder IAM-Benutzeranmeldeinformationen <https://docs.aws.amazon.com/AmazonS3/latest/userguide/AuthUsingAcctOrUserCredentials.html> stellen.

Der folgende Ruby-Codeausschnitt verwendet die Anmeldeinformationen in einer Datei mit gemeinsam genutzten AWS Anmeldeinformationen auf einem lokalen Computer, um eine Anforderung zum Abrufen aller Objektschlüsselnamen in einem bestimmten Bucket zu authentifizieren. Es führt die folgenden Aktionen aus:

1. Erstellt eine Instance der `Aws::S3::Client`-Klasse.
2. Sendet eine Anfrage an Amazon S3 durch Auflisten der Objekte in einem Bucket mit der `list_objects_v2`-Methode von `Aws::S3::Client`. Der Client generiert den erforderlichen Signaturwert aus den Anmeldeinformationen in der AWS Anmeldeinformationsdatei auf Ihrem Computer und nimmt ihn in die Anfrage auf, die er an Amazon S3 sendet.
3. Gibt das Array der Objektschlüsselnamen an das Terminal aus.

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__
```

Wenn Sie keine lokale AWS Anmeldeinformationsdatei haben, können Sie die `Aws::S3::Client` Ressource trotzdem erstellen und Code für Amazon S3-Buckets und -Objekte ausführen. Anforderungen, die mit Version 3 des SDK for Ruby gesendet werden, sind anonym und haben standardmäßig keine Signatur. Amazon S3 gibt einen Fehler zurück, wenn Sie anonyme Anfragen für eine Ressource stellen, die nicht öffentlich verfügbar ist.

Sie können den obigen Codeausschnitt für SDK for Ruby-Anwendungen verwenden und erweitern, wie im folgenden, robusteren Beispiel gezeigt.

```
# 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

Im folgenden Beispiel werden AWS Anmeldeinformationen verwendet, die vom SDK für Go automatisch aus der Datei mit den gemeinsam genutzten Anmeldeinformationen geladen werden.

```

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)
    }
}
}
```

Zugehörige Ressourcen

- [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#)
- [AWS SDK für PHP für Amazon S3 Aws\S3\S3Client-Klasse](#)
- [AWS SDK für PHP -Dokumentation](#)

Anfragen unter Verwendung temporärer Anmeldeinformationen für IAM-Benutzer erstellen

Ein AWS-Konto oder ein IAM-Benutzer kann temporäre Sicherheitsanmeldeinformationen anfordern und diese verwenden, um authentifizierte Anfragen an Amazon S3 zu senden. Dieser Abschnitt enthält Beispiele, wie Sie das AWS SDK für Java, .NET und PHP verwenden können, um temporäre Sicherheitsanmeldeinformationen zu erhalten und diese zur Authentifizierung Ihrer Anfragen an Amazon S3 zu verwenden.

Java

Ein oder mehrere IAM-Benutzer AWS-Konto können mithilfe von temporäre Sicherheitsanmeldeinformationen (siehe [Senden von Anforderungen](#)) anfordern AWS SDK für Java und diese für den Zugriff auf Amazon S3 verwenden. Diese Anmeldeinformationen laufen nach der angegebenen Sitzungsdauer ab.

Die Sitzungsdauer beträgt standardmäßig eine Stunde. Wenn Sie IAM-Benutzer-Anmeldeinformationen verwenden, können Sie als Dauer, nach der die temporären Sicherheitsanmeldeinformationen angefordert werden sollen, eine Zeitspanne zwischen 15 Minuten und der maximalen Sitzungsdauer für die Rolle angeben. Weitere Informationen zu temporären Anmeldeinformationen finden Sie unter [Temporäre Sicherheitsanmeldeinformationen](#) im IAM-Benutzerhandbuch. Weitere Informationen zum Senden von Anfragen finden Sie unter [Senden von Anforderungen](#).

So rufen Sie temporäre Sicherheitsanmeldeinformationen ab und greifen auf Amazon S3 zu

1. Erstellen Sie eine Instance der `AWSSecurityTokenService`-Klasse.
2. Rufen Sie die temporären Sicherheitsanmeldeinformationen für die gewünschte Rolle ab, indem Sie die `assumeRole()`-Methode des Security Token Service (STS) -Clients aufrufen.
3. Bündeln Sie die temporären Sicherheitsanmeldeinformationen in einem `BasicSessionCredentials`-Objekt. Sie verwenden dieses Objekt, um die temporären Sicherheitsanmeldeinformationen für Ihren Amazon-S3-Client bereitzustellen.
4. Erstellen Sie mit den temporären Sicherheitsanmeldeinformationen eine Instance der `AmazonS3Client`-Klasse. Mit diesem Client senden Sie Anfragen an Amazon S3. Wenn Sie beim Senden von Anfragen abgelaufene Anmeldeinformationen verwenden, gibt Amazon S3 einen Fehler zurück.

 Note

Wenn Sie unter Verwendung der Sicherheitsanmeldeinformationen Ihres AWS-Konto temporäre Sicherheitsanmeldeinformationen erhalten haben, sind sie nur eine Stunde lang gültig. Sie können eine Sitzungsdauer nur dann festlegen, wenn Sie IAM-Benutzer-Anmeldeinformationen verwenden, um eine Sitzung anzufordern.

Das folgende Beispiel listet einen Satz von Objektschlüsseln im angegebenen Bucket auf. Das Beispiel fordert temporäre Sicherheitsanmeldeinformationen für eine Sitzung an, die dann zum Senden einer authentifizierten Anfrage an Amazon S3 verwendet werden.

Wenn Sie das Beispiel mit IAM-Benutzer-Anmeldeinformationen testen möchten, müssen Sie einen IAM-Benutzer unter Ihrem AWS-Konto erstellen. Weitere Informationen zum Erstellen eines IAM-Benutzers finden Sie unter [Erstellen Ihrer ersten IAM-Benutzer- und Administratorengruppe](#) im IAM-Benutzerhandbuch.

Anweisungen zum Erstellen und Testen eines funktionierenden Beispiels finden Sie [im AWS SDK für Java Entwicklerhandbuch unter Erste Schritte](#).

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

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



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

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

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

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

    // Verify that assuming the role worked and the permissions are set
correctly
    // by getting a set of object keys from the bucket.
    ObjectListing objects = s3Client.listObjects(bucketName);
```

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

.NET

Ein IAM-Benutzer oder ein IAM-Benutzer AWS-Konto kann mithilfe von temporäre Sicherheitsanmeldeinformationen anfordern AWS SDK für .NET und diese für den Zugriff auf Amazon S3 verwenden. Diese Anmeldeinformationen laufen nach Ende der Sitzung ab.

Die Sitzungsdauer beträgt standardmäßig eine Stunde. Wenn Sie IAM-Benutzer-Anmeldeinformationen verwenden, können Sie als Dauer, nach der die temporären Sicherheitsanmeldeinformationen angefordert werden sollen, eine Zeitspanne zwischen 15 Minuten und der maximalen Sitzungsdauer für die Rolle angeben. Weitere Informationen zu temporären Anmeldeinformationen finden Sie unter [Temporäre Sicherheitsanmeldeinformationen](#) im IAM-Benutzerhandbuch. Weitere Informationen zum Senden von Anfragen finden Sie unter [Senden von Anforderungen](#).

So rufen Sie temporäre Sicherheitsanmeldeinformationen ab und greifen auf Amazon S3 zu

1. Erstellen Sie eine Instanz des AWS -Security-Token-Service Clients, `AmazonSecurityTokenServiceClient`.
2. Starten Sie eine Sitzung durch Aufruf der Methode `GetSessionToken` des STS-Clients, den Sie im vorigen Schritt erstellt haben. Sie stellen für diese Methode Sitzungsdaten mithilfe eines `GetSessionTokenRequest`-Objekts bereit.

Die Methode gibt Ihre temporären Sicherheitsanmeldeinformationen zurück.

3. Bündeln Sie die temporären Sicherheitsanmeldeinformationen in einer Instance des Objekts `SessionAWSCredentials`. Sie verwenden dieses Objekt, um die temporären Sicherheitsanmeldeinformationen für Ihren Amazon-S3-Client bereitzustellen.
4. Erstellen Sie eine Instance der Klasse `AmazonS3Client`, indem Sie die temporären Sicherheitsanmeldeinformationen übergeben. Mit diesem Client senden Sie Anfragen an Amazon S3. Wenn Sie beim Senden der Anfragen abgelaufene Anmeldeinformationen verwenden, gibt Amazon S3 einen Fehler zurück.

 Note

Wenn Sie unter Verwendung der Sicherheitsanmeldeinformationen Ihres AWS-Konto temporäre Sicherheitsanmeldeinformationen erhalten haben, sind diese Anmeldeinformationen nur eine Stunde lang gültig. Sie können die Sitzungsdauer nur dann festlegen, wenn Sie IAM-Benutzer-Anmeldeinformationen verwenden, um eine Sitzung anzufordern.

Das folgende C#-Beispiel listet Objektschlüssel im angegebenen Bucket auf. Zur Veranschaulichung fordert das Beispiel temporäre Sicherheitsanmeldeinformationen für die standardmäßige, einstündige Sitzung an, die dann zum Senden einer authentifizierten Anfrage an Amazon S3 verwendet werden.

Wenn Sie das Beispiel mit IAM-Benutzer-Anmeldeinformationen testen möchten, müssen Sie einen IAM-Benutzer unter Ihrem AWS-Konto erstellen. Weitere Informationen zum Erstellen eines IAM-Benutzers finden Sie unter [Erstellen Ihrer ersten IAM-Benutzer- und Administratorengruppe](#) im IAM-Benutzerhandbuch. Weitere Informationen zum Senden von Anfragen finden Sie unter [Senden von Anforderungen](#).

Informationen zum Einrichten und Ausführen der Codebeispiele finden Sie unter [Getting Started with the AWS SDK for .NET](#) im AWS SDK for .NET-Entwicklerhandbuch.

```
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

Weitere Informationen zur AWS SDK für Ruby-API finden Sie unter [AWS SDK for Ruby — Version 2](#).

Ein IAM-Benutzer oder ein IAM-Benutzer AWS-Konto kann temporäre Sicherheitsanmeldeinformationen mithilfe von Version 3 von anfordern. AWS SDK für PHP Mit

diesen temporären Anmeldeinformationen kann der Benutzer/das Konto dann auf Amazon S3 zugreifen. Die Anmeldeinformationen laufen mit dem Ende der Sitzung ab.

Die Sitzungsdauer beträgt standardmäßig eine Stunde. Wenn Sie IAM-Benutzer-Anmeldeinformationen verwenden, können Sie als Dauer, nach der die temporären Sicherheitsanmeldeinformationen angefordert werden sollen, eine Zeitspanne zwischen 15 Minuten und der maximalen Sitzungsdauer für die Rolle angeben. Weitere Informationen zu temporären Anmeldeinformationen finden Sie unter [Temporäre Sicherheitsanmeldeinformationen](#) im IAM-Benutzerhandbuch. Weitere Informationen zum Senden von Anfragen finden Sie unter [Senden von Anforderungen](#).

 Note

Wenn Sie unter Verwendung Ihrer AWS-Konto -Sicherheitsanmeldeinformationen temporäre Sicherheitsanmeldeinformationen erhalten haben, sind sie lediglich eine Stunde lang gültig. Sie können eine Sitzungsdauer nur dann festlegen, wenn Sie IAM-Benutzer-Anmeldeinformationen verwenden, um eine Sitzung anzufordern.

Example

Das folgende PHP-Beispiel listet die Objektschlüssel im angegebenen Bucket bei Verwendung temporärer Sicherheitsanmeldeinformationen auf. Das Beispiel fordert temporäre Sicherheitsanmeldeinformationen für eine standardmäßige, einstündige Sitzung an, die dann zum Senden einer authentifizierten Anfrage an Amazon S3 verwendet werden. Weitere Informationen zur AWS SDK for Ruby API finden Sie unter [AWS SDK for Ruby — Version 2](#).

Wenn Sie das Beispiel mit IAM-Benutzer-Anmeldeinformationen testen möchten, müssen Sie einen IAM-Benutzer unter Ihrem AWS-Konto erstellen. Weitere Informationen zum Erstellen eines IAM-Benutzers finden Sie unter [Erstellen Ihrer ersten IAM-Benutzer und Administratorengruppe](#) im IAM-Benutzerhandbuch. Ein Beispiel für das Festlegen der Sitzungsdauer beim Anfordern einer Sitzung mit IAM-Benutzer-Anmeldeinformationen finden Sie unter [Anfragen unter Verwendung temporärer Anmeldeinformationen für IAM-Benutzer erstellen](#).

```
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

Ein IAM-Benutzer oder ein IAM-Benutzer AWS-Konto kann temporäre Sicherheitsanmeldeinformationen anfordern AWS SDK für Ruby und diese für den Zugriff auf Amazon S3 verwenden. Diese Anmeldeinformationen laufen nach Ende der Sitzung ab.

Die Sitzungsdauer beträgt standardmäßig eine Stunde. Wenn Sie IAM-Benutzer-Anmeldeinformationen verwenden, können Sie als Dauer, nach der die temporären Sicherheitsanmeldeinformationen angefordert werden sollen, eine Zeitspanne zwischen 15 Minuten und der maximalen Sitzungsdauer für die Rolle angeben. Weitere Informationen zu temporären Anmeldeinformationen finden Sie unter [Temporäre Sicherheitsanmeldeinformationen](#) im IAM-Benutzerhandbuch. Weitere Informationen zum Senden von Anfragen finden Sie unter [Senden von Anforderungen](#).

 Note

Wenn Sie unter Verwendung Ihrer AWS-Konto -Sicherheitsanmeldeinformationen temporäre Sicherheitsanmeldeinformationen erhalten haben, sind sie lediglich eine Stunde lang gültig. Sie können die Sitzungsdauer nur dann festlegen, wenn Sie IAM-Benutzer-Anmeldeinformationen verwenden, um eine Sitzung anzufordern.

Das folgende Ruby-Beispiel erstellt einen temporären Benutzer, damit dieser für eine Stunde die Elemente in einem bestimmten Bucket auflisten kann. Um dieses Beispiel verwenden zu können, benötigen Sie AWS Anmeldeinformationen, die über die erforderlichen Berechtigungen verfügen, um neue AWS -Security-Token-Service (AWS STS) -Clients zu erstellen und Amazon S3-Buckets aufzulisten.

```
# 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

```

```

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
end

```

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

Anforderungen unter Verwendung temporärer Anmeldeinformationen verbundener Benutzer

Sie können temporäre Sicherheitsanmeldeinformationen anfordern und diese Ihren Verbundbenutzern oder Anwendungen zur Verfügung stellen, die auf Ihre AWS Ressourcen zugreifen müssen. In diesem Abschnitt finden Sie Beispiele dafür, wie Sie das AWS SDK verwenden können, um temporäre Sicherheitsanmeldeinformationen für Ihre Verbundbenutzer oder -anwendungen zu erhalten und mithilfe dieser Anmeldeinformationen authentifizierte Anfragen an Amazon S3 zu senden. Eine Liste der verfügbaren AWS SDKs finden Sie unter [Beispielcode und Bibliotheken](#).

Note

AWS-Konto Sowohl der als auch ein IAM-Benutzer können temporäre Sicherheitsanmeldeinformationen für Verbundbenutzer anfordern. Jedoch sollte aus Sicherheitsgründen nur ein IAM-Benutzer mit den notwendigen Berechtigungen diese temporären Anmeldeinformationen anfordern, um sicherzustellen, dass der verbundene Benutzer höchstens die Berechtigungen des anfragenden IAM-Benutzers erhält. In einigen Anwendungen kann es nützlich sein, einen IAM-Benutzer mit den spezifischen Berechtigungen nur zu dem Zweck zu erstellen, temporäre Sicherheitsanmeldeinformationen für Ihre verbundenen Benutzer und Anwendungen zu erteilen.

Java

Sie können temporäre Sicherheitsanmeldeinformationen für Ihre Verbundbenutzer und Anwendungen bereitstellen, damit diese authentifizierte Anfragen für den Zugriff auf Ihre Ressourcen senden können. AWS Wenn Sie diese temporären Anmeldeinformationen anfordern, müssen Sie einen Benutzernamen und eine IAM-Richtlinie bereitstellen, mit der die Ressourcenberechtigungen beschrieben wird, die Sie erteilen möchten. Die Sitzungsdauer beträgt standardmäßig eine Stunde. Sie können explizit einen anderen Wert für die Dauer angeben, wenn Sie temporäre Sicherheitsanmeldeinformationen für verbundene Benutzer und Anwendungen anfordern.

Note

Zur Gewährleistung zusätzlicher Sicherheit beim Anfordern temporärer Sicherheitsanmeldeinformationen für verbundene Benutzer und Anwendungen empfehlen wir die Verwendung eines dedizierten IAM-Benutzers mit nur den dazu

benötigten Zugriffsberechtigungen. Der temporäre Benutzer, den Sie erstellen, darf nie mehr Berechtigungen erhalten als der IAM-Benutzer, der die temporären Sicherheitsanmeldeinformationen angefordert hat. Weitere Informationen finden Sie unter [AWS Identity and Access Management : Häufig gestellte Fragen](#).

Gehen Sie zum Bereitstellen von Sicherheitsanmeldeinformationen und zum Senden einer authentifizierten Anfrage für den Zugriff auf Ressourcen wie folgt vor:

- Erstellen Sie eine Instance der `AWSSecurityTokenServiceClient`-Klasse.
- Starten Sie mit dem Aufruf der Methode `getFederationToken()` des Security Token Service (STS)-Clients eine Sitzung. Stellen Sie Sitzungsinformationen wie beispielsweise den Benutzernamen und eine IAM-Richtlinie bereit, die Sie den temporären Anmeldeinformationen anfügen möchten. Optional können Sie eine Sitzungsdauer angeben. Diese Methode gibt Ihre temporären Sicherheitsanmeldeinformationen zurück.
- Bündeln Sie die temporären Sicherheitsanmeldeinformationen in einer Instance des Objekts `BasicSessionCredentials`. Sie verwenden dieses Objekt, um die temporären Sicherheitsanmeldeinformationen für Ihren Amazon-S3-Client bereitzustellen.
- Erstellen Sie mit den temporären Sicherheitsanmeldeinformationen eine Instance der `AmazonS3Client`-Klasse. Mit diesem Client senden Sie Anfragen an Amazon S3. Wenn Sie beim Senden der Anfragen abgelaufene Anmeldeinformationen verwenden, gibt Amazon S3 einen Fehler zurück.

Example

Das folgende Beispiel listet die Schlüssel im angegebenen S3-Bucket auf. In dem Beispiel erhalten Sie temporäre Sicherheitsanmeldeinformationen für eine zweistündige Sitzung für Ihren verbundenen Benutzer und verwenden sie, um authentifizierte Anfragen an Amazon S3 zu senden. Um das Beispiel auszuführen, müssen Sie einen IAM-Benutzer mit einer angehängten Richtlinie erstellen, die es dem Benutzer ermöglicht, temporäre Sicherheitsanmeldeinformationen anzufordern und Ihre Ressourcen aufzulisten. AWS Dies wird mit der folgenden Richtlinie erreicht:

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

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

Weitere Informationen zum Erstellen eines IAM-Benutzers finden Sie unter [Erstellen Ihrer ersten IAM-Benutzer- und Administratorengruppe](#) im IAM-Benutzerhandbuch.

Nachdem ein IAM-Benutzer erstellt und die vorhergehende Richtlinie angefügt wurde, können Sie das folgende Beispiel ausführen. Anweisungen zum Erstellen und Testen eines funktionierenden Beispiels finden Sie [im AWS SDK für Java Entwicklerhandbuch unter](#) Erste Schritte.

```
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

Sie können temporäre Sicherheitsanmeldeinformationen für Ihre Verbundbenutzer und -anwendungen bereitstellen, damit diese authentifizierte Anfragen für den Zugriff auf Ihre AWS Ressourcen senden können. Wenn Sie diese temporären Anmeldeinformationen anfordern, müssen Sie einen Benutzernamen und eine IAM-Richtlinie bereitstellen, mit der die Ressourcenberechtigungen beschrieben wird, die Sie erteilen möchten. Standardmäßig beträgt die Dauer einer Sitzung eine Stunde. Sie können explizit einen anderen Wert für die Dauer angeben, wenn Sie temporäre Sicherheitsanmeldeinformationen für verbundene Benutzer und Anwendungen anfordern. Informationen zum Senden authentifizierter Anfragen finden Sie unter [Senden von Anforderungen](#).

Note

Wenn Sie temporäre Sicherheitsanmeldeinformationen für Verbundbenutzer und Anwendungen anfordern, verwenden Sie aus Sicherheitsgründen einen dedizierten IAM-Benutzer, der nur über die erforderlichen Zugriffsberechtigungen verfügt. Der temporäre Benutzer, den Sie erstellen, darf nie mehr Berechtigungen erhalten als der IAM-Benutzer, der die temporären Sicherheitsanmeldeinformationen angefordert hat. Weitere Informationen finden Sie unter [AWS Identity and Access Management : Häufig gestellte Fragen](#).

Dazu führen Sie die folgenden Schritte aus:

- Erstellen Sie eine Instanz des AWS -Security-Token-Service Clients, `AmazonSecurityTokenServiceClient` Klasse.

- Starten Sie eine Sitzung durch Aufruf der Methode `GetFederationToken` des STS-Clients. Sie müssen Sitzungsinformationen bereitstellen, wie beispielsweise den Benutzernamen und eine IAM-Richtlinie, die Sie den temporären Anmeldeinformationen zuordnen wollen. Optional können Sie eine Sitzungsdauer angeben. Diese Methode gibt Ihre temporären Sicherheitsanmeldeinformationen zurück.
- Bündeln Sie die temporären Sicherheitsanmeldeinformationen in einer Instance des Objekts `SessionAWSCredentials`. Sie verwenden dieses Objekt, um die temporären Sicherheitsanmeldeinformationen für Ihren Amazon-S3-Client bereitzustellen.
- Erstellen Sie eine Instance der `AmazonS3Client`-Klasse, indem Sie die temporären Sicherheitsanmeldeinformationen übergeben. Sie verwenden diesen Client zum Senden von Anfragen an Amazon S3. Wenn Sie beim Senden der Anfragen abgelaufene Anmeldeinformationen verwenden, gibt Amazon S3 einen Fehler zurück.

Example

Das folgende C#-Beispiel listet die Schlüssel im angegebenen Bucket auf. In dem Beispiel erhalten Sie temporäre Sicherheitsanmeldeinformationen für eine zweistündige Sitzung für Ihren verbundenen Benutzer (User1) und verwenden sie, um authentifizierte Anfragen an Amazon S3 zu senden.

- In dieser Übung erstellen Sie einen IAM-Benutzer mit minimalen Berechtigungen. Unter Verwendung der Anmeldeinformationen dieses IAM-Benutzers fordern Sie temporäre Anmeldeinformationen für andere an. Dieses Beispiel listet nur die Objekte in einem bestimmten Bucket auf. Erstellen Sie einen IAM-Benutzer, dem die folgenden Richtlinie zugeordnet ist:

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

Die Richtlinie ermöglicht es dem IAM-Benutzer, temporäre Sicherheitsanmeldeinformationen und Zugriffsberechtigungen nur für die Auflistung Ihrer AWS Ressourcen anzufordern. Weitere Informationen zum Erstellen eines IAM-Benutzers finden Sie unter [Erstellen Ihrer IAM-Benutzer- und Administratorengruppe](#) im IAM-Benutzerhandbuch.

- Verwenden Sie die Sicherheitsanmeldeinformationen des IAM-Benutzers, um das folgende Beispiel zu testen. Das Beispiel sendet unter Verwendung temporärer Sicherheitsanmeldeinformationen eine authentifizierte Anfrage an Amazon S3. Das Beispiel gibt die folgende Richtlinie an, wenn temporäre Sicherheitsanmeldeinformationen für den verbundenen Benutzer (User1) angefordert werden, womit der Zugriff auf eine Auflistung der Objekte in einem spezifischen Bucket eingeschränkt wird (YourBucketName). Sie müssen die Richtlinie aktualisieren, indem Sie Ihren eigenen bestehenden Bucket-Namen angeben.

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

- Example

Aktualisieren Sie das folgende Beispiel und geben Sie den Bucket-Namen an, den Sie in der vorherigen Zugriffsrichtlinie für verbundene Benutzer angegeben haben. Informationen zum Einrichten und Ausführen der Codebeispiele finden Sie unter [Erste Schritte mit dem AWS SDK for .NET](#) im AWS SDK for .NET-Entwicklerhandbuch.

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

```

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

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

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

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

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

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

```

```

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

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

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

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

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

```

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

PHP

In diesem Thema wird erklärt, wie Klassen aus Version 3 von verwendet werden, AWS SDK für PHP um temporäre Sicherheitsanmeldeinformationen für Verbundbenutzer und Anwendungen anzufordern und diese für den Zugriff auf in Amazon S3 gespeicherte Ressourcen zu verwenden. Weitere Informationen zum AWS SDK für Ruby API finden Sie unter [AWS SDK for Ruby — Version 2](#).

Sie können Ihren Verbundbenutzern und Anwendungen temporäre Sicherheitsanmeldeinformationen zur Verfügung stellen, damit diese authentifizierte Anfragen für den Zugriff auf Ihre AWS Ressourcen senden können. Wenn Sie diese temporären Anmeldeinformationen anfordern, müssen Sie einen Benutzernamen und eine IAM-Richtlinie bereitstellen, mit der die Ressourcenberechtigungen beschrieben wird, die Sie erteilen möchten. Diese Anmeldeinformationen laufen mit dem Ende der Sitzung ab. Die Sitzungsdauer beträgt standardmäßig eine Stunde. Sie können explizit einen anderen Wert für die Dauer angeben, wenn Sie temporäre Sicherheitsanmeldeinformationen für verbundene Benutzer und Anwendungen anfordern. Weitere Informationen zu temporären Anmeldeinformationen finden Sie unter [Temporäre Sicherheitsanmeldeinformationen](#) im IAM-Benutzerhandbuch. Informationen zum Bereitstellen temporärer Sicherheitsanmeldeinformationen für Ihre verbundenen Benutzer und Anwendungen finden Sie unter [Senden von Anforderungen](#).

Für zusätzliche Sicherheit beim Anfordern temporärer Sicherheitsanmeldeinformationen für verbundene Benutzer und Anwendungen empfehlen wir die Verwendung eines dedizierten IAM-Benutzers mit nur den dazu benötigten Zugriffsberechtigungen. Der temporäre Benutzer, den Sie erstellen, darf nie mehr Berechtigungen erhalten als der IAM-Benutzer, der die temporären Sicherheitsanmeldeinformationen angefordert hat. Weitere Informationen über Identitätsverbunde finden Sie unter den [Häufig gestellten Fragen zu AWS Identity and Access Management](#).

Weitere Informationen zur AWS SDK for Ruby API finden Sie unter [AWS SDK for Ruby — Version 2](#).

Example

Das folgende PHP-Beispiel listet Schlüssel im angegebenen Bucket auf. In dem Beispiel erhalten Sie temporäre Sicherheitsanmeldeinformationen für eine einstündige Sitzung für Ihren verbundenen Benutzer (User1). Sie verwenden die temporären Sicherheitsanmeldeinformationen dann zum Senden authentifizierter Anfragen an Amazon S3.

Verwenden Sie zur Gewährleistung zusätzlicher Sicherheit beim Anfordern temporärer Anmeldeinformationen für andere die Sicherheitsanmeldeinformationen eines IAM-Benutzers, der über die Berechtigung zum Anfordern temporärer Sicherheitsanmeldeinformationen verfügt. Um sicherzustellen, dass der IAM-Benutzer dem verbundenen Benutzer nur die anwendungsspezifischen Mindestberechtigungen gewährt, können Sie auch die Zugriffsberechtigungen dieses IAM-Benutzers einschränken. Dieses Beispiel listet nur Objekte in einem bestimmten Bucket auf. Erstellen Sie einen IAM-Benutzer, dem die folgenden Richtlinie zugeordnet ist:

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

Die Richtlinie ermöglicht es dem IAM-Benutzer, temporäre Sicherheitsanmeldeinformationen und Zugriffsberechtigungen nur für die Auflistung Ihrer AWS Ressourcen anzufordern. Weitere Informationen zum Erstellen eines IAM-Benutzers finden Sie unter [Erstellen Ihrer ersten IAM-Benutzer- und Administratorengruppe](#) im IAM-Benutzerhandbuch.

Jetzt können Sie die Sicherheitsanmeldeinformationen des IAM-Benutzers verwenden, um das folgende Beispiel zu testen. Das Beispiel sendet unter Verwendung temporärer Sicherheitsanmeldeinformationen eine authentifizierte Anfrage an Amazon S3. Wenn temporäre Sicherheitsanmeldeinformationen für den verbundenen Benutzer (User1) angefordert werden, gibt das Beispiel die folgenden Richtlinie an, womit der Zugriff auf eine Auflistung der Objekte in einem bestimmten Bucket eingeschränkt wird. Aktualisieren Sie die Richtlinie mit Ihrem Bucket-Namen.

```
{
```

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

Ersetzen Sie im folgenden Beispiel bei der Angabe der Richtlinien-Ressource `YourBucketName` durch den Namen Ihres Buckets:

```
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

Sie können temporäre Sicherheitsanmeldeinformationen für Ihre Verbundbenutzer und Anwendungen bereitstellen, damit diese authentifizierte Anfragen für den Zugriff auf Ihre Ressourcen senden können. AWS Wenn Sie diese temporären Anmeldeinformationen vom IAM-Service anfordern, müssen Sie einen Benutzernamen und eine IAM-Richtlinie bereitstellen, mit der die Ressourcenberechtigungen beschrieben werden, die Sie erteilen möchten. Die Sitzungsdauer beträgt standardmäßig eine Stunde. Wenn Sie temporäre Sicherheitsanmeldeinformationen unter Verwendung von IAM-Benutzer-Anmeldeinformationen anfordern, können Sie explizit einen anderen Wert für die Gültigkeitsdauer festlegen, wenn Sie die temporären Sicherheitsanmeldeinformationen für verbundene Benutzer und Anwendungen anfordern. Informationen zu temporären Sicherheitsanmeldeinformationen für Ihre verbundenen Benutzer und Anwendungen finden Sie unter [Senden von Anforderungen](#).

Note

Verwenden Sie zur Gewährleistung zusätzlicher Sicherheit beim Anfordern temporärer Sicherheitsanmeldeinformationen für verbundene Benutzer und Anwendungen einen dedizierten IAM-Benutzer mit nur den dazu benötigten Zugriffsberechtigungen. Der temporäre Benutzer, den Sie erstellen, darf nie mehr Berechtigungen erhalten als der

IAM-Benutzer, der die temporären Sicherheitsanmeldeinformationen angefordert hat. Weitere Informationen finden Sie unter [AWS Identity and Access Management : Häufig gestellte Fragen](#).

Example

Das folgende Ruby-Codebeispiel gestattet es einem verbundenen Benutzer mit einem limitierten Satz von Berechtigungen Schlüssel im angegebenen Bucket aufzulisten.

```
# 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__
```

Senden von Anforderungen unter Verwendung der REST-API

In diesem Abschnitt finden Sie Informationen zum Senden von Anfragen an Amazon-S3-Endpunkte über die REST-API. Eine Liste der Amazon-S3-Endpunkte finden Sie unter [Regionen und Endpunkte](#) in der Allgemeine AWS-Referenz.

Erstellen von S3-Hostnamen für REST-API-Anforderungen

Amazon-S3-Endpunkte folgen der folgenden Struktur:

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

Amazon-S3-Zugriffspunkte-Endpunkte und Dual-Stack-Endpunkte folgen ebenfalls der Standardstruktur:

- Amazon-S3-Zugriffspunkte -s3-accesspoint.*Region*.amazonaws.com
- Dual-stack - s3.dualstack.*Region*.amazonaws.com

Eine vollständige Liste der Amazon-S3-Regionen und -Endpunkte finden Sie unter [Endpunkte und Kontingente von Amazon S3](#) in der Allgemeine Amazon Web Services-Referenz.

Virtuell gehostete und Pfad-Stil-Anforderungen

Wenn Sie Anforderungen über die REST-API senden, können Sie für die Amazon-S3-Endpunkte URIs im virtuell gehosteten oder im Pfad-Stil verwenden. Weitere Informationen finden Sie unter [Path-style Anfragen](#).

Example Anforderung im virtuell gehosteten Stil

Nachfolgend finden Sie ein Beispiel für eine virtuell gehostete Anforderung zum Löschen der Datei `puppy.jpg` aus dem Bucket `examplebucket` in der Region USA West (Oregon). Weitere Informationen zu Anfragen im virtuellen Hosting-Stil finden Sie unter [Path-style Anfragen](#).

```
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 Path-style Anfrage

Im Folgenden finden Sie ein Beispiel für eine Path-Style-Version derselben Anfrage.

```
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
```

Derzeit unterstützt Amazon S3 den URL-Zugriff in allen AWS-Regionen sowohl im virtuellem Hosting- als auch im Pfadformat. URLs im Pfadformat werden jedoch in naher Zukunft eingestellt. Weitere Informationen finden Sie im folgenden wichtigen Hinweis.

Weitere Informationen zu Anfragen im Pfadstil finden Sie unter [Path-style Anfragen](#).

Important

Update (23. September 2020) – Wir haben das Veralten von URLs im Pfadformat verschoben, um sicherzustellen, dass Kunden genügend Zeit haben, um zu URLs im virtuellen Hosting-Format zu wechseln. Weitere Informationen finden Sie unter [Amazon S3 Path Deprecation Plan – The Rest of the Story](#) im AWS News Blog.

Senden von Anforderungen an Dual-Stack-Endpunkte unter Verwendung der REST-API

Wenn Sie die REST-API verwenden, können Sie direkt auf einen Dual-Stack-Endpunkt zugreifen, indem Sie einen Virtual-Hosted-Style- oder Path-Style-Endpunktnamen (URI) verwenden. Alle Amazon-S3-Dual-Stack-Endpunktnamen enthalten die Region. Im Gegensatz zu den IPv4-only Standardendpunkten verwenden sowohl Endpunkte im virtuellen Hosting-Stil als auch im Pfadstil regionsspezifische Endpunktnamen.

Example Dual-Stack-Endpunkt-Anforderung im virtuell gehosteten Stil

Sie können in Ihrer REST-Anforderung einen virtuell gehosteten Endpunkt wie im folgenden Beispiel gezeigt verwenden, der das Objekt `puppy.jpg` aus dem Bucket `examplebucket` in der Region USA West (Oregon) abrufen.

```
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 Path-style Dual-Stack-Endpunkt-Anfrage

Sie können auch wie im folgenden Beispiel gezeigt einen Path-Style-Endpunkt in Ihrer Anfrage verwenden.

```
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
```

Weitere Informationen zu Dual-Stack-Endpunkten finden Sie unter [Verwenden von Amazon-S3-Dual-Stack-Endpunkten](#).

Weitere Informationen zum Stellen von Anfragen mithilfe der REST-API finden Sie in den folgenden Themen.

Themen

- [Anforderungsumleitung und die REST-API](#)
- [Weiterleitung von Anforderungen](#)

Anforderungsumleitung und die REST-API

Themen

- [Umleitungen und HTTP-Benutzeragenten](#)
- [Umleitungen und 100 Continue](#)
- [Beispiel für eine Umleitung](#)

In diesem Abschnitt wird beschrieben, wie HTTP-Umleitungen unter Verwendung der Amazon-S3-REST-API verarbeitet werden. Allgemeine Informationen zu Amazon-S3-Weiterleitungen finden Sie unter [Senden von Anforderungen](#) in der Amazon-Simple-Storage-Service-API-Referenz.

Umleitungen und HTTP-Benutzeragenten

Programme, die die Amazon S3 REST-API verwenden, sollten Umleitungen auf der Anwendungsschicht oder auf der HTTP-Ebene verarbeiten. Es können viele HTTP-Client-Bibliotheken und Benutzeragenten konfiguriert werden, um Umleitungen automatisch korrekt zu verarbeiten. Viele andere beinhalten jedoch fehlerhafte oder unvollständige Implementierungen für Umleitungen.

Bevor Sie sich darauf verlassen, dass eine Bibliothek die Umleitungsanfrage erfüllt, testen Sie die folgenden Fälle:

- Überprüfen Sie, ob alle HTTP-Anfrageheader korrekt in die Umleitungsanfrage aufgenommen wurden (die zweite Anfrage nach Empfang einer Umleitung), einschließlich der HTTP-Standards, wie beispielsweise Autorisierung und Datum.
- Überprüfen Sie, ob auch andere als GET-Umleitungen ordnungsgemäß funktionieren, wie beispielsweise PUT und DELETE.
- Überprüfen Sie, ob PUT-Anfragen den Umleitungen ordnungsgemäß folgen.
- Überprüfen Sie, ob PUT-anfragenden Umleitungen ordnungsgemäß folgen, wenn die Antwort auf 100 continue lang dauert.

HTTP-Benutzeragenten, die streng konform zu RFC 2616 arbeiten, fordern möglicherweise eine explizite Bestätigung, bevor sie einer Umleitung folgen, wenn die HTTP-Anfragemethode nicht GET

oder HEAD ist. Im Allgemeinen ist es sicher, von Amazon S3 erstellten Umleitungen automatisch zu folgen, weil das System nur Umleitungen auf Hosts innerhalb der Domäne `amazonaws.com` ausgibt, und die Wirkung der umgeleiteten Anfrage dieselbe ist wie die der ursprünglichen Anforderung.

Umleitungen und 100 Continue

Um die Verarbeitung von Umleitungen zu vereinfachen, die Effizienzen zu verbessern und die Kosten zu vermeiden, die beim doppelten Versenden eines umgeleiteten Anfragerumpfs entstehen, konfigurieren Sie Ihre Anwendung so, dass sie 100 continues für PUT-Operationen verwendet. Wenn Ihre Anwendung 100 continue verwendet, sendet sie den Anfragerumpf erst dann, wenn sie eine Bestätigung erhält. Wird die Nachricht basierend auf den Header abgewiesen, wird der Rumpf der Meldung nicht gesendet. Weitere Informationen zu 100-continue finden Sie unter [RFC 2616 Abschnitt 8.2.3](#).

Note

Laut RFC 2616 sollten Sie nicht unendlich lange warten, bevor Sie den Anfragerumpf senden, wenn Sie `Expect: Continue` für einen unbekannten HTTP-Server verwenden. Der Grund dafür ist, dass einige HTTP-Server 100 continue nicht erkennen. Amazon S3 erkennt jedoch nicht, ob Ihre Anforderung ein `Expect: Continue` enthält, und reagiert mit einem vorläufigen 100-continue-Status oder einem endgültigen Statuscode. Darüber hinaus tritt kein Umleitungscode auf, nachdem die provisorische 100-continue-Genehmigung empfangen wurde. Dies hilft Ihnen, den Empfang einer Umleitungsantwort zu vermeiden, wenn Sie noch den Anfragerumpf schreiben.

Beispiel für eine Umleitung

Dieser Abschnitt bietet ein Beispiel für eine Client/Server-Interaktion mit HTTP-Umleitungen und 100 continue.

Nachfolgend finden Sie ein Beispiel für ein PUT in den `quotes.s3.amazonaws.com`-Bucket.

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

Content-Length: 6
```

```
Expect: 100-continue
```

Amazon S3 gibt Folgendes zurück:

```
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>
```

Der Client folgt der Umleitungsantwort und gibt eine neue Anfrage an den temporären Endpunkt `quotes.s3-4c25d83b.amazonaws.com` aus.

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

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

Amazon S3 gibt ein 100 continue zurück, das darauf hinweist, dass der Client damit fortfahren soll, den Anforderungstext zu senden.

```
HTTP/1.1 100 Continue
```

Der Client sendet den Anfragerumpf.

```
ha ha\n
```

Amazon S3 gibt die endgültige Antwort zurück:

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

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

Weiterleitung von Anforderungen

Programme, die Anfragen für Buckets stellen, die mithilfe der [CreateBucket](#) API erstellt wurden und die eine Option enthalten, die Weiterleitungen unterstützen [CreateBucketConfiguration](#) muss. Darüber hinaus entstehen möglicherweise für einige Clients Probleme, die DNS TTLs nicht unterstützen.

Dieser Abschnitt beschreibt Weiterleitungs- und DNS-Aspekte, die Sie beim Entwurf Ihres Service oder Ihrer Anwendung für Amazon S3 berücksichtigen sollten.

Anforderungsumleitung und die REST-API

Amazon S3 verwendet das DNS (Domain Name System), um Anfragen an Einrichtungen weiterzuleiten, die sie verarbeiten können. Dieses System arbeitet effektiv, aber es können temporäre Routing-Fehler auftreten. Trifft eine Anfrage am falschen Amazon-S3-Standort ein, reagiert Amazon S3 mit einer temporären Umleitung, die den Auftraggeber anweist, die Anfrage an einen neuen Endpunkt zu senden. Ist eine Anfrage fehlerhaft aufgebaut, verwendet Amazon S3 permanente Umleitungen, um dem Auftraggeber mitzuteilen, wie die Anfrage korrekt ausgeführt wird.

Important

Um diese Funktion nutzen zu können, benötigen Sie eine Anwendung, die Amazon-S3-Umleitungsantworten verarbeiten kann. Die einzige Ausnahme bilden Anwendungen, die ausschließlich mit Buckets arbeiten, die ohne erstellt wurde `<CreateBucketConfiguration>`. Weitere Informationen zu Standortbeschränkungen finden Sie unter Auf einen Amazon S3-Bucket <https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-bucket-intro.html> zugreifen und ihn auflisten.

Für alle Regionen, die nach dem 20. März 2019 gestartet wurden, gibt Amazon S3 den Fehler „HTTP 400 Bad Request“ zurück, wenn eine Anforderung an einer falschen Amazon-S3-Position empfangen wird.

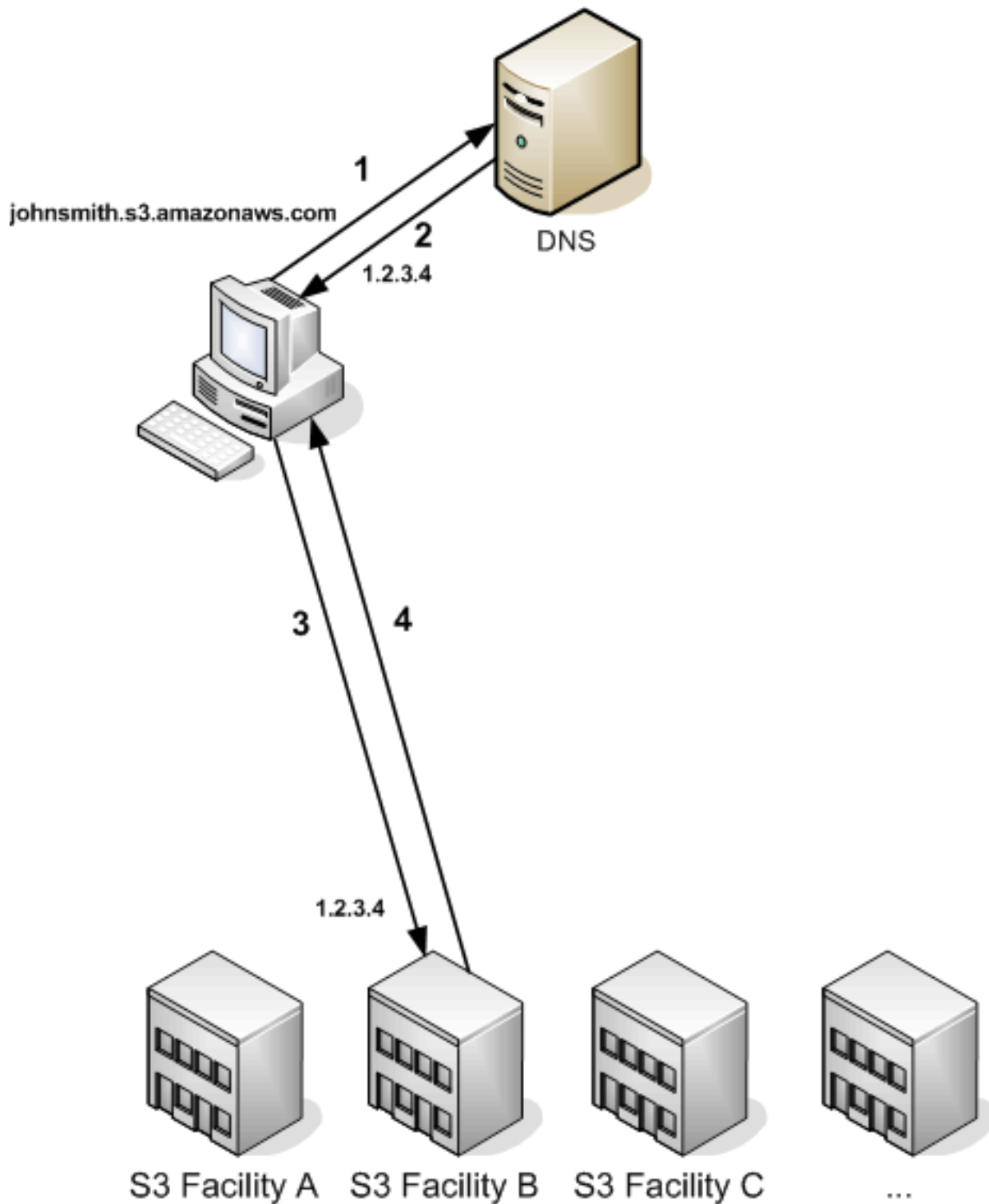
Weitere Informationen zum Aktivieren oder Deaktivieren eines finden Sie AWS-Region unter [AWS-Regionen und Endpunkte](#) im. Allgemeine AWS-Referenz

Themen

- [DNS-Weiterleitung](#)
- [Temporäre Anforderungsumleitung](#)
- [Permanente Anforderungsumleitung](#)
- [Beispiele für Anforderungsumleitung](#)

DNS-Weiterleitung

Die DNS-Weiterleitung leitet Anfragen an geeignete Amazon-S3-Einrichtungen weiter. Die folgende Abbildung und Vorgehensweise zeigen ein Beispiel für DNS-Routing.



DNS-Routing-Anforderungsschritte

1. Der Client stellt eine DNS-Anfrage, um ein Objekt in Amazon S3 speichern zu lassen.

2. Der Client erhält eine oder mehrere IP-Adressen für Einrichtungen, die die Anfrage verarbeiten können. In diesem Beispiel ist die IP-Adresse für die Einrichtung B.
3. Der Client stellt eine Anforderung an Amazon-S3-Einrichtung B.
4. Einrichtung B gibt eine Kopie des Objekts an den Kunden zurück.

Temporäre Anforderungsumleitung

Eine temporäre Umleitung ist eine Art Fehlermeldung, die dem Auftraggeber signalisiert, dass er die Anfrage an einen anderen Endpunkt senden sollte. Aufgrund der verteilten Natur von Amazon S3 können Anfragen temporär in die falsche Einrichtung weitergeleitet werden. Dies tritt am wahrscheinlichsten auf, unmittelbar nachdem Buckets erstellt oder gelöscht wurden.

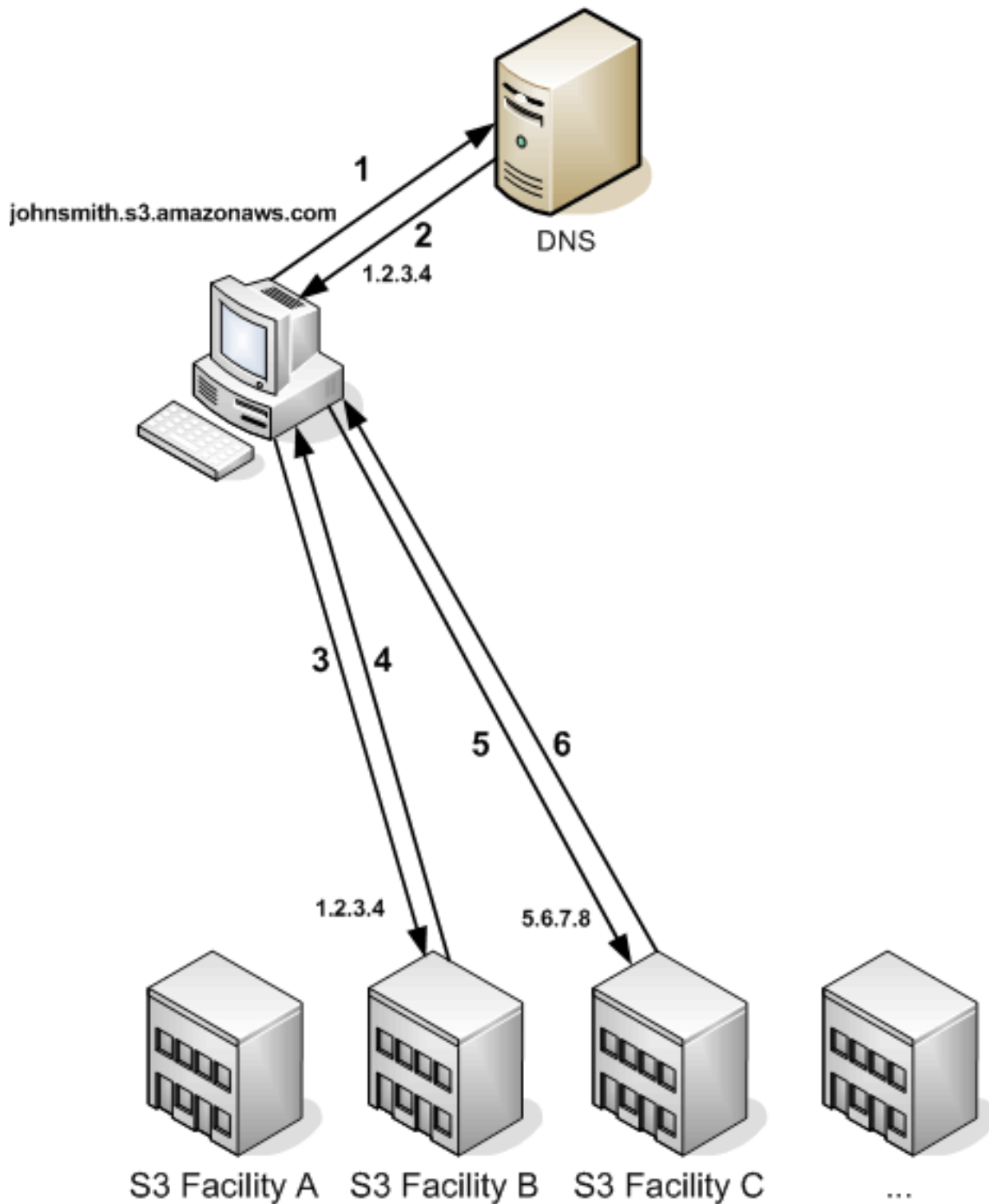
Wenn Sie beispielsweise einen neuen Bucket erstellen und sofort eine Anfrage für den Bucket machen, werden Sie möglicherweise temporär umgeleitet, abhängig von der Standorteinschränkung des Buckets. Wenn Sie den Bucket im Osten der USA (Nord-Virginia) erstellt haben AWS-Region, wird die Umleitung nicht angezeigt, da dies auch der standardmäßige Amazon S3-Endpunkt ist.

Wenn der Bucket jedoch in einer anderen Region erstellt wird, gehen alle Anfragen für den Bucket an den Standard-Endpunkt, während der DNS-Eintrag des Buckets propagiert wird. Der Standard-Endpunkt leitet die Anfrage mit einer HTTP 302-Antwort an den richtigen Endpunkt weiter. Temporäre Umleitungen enthalten eine URI zur richtigen Einrichtung, die Sie verwenden können, um die Anfrage sofort erneut zu senden.

Important

Verwenden Sie keinen Endpunkt wieder, der aus einer vorherigen Umleitungsantwort stammt. Das scheint zu funktionieren (manchmal sogar sehr lange), kann aber unvorhersehbare Ergebnisse erzeugen und schlägt irgendwann ohne weitere Mitteilung fehl.

Die folgende Abbildung und Vorgehensweise zeigen ein Beispiel für eine temporäre Umleitung.



Temporäre Umleitungsschritte für Anfragen

1. Der Client stellt eine DNS-Anfrage, um ein Objekt in Amazon S3 speichern zu lassen.

2. Der Client erhält eine oder mehrere IP-Adressen für Einrichtungen, die die Anfrage verarbeiten können.
3. Der Client stellt eine Anforderung an Amazon-S3-Einrichtung B.
4. Einrichtung B gibt eine Umleitung zurück, die angibt, dass das Objekt an Standort C zur Verfügung steht.
5. Der Client sendet Anfrage noch einmal an Einrichtung C.
6. Einrichtung C gibt eine Kopie des Objekts zurück.

Permanente Anforderungsumleitung

Eine permanente Anfrageumleitung zeigt an, dass Ihre Anfrage eine fehlerhafte Adresse für eine Ressource angegeben hat. Permanente Umleitungen treten beispielsweise auf, wenn Sie eine Anfrage mit Angabe eines Pfads für den Zugriff auf einen Bucket verwenden, der mit erstellt wurde `<CreateBucketConfiguration>`. Weitere Informationen finden Sie unter [Auf einen Amazon S3-Bucket zugreifen und ihn auflisten](#).

Um diese Fehler bei der Entwicklung zu erkennen, enthält diese Art Umleitung keinen HTTP Location-Header, der Ihnen ermöglicht, die Anfrage automatisch an den richtigen Standort zu verfolgen. Weitere Informationen über die Verwendung des richtigen Amazon-S3-Endpunkts finden Sie im resultierenden XML-Fehlerdokument.

Beispiele für Anforderungsumleitung

Im Folgenden finden Sie Beispiele für temporäre Umleitungsantworten.

Temporäre Umleitungsantwort der REST-API

```
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-gztlb4pa9sq.amazonaws.com</Endpoint>  
</Error>
```

Temporäre Umleitungsantwort der SOAP API

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

```
<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>
```

Überlegungen zu DNS

Eine der Entwurfsanforderungen von Amazon S3 ist eine extrem hohe Verfügbarkeit. Unter anderem erfüllen wir diese Anforderung, indem wir die dem Amazon-S3-Endpunkt in DNS zugeordnete IP-Adressen nach Bedarf aktualisieren. Diese Änderungen wirken sich automatisch auf kurzlebige Clients aus, nicht jedoch auf einige langlebige Clients. Long-lived Kunden müssen in regelmäßigen Abständen besondere Maßnahmen ergreifen, um den Amazon S3-Endpunkt erneut zu beheben, um von diesen Änderungen zu profitieren. Weitere Informationen zu virtuellen Maschinen (VMs) finden Sie hier:

- Für Java speichert die JVM von Sun standardmäßig DNS-Lookups für immer im Cache. Informationen dazu, wie Sie dieses Verhalten ändern können, finden Sie im Abschnitt „InetAddressCaching“ [der InetAddress Dokumentation](#).

- Für PHP speichert die persistente PHP VM, die in den gebräuchlichsten Bereitstellungskonfigurationen ausgeführt wird, DNS-Suchen, bis die VM neu gestartet wird. Gehen Sie zum Abschnitt „PHP-Dokumente abrufen“ [. HostByName](#)

Entwickeln mit Amazon S3 unter Verwendung der AWS CLI

Die Amazon AWS CLI S3-Befehle sind in der AWS CLI Befehlsreferenz in verschiedene Sätze unterteilt, und für jeden sind eigene Befehle verfügbar. Wenn Sie den Befehl, nach dem Sie suchen, in einem Satz nicht finden, überprüfen Sie einen der anderen Sätze. Die verschiedenen Sätze lauten wie folgt:

- `s3`— Beschreibt allgemeine Befehle für die Arbeit mit Objekten und Buckets. Dazu gehören Aktionen zum Kopieren, Auflisten und Verschieben. Eine vollständige Liste der Befehle in diesem Satz finden Sie unter [s3](#) in der AWS CLI Befehlsreferenz.
- `s3api`— Beschreibt Befehle auf niedriger Ebene, die S3-Ressourcen wie Buckets, Objekte, Sitzungen und mehrteilige Uploads verwalten. Diese Befehle entsprechen den [Amazon S3-API-Vorgängen](#). Eine vollständige Liste der CLI-Befehle in diesem Satz finden Sie unter [s3api](#) in der AWS CLI Befehlsreferenz.
- `s3control`— Beschreibt Befehle auf niedriger Ebene, die alle anderen S3-Ressourcen wie Access Grants, Storage Lens-Gruppen und Amazon S3 auf Outposts-Buckets verwalten. Diese Befehle entsprechen den [Amazon S3 Control-API-Vorgängen](#). Eine vollständige Liste der CLI-Befehle in diesem Satz finden Sie unter [s3control](#) in der AWS CLI Befehlsreferenz.

Wenn Sie Probleme bei der Einrichtung von haben AWS CLI, finden Sie im AWS re:Post Knowledge Center unter [Wie behebe ich den Fehler „Anmeldeinformationen konnten nicht gefunden werden“](#) in Amazon S3.

Note

Dienste in AWS, wie Amazon S3, erfordern, dass Sie Anmeldeinformationen angeben, wenn Sie darauf zugreifen. Der Service kann dann feststellen, ob Sie über die Berechtigung für den Zugriff auf seine Ressourcen verfügen. Für die Konsole müssen Sie Ihr Passwort eingeben. Sie können Zugriffsschlüssel für den Zugriff AWS-Konto auf die AWS CLI OR-API erstellen. Greifen Sie jedoch nicht AWS mit den Anmeldeinformationen für Ihre zu AWS-Konto. Stattdessen empfehlen wir Ihnen, AWS Identity and Access Management (IAM) zu verwenden. Erstellen Sie einen IAM-Benutzer und fügen Sie den Benutzer zu einer IAM-

Gruppe mit Administrator-Berechtigungen hinzu. Anschließend gewähren Sie dem von Ihnen erstellten IAM-Benutzer administrative Berechtigungen. Sie können dann mithilfe einer speziellen URL und mit den Anmeldeinformationen für den IAM-Benutzer auf AWS zugreifen. Weitere Anweisungen finden Sie unter [Erstellen Ihrer ersten IAM-Benutzer- und Administratorengruppe](#) im IAM-Benutzerhandbuch.

Erfahren Sie mehr über die AWS CLI

Weitere Informationen zu den AWS finden Sie in den folgenden Ressourcen:

- [AWS Command Line Interface](#)
- [AWS Command Line Interface Benutzerhandbuch für Version 2](#)

Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs

AWS Software Development Kits (SDKs) sind für viele gängige Programmiersprachen verfügbar. Jedes SDK bietet eine API, Codebeispiele und Dokumentation, die es Entwicklern erleichtern, Anwendungen in ihrer bevorzugten Sprache zu erstellen.

Note

Sie können es AWS Amplify für die komplette Fullstack-Entwicklung von Web- und mobilen Apps verwenden. Amplify Storage integriert Dateispeicher- und Verwaltungsfunktionen nahtlos in Frontend-Web- und mobile Apps, die auf Amazon S3 aufbauen. Weitere Informationen finden Sie unter [Storage](#) im Amplify-Benutzerhandbuch.

SDK-Dokumentation	Codebeispiele
AWS SDK für C++	AWS SDK für C++ Code-Beispiele
AWS CLI	AWS CLI Code-Beispiele
AWS SDK für Go	AWS SDK für Go Code-Beispiele
AWS SDK für Java	AWS SDK für Java Code-Beispiele

SDK-Dokumentation	Codebeispiele
AWS SDK für JavaScript	AWS SDK für JavaScript Code-Beispiele
AWS SDK für Kotlin	AWS SDK für Kotlin Code-Beispiele
AWS SDK für .NET	AWS SDK für .NET Code-Beispiele
AWS SDK für PHP	AWS SDK für PHP Code-Beispiele
AWS -Tools für PowerShell	Tools für PowerShell Codebeispiele
AWS SDK für Python (Boto3)	AWS SDK für Python (Boto3) Code-Beispiele
AWS SDK für Ruby	AWS SDK für Ruby Code-Beispiele
AWS SDK für Rust	AWS SDK für Rust Code-Beispiele
AWS SDK für SAP ABAP	AWS SDK für SAP ABAP Code-Beispiele
AWS SDK für Swift	AWS SDK für Swift Code-Beispiele

Spezifische Beispiele finden Sie unter [Codebeispiele für Amazon S3 mit AWS SDKs](#).

SDK-Programmierschnittstellen

Jedes AWS SDK bietet eine oder mehrere programmatische Schnittstellen für die Arbeit mit Amazon S3. Jedes SDK bietet eine Low-Level-Schnittstelle für Amazon S3 mit Methoden, die API-Operationen sehr ähnlich sind. Einige SDKs bieten High-Level-Schnittstellen für Amazon S3, bei denen es sich um Abstraktionen handelt, die allgemeine Anwendungsfälle vereinfachen sollen.

Wenn Sie beispielsweise einen mehrteiligen Upload mithilfe der API-Operationen auf niedriger Ebene durchführen, verwenden Sie eine Operation, um den Upload zu starten, eine weitere Operation, um Teile hochzuladen, und eine letzte Operation, um den Upload abzuschließen. Bei einem API-Vorgang auf hoher Ebene können Sie alle für den Upload erforderlichen Vorgänge in einem einzigen API-Aufruf ausführen. Beispiele finden Sie unter [Hochladen eines Objekts mithilfe eines mehrteiligen Uploads](#) im Amazon S3-Benutzerhandbuch.

Low-level API-Operationen ermöglichen eine bessere Kontrolle über den Upload. Wir empfehlen, die Low-Level-API-Operationen zu verwenden, wenn Sie Uploads unterbrechen und fortsetzen, die

Teilgrößen während des Uploads variieren oder mit Uploads beginnen müssen, wenn Sie die Größe der Daten nicht im Voraus kennen.

Angabe der Signaturversion bei der Anforderungsauthentifizierung

Amazon S3 unterstützt in den meisten AWS-Regionen Fällen nur AWS Signature Version 4. In einigen älteren Versionen AWS-Regionen unterstützt Amazon S3 sowohl Signature Version 4 als auch Signature Version 2. Signature Version 2 wird jedoch deaktiviert (veraltet). Weitere Informationen zum Ende der Unterstützung für Signature Version 2 finden Sie unter [AWS Signature Version 2 ist für Amazon S3 deaktiviert \(veraltet\)](#).

Eine Liste aller Amazon-S3-Regionen und der von ihnen unterstützten Signature-Versionen finden Sie unter [Regions and Endpoints \(Regionen und Endpunkte\)](#) in der Allgemeinen AWS -Referenz.

Für alle AWS-Regionen verwenden AWS SDKs standardmäßig Signature Version 4, um Anfragen zu authentifizieren. Wenn Sie AWS SDKs verwenden, die vor Mai 2016 veröffentlicht wurden, müssen Sie möglicherweise Signature Version 4 anfordern, wie in der folgenden Tabelle dargestellt.

SDK	Anforderung von Signature Version 4 für die Anforderungs-Authentifizierung
AWS CLI	Für das Standardprofil führen Sie den folgenden Befehl aus: <pre>\$ aws configure set default.s3.signature_version s3v4</pre> Für ein benutzerdefiniertes Profil führen Sie den folgenden Befehl aus: <pre>\$ aws configure set profile.your_profile_name.s3.signature_version s3v4</pre>
Java-SDK	Fügen Sie Ihrem Code Folgendes hinzu: <pre>System.setProperty(SDKGlobalConfiguration.ENABLE_S3_SIGV4_SYSTEM_PROPERTY, "true");</pre> Oder geben Sie in der Befehlszeile Folgendes an:

SDK	Anforderung von Signature Version 4 für die Anforderungs-Authentifizierung
	<pre>-Dcom.amazonaws.services.s3.enableV4</pre>
PHP SDK	Setzen Sie den Parameter <code>signature</code> auf v4, wenn Sie den Amazon S3 Service-Client für PHP SDK v2 erstellen: <pre><?php \$client = S3Client::factory(['region' => 'YOUR-REGION', 'version' => 'latest', 'signature' => 'v4']);</pre> Setzen Sie den Parameter <code>signature_version</code> bei Verwendung des PHP SDK v3 im Rahmen der Erstellung des Amazon S3 Service-Clients auf v4: <pre><?php \$s3 = new Aws\S3\S3Client(['version' => '2006-03-01', 'region' => 'YOUR-REGION', 'signature_version' => 'v4']);</pre>
Python-Boto SDK	Geben Sie Folgendes in der boto-Standardkonfigurationsdatei an: <pre>[s3] use-sigv4 = True</pre>

SDK	Anforderung von Signature Version 4 für die Anforderungs-Authentifizierung
Ruby SDK	Ruby SDK – Version 1: Setzen Sie den <code>:s3_signature_version</code> -Parameter auf <code>:v4</code>, wenn Sie den Client konstruieren: <pre>s3 = AWS::S3::Client.new(:s3_signature_version => :v4)</pre> Ruby SDK – Version 3: Setzen Sie den <code>signature_version</code> -Parameter auf <code>v4</code>, wenn Sie den Client konstruieren: <pre>s3 = Aws::S3::Client.new(signature_version: 'v4')</pre>
.NET SDK	Fügen Sie den folgenden Code hinzu, bevor Sie den Amazon-S3-Client erstellen: <pre>AWSConfigsS3.UseSignatureVersion4 = true;</pre> Oder fügen Sie der Konfigurationsdatei Folgendes hinzu: <pre><appSettings> <add key="AWS.S3.UseSignatureVersion4" value="true" /> </appSettings></pre>

AWS Signature Version 2 ist für Amazon S3 deaktiviert (veraltet)

Signature Version 2 wird unter Amazon S3 deaktiviert (veraltet). Danach akzeptiert Amazon S3 nur noch API-Anforderungen, die mithilfe der Signature Version 4 signiert wurden.

Dieser Abschnitt umfasst Antworten auf häufig gestellte Fragen im Bezug auf das Supportende für Signature Version 2.

Was ist die Signaturversion und was bedeutet es 2/4, Anfragen zu signieren?

Der Signiervorgang mittels Signature Version 2 oder Signature Version 4 authentifiziert Ihre API-Anforderungen in Amazon S3. Durch das Signieren von Anforderungen ist Amazon S3 in der Lage, den Absender der Anforderung zu identifizieren und Ihre Anforderungen vor Angreifern zu schützen.

Weitere Informationen zum Signieren von AWS Anfragen finden Sie unter [Signieren von AWS API-Anfragen](#) im Allgemeine AWS-Referenz.

Welche Aktualisierung nehmen Sie vor?

Derzeit unterstützen wir Amazon S3 API-Anforderungen, die mithilfe von Signature Version 2 und Signature Version 4 signiert wurden. Später akzeptiert Amazon S3 nur noch Anforderungen, die mit Signature Version 4 signiert wurden.

Weitere Informationen zum Signieren von AWS Anfragen finden Sie unter [Änderungen in Signaturversion 4](#) im Allgemeine AWS-Referenz.

Warum nehmen Sie die Aktualisierung vor?

Signature Version 4 bietet mehr Sicherheit, da hierbei im Gegensatz zum geheimen Zugriffsschlüssel ein Signaturschlüssel verwendet wird. Signature Version 4 wird derzeit in allen unterstützten AWS-Regionen, wohingegen Signature Version 2 nur in Regionen unterstützt wird, die vor Januar 2014 eingeführt wurden. Dank dieser Aktualisierung sind wir in der Lage, in allen Regionen eine einheitlichere Erfahrung bereitzustellen.

Wie kann ich sicherstellen, dass ich Signature Version 4 verwende und welche Aktualisierungen benötige ich?

Normalerweise entscheidet das Tool oder das SDK auf Clientseite, welche Signature Version zum Signieren der Anforderungen verwendet wird. Standardmäßig verwenden die neuesten Versionen unserer AWS SDKs Signature Version 4. Setzen Sie sich mit dem entsprechenden Supportteam für Ihre Drittanbietersoftware in Verbindung, um zu erfahren, welche Version Sie benötigen. Um direkte REST-Aufrufe an Amazon S3 zu senden, müssen Sie Ihre Anwendung so anpassen, dass der Signiervorgang mithilfe von Signature Version 4 ausgeführt wird.

Informationen darüber, welche Version der AWS SDKs bei der Umstellung auf Signature Version 4 verwendet werden soll, finden Sie unter [Von Signature Version# 2 zur Signature Version 4 wechseln](#).

Weitere Informationen zur Verwendung von Signature Version 4 mit der Amazon S3 REST-API finden Sie unter [Authenticating Requests \(Authentifizieren von Anforderungen\) \(AWS Signature Version 4\)](#) in der API-Referenz zu Amazon Simple Storage Service.

Was passiert, wenn ich keine Aktualisierung durchführe?

Anforderungen, mit Signature Version 2 signiert wurden und zu einem späteren Zeitpunkt gesendet werden, können dann nicht mehr mit Amazon S3 authentifiziert werden. Anforderer sehen eine Fehlermeldung, dass die Anforderung mit Signature Version 4 signiert sein muss.

Sollte ich Änderungen vornehmen, selbst wenn ich eine vorsignierte URL verwende, die es erforderlich macht, dass ich für mehr als sieben Tage signiere?

Wenn sie eine vorsignierte URL verwenden, für die Sie für mehr als sieben Tage signieren müssen, ist keine Maßnahme erforderlich. Sie können AWS Signature Version 2 weiterhin verwenden, um die vorsignierte URL zu signieren und zu authentifizieren. Weitere Informationen zur Migration auf Signature Version 4 für ein Szenario mit einer vorsignierten URL finden Sie in der Amazon S3-Dokumentation.

Weitere Infos

- Weitere Informationen zur Verwendung von Signature Version 4 finden Sie unter [Signieren von AWS API-Anfragen](#).
- Sehen Sie sich die Änderungen zwischen Signature Version 2 und Signature Version 4 unter [Änderungen in Signature Version 4](#) an.
- Sehen Sie sich den Beitrag [AWS Signature Version 4 an, um AWS Signature Version 2 für das Signieren von Amazon S3-API-Anfragen](#) in den AWS Foren zu ersetzen.
- Wenden Sie sich bei weiteren Fragen oder Bedenken unter [Support](#) an uns.

Von Signature Version# 2 zur Signature Version 4 wechseln.

Wenn Sie aktuell Signature Version 2 für die Authentifizierung der API-Anforderungen in Amazon S3 verwenden, sollten Sie möglichst bald auf Signature Version 4 umwechseln. Das Ende der Unterstützung für Signature Version 2 wird unter beschriebene [AWS Signature Version 2 ist für Amazon S3 deaktiviert \(veraltet\)](#).

Weitere Informationen zur Verwendung von Signature Version 4 mit der Amazon S3 REST-API finden Sie unter [Authenticating Requests \(Authentifizieren von Anforderungen\) \(AWS Signature Version 4\)](#) in der API-Referenz zu Amazon Simple Storage Service.

Die folgende Liste stellt die SDKs mit der erforderlichen Mindestversion zur Verwendung von Signature Version 4 (SigV4) dar. Wenn Sie vorsignierte URLs mit den AWS Java-, JavaScript

(Node.js) - oder Python (Boto/CLI) -SDKs verwenden, müssen Sie die richtige AWS-Region und Signature Version 4 in der Client-Konfiguration festlegen. Weitere Informationen zum Einrichten von SigV4 in der Client-Konfiguration finden Sie unter [Angabe der Signaturversion bei der Anforderungsauthentifizierung](#).

Wenn Sie das verwenden SDK/Product	Aktualisieren Sie auf diese SDK-Version	Bei Verwendung von Sigv4 Codeänderung am Client erforderlich?	Link zur SDK-Dokumentation
AWS SDK für Java v1	Upgrade auf Java 1.11.201+ oder v2.	Ja	Angabe der Signaturversion bei der Anforderungsauthentifizierung
AWS SDK für Java v2	Keine SDK-Aktualisierung erforderlich.	Nein	AWS SDK für Java
AWS SDK für .NET v1	Auf 3.1.10 oder höher aktualisieren.	Ja	AWS SDK für .NET
AWS SDK für .NET v2	Auf 3.1.10 oder höher aktualisieren.	Nein	AWS SDK für .NET v2
AWS SDK für .NET v3	Auf 3.3.0.0 oder höher aktualisieren.	Ja	AWS SDK für .NET v3
AWS SDK für JavaScript v1	Auf 2.68.0 oder höher aktualisieren.	Ja	AWS SDK für JavaScript

Wenn Sie das verwenden SDK/Product	Aktualisieren Sie auf diese SDK-Version	Bei Verwendung von Sigv4 Codeänderung am Client erforderlich?	Link zur SDK-Dokumentation
AWS SDK für JavaScript v2	Auf 2.68.0 oder höher aktualisieren.	Ja	AWS SDK für JavaScript
AWS SDK für JavaScript v3	Derzeit ist keine weitere Maßnahme erforderlich. Im Q3 2019 auf Hauptversion V3 aktualisieren.	Nein	AWS SDK für JavaScript
AWS SDK für PHP v1	Empfohlen wird ein Upgrade auf die aktuelle Version von PHP oder mindestens v2.7.4 und der Wert v4 für den Signature-Parameter in der Konfiguration des S3-Clients.	Ja	AWS SDK für PHP

Wenn Sie das verwenden SDK/Product	Aktualisieren Sie auf diese SDK-Version	Bei Verwendung von Sigv4 Codeänderung am Client erforderlich?	Link zur SDK-Dokumentation
AWS SDK für PHP v2	Empfohlen wird ein Upgrade auf die aktuelle Version von PHP oder mindestens v2.7.4 und der Wert v4 für den Signature-Parameter in der Konfiguration des S3-Clients.	Nein	AWS SDK für PHP
AWS SDK für PHP v3	Keine SDK-Aktualisierung erforderlich.	Nein	AWS SDK für PHP
Boto2	Auf Boto2 v2.49.0. aktualisieren.	Ja	Boto 2-Aktualisierung
Boto3	Auf 1.5.71 (Botocore), 1.4.6 (Boto3) aktualisieren.	Ja	Boto 3 — AWS SDK für Python

Wenn Sie das verwenden SDK/Product	Aktualisieren Sie auf diese SDK-Version	Bei Verwendung von Sigv4 Codeänderung am Client erforderlich?	Link zur SDK-Dokumentation
AWS CLI	Auf 1.11.108 aktualisieren.	Ja	AWS Command Line Interface
AWS CLI v2 (Vorschau)	Keine SDK-Aktualisierung erforderlich.	Nein	AWS Command Line Interface Version 2
AWS SDK für Ruby v1	Auf Ruby V3 aktualisieren.	Ja	Ruby V3 für AWS
AWS SDK für Ruby v2	Auf Ruby V3 aktualisieren.	Ja	Ruby V3 für AWS
AWS SDK für Ruby v3	Keine SDK-Aktualisierung erforderlich.	Nein	Ruby V3 für AWS
Go	Keine SDK-Aktualisierung erforderlich.	Nein	AWS SDK für Go
C++	Keine SDK-Aktualisierung erforderlich.	Nein	AWS SDK für C++

AWS Tools for Windows PowerShell oder AWS Tools for PowerShell Core

Wenn Sie Modulversionen verwenden, die älter als 3.3.0.0 sind, müssen Sie auf 3.3.0.0 aktualisieren.

Verwenden Sie `Get-Module` cmdlet, um die Versionsinformationen zu erhalten:

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

Um auf die Version 3.3.0.0 zu aktualisieren, verwenden Sie das cmdlet `Update-Module`:

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

Für Signature Version 2-Datenverkehr können Sie vorab signierte URLs verwenden, die länger als sieben Tage gültig sind.

Abrufen von Amazon S3-Anforderungs-IDs für AWS Support

Wann immer Sie Kontakt aufnehmen AWS Support , weil Sie in Amazon S3 auf Fehler oder unerwartetes Verhalten gestoßen sind, müssen Sie die Anforderungs-IDs angeben, die der fehlgeschlagenen Aktion zugeordnet sind. AWS Support verwendet diese Anforderungs-IDs, um die aufgetretenen Probleme zu lösen.

Anfrage-IDs werden paarweise vergeben. Sie werden in jeder von Amazon S3 verarbeiteten Antwort zurückgegeben (auch in den fehlerhaften). Der Zugriff darauf erfolgt über Verbose-Protokolle. Es gibt eine Reihe gängiger Methoden, um Ihre Anforderungs-IDs abzurufen, darunter S3-Serverzugriffsprotokolle und AWS CloudTrail -ereignisse oder Datenereignisse.

Nachdem Sie diese Protokolle wiederhergestellt haben, kopieren Sie diese beiden Werte und behalten Sie sie bei, da Sie sie benötigen, wenn Sie Kontakt aufnehmen AWS Support. Informationen zur Kontaktaufnahme finden Sie AWS Support unter [Kontakt AWS](#) oder in der [AWS Support Dokumentation](#).

Themen

- [Verwendung der AWS CLI um Anfrage-IDs zu erhalten](#)

- [Verwenden von Windows PowerShell zum Abrufen von Anforderungs-IDs](#)
- [Verwenden AWS CloudTrail Datenereignisse zum Abrufen von Anforderungs-IDs](#)
- [Abrufen der Anforderungs-IDs mithilfe der S3-Server-Zugriffsprotokollierung](#)
- [Abrufen der Anforderungs-IDs mithilfe von HTTP](#)
- [Abrufen der Anforderungs-IDs mithilfe eines Webbrowsers](#)
- [Verwendung der AWS SDKs zum Abrufen von Anforderungs-IDs](#)

Verwendung der AWS CLI um Anfrage-IDs zu erhalten

Um Ihre Anforderungs-IDs zu erhalten, wenn Sie die AWS Command Line Interface (AWS CLI) verwenden, fügen `--debug` Sie Ihrem Befehl hinzu. Zum Beispiel solltest du `x-amz-request-id` und `x-amz-id-2` im Debug-Log wie unten gezeigt sehen:

```
...
2025-04-30 14:35:28,572 - MainThread - botocore.parsers - DEBUG - Response headers:
{'x-amz-id-2': 'a+zm50vDJH3LLmCiXvwEo0u0PtPS/qCJaBvB2ZMH9dzyzTiJhiLZkBFFoRfsPfOKztUKT/
garCI=', 'x-amz-request-id': 'N4NMN0MJ4VDFZMX9', 'Date': 'Wed, 30 Apr 2025 21:35:29
GMT', 'Content-Type': 'application/xml', 'Transfer-Encoding': 'chunked', 'Server':
'AmazonS3'}
...
```

Verwenden von Windows PowerShell zum Abrufen von Anforderungs-IDs

Informationen zum Wiederherstellen von Protokollen mit Windows PowerShell finden Sie unter [Response Logging in AWS Tools for Windows PowerShell](#) im AWS Developer Tools-Blog.

Verwenden AWS CloudTrail Datenereignisse zum Abrufen von Anforderungs-IDs

Ein Amazon S3-Bucket, der mit CloudTrail Datenereignissen konfiguriert ist, um API-Operationen auf S3-Objektebene zu protokollieren, bietet detaillierte Informationen über Aktionen, die von einem Benutzer, einer Rolle oder einem AWS Dienst in Amazon S3 ausgeführt werden. Sie können S3-Anforderungs-IDs identifizieren, indem Sie CloudTrail Ereignisse mit Athena abfragen.

Weitere Informationen finden Sie unter [Identifizieren von Amazon S3-Anfragen mithilfe CloudTrail](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Abrufen der Anforderungs-IDs mithilfe der S3-Server-Zugriffsprotokollierung

Ein Amazon-S3-Bucket, der für die S3-Server-Zugriffsprotokollierung konfiguriert ist, stellt detaillierte Aufzeichnungen zu allen an den Bucket gesendeten Anforderungen bereit. Sie können S3-Anforderungs-IDs identifizieren, indem Sie die Serverzugriffsprotokolle mit Athena abfragen.

Weitere Informationen finden Sie unter [Abfragen von Zugriffsprotokollen für Anfragen mithilfe Amazon Athena](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Abrufen der Anforderungs-IDs mithilfe von HTTP

Sie können Ihre Anfrage-IDs `x-amz-request-id` und `x-amz-id-2` ermitteln, indem Sie die Abschnitte einer HTTP-Anfrage protokollieren, bevor sie die Zielanwendung erreicht. Es gibt verschiedene Tools von Drittanbietern, mit denen Verbose-Protokolle für HTTP-Anfragen wiederhergestellt werden können. Wählen Sie ein Tool, dem Sie vertrauen, führen Sie es aus und überwachen Sie den Port, über den Ihr Amazon-S3-Datenverkehr läuft, während Sie eine weitere Amazon-S3-HTTP-Anforderung senden.

Bei HTTP-Anforderungen sieht das Paar der Anforderungs-IDs wie in den folgenden Beispielen gezeigt aus:

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

Note

HTTPS-Anfragen werden in den meisten Paketerfassungen verschlüsselt und verborgen.

Abrufen der Anforderungs-IDs mithilfe eines Webbrowsers

Die meisten Webbrowser beinhalten Entwickler-Tools, mit denen Sie Anforderungs-Header anzeigen können.

Für auf einem Webbrowser basierte Anfragen, die einen Fehler zurückgeben, sieht das Paar der Anfrage-IDs wie in den folgenden Beispielen gezeigt aus.

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



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

Zeigen Sie die HTTP-Anforderungs-Header mithilfe der Entwickler-Tools zu Ihrem Browser an, um das Paar der Anforderungs-IDs aus erfolgreichen Anforderungen abzurufen.

Verwendung der AWS SDKs zum Abrufen von Anforderungs-IDs

Die folgenden Abschnitte enthalten Informationen zur Konfiguration der Protokollierung mithilfe verschiedener AWS SDKs.

Note

Sie können zwar die ausführliche Protokollierung für jede Anfrage und Antwort aktivieren, empfehlen jedoch nicht, die Protokollierung in Produktionssystemen zu aktivieren, da umfangreiche Anfragen oder Antworten eine Anwendung erheblich verlangsamen können.

Bei AWS SDK-Anfragen sieht das Paar von Anforderungs-IDs wie folgt aus:

```
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++

Der Loggertyp und die Ausführlichkeit werden bei der SDK-Initialisierung im Argument angegeben. `SDKOptions` Im folgenden Beispiel wird die Ausführlichkeitsstufe als angegeben. `LogLevel::Debug`

Der Standard-Logger schreibt in das Dateisystem und die Datei wird nach der folgenden Konvention benannt. `aws_sdk_YYYY-MM-DD-HH.log` Der Logger erstellt zu jeder Stunde eine neue Datei.

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

Weitere Informationen finden Sie unter [Wie aktiviere ich die Protokollierung?](#) im AWS SDK für C++-Wiki auf GitHub.

Go

Sie können die Protokollierung mithilfe von SDK for Go konfigurieren. Weitere Informationen finden Sie [im AWS SDK für Go v2-Entwicklerhandbuch unter Logging](#).

Java

Sie können die Protokollierung für bestimmte Anfragen oder Antworten aktivieren, um nur relevante Header abzufangen und zurückzugeben. Dazu importieren Sie die Klasse `com.amazonaws.services.s3.S3ResponseMetadata`. Anschließend können Sie die Anforderung in einer Variablen speichern, bevor Sie die eigentliche Anforderung ausführen. Rufen Sie `getCachedResponseMetadata(AmazonWebServiceRequest request).getRequestID()` auf, um die protokollierte Anforderung oder Antwort zu erhalten.

```
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());
```

Alternativ können Sie eine Verbose-Protokollierung jeder Java-Anfrage und -Antwort verwenden. Weitere Informationen finden Sie unter [Verbose-Protokollierung des Netzwerkverkehrs](#) im Entwicklerhandbuch zu AWS SDK für Java .

JavaScript

Der AWS SDK für JavaScript hat einen eingebauten Logger, sodass Sie API-Aufrufe, die Sie damit tätigen, protokollieren können. Um den Logger einzuschalten und Logeinträge in der Konsole zu drucken, fügen Sie Ihrem Code die folgende Anweisung hinzu:

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

Weitere Informationen finden Sie [im AWS SDK für JavaScript Entwicklerhandbuch unter Logging AWS SDK für JavaScript Calls](#).

Kotlin

Mit dem AWS SDK für Kotlin können Sie den Protokollmodus für Nachrichten auf Wire-Level-Ebene mithilfe von Code- oder Umgebungseinstellungen angeben. Sie können den Protokollmodus für HTTP-Anfragen und HTTP-Antworten festlegen.

Um zusätzliche Protokollierung zu aktivieren, legen Sie die `logMode` Eigenschaft fest, wenn Sie einen Service-Client erstellen:

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

// ...

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

Alternativ können Sie den Protokollmodus mithilfe einer Umgebungsvariablen festlegen:

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

Weitere Informationen finden Sie [im AWS SDK für Kotlin Entwicklerhandbuch unter Logging](#).

.NET

Sie können die Protokollierung mit dem konfigurieren, SDK für .NET indem Sie das integrierte `System.Diagnostics` Protokollierungstool verwenden. Weitere Informationen finden Sie im Blogbeitrag [Logging with the SDK für .NET](#) AWS Developer.

Note

Standardmäßig enthält das zurückgegebene Protokoll nur Fehlerinformationen. Zum Abrufen der Anforderungs-IDs muss der Konfigurationsdatei `AWSLogMetrics` (und optional `AWSResponseLogging`) hinzugefügt werden.

PHP

Sie können Debug-Informationen, einschließlich der über die Leitung gesendeten Daten, abrufen, indem Sie die Debug-Option `true` in einem Client-Konstruktor auf setzen.

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

```
]);
```

Weitere Informationen finden Sie unter [Wie kann ich sehen, welche Daten übertragen wurden?](#) im Entwicklerhandbuch zum AWS SDK für PHP .

Python (Boto3)

Mit dem AWS SDK für Python (Boto3) können Sie bestimmte Antworten protokollieren. Sie können diese Funktion verwenden, um nur die relevanten Header zu erfassen. Der folgende Code zeigt, wie Teile der Antwort in einer Datei protokolliert werden:

```
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'])
```

Sie können auch Ausnahmen abfangen und relevante Informationen protokollieren, wenn eine Ausnahme ausgelöst wird. Weitere Informationen finden Sie unter [Erkennen nützlicher Informationen in Fehlerantworten](#) in der API-Referenz zum AWS SDK für Python (Boto).

Darüber hinaus können Sie Boto3 mit dem folgenden Code für die Ausgabe ausführlicher Debugging-Protokolle konfigurieren:

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

Weitere Informationen finden Sie unter [set_stream_logger](#) in der API-Referenz zum AWS SDK für Python (Boto).

Ruby

Sie können Ihre Anforderungs-IDs mit dem SDK für Ruby Version 3 abrufen. Sie können das HTTP-Wire-Logging in Ihrem Client mit dem folgenden Code aktivieren:

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

 Note

Die Drahtprotokollierung kann vertrauliche Informationen enthalten, wie z. B. Ihre Zugangsschlüssel-ID. Vertrauliche Informationen sollten bereinigt werden, bevor sie an den Support weitergegeben AWS werden.

Die Anforderungs-ID des Anforderungskontextobjekts finden Sie auch in der Anforderungsantwort oder im Fehler:

```
# 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]
```

Weitere Informationen finden Sie im AWS SDK für Ruby Entwicklerhandbuch unter [Debuggen mithilfe von Wire-Trace-Informationen von einem AWS SDK für Ruby Client](#).

Rust

Um die Protokollierung zu aktivieren, fügen Sie die `tracing-subscriber` Crate hinzu und initialisieren Sie sie in Ihrer Rust-Anwendung.

Fügen Sie die Tracing-Bibliothek zu Ihrer Datei hinzu: `Cargo.toml`

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

Initialisieren Sie dann in Ihrem Rust-Code den Logger in der Hauptfunktion, bevor Sie eine SDK-Operation aufrufen:

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

Weitere Informationen finden Sie unter [Konfiguration und Verwendung der Protokollierung AWS SDK für Rust](#) im AWS SDK für Rust Entwicklerhandbuch.

Unterstützte Amazon S3-API-Operationen auf Objektebene für S3-Tabellen

Die folgende Tabelle enthält unterstützte S3-API-Operationen auf Objektebene und die entsprechenden Header für S3-Tabellen. Eine Liste der Amazon S3-Tabellen-APIs finden Sie unter [Amazon S3-Tabellen](#). Weitere Informationen zu Amazon S3-Tabellen finden Sie unter [Arbeiten mit Amazon S3-Tabellen und Tabellen-Buckets](#) im Amazon S3-Benutzerhandbuch.

Object-level API	Unterstützte Header	Hinweise	
AbortMultipartUpload	x-amz-expected-bucket-owner	Keine	
CompleteMultipartUpload	x-amz-checksum-crc32	Keine	
	x-amz-checksum-crc32c	Keine	
	x-amz-checksum-sha1	Keine	
	x-amz-checksum-sha256	Keine	
	x-amz-expected-bucket-owner	Keine	
	If-Match	Keine	
	If-None-Match	Keine	

Object-level API	Unterstützte Header	Hinweise	
CreateMultipartUpload	x-amz-acl: ACL	Für S3-Tabellen lautet der Standardwert <code>bucket-owner-full-control</code> und er kann nicht geändert werden.	
	Cache-Control	Keine	
	Content-Disposition	Keine	
	Content-Encoding	Keine	
	Content-Language	Keine	
	Content-Type	Keine	
	Expires	Keine	
	x-amz-checksum-algorithm	Keine	
	x-amz-storage-class	Für S3-Tabellen lautet der Standardwert <code>STANDARD</code> und er kann nicht geändert werden.	
	x-amz-server-side-encryption	Für S3-Tabellen ist der Standardwert (SSE-S3) (AES256) und er kann nicht geändert werden.	
GetObject	If-Match	Keine	
	If-Modified-Since	Keine	

Object-level API	Unterstützte Header	Hinweise	
	If-None-Match	Keine	
	If-Unmodified-Since	Keine	
	Range	Keine	
	x-amz-expected-bucket-owner	Keine	
	x-amz-checksum-mode	Keine	
HeadObject	If-Match	Keine	
	If-Modified-Since	Keine	
	If-None-Match	Keine	
	If-Unmodified-Since	Keine	
	Range	Keine	
	x-amz-checksum-mode	Keine	
	x-amz-expected-bucket-owner	Keine	
ListParts	x-amz-expected-bucket-owner	Keine	

Object-level API	Unterstützte Header	Hinweise	
PutObject	x-amz-acl: ACL	Für S3-Tabellen ist der Standardwert <code>bucket-owner-full-control</code> und er kann nicht geändert werden.	
	Cache-Control	Keine	
	Content-Disposition	Keine	
	Content-Encoding	Keine	
	Content-Language	Keine	
	Content-Length	Keine	
	Content-MD5	Keine	
	Content-Type	Keine	
	x-amz-sdk-checksum-algorithm	Keine	
	x-amz-checksum-crc32	Keine	
	x-amz-checksum-crc32c	Keine	
	x-amz-checksum-sha1	Keine	
	x-amz-checksum-sha256	Keine	

Object-level API	Unterstützte Header	Hinweise	
	Expires	Keine	
	If-None-Match	Keine	
	x-amz-expected-bucket-owner	Keine	
	x-amz-storage-class	Für S3-Tabellen lautet der Standardwert STANDARD und er kann nicht geändert werden.	
	x-amz-server-side-encryption	Für S3-Tabellen ist der Standardwert (SSE-S3) (AES256) und er kann nicht geändert werden.	
UploadPart	Content-Length	Keine	
	Content-MD5	Keine	
	x-amz-checksum-crc32	Keine	
	x-amz-checksum-crc32c	Keine	
	x-amz-checksum-sha1	Keine	
	x-amz-checksum-sha256	Keine	
	x-amz-expected-bucket-owner	Keine	

Codebeispiele für Amazon S3 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Amazon S3 mit einem AWS Software Development Kit (SDK) verwendet wird.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Codebeispiele

- [Codebeispiele für Amazon S3 mit AWS SDKs](#)
 - [Grundlegende Beispiele für die Verwendung von Amazon S3 AWS SDKs](#)
 - [Hello Amazon S3](#)
 - [Lernen Sie die Grundlagen von Amazon S3 mit einem AWS SDK](#)
 - [Aktionen für Amazon S3 mit AWS SDKs](#)
 - [Verwenden Sie AbortMultipartUpload mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CompleteMultipartUpload mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CopyObject mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CreateBucket mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CreateMultiRegionAccessPoint mit einem AWS SDK](#)
 - [Verwenden Sie CreateMultipartUpload mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CreatePresignedPost mit einem AWS SDK](#)
 - [Verwenden Sie DeleteBucket mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketAnalyticsConfiguration mit einer CLI](#)
 - [Verwenden Sie DeleteBucketCors mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketEncryption mit einer CLI](#)
 - [Verwendung von DeleteBucketInventoryConfiguration mit einer CLI](#)
 - [Verwenden Sie DeleteBucketLifecycle mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketMetricsConfiguration mit einer CLI](#)
 - [Verwenden Sie DeleteBucketPolicy mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketReplication mit einer CLI](#)
 - [Verwendung von DeleteBucketTagging mit einer CLI](#)

- [Verwenden Sie DeleteBucketWebsite mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DeleteObject mit einem AWS SDK oder CLI](#)
- [Verwendung von DeleteObjectTagging mit einer CLI](#)
- [Verwenden Sie DeleteObjects mit einem AWS SDK oder CLI](#)
- [Verwendung von DeletePublicAccessBlock mit einer CLI](#)
- [Verwendung von GetBucketAccelerateConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketAcl mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketAnalyticsConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketCors mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetBucketEncryption mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketInventoryConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketLifecycleConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetBucketLocation mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketLogging mit einer CLI](#)
- [Verwendung von GetBucketMetricsConfiguration mit einer CLI](#)
- [Verwendung von GetBucketNotification mit einer CLI](#)
- [Verwenden Sie GetBucketPolicy mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketPolicyStatus mit einer CLI](#)
- [Verwenden Sie GetBucketReplication mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketRequestPayment mit einer CLI](#)
- [Verwendung von GetBucketTagging mit einer CLI](#)
- [Verwendung von GetBucketVersioning mit einer CLI](#)
- [Verwenden Sie GetBucketWebsite mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObject mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectAcl mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectAttributes mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectLegalHold mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectLockConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectRetention mit einem AWS SDK oder CLI](#)
- [Verwendung von GetObjectTagging mit einer CLI](#)

- [Verwendung von GetPublicAccessBlock mit einer CLI](#)
- [Verwenden Sie HeadBucket mit einem AWS SDK oder CLI](#)
- [Verwenden Sie HeadObject mit einem AWS SDK oder CLI](#)
- [Verwendung von ListBucketAnalyticsConfigurations mit einer CLI](#)
- [Verwendung von ListBucketInventoryConfigurations mit einer CLI](#)
- [Verwenden Sie ListBuckets mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ListMultipartUploads mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ListObjectVersions mit einem AWS SDK oder CLI](#)
- [Verwendung von ListObjects mit einer CLI](#)
- [Verwenden Sie ListObjectsV2 mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketAccelerateConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketAcl mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketCors mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketEncryption mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketLifecycleConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketLogging mit einem AWS SDK oder CLI](#)
- [Verwendung von PutBucketNotification mit einer CLI](#)
- [Verwenden Sie PutBucketNotificationConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketPolicymit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketReplication mit einem AWS SDK oder CLI](#)
- [Verwendung von PutBucketRequestPayment mit einer CLI](#)
- [Verwendung von PutBucketTagging mit einer CLI](#)
- [Verwenden Sie PutBucketVersioning mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketWebsite mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObject mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectAcl mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectLegalHold mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectLockConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectRetention mit einem AWS SDK oder CLI](#)
- [Verwenden Sie RestoreObject mit einem AWS SDK oder CLI](#)

- [Verwenden Sie SelectObjectContent mit einem AWS SDK oder CLI](#)
- [Verwenden Sie UploadPart mit einem AWS SDK oder CLI](#)
- [Verwenden Sie UploadPartCopy mit einem AWS SDK oder CLI](#)
- [Szenarien für die Verwendung von Amazon S3 AWS SDKs](#)
 - [Prüfen, ob ein Bucket vorhanden ist](#)
 - [Prüfen Sie, ob ein Objekt existiert](#)
 - [Konvertieren Sie Text in Sprache und zurück in Text mithilfe eines AWS SDK](#)
 - [Erstellen Sie eine vorsignierte URL für Amazon S3 mit einem AWS SDK](#)
 - [Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können](#)
 - [Eine Webseite, die Amazon S3 S3-Objekte auflistet, und zwar mithilfe eines AWS SDK](#)
 - [Erstellen Sie eine Amazon-Textextract-Explorer-Anwendung](#)
 - [Löschen Sie alle Objekte in einem bestimmten Amazon S3 S3-Bucket mit einem AWS SDK](#)
 - [Löschen Sie unvollständige mehrteilige Uploads auf Amazon S3 mit einem AWS SDK](#)
 - [Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK](#)
 - [Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mithilfe eines AWS SDK](#)
 - [Erkennen Sie Gesichter in einem Bild mit einem AWS SDK](#)
 - [Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK](#)
 - [Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK](#)
 - [Herunterladen von S3-Verzeichnissen aus einem Bucket in Amazon Simple Storage Service \(Amazon S3\)](#)
 - [Herunterladen aller Objekte aus einem Amazon Simple Storage Service \(Amazon S3\)-Bucket in ein lokales Verzeichnis](#)
 - [Laden Sie einen Stream unbekannter Größe von einem Amazon S3 S3-Objekt herunter, indem Sie eine AWS SDK](#)
 - [Rufen Sie ein Amazon S3 S3-Objekt von einem Multi-Region Access Point ab, indem Sie eine AWS SDK](#)
 - [Holen Sie sich ein Objekt aus einem Amazon S3 S3-Bucket mit einem AWS SDK, mit Angabe eines If-Modified-Since Headers](#)

- [Erste Schritte mit der Verschlüsselung von Amazon S3 S3-Objekten mithilfe eines AWS SDK](#)
- [Erste Schritte mit Tags für Amazon S3 S3-Objekte mithilfe eines AWS SDK](#)
- [Erste Schritte mit Clustern zur Verarbeitung großer Datenmengen](#)
- [Erste Schritte mit der Konfigurationsverwaltung](#)
- [Erste Schritte mit der Textextraktion von Dokumenten](#)
- [Erste Schritte mit Feature-Stores für maschinelles Lernen](#)
- [Erste Schritte mit Objektspeicher](#)
- [Erste Schritte mit Query Analytics](#)
- [Arbeiten Sie mit den Objektsperreffunktionen von Amazon S3 mithilfe eines AWS SDK](#)
- [Stellen Sie bedingte Amazon S3 S3-Anfragen mit einem AWS SDK](#)
- [Verwalten Sie Zugriffskontrolllisten \(ACLs\) für Amazon S3 S3-Buckets mithilfe eines AWS SDK](#)
- [Verwalten Sie große Amazon SQS SQS-Nachrichten mithilfe von Amazon S3 mit einem AWS SDK](#)
- [Verwalten Sie versionierte Amazon S3 S3-Objekte stapelweise mit einer Lambda-Funktion mithilfe eines AWS SDK](#)
- [Analysieren Sie Amazon S3 S3-URLs mit einem AWS SDK](#)
- [Erstellen Sie eine mehrteilige Kopie eines Amazon S3 S3-Objekts mit einem AWS SDK](#)
- [Empfangen und verarbeiten Sie Amazon S3 S3-Ereignisbenachrichtigungen mithilfe eines AWS SDK](#)
- [Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK](#)
- [Senden Sie S3-Ereignisbenachrichtigungen an Amazon EventBridge mit einem AWS SDK](#)
- [Verfolgen Sie den Upload oder Download eines Amazon S3 S3-Objekts mithilfe eines AWS SDK](#)
- [Transformieren von Daten für Ihre Anwendung mit S3 Object Lambda](#)
- [Beispielansätze für Einheiten- und Integrationstests mit einem AWS SDK](#)
- [Rekursives Hochladen eines lokalen Verzeichnisses in einen Amazon Simple Storage Service \(Amazon S3\)-Bucket](#)
- [Laden Sie große Dateien zu und von Amazon S3 hoch oder laden Sie sie herunter mit einem AWS SDK](#)

- [Laden Sie einen Stream unbekannter Größe in ein Amazon S3 S3-Objekt hoch, indem Sie eine AWS SDK](#)
- [Verwenden Sie Prüfsummen, um mit einem Amazon S3 S3-Objekt zu arbeiten, indem Sie AWS SDK](#)
- [Arbeiten Sie mit den Objektintegritätsfunktionen von Amazon S3 mithilfe eines AWS SDK](#)
- [Arbeiten Sie mit Objekten mit Amazon S3 S3-Version unter Verwendung eines AWS SDK](#)
- [Serverless-Beispiele für Amazon S3](#)
 - [Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser](#)
- [Codebeispiele für Amazon S3 Control mit AWS SDKs](#)
- [Grundlegende Beispiele für die Verwendung von Amazon S3 Control AWS SDKs](#)
 - [Hello Amazon S3 Control](#)
 - [Lernen Sie die Grundlagen von Amazon S3 Control mit einem AWS SDK](#)
 - [Aktionen für Amazon S3 Control mit AWS SDKs](#)
 - [Verwenden Sie CreateJobmit einem AWS SDK oder CLI](#)
 - [Verwenden Sie DeleteJobTaggingmit einem AWS SDK](#)
 - [Verwenden Sie DescribeJobmit einem AWS SDK oder CLI](#)
 - [Verwenden Sie GetJobTaggingmit einem AWS SDK](#)
 - [Verwenden Sie PutJobTaggingmit einem AWS SDK](#)
 - [Verwenden Sie UpdateJobPrioritymit einem AWS SDK oder CLI](#)
 - [Verwenden Sie UpdateJobStatusmit einem AWS SDK oder CLI](#)
- [Codebeispiele für S3-Verzeichnis-Buckets mit AWS SDKs](#)
- [Grundlegende Beispiele für S3-Verzeichnis-Buckets mit AWS SDKs](#)
 - [Hello Amazon-S3-Verzeichnis-Buckets](#)
 - [Lernen Sie die Grundlagen von S3 Directory Buckets mit einem AWS SDK](#)
 - [Aktionen für S3-Verzeichnis-Buckets mit AWS SDKs](#)
 - [Verwenden Sie AbortMultipartUploadmit einem AWS SDK](#)
 - [Verwenden Sie CompleteMultipartUploadmit einem AWS SDK](#)
 - [Verwenden Sie CopyObjectmit einem AWS SDK](#)
 - [Verwenden Sie CreateBucketmit einem AWS SDK](#)
 - [Verwenden Sie CreateMultipartUploadmit einem AWS SDK](#)
 - [Verwenden Sie CreateSessionmit einem AWS SDK](#)

- [Verwenden Sie DeleteBucket mit einem AWS SDK](#)
- [Verwenden Sie DeleteBucketEncryption mit einem AWS SDK](#)
- [Verwenden Sie DeleteBucketPolicy mit einem AWS SDK](#)
- [Verwenden Sie DeleteObject mit einem AWS SDK](#)
- [Verwenden Sie DeleteObjects mit einem AWS SDK](#)
- [Verwenden Sie GetBucketEncryption mit einem AWS SDK](#)
- [Verwenden Sie GetBucketPolicy mit einem AWS SDK](#)
- [Verwenden Sie GetObject mit einem AWS SDK](#)
- [Verwenden Sie GetObjectAttributes mit einem AWS SDK](#)
- [Verwenden Sie HeadBucket mit einem AWS SDK](#)
- [Verwenden Sie HeadObject mit einem AWS SDK](#)
- [Verwenden Sie ListDirectoryBuckets mit einem AWS SDK](#)
- [Verwenden Sie ListMultipartUploads mit einem AWS SDK](#)
- [Verwenden Sie ListObjectsV2 mit einem AWS SDK](#)
- [Verwenden Sie ListParts mit einem AWS SDK](#)
- [Verwenden Sie PutBucketEncryption mit einem AWS SDK](#)
- [Verwenden Sie PutBucketPolicy mit einem AWS SDK](#)
- [Verwenden Sie PutObject mit einem AWS SDK](#)
- [Verwenden Sie UploadPart mit einem AWS SDK](#)
- [Verwenden Sie UploadPartCopy mit einem AWS SDK](#)
- [Szenarien für S3-Verzeichnis-Buckets mit AWS SDKs](#)
 - [Erstellen Sie eine vorsignierte URL für Amazon S3 S3-Verzeichnis-Buckets, um ein Objekt mithilfe eines AWS SDK](#)

Codebeispiele für Amazon S3 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Amazon S3 mit einem AWS Software Development Kit (SDK) verwendet wird.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Amazon S3

- [Grundlegende Beispiele für die Verwendung von Amazon S3 AWS SDKs](#)
 - [Hello Amazon S3](#)
 - [Lernen Sie die Grundlagen von Amazon S3 mit einem AWS SDK](#)
 - [Aktionen für Amazon S3 mit AWS SDKs](#)
 - [Verwenden Sie AbortMultipartUpload mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CompleteMultipartUpload mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CopyObject mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CreateBucket mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CreateMultiRegionAccessPoint mit einem AWS SDK](#)
 - [Verwenden Sie CreateMultipartUpload mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CreatePresignedPost mit einem AWS SDK](#)
 - [Verwenden Sie DeleteBucket mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketAnalyticsConfiguration mit einer CLI](#)
 - [Verwenden Sie DeleteBucketCors mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketEncryption mit einer CLI](#)
 - [Verwendung von DeleteBucketInventoryConfiguration mit einer CLI](#)
 - [Verwenden Sie DeleteBucketLifecycle mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketMetricsConfiguration mit einer CLI](#)
 - [Verwenden Sie DeleteBucketPolicy mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketReplication mit einer CLI](#)
 - [Verwendung von DeleteBucketTagging mit einer CLI](#)

- [Verwenden Sie DeleteBucketWebsite mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DeleteObject mit einem AWS SDK oder CLI](#)
- [Verwendung von DeleteObjectTagging mit einer CLI](#)
- [Verwenden Sie DeleteObjects mit einem AWS SDK oder CLI](#)
- [Verwendung von DeletePublicAccessBlock mit einer CLI](#)
- [Verwendung von GetBucketAccelerateConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketAcl mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketAnalyticsConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketCors mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetBucketEncryption mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketInventoryConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketLifecycleConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetBucketLocation mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketLogging mit einer CLI](#)
- [Verwendung von GetBucketMetricsConfiguration mit einer CLI](#)
- [Verwendung von GetBucketNotification mit einer CLI](#)
- [Verwenden Sie GetBucketPolicy mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketPolicyStatus mit einer CLI](#)
- [Verwenden Sie GetBucketReplication mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketRequestPayment mit einer CLI](#)
- [Verwendung von GetBucketTagging mit einer CLI](#)
- [Verwendung von GetBucketVersioning mit einer CLI](#)
- [Verwenden Sie GetBucketWebsite mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObject mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectAcl mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectAttributes mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectLegalHold mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectLockConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectRetention mit einem AWS SDK oder CLI](#)
- [Verwendung von GetObjectTagging mit einer CLI](#)

- [Verwendung von GetPublicAccessBlock mit einer CLI](#)
- [Verwenden Sie HeadBucket mit einem AWS SDK oder CLI](#)
- [Verwenden Sie HeadObject mit einem AWS SDK oder CLI](#)
- [Verwendung von ListBucketAnalyticsConfigurations mit einer CLI](#)
- [Verwendung von ListBucketInventoryConfigurations mit einer CLI](#)
- [Verwenden Sie ListBuckets mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ListMultipartUploads mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ListObjectVersions mit einem AWS SDK oder CLI](#)
- [Verwendung von ListObjects mit einer CLI](#)
- [Verwenden Sie ListObjectsV2 mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketAccelerateConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketAcl mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketCors mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketEncryption mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketLifecycleConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketLogging mit einem AWS SDK oder CLI](#)
- [Verwendung von PutBucketNotification mit einer CLI](#)
- [Verwenden Sie PutBucketNotificationConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketPolicymit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketReplication mit einem AWS SDK oder CLI](#)
- [Verwendung von PutBucketRequestPayment mit einer CLI](#)
- [Verwendung von PutBucketTagging mit einer CLI](#)
- [Verwenden Sie PutBucketVersioning mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketWebsite mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObject mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectAcl mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectLegalHold mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectLockConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectRetention mit einem AWS SDK oder CLI](#)
- [Verwenden Sie RestoreObject mit einem AWS SDK oder CLI](#)

- [Verwenden Sie SelectObjectContent mit einem AWS SDK oder CLI](#)
- [Verwenden Sie UploadPart mit einem AWS SDK oder CLI](#)
- [Verwenden Sie UploadPartCopy mit einem AWS SDK oder CLI](#)
- [Szenarien für die Verwendung von Amazon S3 AWS SDKs](#)
 - [Prüfen, ob ein Bucket vorhanden ist](#)
 - [Prüfen Sie, ob ein Objekt existiert](#)
 - [Konvertieren Sie Text in Sprache und zurück in Text mithilfe eines AWS SDK](#)
 - [Erstellen Sie eine vorsignierte URL für Amazon S3 mit einem AWS SDK](#)
 - [Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können](#)
 - [Eine Webseite, die Amazon S3 S3-Objekte auflistet, und zwar mithilfe eines AWS SDK](#)
 - [Erstellen Sie eine Amazon-Textextract-Explorer-Anwendung](#)
 - [Löschen Sie alle Objekte in einem bestimmten Amazon S3 S3-Bucket mit einem AWS SDK](#)
 - [Löschen Sie unvollständige mehrteilige Uploads auf Amazon S3 mit einem AWS SDK](#)
 - [Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK](#)
 - [Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mithilfe eines AWS SDK](#)
 - [Erkennen Sie Gesichter in einem Bild mit einem AWS SDK](#)
 - [Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK](#)
 - [Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK](#)
 - [Herunterladen von S3-Verzeichnissen aus einem Bucket in Amazon Simple Storage Service \(Amazon S3\)](#)
 - [Herunterladen aller Objekte aus einem Amazon Simple Storage Service \(Amazon S3\)-Bucket in ein lokales Verzeichnis](#)
 - [Laden Sie einen Stream unbekannter Größe von einem Amazon S3 S3-Objekt herunter, indem Sie eine AWS SDK](#)
 - [Rufen Sie ein Amazon S3 S3-Objekt von einem Multi-Region Access Point ab, indem Sie eine AWS SDK](#)
 - [Holen Sie sich ein Objekt aus einem Amazon S3 S3-Bucket mit einem AWS SDK, mit Angabe eines If-Modified-Since Headers](#)
- [Erste Schritte mit der Verschlüsselung von Amazon S3 S3-Objekten mithilfe eines AWS SDK](#)

- [Erste Schritte mit Tags für Amazon S3 S3-Objekte mithilfe eines AWS SDK](#)
- [Erste Schritte mit Clustern zur Verarbeitung großer Datenmengen](#)
- [Erste Schritte mit der Konfigurationsverwaltung](#)
- [Erste Schritte mit der Textextraktion von Dokumenten](#)
- [Erste Schritte mit Feature-Stores für maschinelles Lernen](#)
- [Erste Schritte mit Objektspeicher](#)
- [Erste Schritte mit Query Analytics](#)
- [Arbeiten Sie mit den Objektsperreffunktionen von Amazon S3 mithilfe eines AWS SDK](#)
- [Stellen Sie bedingte Amazon S3 S3-Anfragen mit einem AWS SDK](#)
- [Verwalten Sie Zugriffskontrolllisten \(ACLs\) für Amazon S3 S3-Buckets mithilfe eines AWS SDK](#)
- [Verwalten Sie große Amazon SQS SQS-Nachrichten mithilfe von Amazon S3 mit einem AWS SDK](#)
- [Verwalten Sie versionierte Amazon S3 S3-Objekte stapelweise mit einer Lambda-Funktion mithilfe eines AWS SDK](#)
- [Analysieren Sie Amazon S3 S3-URLs mit einem AWS SDK](#)
- [Erstellen Sie eine mehrteilige Kopie eines Amazon S3 S3-Objekts mit einem AWS SDK](#)
- [Empfangen und verarbeiten Sie Amazon S3 S3-Ereignisbenachrichtigungen mithilfe eines AWS SDK](#)
- [Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK](#)
- [Senden Sie S3-Ereignisbenachrichtigungen an Amazon EventBridge mit einem AWS SDK](#)
- [Verfolgen Sie den Upload oder Download eines Amazon S3 S3-Objekts mithilfe eines AWS SDK](#)
- [Transformieren von Daten für Ihre Anwendung mit S3 Object Lambda](#)
- [Beispielansätze für Einheiten- und Integrationstests mit einem AWS SDK](#)
- [Rekursives Hochladen eines lokalen Verzeichnisses in einen Amazon Simple Storage Service \(Amazon S3\)-Bucket](#)
- [Laden Sie große Dateien zu und von Amazon S3 hoch oder laden Sie sie herunter mit einem AWS SDK](#)
- [Laden Sie einen Stream unbekannter Größe in ein Amazon S3 S3-Objekt hoch, indem Sie eine AWS SDK](#)
- [Verwenden Sie Prüfsummen, um mit einem Amazon S3 S3-Objekt zu arbeiten, indem Sie AWS](#)

- [Arbeiten Sie mit den Objektintegritätsfunktionen von Amazon S3 mithilfe eines AWS SDK](#)
- [Arbeiten Sie mit Objekten mit Amazon S3 S3-Version unter Verwendung eines AWS SDK](#)
- [Serverless-Beispiele für Amazon S3](#)
 - [Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser](#)

Grundlegende Beispiele für die Verwendung von Amazon S3 AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon Simple Storage Service mit AWS SDKs verwendet werden.

Beispiele

- [Hello Amazon S3](#)
- [Lernen Sie die Grundlagen von Amazon S3 mit einem AWS SDK](#)
- [Aktionen für Amazon S3 mit AWS SDKs](#)
 - [Verwenden Sie AbortMultipartUpload mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CompleteMultipartUpload mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CopyObject mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CreateBucket mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CreateMultiRegionAccessPoint mit einem AWS SDK](#)
 - [Verwenden Sie CreateMultipartUpload mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie CreatePresignedPost mit einem AWS SDK](#)
 - [Verwenden Sie DeleteBucket mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketAnalyticsConfiguration mit einer CLI](#)
 - [Verwenden Sie DeleteBucketCors mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketEncryption mit einer CLI](#)
 - [Verwendung von DeleteBucketInventoryConfiguration mit einer CLI](#)
 - [Verwenden Sie DeleteBucketLifecycle mit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketMetricsConfiguration mit einer CLI](#)
 - [Verwenden Sie DeleteBucketPolicymit einem AWS SDK oder CLI](#)
 - [Verwendung von DeleteBucketReplication mit einer CLI](#)
 - [Verwendung von DeleteBucketTagging mit einer CLI](#)
 - [Verwenden Sie DeleteBucketWebsite mit einem AWS SDK oder CLI](#)

- [Verwenden Sie DeleteObject mit einem AWS SDK oder CLI](#)
- [Verwendung von DeleteObjectTagging mit einer CLI](#)
- [Verwenden Sie DeleteObjects mit einem AWS SDK oder CLI](#)
- [Verwendung von DeletePublicAccessBlock mit einer CLI](#)
- [Verwendung von GetBucketAccelerateConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketAcl mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketAnalyticsConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketCors mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetBucketEncryption mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketInventoryConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketLifecycleConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetBucketLocation mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketLogging mit einer CLI](#)
- [Verwendung von GetBucketMetricsConfiguration mit einer CLI](#)
- [Verwendung von GetBucketNotification mit einer CLI](#)
- [Verwenden Sie GetBucketPolicymit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketPolicyStatus mit einer CLI](#)
- [Verwenden Sie GetBucketReplication mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketRequestPayment mit einer CLI](#)
- [Verwendung von GetBucketTagging mit einer CLI](#)
- [Verwendung von GetBucketVersioning mit einer CLI](#)
- [Verwenden Sie GetBucketWebsite mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObject mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectAcl mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectAttributes mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectLegalHold mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectLockConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectRetention mit einem AWS SDK oder CLI](#)
- [Verwendung von GetObjectTagging mit einer CLI](#)
- [Verwendung von GetPublicAccessBlock mit einer CLI](#)

- [Verwenden Sie HeadBucket mit einem AWS SDK oder CLI](#)
- [Verwenden Sie HeadObject mit einem AWS SDK oder CLI](#)
- [Verwendung von ListBucketAnalyticsConfigurations mit einer CLI](#)
- [Verwendung von ListBucketInventoryConfigurations mit einer CLI](#)
- [Verwenden Sie ListBuckets mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ListMultipartUploads mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ListObjectVersions mit einem AWS SDK oder CLI](#)
- [Verwendung von ListObjects mit einer CLI](#)
- [Verwenden Sie ListObjectsV2 mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketAccelerateConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketAcl mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketCors mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketEncryption mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketLifecycleConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketLogging mit einem AWS SDK oder CLI](#)
- [Verwendung von PutBucketNotification mit einer CLI](#)
- [Verwenden Sie PutBucketNotificationConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketPolicymit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketReplication mit einem AWS SDK oder CLI](#)
- [Verwendung von PutBucketRequestPayment mit einer CLI](#)
- [Verwendung von PutBucketTagging mit einer CLI](#)
- [Verwenden Sie PutBucketVersioning mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketWebsite mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObject mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectAcl mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectLegalHold mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectLockConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectRetention mit einem AWS SDK oder CLI](#)
- [Verwenden Sie RestoreObject mit einem AWS SDK oder CLI](#)
- [Verwenden Sie SelectObjectContent mit einem AWS SDK oder CLI](#)

- [Verwenden Sie UploadPart mit einem AWS SDK oder CLI](#)
- [Verwenden Sie UploadPartCopy mit einem AWS SDK oder CLI](#)

Hello Amazon S3

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon S3 beginnen.

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Code für die CMakeLists.txt CMake-Datei.

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

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

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

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

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

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

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

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

```

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

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

add_executable(${PROJECT_NAME}
    hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

Code für die Quelldatei „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;
}

```

- Einzelheiten zur API finden Sie [ListBuckets](#) unter AWS SDK für C++ API-Referenz.

Go

SDK für Go V2

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
package main

import (
    "context"
    "errors"
    "fmt"

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

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

```

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

```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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;  
    }  
}  
};
```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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()
```

- Einzelheiten zur API finden Sie [ListBuckets](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
# 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
```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// S3 Hello World Example using the AWS SDK for Rust.
///
/// This example lists the objects in a bucket, uploads an object to that bucket,
/// and then retrieves the object and prints some S3 information about the
/// object.
/// This shows a number of S3 features, including how to use built-in paginators
/// for large data sets.
///
/// # Arguments
///
/// * `client` - an S3 client configured appropriately for the environment.
/// * `bucket` - the bucket name that the object will be uploaded to. Must be
///   present in the region the `client` is configured to use.
/// * `filename` - a reference to a path that will be read and uploaded to S3.
/// * `key` - the string key that the object will be uploaded as inside the
///   bucket.
async fn list_bucket_and_upload_object(
    client: &aws_sdk_s3::Client,
    bucket: &str,
    filepath: &Path,
    key: &str,
) -> Result<(), S3ExampleError> {
    // List the buckets in this account
    let mut objects = client
        .list_objects_v2()
```

```

        .bucket(bucket)
        .into_paginator()
        .send();

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

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

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

```

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

Ok(())
}
```

S3ExampleError Versorgungsunternehmen.

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

- Einzelheiten zur API finden Sie [ListBuckets](#) in der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Lernen Sie die Grundlagen von Amazon S3 mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die Amazon-S3-Features gezeigt werden.

```
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;
    }
}

```

Eine Wrapper-Klasse für Methoden des Amazon S3 SDK.

```

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;
        }
    }
}

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

#####
# 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
}

```

Die in diesem Szenario verwendeten Amazon-S3-Funktionen.

```

#####
# 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
}

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

C++

SDK für C++

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
#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();
}

```


- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Go

SDK für Go V2

 Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Definieren Sie eine Struktur, die die vom Szenario verwendete Bucket- und Objektaktionen umschließt.

```
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)
            }
        }
    }
    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
        }
    }
    if err == nil {
        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
}
```

Führen Sie ein interaktives Szenario aus, das Ihnen zeigt, wie Sie mit S3-Buckets und Objekten arbeiten.

```
import (
    "context"
    "fmt"
    "log"
    "os"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunGetStartedScenario is an interactive example that shows you how to use
// Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
// objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Download an object to a local file.
// 4. Copy an object to a different folder in the bucket.
// 5. List objects in the bucket.
// 6. Delete all objects in the bucket.
// 7. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig
// so that
// you can replace it with a mocked or stubbed config for unit testing.
//
```

```
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../../demotools folder of this repo.
func RunGetStartedScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner) {
defer func() {
    if r := recover(); r != nil {
        log.Println("Something went wrong with the demo.")
        _, isMock := questioner.(*demotools.MockQuestioner)
        if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
            log.Println(r)
        }
    }
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 getting started demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}

count := 10
log.Printf("Let's list up to %v buckets for your account:", count)
buckets, err := bucketBasics.ListBuckets(ctx)
if err != nil {
    panic(err)
}
if len(buckets) == 0 {
    log.Println("You don't have any buckets!")
} else {
    if count > len(buckets) {
        count = len(buckets)
    }
    for _, bucket := range buckets[:count] {
        log.Printf("\t\t%v\n", *bucket.Name)
    }
}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
    demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
```

```
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
const smallKey = "doc-example-key"
err = bucketBasics.UploadFile(ctx, bucketName, smallKey, smallFile)
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download %v to a file.", smallKey)
downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
    demotools.NotEmpty{})
err = bucketBasics.DownloadFile(ctx, bucketName, smallKey, downloadFileName)
if err != nil {
    panic(err)
}
log.Printf("File %v downloaded.", downloadFileName)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{})
err = bucketBasics.CopyToFolder(ctx, bucketName, smallKey, folderName)
if err != nil {
    panic(err)
}
log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
log.Println(strings.Repeat("-", 88))

log.Println("Let's list the objects in your bucket.")
```

```
questioner.Ask("Press Enter when you're ready.")
objects, err := bucketBasics.ListObjects(ctx, bucketName)
if err != nil {
    panic(err)
}
log.Printf("Found %v objects.\n", len(objects))
var objKeys []string
for _, object := range objects {
    objKeys = append(objKeys, *object.Key)
    log.Printf("\t\t%v\n", *object.Key)
}
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
    "contents? (y/n)", "y") {
    log.Println("Deleting objects.")
    err = bucketBasics.DeleteObjects(ctx, bucketName, objKeys)
    if err != nil {
        panic(err)
    }
    log.Println("Deleting bucket.")
    err = bucketBasics.DeleteBucket(ctx, bucketName)
    if err != nil {
        panic(err)
    }
    log.Printf("Deleting downloaded file %v.\n", downloadFileName)
    err = os.Remove(downloadFileName)
    if err != nil {
        panic(err)
    }
} else {
    log.Println("Okay. Don't forget to delete objects from your bucket to avoid
charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#)einrichten und ausführen.

Ein Beispielszenario.

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

```

Eine Wrapper-Klasse, die die Operationen enthält.

```

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;
}
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Importieren Sie zunächst alle erforderlichen Module.

```
// 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";
```

Die vorherigen Importe verweisen auf einige hilfreiche Dienstprogramme. Diese Dienstprogramme befinden sich lokal in dem GitHub Repository, auf das am Anfang dieses Abschnitts verwiesen wird. Zu Ihrer Information finden Sie im Folgenden Implementierungen dieser Dienstprogramme.

```
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`;
};
```

Objekte werden in „Buckets“ gespeichert. Definieren wir eine Funktion zum Erstellen eines neuen Buckets.

```
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;
};
```

Buckets enthalten „Objekte“. Diese Funktion lädt den Inhalt eines Verzeichnisses als Objekte in Ihren Bucket hoch.

```
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.`);
  }
};
```

Überprüfen Sie nach dem Hochladen von Objekten, ob sie korrekt hochgeladen wurden. Sie können dafür verwenden `ListObjects`. Sie werden die Eigenschaft „Key“ verwenden, die Antwort enthält jedoch noch weitere nützliche Eigenschaften.

```
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`);
};
```

Gelegentlich möchten Sie vielleicht ein Objekt von einem Bucket in einen anderen kopieren. Verwenden Sie dafür den `CopyObject` Befehl.

```
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();
};
```

Es gibt keine SDK-Methode zum Abrufen mehrerer Objekte aus einem Bucket. Stattdessen erstellen Sie eine Liste von Objekten, die Sie herunterladen und durchlaufen können.

```
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");
};
```

Es ist Zeit, Ihre Ressourcen zu bereinigen. Ein Bucket muss leer sein, bevor er gelöscht werden kann. Mit diesen beiden Funktionen leeren und löschen Sie den 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`);
};
```

Die Funktion „main“ fasst alles zusammen. Wenn Sie diese Datei direkt ausführen, wird die Funktion „main“ aufgerufen.

```
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);
}
};
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Kotlin

SDK für Kotlin

Note

Da ist noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <bucketName> <key> <objectPath> <savePath> <toBucket>

    Where:
        bucketName - The Amazon S3 bucket to create.
        key - The key to use.
        objectPath - The path where the file is located (for example, C:/AWS/
book2.pdf).
        savePath - The path where the file is saved after it's downloaded (for
example, C:/AWS/book2.pdf).
        toBucket - An Amazon S3 bucket to where an object is copied to (for
example, C:/AWS/book2.pdf).
    """

    if (args.size != 4) {
        println(usage)
        exitProcess(1)
    }

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

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

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

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

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

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

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

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

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

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

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

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

```
    }

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

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

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

suspend fun listBucketObs(bucketName: String) {
    val request =
        ListObjectsRequest {
            bucket = bucketName
        }

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

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

```
suspend fun copyBucketOb(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}

suspend fun deleteBucketObs(
    bucketName: String,
    objectName: String,
) {
    val objectId =
        ObjectIdentifier {
            key = objectName
        }

    val delOb =
        Delete {
            objects = listOf(objectId)
        }

    val request =
        DeleteObjectsRequest {
            bucket = bucketName
            delete = delOb
        }
}
```

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

suspend fun deleteBucket(bucketName: String?) {
    val request =
        DeleteBucketRequest {
            bucket = bucketName
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.deleteBucket(request)
        println("The $bucketName was successfully deleted!")
    }
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS-SDK für Kotlin.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

PHP

SDK für PHP

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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";

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP - API-Referenz.
 - [CopyObject](#)
 - [CreateBucket](#)

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

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#)einrichten und ausführen.

```
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"))

```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS-SDK für Python (Boto3).
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Ruby

SDK für Ruby

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#)einrichten und ausführen.

```
require 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
```

```

        create_bucket_configuration: {
          location_constraint: 'us-east-1' # NOTE: only certain regions permitted
        }
      )
      puts("Created demo bucket named #{bucket.name}.")
    rescue Aws::Errors::ServiceError => e
      puts('Tried and failed to create demo bucket.')
      puts("\t#{e.code}: #{e.message}")
      puts("\nCan't continue the demo without a bucket!")
      raise
    else
      bucket
    end

    # Requests a file name from the user.
    #
    # @return The name of the file.
    def create_file
      File.open('demo.txt', w) { |f| f.write('This is a demo file.') }
    end

    # Uploads a file to an Amazon S3 bucket.
    #
    # @param bucket [Aws::S3::Bucket] The bucket object representing the upload
    destination
    # @return [Aws::S3::Object] The Amazon S3 object that contains the uploaded
    file.
    def upload_file(bucket)
      File.open('demo.txt', 'w+') { |f| f.write('This is a demo file.') }
      s3_object = bucket.object(File.basename('demo.txt'))
      s3_object.upload_file('demo.txt')
      puts("Uploaded file demo.txt into bucket #{bucket.name} with key
      #{s3_object.key}.")
    rescue Aws::Errors::ServiceError => e
      puts("Couldn't upload file demo.txt to #{bucket.name}.")
      puts("\t#{e.code}: #{e.message}")
      raise
    else
      s3_object
    end

    # Downloads an Amazon S3 object to a file.
    #
    # @param s3_object [Aws::S3::Object] The object to download.

```

```

def download_file(s3_object)
  puts("\nDo you want to download #{s3_object.key} to a local file (y/n)? ")
  answer = gets.chomp.downcase
  if answer == 'y'
    puts('Enter a name for the downloaded file: ')
    file_name = gets.chomp
    s3_object.download_file(file_name)
    puts("Object #{s3_object.key} successfully downloaded to #{file_name}.")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't download #{s3_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Copies an Amazon S3 object to a subfolder within the same bucket.
#
# @param source_object [Aws::S3::Object] The source object to copy.
# @return [Aws::S3::Object, nil] The destination object.
def copy_object(source_object)
  dest_object = nil
  puts("\nDo you want to copy #{source_object.key} to a subfolder in your
bucket (y/n)? ")
  answer = gets.chomp.downcase
  if answer == 'y'
    dest_object = source_object.bucket.object("demo-folder/
#{source_object.key}")
    dest_object.copy_from(source_object)
    puts("Copied #{source_object.key} to #{dest_object.key}.")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't copy #{source_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
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
  end
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__
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Ruby - API-Referenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Code für die binäre Crate-Datei, die das Szenario ausführt.

```
#![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(())
}

```

Allgemeine Aktionen, die im Szenario verwendet werden.

```

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))  
            }  
        }  
    }  
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für Rust.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

SAP ABAP

SDK für SAP ABAP

 Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

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

    lo_s3->createbucket(
        iv_bucket = iv_bucket_name
        io_createbucketconfiguration = lo_constraint ).
    MESSAGE 'S3 bucket created.' TYPE 'I'.
CATCH /aws1/cx_s3_bucketalrddyexists.
    MESSAGE 'Bucket name already exists.' TYPE 'E'.
CATCH /aws1/cx_s3_bktalrddyownedbyyou.
    MESSAGE 'Bucket already exists and is owned by you.' TYPE 'E'.
ENDTRY.

"Upload an object to an S3 bucket."
TRY.
    "Get contents of file from application server."
    DATA lv_file_content TYPE xstring.
```

```
OPEN DATASET iv_key FOR INPUT IN BINARY MODE.
READ DATASET iv_key INTO lv_file_content.
CLOSE DATASET iv_key.

lo_s3->putobject(
    iv_bucket = iv_bucket_name
    iv_key = iv_key
    iv_body = lv_file_content ).
MESSAGE 'Object uploaded to S3 bucket.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.

" Get an object from a bucket. "
TRY.
    DATA(lo_result) = lo_s3->getobject(
        iv_bucket = iv_bucket_name
        iv_key = iv_key ).
    DATA(lv_object_data) = lo_result->get_body( ).
    MESSAGE 'Object retrieved from S3 bucket.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" Copy an object to a subfolder in a bucket. "
TRY.
    lo_s3->copyobject(
        iv_bucket = iv_bucket_name
        iv_key = |{ iv_copy_to_folder }/{ iv_key }|
        iv_copysource = |{ iv_bucket_name }/{ iv_key }| ).
    MESSAGE 'Object copied to a subfolder.' TYPE 'I'.
    CATCH /aws1/cx_s3_nosuchbucket.
        MESSAGE 'Bucket does not exist.' TYPE 'E'.
    CATCH /aws1/cx_s3_nosuchkey.
        MESSAGE 'Object key does not exist.' TYPE 'E'.
ENDTRY.

" List objects in the bucket. "
TRY.
    DATA(lo_list) = lo_s3->listobjects(
        iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved list of objects in S3 bucket.' TYPE 'I'.
```



```

    CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.
ENDTRY.
DATA text TYPE string VALUE 'Object List - '.
DATA lv_object_key TYPE /aws1/s3_objectkey.
LOOP AT lo_list->get_contents( ) INTO DATA(lo_object).
    lv_object_key = lo_object->get_key( ).
    CONCATENATE lv_object_key ', ' INTO text.
ENDLOOP.
MESSAGE text TYPE 'I'.

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

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

```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS -SDK für SAP ABAP.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)

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

Swift

SDK für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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)
        }
    }
}
}

```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Swift.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)

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

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Aktionen für Amazon S3 mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon S3 S3-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die Amazon-S3-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter [Szenarien für die Verwendung von Amazon S3 AWS SDKs](#) anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der [API-Referenz zu Amazon Simple Storage Service](#).

Beispiele

- [Verwenden Sie AbortMultipartUploadmit einem AWS SDK oder CLI](#)
- [Verwenden Sie CompleteMultipartUploadmit einem AWS SDK oder CLI](#)
- [Verwenden Sie CopyObjectmit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreateBucketmit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreateMultiRegionAccessPointmit einem AWS SDK](#)
- [Verwenden Sie CreateMultipartUploadmit einem AWS SDK oder CLI](#)
- [Verwenden Sie CreatePresignedPostmit einem AWS SDK](#)
- [Verwenden Sie DeleteBucketmit einem AWS SDK oder CLI](#)
- [Verwendung von DeleteBucketAnalyticsConfiguration mit einer CLI](#)
- [Verwenden Sie DeleteBucketCorsmit einem AWS SDK oder CLI](#)
- [Verwendung von DeleteBucketEncryption mit einer CLI](#)
- [Verwendung von DeleteBucketInventoryConfiguration mit einer CLI](#)
- [Verwenden Sie DeleteBucketLifecyclemit einem AWS SDK oder CLI](#)

- [Verwendung von DeleteBucketMetricsConfiguration mit einer CLI](#)
- [Verwenden Sie DeleteBucketPolicy mit einem AWS SDK oder CLI](#)
- [Verwendung von DeleteBucketReplication mit einer CLI](#)
- [Verwendung von DeleteBucketTagging mit einer CLI](#)
- [Verwenden Sie DeleteBucketWebsite mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DeleteObject mit einem AWS SDK oder CLI](#)
- [Verwendung von DeleteObjectTagging mit einer CLI](#)
- [Verwenden Sie DeleteObjects mit einem AWS SDK oder CLI](#)
- [Verwendung von DeletePublicAccessBlock mit einer CLI](#)
- [Verwendung von GetBucketAccelerateConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketAcl mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketAnalyticsConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketCors mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetBucketEncryption mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketInventoryConfiguration mit einer CLI](#)
- [Verwenden Sie GetBucketLifecycleConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetBucketLocation mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketLogging mit einer CLI](#)
- [Verwendung von GetBucketMetricsConfiguration mit einer CLI](#)
- [Verwendung von GetBucketNotification mit einer CLI](#)
- [Verwenden Sie GetBucketPolicy mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketPolicyStatus mit einer CLI](#)
- [Verwenden Sie GetBucketReplication mit einem AWS SDK oder CLI](#)
- [Verwendung von GetBucketRequestPayment mit einer CLI](#)
- [Verwendung von GetBucketTagging mit einer CLI](#)
- [Verwendung von GetBucketVersioning mit einer CLI](#)
- [Verwenden Sie GetBucketWebsite mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObject mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectAcl mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectAttributes mit einem AWS SDK oder CLI](#)

- [Verwenden Sie GetObjectLegalHold mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectLockConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetObjectRetention mit einem AWS SDK oder CLI](#)
- [Verwendung von GetObjectTagging mit einer CLI](#)
- [Verwendung von GetPublicAccessBlock mit einer CLI](#)
- [Verwenden Sie HeadBucket mit einem AWS SDK oder CLI](#)
- [Verwenden Sie HeadObject mit einem AWS SDK oder CLI](#)
- [Verwendung von ListBucketAnalyticsConfigurations mit einer CLI](#)
- [Verwendung von ListBucketInventoryConfigurations mit einer CLI](#)
- [Verwenden Sie ListBuckets mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ListMultipartUploads mit einem AWS SDK oder CLI](#)
- [Verwenden Sie ListObjectVersions mit einem AWS SDK oder CLI](#)
- [Verwendung von ListObjects mit einer CLI](#)
- [Verwenden Sie ListObjectsV2 mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketAccelerateConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketAcl mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketCors mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketEncryption mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketLifecycleConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketLogging mit einem AWS SDK oder CLI](#)
- [Verwendung von PutBucketNotification mit einer CLI](#)
- [Verwenden Sie PutBucketNotificationConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketPolicy mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketReplication mit einem AWS SDK oder CLI](#)
- [Verwendung von PutBucketRequestPayment mit einer CLI](#)
- [Verwendung von PutBucketTagging mit einer CLI](#)
- [Verwenden Sie PutBucketVersioning mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutBucketWebsite mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObject mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectAcl mit einem AWS SDK oder CLI](#)

- [Verwenden Sie PutObjectLegalHold mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectLockConfiguration mit einem AWS SDK oder CLI](#)
- [Verwenden Sie PutObjectRetention mit einem AWS SDK oder CLI](#)
- [Verwenden Sie RestoreObject mit einem AWS SDK oder CLI](#)
- [Verwenden Sie SelectObjectContent mit einem AWS SDK oder CLI](#)
- [Verwenden Sie UploadPart mit einem AWS SDK oder CLI](#)
- [Verwenden Sie UploadPartCopy mit einem AWS SDK oder CLI](#)

Verwenden Sie **AbortMultipartUpload** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie AbortMultipartUpload verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- [Löschen von unvollständigen mehrteiligen Uploads](#)
- [Arbeiten mit Amazon-S3-Objektintegrität](#)

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
//! 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();
}

```

- Einzelheiten zur API finden Sie [AbortMultipartUpload](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So brechen Sie den angegebenen mehrteiligen Upload ab

Der folgende `abort-multipart-upload`-Befehl bricht einen mehrteiligen Upload für den Schlüssel `multipart/01` im Bucket `amzn-s3-demo-bucket` ab.

```

aws s3api abort-multipart-upload \
  --bucket amzn-s3-demo-bucket \
  --key multipart/01 \
  --upload-
id dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P

```

Die für diesen Befehl benötigte Upload-ID wird von `create-multipart-upload` ausgegeben und kann auch mit `list-multipart-uploads` abgerufen werden.

- Einzelheiten zur API finden Sie [AbortMultipartUpload](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadRequest;
import software.amazon.awssdk.services.s3.model.AbortMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.CompletedMultipartUpload;
import software.amazon.awssdk.services.s3.model.CompletedPart;
import software.amazon.awssdk.services.s3.model.CreateMultipartUploadResponse;
import software.amazon.awssdk.services.s3.model.LifecycleRule;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsRequest;
import software.amazon.awssdk.services.s3.model.ListMultipartUploadsResponse;
import software.amazon.awssdk.services.s3.model.MultipartUpload;
import
    software.amazon.awssdk.services.s3.model.PutBucketLifecycleConfigurationResponse;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.UploadPartRequest;
import software.amazon.awssdk.services.s3.model.UploadPartResponse;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.utils.builder.SdkBuilder;

import java.io.IOException;
import java.io.RandomAccessFile;
import java.net.URISyntaxException;
import java.net.URL;
import java.nio.ByteBuffer;
import java.time.Duration;
import java.time.Instant;
import java.util.ArrayList;
```

```
import java.util.Collection;
import java.util.List;
import java.util.Objects;
import java.util.UUID;

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

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

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

    public static void main(String[] args) {
        String bucketName = "amzn-s3-demo-bucket"; // Replace with your bucket
        name.

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

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

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

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

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

        AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
        if (abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
            logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
        }
    }
}

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

```

        ListMultipartUploadsRequest listMultipartUploadsRequest =
ListMultipartUploadsRequest.builder()
    .bucket(bucketName)
    .build();

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

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

                AbortMultipartUploadResponse abortMultipartUploadResponse =
s3Client.abortMultipartUpload(abortMultipartUploadRequest);
                if
(abortMultipartUploadResponse.sdkHttpResponse().isSuccessful()) {
                    logger.info("Upload ID [{}] to bucket [{}] successfully
aborted.", upload.uploadId(), bucketName);
                }
            }
        }
    }

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

```

```

        .uploadId(uploadId)
        .bucket(bucketName)
        .key(key));

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

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

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

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

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

```

```

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

    static Instant initiateAndInterruptTwoUploads(String bucketName, String key)
{
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread1");
        try {
            Thread.sleep(Duration.ofSeconds(5).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
        }
        Instant secondUploadInstant = Instant.now();
        initiateAndInterruptMultiPartUpload(bucketName, key, "uploadThread2");
        return secondUploadInstant;
    }

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

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

```

```
        return createMultipartUploadResponse.uploadId();
    }

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

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

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

                UploadPartResponse partResponse = s3Client.uploadPart(
                    uploadPartRequest,
                    RequestBody.fromByteBuffer(bb));

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

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

```

    }

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

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

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

    static void doMultipartUploadWithUploadId(String bucketName, String key,
String uploadId) {
        new Thread(() -> {
            try {
                List<CompletedPart> completedParts = step2UploadParts(bucketName,
key, uploadId);
                step3CompleteMultipartUpload(bucketName, key, uploadId,
completedParts);
            } catch (SdkException e) {
                logger.error(e.getMessage());
            }
        }, "upload thread").start();
        try {
            Thread.sleep(Duration.ofSeconds(2L).toMillis());
        } catch (InterruptedException e) {
            logger.error(e.getMessage());
            System.exit(1);
        }
    }
}

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

static void createBucket(String bucketName) {
    logger.info("Creating bucket: [{}]", bucketName);
}

```



```

        s3Client.createBucket(b -> b.bucket(bucketName));
        try (S3Waiter s3Waiter = s3Client.waiter()) {
            s3Waiter.waitUntilBucketExists(b -> b.bucket(bucketName));
        }
        logger.info("Bucket created.");
    }

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

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

    static String getFullFilePath(String filePath) {
        URL uploadDirectoryURL =
PerformMultiPartUpload.class.getResource(filePath);
        String fullFilePath;
        try {
            fullFilePath =
Objects.requireNonNull(uploadDirectoryURL).toURI().getPath();
        } catch (URISyntaxException e) {
            throw new RuntimeException(e);
        }
        return fullFilePath;
    }
}

```

- Einzelheiten zur API finden Sie [AbortMultipartUpload](#) in der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl bricht mehrteilige Uploads ab, die vor mehr als 5 Tagen erstellt wurden.

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

Beispiel 2: Dieser Befehl bricht mehrteilige Uploads ab, die vor dem 2. Januar 2014 erstellt wurden.

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

Beispiel 2: Dieser Befehl bricht mehrteilige Uploads ab, die vor dem 2. Januar 2014, 10:45:37, erstellt wurden.

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

- Einzelheiten zur API finden Sie unter [AbortMultipartUpload AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl bricht mehrteilige Uploads ab, die vor mehr als 5 Tagen erstellt wurden.

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

Beispiel 2: Dieser Befehl bricht mehrteilige Uploads ab, die vor dem 2. Januar 2014 erstellt wurden.

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

Beispiel 2: Dieser Befehl bricht mehrteilige Uploads ab, die vor dem 2. Januar 2014, 10:45:37, erstellt wurden.

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

- Einzelheiten zur API finden Sie unter [AbortMultipartUpload AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CompleteMultipartUpload** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie CompleteMultipartUpload verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- [Erstellen einer mehrteiligen Kopie](#)
- [Verwenden der Prüfsummen](#)
- [Arbeiten mit Amazon-S3-Objektintegrität](#)

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
///! 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;
}

```

- Einzelheiten zur API finden Sie [CompleteMultipartUpload](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl schließt einen mehrteiligen Upload für den Schlüssel `multipart/01` im Bucket `amzn-s3-demo-bucket` ab:

```
aws s3api complete-multipart-upload --multipart-upload file://
mpustruct --bucket amzn-s3-demo-bucket --key 'multipart/01' --upload-
id dfRtDYU0WMCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZljF.Yxwh6XG7WfS2vC4to6HiV6YjlX.cph0gtNBtJ8P
```

Die für diesen Befehl benötigte Upload-ID wird von `create-multipart-upload` ausgegeben und kann auch mit `list-multipart-uploads` abgerufen werden.

Die Option für mehrteilige Uploads im obigen Befehl erwartet eine JSON-Struktur, die die Teile des mehrteiligen Uploads beschreibt, die zu einer vollständigen Datei zusammengesetzt werden sollen. In diesem Beispiel wird das Präfix `file://` verwendet, um die JSON-Struktur aus einer Datei im lokalen Ordner mit dem Namen `mpustruct` zu laden.

`mpustruct`:

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

Der ETag-Wert für jeden Teil, der hochgeladen wird, wird jedes Mal ausgegeben, wenn Sie einen Teil mit dem `upload-part`-Befehl hochladen. Er kann auch durch Aufrufen von `list-parts` abgerufen werden, indem die MD5-Prüfsumme für jeden Teils kalkuliert wird.

Ausgabe:

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

```
"Key": "multipart/01"
}
```

- Einzelheiten zur API finden Sie [CompleteMultipartUpload](#) in der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// 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(),
    );
}
```

- Einzelheiten zur API finden Sie [CompleteMultipartUpload](#) in der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CopyObject** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie CopyObject verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Erste Schritte mit der Verschlüsselung](#)
- [Erste Schritte mit Objektspeicher](#)
- [Senden von bedingten Anforderungen](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <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;
        }
    }
}

```

- Einzelheiten zur API finden Sie [CopyObject](#) in der AWS SDK für .NET API-Referenz.

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Kopieren Sie ein Objekt mithilfe einer bedingten Anforderung.

```

/// <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;
}
}

```

- Einzelheiten zur API finden Sie [CopyObject](#) in der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

#####
# 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
}

```

- Einzelheiten zur API finden Sie [CopyObject](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
    }
}
```

- Einzelheiten zur API finden Sie [CopyObject](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl kopiert ein Objekt von bucket-1 nach bucket-2:

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

Ausgabe:

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

- Einzelheiten zur API finden Sie [CopyObject](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}
```

- Einzelheiten zur API finden Sie [CopyObject](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Kopieren eines Objekts mit einem [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);
    }

```

Verwenden Sie an [S3TransferManager](#), um [ein Objekt von einem Bucket in einen anderen zu kopieren](#). Sehen Sie sich die [vollständige Datei](#) an und [testen](#) Sie sie.

```

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();
}
```

- Einzelheiten zur API finden Sie [CopyObject](#) unter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Kopieren Sie das Objekt.

```
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;
  }
}
};
```

Kopieren Sie das Objekt unter der Bedingung, dass sein ETag nicht mit dem angegebenen übereinstimmt.

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

Kopieren Sie das Objekt unter der Bedingung, dass sein ETag nicht mit dem angegebenen übereinstimmt.

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

Kopieren Sie das Objekt unter der Bedingung, dass es in einem bestimmten Zeitraum erstellt oder geändert wurde.

```
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"));
  }
}

```

Kopiert das Objekt unter der Bedingung, dass es in einem bestimmten Zeitraum nicht erstellt oder geändert wurde.

```

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"));
  }
}
```

- Einzelheiten zur API finden Sie [CopyObject](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun copyBucketObject(
    fromBucket: String,
    objectKey: String,
    toBucket: String,
) {
    var encodedUrl = ""
    try {
        encodedUrl = URLEncoder.encode("$fromBucket/$objectKey",
StandardCharsets.UTF_8.toString())
    } catch (e: UnsupportedOperationException) {
        println("URL could not be encoded: " + e.message)
    }

    val request =
        CopyObjectRequest {
            copySource = encodedUrl
            bucket = toBucket
            key = objectKey
        }
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
        s3.copyObject(request)
    }
}
```

- Einzelheiten zur API finden Sie [CopyObject](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Einfache Kopie eines Objekts.

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

- Einzelheiten zur API finden Sie [CopyObject](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl kopiert das Objekt „sample.txt“ aus dem Bucket „test-files“ in denselben Bucket, jedoch mit einem neuen Schlüssel „sample-copy.txt“.

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

Beispiel 2: Dieser Befehl kopiert das Objekt „sample.txt“ aus dem Bucket „test-files“ in den Bucket „backup-files“ mit dem Schlüssel „sample-copy.txt“.

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

Beispiel 3: Dieser Befehl lädt das Objekt „sample.txt“ aus dem Bucket „test-files“ in eine lokale Datei mit dem Namen „local-sample.txt“ herunter.

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

Beispiel 4: Lädt das einzelne Objekt in die angegebene Datei herunter. Die heruntergeladene Datei befindet sich unter c:\downloads\data\archive.zip

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

Beispiel 5: Lädt alle Objekte, die mit dem angegebenen Schlüsselpräfix übereinstimmen, in den lokalen Ordner herunter. Die relative Schlüsselhierarchie wird als Unterordner im gesamten Download-Speicherort beibehalten.

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

- Einzelheiten zur API finden Sie unter [CopyObject AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl kopiert das Objekt „sample.txt“ aus dem Bucket „test-files“ in denselben Bucket, jedoch mit einem neuen Schlüssel „sample-copy.txt“.

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

Beispiel 2: Dieser Befehl kopiert das Objekt „sample.txt“ aus dem Bucket „test-files“ in den Bucket „backup-files“ mit dem Schlüssel „sample-copy.txt“.

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

Beispiel 3: Dieser Befehl lädt das Objekt „sample.txt“ aus dem Bucket „test-files“ in eine lokale Datei mit dem Namen „local-sample.txt“ herunter.

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

Beispiel 4: Lädt das einzelne Objekt in die angegebene Datei herunter. Die heruntergeladene Datei befindet sich unter c:\downloads\data\archive.zip

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

Beispiel 5: Lädt alle Objekte, die mit dem angegebenen Schlüsselpräfix übereinstimmen, in den lokalen Ordner herunter. Die relative Schlüsselhierarchie wird als Unterordner im gesamten Download-Speicherort beibehalten.

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

Beispiel 6: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen parallel Downloads in eine lokale Datei herunter, um die Leistung bei großen Dateien zu verbessern.

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

Beispiel 7: Lädt ein großes Objekt im RANGE-Modus mit einer benutzerdefinierten Teilgröße von 64 MB herunter. Der RANGE-Modus verwendet HTTP-Bytebereichsanforderungen und funktioniert für alle Objekte, unabhängig davon, wie sie hochgeladen wurden.

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


Beispiel 8: Lädt alle Objekte, die dem key prefix entsprechen, mithilfe eines mehrteiligen Downloads mit gleichzeitigen Dateidownloads in einen lokalen Ordner herunter.

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

Beispiel 9: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen Downloads mit 20 gleichzeitigen Verbindungen herunter, um den Durchsatz in Netzwerken mit hoher Bandbreite zu maximieren.

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

Beispiel 10: Lädt alle Objekte, die dem key prefix entsprechen, mithilfe eines mehrteiligen Downloads in einen lokalen Ordner herunter, wobei einzelne Dateifehler überwunden werden, anstatt den Vorgang abubrechen.

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

- Einzelheiten zur API finden Sie unter [CopyObject AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
```

Kopieren Sie ein Objekt mithilfe einer bedingten Anforderung.

```
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

```

- Einzelheiten zur API finden Sie [CopyObject](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Kopieren Sie ein Objekt.

```

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__

```

Kopieren Sie ein Objekt und fügen Sie dem Zielobjekt eine serverseitige Verschlüsselung hinzu.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
  # used as the source object for copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the
  # target key, and encrypt it.
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  # nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
      server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's
  why: #{e.message}"
  end
end

# Example usage:
def run_demo
  source_bucket_name = "amzn-s3-demo-bucket1"
  source_key = "my-source-file.txt"
  target_bucket_name = "amzn-s3-demo-bucket2"
  target_key = "my-target-file.txt"
  target_encryption = "AES256"

  source_bucket = Aws::S3::Bucket.new(source_bucket_name)
  wrapper = ObjectCopyEncryptWrapper.new(source_bucket.object(source_key))
  target_bucket = Aws::S3::Bucket.new(target_bucket_name)
```

```

    target_object = wrapper.copy_object(target_bucket, target_key,
    target_encryption)
    return unless target_object

    puts "Copied #{source_key} from #{source_bucket_name} to
    #{target_object.bucket_name}:#{target_object.key} and "\
        "encrypted the target with #{target_object.server_side_encryption}
    encryption."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Einzelheiten zur API finden Sie [CopyObject](#) in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

/// 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(())
}
```

- Einzelheiten zur API finden Sie [CopyObject](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [CopyObject](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
import AWSS3

public func copyFile(from sourceBucket: String, name: String, to destBucket:
String) async throws {
    let srcUrl = ("\"(sourceBucket)/
\"(name)\"").addingPercentEncoding(withAllowedCharacters: .urlPathAllowed)

    let input = CopyObjectInput(
        bucket: destBucket,
        copySource: srcUrl,
        key: name
    )
    do {
        _ = try await client.copyObject(input: input)
    }
    catch {
        print("ERROR: ", dump(error, name: "Copying an object."))
        throw error
    }
}
```

- Einzelheiten zur API finden Sie [CopyObject](#) in der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreateBucket** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie CreateBucket verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Erste Schritte mit dem Konfigurationsmanagement](#)
- [Erste Schritte mit Feature-Stores für maschinelles Lernen](#)
- [Erste Schritte mit Objektspeicher](#)
- [Verwalten großer Nachrichten mit S3](#)
- [Mit versionierten Objekten arbeiten](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <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;
    }
}

```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS SDK für .NET API-Referenz.

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie einen neuen Bucket mit aktivierter Object Lock.

```

/// <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;
    }
}

```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

#####
# 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
                ;;
        esac
    done
}

```

```

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

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

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

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

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

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

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

```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
}
```

```
}
```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So erstellen Sie einen Bucket

Im folgenden Beispiel für `create-bucket` wird ein Bucket mit dem Namen `amzn-s3-demo-bucket` erstellt:

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

Ausgabe:

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

Weitere Informationen finden Sie unter [Erstellen eines Buckets](#) im Amazon-S3-Benutzerhandbuch.

Beispiel 2: So erstellen Sie einen Bucket, bei dem der Besitzer erzwungen wird

Im folgenden Beispiel für `create-bucket` wird ein Bucket namens `amzn-s3-demo-bucket` erstellt, der die vom Bucket-Eigentümer erzwungene Einstellung für S3 Object Ownership verwendet.

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

Ausgabe:


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

Weitere Informationen finden Sie unter [Steuern des Eigentums an Objekten und Deaktivieren von ACLs](#) im Benutzerhandbuch für Amazon S3.

Beispiel 3: So erstellen Sie einen Bucket außerhalb der Region „us-east“

Im folgenden Beispiel für `create-bucket` wird ein Bucket namens `amzn-s3-demo-bucket` in der Region `eu-west-1` erstellt. Für Regionen außerhalb von `us-east-1` müssen die entsprechenden `LocationConstraint` angegeben werden, um den Bucket in der gewünschten Region zu erstellen.

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

Ausgabe:

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

Weitere Informationen finden Sie unter [Erstellen eines Buckets](#) im Amazon-S3-Benutzerhandbuch.

- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen eines Buckets mit Standardkonfiguration.

```
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
}
```

Erstellen eines Buckets mit Object Lock und warten, bis er existiert.

```
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
}
```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie einen 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);  
        }  
    });  
}
```

Erstellen Sie einen neuen Bucket mit aktivierter Object Lock.

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

- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie den 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;  
    }  
}  
};
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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")  
    }  
}
```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie einen 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.");
}
```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie einen Bucket mit Standardeinstellungen.

```

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

```

Erstellen Sie einen versionierten Bucket mit einer Lebenszyklus-Konfiguration.

```
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

```

- Einzelheiten zur API finden Sie [CreateBucket](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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__

```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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))
    }
}))
}

```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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.

```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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
        }
    }
}

```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreateMultiRegionAccessPoint** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird **CreateMultiRegionAccessPoint**.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Konfigurieren Sie den S3-Steuerungs-Client so, dass er eine Anfrage an die Region us-west-2 sendet.

```
suspend fun createS3ControlClient(): S3ControlClient {
    // Configure your S3ControlClient to send requests to US West
    (Oregon).
        val s3Control = S3ControlClient.fromEnvironment {
            region = "us-west-2"
        }
        return s3Control
}
```

Erstellen Sie den Multi-Region Access Point.

```
suspend fun createMrap(
    s3Control: S3ControlClient,
    accountIdParam: String,
    bucketName1: String,
    bucketName2: String,
    mrpName: String,
): String {
    println("Creating MRAP ...")
    val createMrapResponse: CreateMultiRegionAccessPointResponse =
        s3Control.createMultiRegionAccessPoint {
            accountId = accountIdParam
            clientToken = UUID.randomUUID().toString()
            details {
                name = mrpName
                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:::$accountIdParam:accesspoint/$mrpAlias"
}

```

Warten Sie, bis der Multi-Region Access Point verfügbar ist.

```

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)
}
}
```

- Weitere Informationen finden Sie im [Entwicklerhandbuch zum AWS SDK für Kotlin](#).
- Einzelheiten zur API finden Sie [CreateMultiRegionAccessPoint](#) in der API-Referenz zum AWS SDK für Kotlin.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreateMultipartUpload** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie CreateMultipartUpload verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- [Erstellen einer mehrteiligen Kopie](#)
- [Verwenden der Prüfsummen](#)
- [Arbeiten mit Amazon-S3-Objektintegrität](#)

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

//! 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;
}
```

- Einzelheiten zur API finden Sie [CreateMultipartUpload](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl erstellt einen mehrteiligen Upload im Bucket `amzn-s3-demo-bucket` mit dem Schlüssel `multipart/01`:

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

Ausgabe:

```
{
  "Bucket": "amzn-s3-demo-bucket",
  "UploadId":
  "dfRtDYU0WWCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZ1jF.Yxwh6XG7WfS2vC4to6HiV6Yj1x.cph0gtNBtJ8P3
  "Key": "multipart/01"
}
```

Die fertige Datei in einem Ordner namens `multipart` im Bucket `amzn-s3-demo-bucket` wird `01` benannt. Speichern Sie die Upload-ID, den Schlüssel und den Bucket-Namen zur Verwendung mit dem `upload-part`-Befehl.

- Einzelheiten zur API finden Sie [CreateMultipartUpload](#) in der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// Create a multipart upload. Use UploadPart and CompleteMultipartUpload to
// upload the file.
let multipart_upload_res: CreateMultipartUploadOutput = client
    .create_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .send()
    .await?;

let upload_id = multipart_upload_res.upload_id().ok_or(S3ExampleError::new(
    "Missing upload_id after CreateMultipartUpload",
))?;
```

```
let mut upload_parts: Vec<aws_sdk_s3::types::CompletedPart> = Vec::new();

for chunk_index in 0..chunk_count {
    let this_chunk = if chunk_count - 1 == chunk_index {
        size_of_last_chunk
    } else {
        CHUNK_SIZE
    };
    let stream = ByteStream::read_from()
        .path(path)
        .offset(chunk_index * CHUNK_SIZE)
        .length(Length::Exact(this_chunk))
        .build()
        .await
        .unwrap();

    // Chunk index needs to start at 0, but part numbers start at 1.
    let part_number = (chunk_index as i32) + 1;
    let upload_part_res = client
        .upload_part()
        .key(&key)
        .bucket(&bucket_name)
        .upload_id(upload_id)
        .body(stream)
        .part_number(part_number)
        .send()
        .await?;

    upload_parts.push(
```

```

        CompletedPart::builder()
            .e_tag(upload_part_res.e_tag.unwrap_or_default())
            .part_number(part_number)
            .build(),
    );
}

```

```

// upload_parts: Vec<aws_sdk_s3::types::CompletedPart>
let completed_multipart_upload: CompletedMultipartUpload =
CompletedMultipartUpload::builder()
    .set_parts(Some(upload_parts))
    .build();

let _complete_multipart_upload_res = client
    .complete_multipart_upload()
    .bucket(&bucket_name)
    .key(&key)
    .multipart_upload(completed_multipart_upload)
    .upload_id(upload_id)
    .send()
    .await?;

```

- Einzelheiten zur API finden Sie [CreateMultipartUpload](#) in der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreatePresignedPost** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `CreatePresignedPost`.

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie eine vorsignierte POST URL.

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

- Einzelheiten zur API finden Sie [CreatePresignedPost](#) in der AWS SDK für .NET API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteBucket** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie DeleteBucket verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Erste Schritte mit dem Konfigurationsmanagement](#)
- [Erste Schritte mit Feature-Stores für maschinelles Lernen](#)
- [Erste Schritte mit Objektspeicher](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

 Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
#####
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
#####
function errecho() {
    printf "%s\n" "$*" 1>&2
}

#####
# function delete_bucket
#
# This function deletes the specified bucket.
#
# Parameters:
#     $1 - The name of the bucket.

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

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

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

```
fi
}
```

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht einen Bucket mit dem Namen `amzn-s3-demo-bucket`:

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

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}
```

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie den 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;  
  }  
}  
};
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [DeleteBucket](#) in der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie einen leeren Bucket.

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

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl entfernt alle Objekte und Objektversionen aus dem Bucket „test-files“ und löscht anschließend den Bucket. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Schalter -Force hinzu, um die Bestätigung zu unterdrücken. Beachten Sie, dass nicht leere Buckets nicht gelöscht werden können.

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

- Einzelheiten zur API finden Sie unter [DeleteBucket AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl entfernt alle Objekte und Objektversionen aus dem Bucket „test-files“ und löscht anschließend den Bucket. Der Befehl fordert Sie auf, den Vorgang zu bestätigen, bevor er ausgeführt wird. Fügen Sie den Schalter -Force hinzu, um die Bestätigung zu unterdrücken. Beachten Sie, dass nicht leere Buckets nicht gelöscht werden können.

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

- Einzelheiten zur API finden Sie unter [DeleteBucket AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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

```

- Einzelheiten zur API finden Sie [DeleteBucket](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

# 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

```

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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))
    }
}
}
```

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
    }
}
```

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteBucketAnalyticsConfiguration** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteBucketAnalyticsConfiguration verwendet wird.

CLI

AWS CLI

So löschen Sie eine Analytics-Konfiguration für einen Bucket

Im folgenden Beispiel für `delete-bucket-analytics-configuration` wird die Analytics-Konfiguration für den angegebenen Bucket und die angegebene ID entfernt.

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [DeleteBucketAnalyticsConfiguration](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl entfernt den Analysefilter mit dem Namen „testfilter“ im angegebenen S3-Bucket.

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

- Einzelheiten zur API finden Sie unter [DeleteBucketAnalyticsConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl entfernt den Analysefilter mit dem Namen „testfilter“ im angegebenen S3-Bucket.

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

- Einzelheiten zur API finden Sie unter [DeleteBucketAnalyticsConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteBucketCors** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie DeleteBucketCors verwendet wird.

.NET

SDK für .NET

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [DeleteBucketCors](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht eine Cross-Origin Resource Sharing-Konfiguration aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

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

- Einzelheiten zur API finden Sie [DeleteBucketCors](#) in der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie [DeleteBucketCors](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie [DeleteBucketCors](#) in der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [DeleteBucketCors](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteBucketEncryption** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteBucketEncryption verwendet wird.

CLI

AWS CLI

So löschen Sie die serverseitige Verschlüsselungskonfiguration eines Buckets

Im folgenden Beispiel für `delete-bucket-encryption` wird die serverseitige Verschlüsselungskonfiguration des angegebenen Buckets gelöscht.

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [DeleteBucketEncryption](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dadurch wird die für den angegebenen S3-Bucket aktivierte Verschlüsselung deaktiviert.

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

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DeleteBucketEncryption AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dadurch wird die für den angegebenen S3-Bucket aktivierte Verschlüsselung deaktiviert.

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

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DeleteBucketEncryption AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteBucketInventoryConfiguration** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteBucketInventoryConfiguration verwendet wird.

CLI

AWS CLI

So löschen Sie die Inventarkonfiguration eines Buckets

Im folgenden Beispiel für delete-bucket-inventory-configuration wird die Inventarkonfiguration mit der ID 1 für den angegebenen Bucket gelöscht.

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [DeleteBucketInventoryConfiguration](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl entfernt das Inventar mit dem Namen InventoryName 'test', das dem angegebenen S3-Bucket entspricht.

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

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DeleteBucketInventoryConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl entfernt das Inventar mit dem Namen InventoryName 'test', das dem angegebenen S3-Bucket entspricht.

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

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DeleteBucketInventoryConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteBucketLifecycle** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie DeleteBucketLifecycle verwendet wird.

.NET

SDK für .NET

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [DeleteBucketLifecycle](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht eine Lebenszykluskonfiguration aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

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

- Einzelheiten zur API finden Sie [DeleteBucketLifecycle](#) in der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie [DeleteBucketLifecycle](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [DeleteBucketLifecycle](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteBucketMetricsConfiguration** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteBucketMetricsConfiguration verwendet wird.

CLI

AWS CLI

So löschen Sie eine Metrik-Konfiguration für einen Bucket

Im folgenden Beispiel für delete-bucket-metrics-configuration wird die Metrik-Konfiguration für den angegebenen Bucket und die angegebene ID entfernt.

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

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [DeleteBucketMetricsConfiguration](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl entfernt den Metrikfilter mit dem Namen „testmetrics“ im angegebenen S3-Bucket.

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

- Einzelheiten zur API finden Sie unter [DeleteBucketMetricsConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl entfernt den Metrikfilter mit dem Namen „testmetrics“ im angegebenen S3-Bucket.

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

- Einzelheiten zur API finden Sie unter [DeleteBucketMetricsConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteBucketPolicy** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie DeleteBucketPolicy verwendet wird.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie [DeleteBucketPolicy](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht eine Bucket-Richtlinie aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

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

- Einzelheiten zur API finden Sie [DeleteBucketPolicy](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);
    }
}
```

- Einzelheiten zur API finden Sie [DeleteBucketPolicy](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie die Bucket-Richtlinie.

```
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;
  }
}
};
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [DeleteBucketPolicy](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.


```
suspend fun deleteS3BucketPolicy(bucketName: String?) {  
    val request =  
        DeleteBucketPolicyRequest {  
            bucket = bucketName  
        }  
  
    S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->  
        s3.deleteBucketPolicy(request)  
        println("Done!")  
    }  
}
```

- Einzelheiten zur API finden Sie [DeleteBucketPolicy](#) in der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Der Befehl entfernt die mit dem angegebenen S3-Bucket verknüpfte Bucket-Richtlinie.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [DeleteBucketPolicy AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl entfernt die mit dem angegebenen S3-Bucket verknüpfte Bucket-Richtlinie.

```
Remove-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [DeleteBucketPolicy AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie [DeleteBucketPolicy](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
# 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
```

- Einzelheiten zur API finden Sie [DeleteBucketPolicy](#) in der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [DeleteBucketPolicy](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteBucketReplication** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteBucketReplication verwendet wird.

CLI

AWS CLI

Der folgende Befehl löscht eine Replikationskonfiguration aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

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

- Einzelheiten zur API finden Sie [DeleteBucketReplication](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Löscht die Replikationskonfiguration, die mit dem Bucket namens „amzn-s3-demo-bucket“ verknüpft ist. Beachten Sie, dass für diesen Vorgang eine Genehmigung für die DeleteReplicationConfiguration Aktion s3: erforderlich ist. Sie werden vor der Ausführung des Vorgangs zur Bestätigung aufgefordert. Um die Bestätigung zu unterdrücken, verwenden Sie den Schalter -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter [DeleteBucketReplication AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Löscht die Replikationskonfiguration, die mit dem Bucket namens „amzn-s3-demo-bucket“ verknüpft ist. Beachten Sie, dass für diesen Vorgang eine Genehmigung für die DeleteReplicationConfiguration Aktion s3: erforderlich ist. Sie werden vor der Ausführung des Vorgangs zur Bestätigung aufgefordert. Um die Bestätigung zu unterdrücken, verwenden Sie den Schalter -Force.

```
Remove-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter [DeleteBucketReplication AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteBucketTagging** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteBucketTagging verwendet wird.

CLI

AWS CLI

Der folgende Befehl löscht eine Tagging-Konfiguration aus einem Bucket namens `amzn-s3-demo-bucket`:

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

- Einzelheiten zur API finden Sie [DeleteBucketTagging](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl entfernt alle Tags, die mit dem angegebenen Bucket verknüpft sind.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DeleteBucketTagging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl entfernt alle Tags, die mit dem angegebenen Bucket verknüpft sind.

```
Remove-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DeleteBucketTagging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteBucketWebsite** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie DeleteBucketWebsite verwendet wird.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();  
}
```

- Einzelheiten zur API finden Sie [DeleteBucketWebsite](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht eine Website-Konfiguration aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

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

- Einzelheiten zur API finden Sie [DeleteBucketWebsite](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);
        }
    }
}
```

- Einzelheiten zur API finden Sie [DeleteBucketWebsite](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie die Website-Konfiguration aus dem 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;
    }
  }
};
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [DeleteBucketWebsite](#) in der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl deaktiviert die statische Website-Hosting-Eigenschaft des angegebenen S3-Buckets.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DeleteBucketWebsite AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl deaktiviert die statische Website-Hosting-Eigenschaft des angegebenen S3-Buckets.

```
Remove-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DeleteBucketWebsite AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteObject** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie DeleteObject verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Arbeiten mit Amazon-S3-Objektintegrität](#)
- [Mit versionierten Objekten arbeiten](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie ein Objekt in einem nicht versionierten S3-Bucket.

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

```
    }  
  }  
}
```

Löschen Sie ein Objekt in einem versionierten S3-Bucket.

```
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.USWest2;  
        IAmazonS3 client = new AmazonS3Client();  
  
        await CreateAndDeleteObjectVersionAsync(client, bucketName, keyName);  
    }  
  
    /// <summary>  
    /// This method creates and then deletes a versioned object.  
    /// </summary>  
    /// <param name="client">The initialized Amazon S3 client object used to  
    /// create and delete the object.</param>  
    /// <param name="bucketName">The name of the Amazon S3 bucket where the  
    /// object will be created and deleted.</param>  
    /// <param name="keyName">The key name of the object to create.</param>
```

```

        public static async Task CreateAndDeleteObjectVersionAsync(IAmazonS3
client, string bucketName, string keyName)
        {
            try
            {
                // Add a sample object.
                string versionID = await PutAnObject(client, bucketName,
keyName);

                // Delete the object by specifying an object key and a version
ID.

                DeleteObjectRequest request = new DeleteObjectRequest()
                {
                    BucketName = bucketName,
                    Key = keyName,
                    VersionId = versionID,
                };

                Console.WriteLine("Deleting an object");
                await client.DeleteObjectAsync(request);
            }
            catch (AmazonS3Exception ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
            }
        }

        /// <summary>
        /// This method is used to create the temporary Amazon S3 object.
        /// </summary>
        /// <param name="client">The initialized Amazon S3 object which will be
used
        /// to create the temporary Amazon S3 object.</param>
        /// <param name="bucketName">The name of the Amazon S3 bucket where the
object
        /// will be created.</param>
        /// <param name="objectKey">The name of the Amazon S3 object co create.</
param>
        /// <returns>The Version ID of the created object.</returns>
        public static async Task<string> PutAnObject(IAmazonS3 client, string
bucketName, string objectKey)
        {
            PutObjectRequest request = new PutObjectRequest()
            {

```



```

        BucketName = bucketName,
        Key = objectKey,
        ContentBody = "This is the content body!",
    };

    PutObjectResponse response = await client.PutObjectAsync(request);
    return response.VersionId;
}
}

```

- Einzelheiten zur API finden Sie [DeleteObject](#) in der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

#####
# 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
}
```

- Einzelheiten zur API finden Sie [DeleteObject](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie [DeleteObject](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht ein Objekt namens `test.txt` aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

```
aws s3api delete-object --bucket amzn-s3-demo-bucket --key test.txt
```

Wenn die Bucket-Versionierung aktiviert ist, enthält die Ausgabe die Versions-ID der Löschmarkierung:

```
{
  "VersionId": "9_gKg5vG56F.TTEUdwkxGpJ3tND1WlGq",
  "DeleteMarker": true
}
```

Weitere Informationen über das Löschen von Objekten finden Sie unter [Löschen von Objekten](#) im Entwicklerhandbuch für Amazon S3.

- Einzelheiten zur API finden Sie [DeleteObject](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}

```

- Einzelheiten zur API finden Sie [DeleteObject](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [DeleteObject](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie ein Objekt.

```
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;
        }
    }
};

```

- Einzelheiten zur API finden Sie [DeleteObject](#) in der AWS SDK für JavaScript API-Referenz.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

    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;
}
}

```

- Einzelheiten zur API finden Sie [DeleteObject](#) in der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie ein Objekt.

```

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

```

Setzen Sie ein Objekt auf eine vorherige Version zurück, indem Sie spätere Versionen des Objekts löschen.

```

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}."
    )

```

Reaktivieren Sie ein gelöschttes Objekt, indem Sie die aktive Löschmarke des Objekts entfernen.

```

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)
```

Erstellen Sie einen Lambda-Handler, der eine Löschmarke aus einem S3-Objekt entfernt. Dieser Handler kann verwendet werden, um irrelevante Löschmarkierungen in einem versionierten Bucket effizient zu bereinigen.

```
import logging
from urllib import parse
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
logger.setLevel("INFO")

s3 = boto3.client("s3")

def lambda_handler(event, context):
    """
    Removes a delete marker from the specified versioned object.

    :param event: The S3 batch event that contains the ID of the delete marker
                  to remove.
    :param context: Context about the event.
    :return: A result structure that Amazon S3 uses to interpret the result of
    the
           operation. When the result code is TemporaryFailure, S3 retries the
           operation.
    """
    # Parse job parameters from Amazon S3 batch operations
    invocation_id = event["invocationId"]
    invocation_schema_version = event["invocationSchemaVersion"]

    results = []
    result_code = None
    result_string = None

    task = event["tasks"][0]
    task_id = task["taskId"]

    try:
        obj_key = parse.unquote_plus(task["s3Key"], encoding="utf-8")
        obj_version_id = task["s3VersionId"]
        bucket_name = task["s3BucketArn"].split(":")[-1]

        logger.info(
            "Got task: remove delete marker %s from object %s.", obj_version_id,
            obj_key
        )
```

```

        try:
            # If this call does not raise an error, the object version is not a
delete            # marker and should not be deleted.
            response = s3.head_object(
                Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
            )
            result_code = "PermanentFailure"
            result_string = (
                f"Object {obj_key}, ID {obj_version_id} is not " f"a delete
marker."
            )

            logger.debug(response)
            logger.warning(result_string)
        except ClientError as error:
            delete_marker = error.response["ResponseMetadata"]
["HTTPHeaders"].get(
                "x-amz-delete-marker", "false"
            )
            if delete_marker == "true":
                logger.info(
                    "Object %s, version %s is a delete marker.", obj_key,
obj_version_id
                )
                try:
                    s3.delete_object(
                        Bucket=bucket_name, Key=obj_key, VersionId=obj_version_id
                    )
                    result_code = "Succeeded"
                    result_string = (
                        f"Successfully removed delete marker "
                        f"{obj_version_id} from object {obj_key}."
                    )
                    logger.info(result_string)
                except ClientError as error:
                    # Mark request timeout as a temporary failure so it will be
retried.

                    if error.response["Error"]["Code"] == "RequestTimeout":
                        result_code = "TemporaryFailure"
                        result_string = (
                            f"Attempt to remove delete marker from "
                            f"object {obj_key} timed out."

```

```

        )
        logger.info(result_string)
    else:
        raise
    else:
        raise ValueError(
            f"The x-amz-delete-marker header is either not "
            f"present or is not 'true'."
        )
except Exception as error:
    # Mark all other exceptions as permanent failures.
    result_code = "PermanentFailure"
    result_string = str(error)
    logger.exception(error)
finally:
    results.append(
        {
            "taskId": task_id,
            "resultCode": result_code,
            "resultString": result_string,
        }
    )
return {
    "invocationSchemaVersion": invocation_schema_version,
    "treatMissingKeysAs": "PermanentFailure",
    "invocationId": invocation_id,
    "results": results,
}

```

- Einzelheiten zur API finden Sie [DeleteObject](#) in AWS SDK for Python (Boto3) API Reference.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

/// 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(())
}

```

- Einzelheiten zur API finden Sie [DeleteObject](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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.

```


- Einzelheiten zur API finden Sie [DeleteObject](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
    }
}
```

- Einzelheiten zur API finden Sie [DeleteObject](#) in der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeleteObjectTagging** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie DeleteObjectTagging verwendet wird.

CLI

AWS CLI

So löschen Sie die Tag-Sets eines Objekts

Im folgenden Beispiel für `delete-object-tagging` wird das Tag mit dem angegebenen Schlüssel aus dem Objekt `doc1.rtf` gelöscht.

```
aws s3api delete-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc1.rtf
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [DeleteObjectTagging](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl entfernt alle Tags, die mit dem Objekt mit dem Schlüssel „testfile.txt“ im angegebenen S3-Bucket verknüpft sind.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DeleteObjectTagging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl entfernt alle Tags, die mit dem Objekt mit dem Schlüssel „testfile.txt“ im angegebenen S3-Bucket verknüpft sind.

```
Remove-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket' -  
Select '^Key'
```

Ausgabe:

```
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
```

- Einzelheiten zur API finden Sie unter [DeleteObjectTagging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteObjects** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie DeleteObjects verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Löschen Sie alle Objekte im Bucket.](#)
- [Erste Schritte mit Objektspeicher](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

```
}
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS SDK für .NET API-Referenz.

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie mehrere Objekte in einem nicht versionierten S3-Bucket.

```
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
        // 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;
}
}

```

Löschen Sie mehrere Objekte in einem versionierten S3-Bucket.

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to delete objects in a version-enabled Amazon
/// Simple StorageService (Amazon S3) bucket.
/// </summary>
public class DeleteMultipleObjects
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";

        // If the AWS Region for your Amazon S3 bucket is different from
        // the AWS Region of the default user, define the AWS Region for
        // the Amazon S3 bucket and pass it to the client constructor
    }
}

```



```

        // like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USSouth1;
        IAmazonS3 s3Client;

        s3Client = new AmazonS3Client();
        await DeleteMultipleObjectsFromVersionedBucketAsync(s3Client,
bucketName);
    }

    /// <summary>
    /// This method removes multiple versions and objects from a
    /// version-enabled Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    public static async Task
DeleteMultipleObjectsFromVersionedBucketAsync(IAmazonS3 client, string
bucketName)
    {
        // Delete objects (specifying object version in the request).
        await DeleteObjectVersionsAsync(client, bucketName);

        // Delete objects (without specifying object version in the request).
        var deletedObjects = await DeleteObjectsAsync(client, bucketName);

        // Additional exercise - remove the delete markers Amazon S3 returned
from
        // the preceding response. This results in the objects reappearing
        // in the bucket (you can verify the appearance/disappearance of
        // objects in the console).
        await RemoveDeleteMarkersAsync(client, bucketName, deletedObjects);
    }

    /// <summary>
    /// Creates and then deletes non-versioned Amazon S3 objects and then
deletes
    /// them again. The method returns a list of the Amazon S3 objects
deleted.
    /// </summary>

```

```

call
    /// <param name="client">The initialized Amazon S3 client object used to
    /// PubObjectsAsync and NonVersionedDeleteAsync.</param>
    /// <param name="bucketName">The name of the bucket where the objects
    /// will be created and then deleted.</param>
    /// <returns>A list of DeletedObjects.</returns>
    public static async Task<List<DeletedObject>>
DeleteObjectsAsync(IAmazonS3 client, string bucketName)
    {
        // Upload the sample objects.
        var keysAndVersions2 = await PutObjectsAsync(client, bucketName, 3);

        // Delete objects using only keys. Amazon S3 creates a delete marker
        and
        // returns its version ID in the response.
        List<DeletedObject> deletedObjects = await
NonVersionedDeleteAsync(client, bucketName, keysAndVersions2);
        return deletedObjects;
    }

    /// <summary>
    /// This method creates several temporary objects and then deletes them.
    /// </summary>
    /// <param name="client">The S3 client.</param>
    /// <param name="bucketName">Name of the bucket.</param>
    /// <returns>Async task.</returns>
    public static async Task DeleteObjectVersionsAsync(IAmazonS3 client,
string bucketName)
    {
        // Upload the sample objects.
        var keysAndVersions1 = await PutObjectsAsync(client, bucketName, 3);

        // Delete the specific object versions.
        await VersionedDeleteAsync(client, bucketName, keysAndVersions1);
    }

    /// <summary>
    /// Displays the list of information about deleted files to the console.
    /// </summary>
    /// <param name="e">Error information from the delete process.</param>
    private static void DisplayDeletionErrors(DeleteObjectsException e)
    {
        var errorResponse = e.Response;

```

```

        Console.WriteLine($"No. of objects successfully deleted =
{errorResponse.DeletedObjects.Count}");
        Console.WriteLine($"No. of objects failed to delete =
{errorResponse.DeleteErrors.Count}");
        Console.WriteLine("Printing error data...");
        foreach (var deleteError in errorResponse.DeleteErrors)
        {
            Console.WriteLine($"Object Key:
{deleteError.Key}\\t{deleteError.Code}\\t{deleteError.Message}");
        }
    }

    /// <summary>
    /// Delete multiple objects from a version-enabled bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
    /// RemoveDeleteMarkersAsync.</param>
    /// <param name="bucketName">The name of the bucket from which to delete
    /// objects.</param>
    /// <param name="keys">A list of key names for the objects to delete.</
param>
    private static async Task VersionedDeleteAsync(IAmazonS3 client, string
bucketName, List<KeyVersion> keys)
    {
        var multiObjectDeleteRequest = new DeleteObjectsRequest
        {
            BucketName = bucketName,
            Objects = keys, // This includes the object keys and specific
version IDs.
        };

        try
        {
            Console.WriteLine("Executing VersionedDelete...");
            DeleteObjectsResponse response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted all the
{response.DeletedObjects.Count} items");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

```

```

    }
}

/// <summary>
/// Deletes multiple objects from a non-versioned Amazon S3 bucket.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
/// RemoveDeleteMarkersAsync.</param>
/// <param name="bucketName">The name of the bucket from which to delete
/// objects.</param>
/// <param name="keys">A list of key names for the objects to delete.</
param>
/// <returns>A list of the deleted objects.</returns>
private static async Task<List<DeletedObject>>
NonVersionedDeleteAsync(IAmazonS3 client, string bucketName, List<KeyVersion>
keys)
{
    // Create a request that includes only the object key names.
    DeleteObjectsRequest multiObjectDeleteRequest = new
DeleteObjectsRequest();
    multiObjectDeleteRequest.BucketName = bucketName;

    foreach (var key in keys)
    {
        multiObjectDeleteRequest.AddKey(key.Key);
    }

    // Execute DeleteObjectsAsync.
    // The DeleteObjectsAsync method adds a delete marker for each
    // object deleted. You can verify that the objects were removed
    // using the Amazon S3 console.
    DeleteObjectsResponse response;
    try
    {
        Console.WriteLine("Executing NonVersionedDelete...");
        response = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
        Console.WriteLine("Successfully deleted all the {0} items",
response.DeletedObjects.Count);
    }
    catch (DeleteObjectsException ex)
    {

```

```

        DisplayDeletionErrors(ex);
        throw; // Some deletions failed. Investigate before continuing.
    }

    // This response contains the DeletedObjects list which we use to
delete the delete markers.
    return response.DeletedObjects;
}

/// <summary>
/// Deletes the markers left after deleting the temporary objects.
/// </summary>
/// <param name="client">The initialized Amazon S3 client object used to
call
/// DeleteObjectVersionsAsync, DeleteObjectsAsync, and
/// RemoveDeleteMarkersAsync.</param>
/// <param name="bucketName">The name of the bucket from which to delete
/// objects.</param>
/// <param name="deletedObjects">A list of the objects that were
deleted.</param>
private static async Task RemoveDeleteMarkersAsync(IAmazonS3 client,
string bucketName, List<DeletedObject> deletedObjects)
{
    var keyVersionList = new List<KeyVersion>();

    foreach (var deletedObject in deletedObjects)
    {
        KeyVersion keyVersion = new KeyVersion
        {
            Key = deletedObject.Key,
            VersionId = deletedObject.DeleteMarkerVersionId,
        };
        keyVersionList.Add(keyVersion);
    }

    // Create another request to delete the delete markers.
    var multiObjectDeleteRequest = new DeleteObjectsRequest
    {
        BucketName = bucketName,
        Objects = keyVersionList,
    };

    // Now, delete the delete marker to bring your objects back to the
bucket.

```

```

        try
        {
            Console.WriteLine("Removing the delete markers .....");
            var deleteObjectResponse = await
client.DeleteObjectsAsync(multiObjectDeleteRequest);
            Console.WriteLine($"Successfully deleted the
{deleteObjectResponse.DeletedObjects.Count} delete markers");
        }
        catch (DeleteObjectsException ex)
        {
            DisplayDeletionErrors(ex);
        }
    }

    /// <summary>
    /// Create temporary Amazon S3 objects to show how object deletion works
in an
    /// Amazon S3 bucket with versioning enabled.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// PutObjectAsync to create temporary objects for the example.</param>
    /// <param name="bucketName">A string representing the name of the S3
    /// bucket where we will create the temporary objects.</param>
    /// <param name="number">The number of temporary objects to create.</
param>
    /// <returns>A list of the KeyVersion objects.</returns>
    private static async Task<List<KeyVersion>> PutObjectsAsync(IAmazonS3
client, string bucketName, int number)
    {
        var keys = new List<KeyVersion>();

        for (var i = 0; i < number; i++)
        {
            string key = "ObjectToDelete-" + new System.Random().Next();
            PutObjectRequest request = new PutObjectRequest
            {
                BucketName = bucketName,
                Key = key,
                ContentBody = "This is the content body!",
            };

            var response = await client.PutObjectAsync(request);
            KeyVersion keyVersion = new KeyVersion

```

```

        {
            Key = key,
            VersionId = response.VersionId,
        };

        keys.Add(keyVersion);
    }

    return keys;
}
}

```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

#####
# 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
}

```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl löscht ein Objekt aus einem Bucket mit dem Namen `amzn-s3-demo-bucket`:

```
aws s3api delete-objects --bucket amzn-s3-demo-bucket --delete file://delete.json
```

`delete.json` ist ein JSON-Dokument im aktuellen Verzeichnis, in dem das zu löschende Objekt angegeben wird:

```
{
  "Objects": [
    {
      "Key": "test1.txt"
    }
  ],
  "Quiet": false
}
```

Ausgabe:

```
{
  "Deleted": [
    {
      "DeleteMarkerVersionId": "mYAT5Mc6F7aeUL8SS7FAAqUP01koHwzU",
      "Key": "test1.txt",
      "DeleteMarker": true
    }
  ]
}
```

```
}
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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 specfied 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);
    }
}
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie mehrere Objekte.

```
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;  
    }  
}  
};
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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")
    }
}

```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie eine Reihe von Objekten aus einer Schlüsselliste.

```

$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.");
    }
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl entfernt das Objekt „sample.txt“ aus dem Bucket „test-files“. Sie werden vor der Ausführung des Befehls um eine Bestätigung gebeten. Um den Prompt zu unterdrücken, verwenden Sie den -Force-Schalter.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Beispiel 2: Dieser Befehl entfernt die angegebene Version des Objekts „sample.txt“ aus dem Bucket „test-files“, vorausgesetzt, der Bucket wurde so konfiguriert, dass Objektversionen aktiviert sind.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Beispiel 3: Dieser Befehl entfernt die Objekte „sample1.txt“, „sample2.txt“ und „sample3.txt“ als einzelne Stapelverarbeitung aus dem Bucket „test-files“. Die Antwort des Services listet alle verarbeiteten Schlüssel auf, unabhängig davon, ob die Löschung erfolgreich war oder fehlgeschlagen ist. Um nur Fehler für Schlüssel zu erhalten, die vom Dienst nicht verarbeitet werden konnten, fügen Sie den ReportErrorsOnly Parameter - hinzu (dieser Parameter kann auch mit dem Alias -Quiet angegeben werden).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",
"sample2.txt", "sample3.txt" )
```

Beispiel 4: In diesem Beispiel wird ein Inline-Ausdruck mit dem KeyCollection Parameter - verwendet, um die Schlüssel der zu löschenden Objekte abzurufen. Get-S3Object gibt eine

Sammlung von Amazon.S3.Model.S3Object Instanzen zurück, von denen jede ein Key-Element vom Typ string hat, das das Objekt identifiziert.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Beispiel 5: In diesem Beispiel werden alle Objekte abgerufen, die ein key prefix "prefix/subprefix" im Bucket haben, und sie werden gelöscht. Beachten Sie, dass die eingehenden Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |
Remove-S3Object -Force
```

Beispiel 6: In diesem Beispiel wird eine Sammlung von Amazon.S3.Model.S3ObjectVersion Instanzen, die Löschmarkierungen darstellen, zur Löschung an das Cmdlet übergeben. Beachten Sie, dass die eingehenden Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Beispiel 7: Dieses Skript zeigt, wie eine Gruppe von Objekten (in diesem Fall Löschmarken) im Batch-Modus gelöscht werden kann, indem ein Array von Objekten erstellt wird, die mit dem Parameter - verwendet werden sollen. 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
```

- Einzelheiten zur API finden Sie unter [DeleteObjects AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl entfernt das Objekt „sample.txt“ aus dem Bucket „test-files“. Sie werden vor der Ausführung des Befehls um eine Bestätigung gebeten. Um den Prompt zu unterdrücken, verwenden Sie den -Force-Schalter.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Beispiel 2: Dieser Befehl entfernt die angegebene Version des Objekts „sample.txt“ aus dem Bucket „test-files“, vorausgesetzt, der Bucket wurde so konfiguriert, dass Objektversionen aktiviert sind.

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -VersionId  
HLbxnx6V9omT6AQYVpks8mmFKQcejpqt
```

Beispiel 3: Dieser Befehl entfernt die Objekte „sample1.txt“, „sample2.txt“ und „sample3.txt“ als einzelne Stapelverarbeitung aus dem Bucket „test-files“. Die Antwort des Services listet alle verarbeiteten Schlüssel auf, unabhängig davon, ob die Löschung erfolgreich war oder fehlgeschlagen ist. Um nur Fehler für Schlüssel zu erhalten, die vom Dienst nicht verarbeitet werden konnten, fügen Sie den ReportErrorsOnly Parameter - hinzu (dieser Parameter kann auch mit dem Alias -Quiet angegeben werden).

```
Remove-S3Object -BucketName amzn-s3-demo-bucket -KeyCollection @( "sample1.txt",  
"sample2.txt", "sample3.txt" )
```

Beispiel 4: In diesem Beispiel wird ein Inline-Ausdruck mit dem KeyCollection Parameter - verwendet, um die Schlüssel der zu löschenden Objekte abzurufen. Get-S3Object gibt eine Sammlung von Amazon.S3.Model.S3Object Instanzen zurück, von denen jede ein Key-Element vom Typ string hat, das das Objekt identifiziert.

```
Remove-S3Object -bucketname "amzn-s3-demo-bucket" -KeyCollection (Get-S3Object  
"test-files" -KeyPrefix "prefix/subprefix" | select -ExpandProperty Key)
```

Beispiel 5: In diesem Beispiel werden alle Objekte abgerufen, die ein key prefix "prefix/subprefix" im Bucket haben, und sie werden gelöscht. Beachten Sie, dass die eingehenden

Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
Get-S3Object -BucketName "amzn-s3-demo-bucket" -KeyPrefix "prefix/subprefix" |  
Remove-S3Object -Force
```

Beispiel 6: In diesem Beispiel wird eine Sammlung von Amazon.S3.Model.S3ObjectVersion Instanzen, die Löschmarkierungen darstellen, zur Löschung an das Cmdlet übergeben. Beachten Sie, dass die eingehenden Objekte nacheinander verarbeitet werden. Bei großen Sammlungen sollten Sie erwägen, die Sammlung an den Parameter - InputObject (alias -S3ObjectCollection) des Cmdlets zu übergeben, damit das Löschen als Batch mit einem einzigen Aufruf des Dienstes erfolgen kann.

```
(Get-S3Version -BucketName "amzn-s3-demo-bucket").Versions | Where  
{$_ .IsDeleteMarker -eq "True"} | Remove-S3Object -Force
```

Beispiel 7: Dieses Skript zeigt, wie eine Gruppe von Objekten (in diesem Fall Löschmarken) im Batch-Modus gelöscht werden kann, indem ein Array von Objekten erstellt wird, die mit dem Parameter - verwendet werden sollen. 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
```

- Einzelheiten zur API finden Sie unter [DeleteObjects AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie eine Reihe von Objekten mithilfe einer Liste von Objektschlüsseln.

```
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

```

Löschen Sie alle Objekte im 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

```

Löschen Sie ein versioniertes Objekt dauerhaft, indem Sie alle seine Versionen löschen.

```

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

```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
# 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
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

/// 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(())
}

```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
    }
}
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **DeletePublicAccessBlock** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `DeletePublicAccessBlock` verwendet wird.

CLI

AWS CLI

So löschen Sie die Konfiguration zum Blockieren des öffentlichen Zugriffs für einen Bucket

Im folgenden Beispiel für `delete-public-access-block` wird die Konfiguration zum Blockieren des öffentlichen Zugriffs für den angegebenen Bucket entfernt.

```
aws s3api delete-public-access-block \
    --bucket amzn-s3-demo-bucket
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [DeletePublicAccessBlock](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl deaktiviert die Einstellung „Öffentlicher Zugriff blockieren“ für den angegebenen Bucket.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

Ausgabe:

```
amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter [DeletePublicAccessBlock AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl deaktiviert die Einstellung „Öffentlicher Zugriff blockieren“ für den angegebenen Bucket.

```
Remove-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket' -Force -Select  
'^BucketName'
```

Ausgabe:

```
amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter [DeletePublicAccessBlock AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetBucketAccelerateConfiguration** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketAccelerateConfiguration verwendet wird.

CLI

AWS CLI

So rufen Sie die Accelerate-Konfiguration eines Buckets ab

Im folgenden Beispiel für `get-bucket-accelerate-configuration` wird die Accelerate-Konfiguration für den angegebenen Bucket abgerufen.

```
aws s3api get-bucket-accelerate-configuration \
  --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "Status": "Enabled"
}
```

- Einzelheiten zur API finden Sie [GetBucketAccelerateConfiguration](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt den Wert „Enabled“ zurück, wenn die Übertragungsbeschleunigungseinstellungen für den angegebenen Bucket aktiviert sind.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Value
-----
Enabled
```

- Einzelheiten zur API finden Sie unter [GetBucketAccelerateConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt den Wert „Enabled“ zurück, wenn die Übertragungsbeschleunigungseinstellungen für den angegebenen Bucket aktiviert sind.

```
Get-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Value
-----
Enabled
```

- Einzelheiten zur API finden Sie unter [GetBucketAccelerateConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetBucketAc1** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketAc1 verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Prüfen, ob ein Bucket vorhanden ist](#)
- [Verwalten von Zugriffssteuerungslisten \(ACL\)](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.


```

    /// <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;
    }

```

- Einzelheiten zur API finden Sie [GetBucketAcl](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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";
}

```

- Einzelheiten zur API finden Sie [GetBucketAcl](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Zugriffskontrollliste für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-acl --bucket amzn-s3-demo-bucket
```

Ausgabe:

```

{
    "Owner": {
        "DisplayName": "my-username",
        "ID": "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
    },

```

```

    "Grants": [
      {
        "Grantee": {
          "DisplayName": "my-username",
          "ID":
            "7009a8971cd538e11f6b6606438875e7c86c5b672f46db45460ddcd087d36c32"
        },
        "Permission": "FULL_CONTROL"
      }
    ]
  }
}

```

- Einzelheiten zur API finden Sie [GetBucketAcl](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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 "";
    }
}
```

- Einzelheiten zur API finden Sie [GetBucketAcl](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie die ACL-Berechtigungen ab.

```
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;
    }
  }
};
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).

- Einzelheiten zur API finden Sie [GetBucketAcl](#) in der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V5

Beispiel 1: Der Befehl ruft die Details des Objektbesitzers des S3-Objekts ab.

```
(Get-S3BucketACL -BucketName 'amzn-s3-demo-bucket' -Select *).Owner
```

Ausgabe:

```
DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544
```

- Einzelheiten zur API finden Sie unter [GetBucketAcl AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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

```

- Einzelheiten zur API finden Sie [GetBucketAcl](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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.
```

- Einzelheiten zur API finden Sie [GetBucketAcl](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetBucketAnalyticsConfiguration** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketAnalyticsConfiguration` verwendet wird.

CLI

AWS CLI

So rufen Sie die Analytics-Konfiguration für einen Bucket mit einer bestimmten ID ab

Im folgenden Beispiel für `get-bucket-analytics-configuration` wird die Analytics-Konfiguration für den angegebenen Bucket und die angegebene ID angezeigt.

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

Ausgabe:

```
{
  "AnalyticsConfiguration": {
    "StorageClassAnalysis": {},
    "Id": "1"
  }
}
```

- Einzelheiten zur API finden Sie [GetBucketAnalyticsConfiguration](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Details des Analysefilters mit dem Namen „testfilter“ im angegebenen S3-Bucket zurück.

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

- Einzelheiten zur API finden Sie unter [GetBucketAnalyticsConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Details des Analysefilters mit dem Namen „testfilter“ im angegebenen S3-Bucket zurück.

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

- Einzelheiten zur API finden Sie unter [GetBucketAnalyticsConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetBucketCors** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketCors verwendet wird.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

    /// <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;
    }

```

- Einzelheiten zur API finden Sie [GetBucketCors](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Cross-Origin Resource Sharing-Konfiguration für einen Bucket mit dem Namen `amzn-s3-demo-bucket` abgerufen:

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

Ausgabe:

```

{
  "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": [
            "*"
        ]
    }
]
```

- Einzelheiten zur API finden Sie [GetBucketCors](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie die CORS-Richtlinie für den Bucket ab.

```
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;
    }
}
};

```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [GetBucketCors](#) in der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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

```

- Einzelheiten zur API finden Sie [GetBucketCors](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with
  an existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end
end

```

```

# 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

```

- Einzelheiten zur API finden Sie [GetBucketCors](#) in der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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.

```

- Einzelheiten zur API finden Sie [GetBucketCors](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetBucketEncryption** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketEncryption verwendet wird.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <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(
                $"{Environment.NewLine}Algorithm:
{encryptionSettings.ServerSideEncryptionByDefault.ServerSideEncryptionAlgorithm}");
        }
    }
}
```

```
        Console.WriteLine(
            $"\\tKey:
{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}");
    }
}
```

- Einzelheiten zur API finden Sie [GetBucketEncryption](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie die serverseitige Verschlüsselungskonfiguration für einen Bucket ab

Im folgenden Beispiel für `get-bucket-encryption` wird die serverseitige Verschlüsselungskonfiguration für den Bucket `amzn-s3-demo-bucket` abgerufen.

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

Ausgabe:

```
{
  "ServerSideEncryptionConfiguration": {
    "Rules": [
      {
        "ApplyServerSideEncryptionByDefault": {
          "SSEAlgorithm": "AES256"
        }
      }
    ]
  }
}
```

```
}  
}
```

- Einzelheiten zur API finden Sie [GetBucketEncryption](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt alle serverseitigen Verschlüsselungsregeln zurück, die dem angegebenen Bucket zugeordnet sind.

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

- Einzelheiten zur API finden Sie unter [GetBucketEncryption AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt alle serverseitigen Verschlüsselungsregeln zurück, die dem angegebenen Bucket zugeordnet sind.

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

- Einzelheiten zur API finden Sie unter [GetBucketEncryption AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetBucketInventoryConfiguration** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketInventoryConfiguration verwendet wird.

CLI

AWS CLI

So rufen Sie die Bestandskonfiguration für einen Bucket ab

Im folgenden Beispiel für `get-bucket-inventory-configuration` wird die Bestandskonfiguration für den angegebenen Bucket mit der ID 1 abgerufen.

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

Ausgabe:

```
{
  "InventoryConfiguration": {
    "IsEnabled": true,
    "Destination": {
      "S3BucketDestination": {
        "Format": "ORC",
        "Bucket": "arn:aws:s3:::amzn-s3-demo-bucket",
        "AccountId": "123456789012"
      }
    },
    "IncludedObjectVersions": "Current",
    "Id": "1",
    "Schedule": {
      "Frequency": "Weekly"
    }
  }
}
```

- Einzelheiten zur API finden Sie [GetBucketInventoryConfiguration](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Details des Inventars mit dem Namen „testinventory“ für den angegebenen S3-Bucket zurück.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId
'testinventory'
```

- Einzelheiten zur API finden Sie unter [GetBucketInventoryConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Details des Inventars mit dem Namen „testinventory“ für den angegebenen S3-Bucket zurück.

```
Get-S3BucketInventoryConfiguration -BucketName 'amzn-s3-demo-bucket' -InventoryId 'testinventory'
```

- Einzelheiten zur API finden Sie unter [GetBucketInventoryConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetBucketLifecycleConfiguration** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketLifecycleConfiguration` verwendet wird.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [GetBucketLifecycleConfiguration](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Lebenszyklus-Konfiguration für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "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"
    }
]
}

```

- Einzelheiten zur API finden Sie [GetBucketLifecycleConfiguration](#) in der AWS CLI Befehlsreferenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

class BucketWrapper:
    """Encapsulates S3 bucket actions."""

    def __init__(self, bucket):
        """
        :param bucket: A Boto3 Bucket resource. This is a high-level resource in
        Boto3
                        that wraps bucket actions in a class-like structure.
        """
        self.bucket = bucket
        self.name = bucket.name

    def get_lifecycle_configuration(self):
        """

```

```

    Get the lifecycle configuration of the bucket.

    :return: The lifecycle rules of the specified bucket.
    """
    try:
        config = self.bucket.LifecycleConfiguration()
        logger.info(
            "Got lifecycle rules %s for bucket '%s'.",
            config.rules,
            self.bucket.name,
        )
    except:
        logger.exception(
            "Couldn't get lifecycle rules for bucket '%s'.", self.bucket.name
        )
        raise
    else:
        return config.rules

```

- Einzelheiten zur API finden Sie [GetBucketLifecycleConfiguration](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

TRY.
    oo_result = lo_s3->getbucketlifecycleconf(          " oo_result is
returned for testing purposes. "
    iv_bucket = iv_bucket_name ).
    MESSAGE 'Retrieved bucket lifecycle configuration.' TYPE 'I'.
CATCH /aws1/cx_s3_nosuchbucket.
    MESSAGE 'Bucket does not exist.' TYPE 'E'.

```

```
ENDTRY.
```

- Einzelheiten zur API finden Sie [GetBucketLifecycleConfiguration](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetBucketLocation** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketLocation` verwendet wird.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Standortbeschränkung für einen Bucket mit dem Namen `amzn-s3-demo-bucket` abgerufen, falls eine Einschränkung besteht:

```
aws s3api get-bucket-location --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "LocationConstraint": "us-west-2"
}
```

- Einzelheiten zur API finden Sie [GetBucketLocation](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Standortbeschränkung für den Bucket „`amzn-s3-demo-bucket`“ zurück, sofern eine Beschränkung vorhanden ist.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Value
-----
ap-south-1
```

- Einzelheiten zur API finden Sie unter [GetBucketLocation AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Standortbeschränkung für den Bucket „amzn-s3-demo-bucket“ zurück, sofern eine Beschränkung vorhanden ist.

```
Get-S3BucketLocation -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Value
-----
ap-south-1
```

- Einzelheiten zur API finden Sie unter [GetBucketLocation AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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(())
}

```

- Einzelheiten zur API finden Sie [GetBucketLocation](#) in der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetBucketLogging** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketLogging verwendet wird.

CLI

AWS CLI

So rufen Sie den Protokollierungsstatus für einen Bucket ab

Im folgenden Beispiel für `get-bucket-logging` wird der Protokollierungsstatus für den angegebenen Bucket abgerufen.

```
aws s3api get-bucket-logging \
  --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "LoggingEnabled": {
    "TargetPrefix": "",
    "TargetBucket": "amzn-s3-demo-bucket-logs"
  }
}
```

- Einzelheiten zur API finden Sie [GetBucketLogging](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt den Protokollierungsstatus für den angegebenen Bucket zurück.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- Einzelheiten zur API finden Sie unter [GetBucketLogging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt den Protokollierungsstatus für den angegebenen Bucket zurück.

```
Get-S3BucketLogging -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

TargetBucketName	Grants	TargetPrefix
-----	-----	-----
testbucket1	{}	testprefix

- Einzelheiten zur API finden Sie unter [GetBucketLogging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetBucketMetricsConfiguration** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketMetricsConfiguration verwendet wird.

CLI

AWS CLI

So rufen Sie die Metrik-Konfiguration für einen Bucket mit einer bestimmten ID ab

Im folgenden Beispiel für get-bucket-metrics-configuration wird die Metrik-Konfiguration für den angegebenen Bucket und die angegebene ID angezeigt.

```
aws s3api get-bucket-metrics-configuration \
```

```
--bucket amzn-s3-demo-bucket \  
--id 123
```

Ausgabe:

```
{  
  "MetricsConfiguration": {  
    "Filter": {  
      "Prefix": "logs"  
    },  
    "Id": "123"  
  }  
}
```

- Einzelheiten zur API finden Sie [GetBucketMetricsConfiguration](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Details zum Metrikfilter mit dem Namen „testfilter“ für den angegebenen S3-Bucket zurück.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- Einzelheiten zur API finden Sie unter [GetBucketMetricsConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Details zum Metrikfilter mit dem Namen „testfilter“ für den angegebenen S3-Bucket zurück.

```
Get-S3BucketMetricsConfiguration -BucketName 'amzn-s3-demo-bucket' -MetricsId  
'testfilter'
```

- Einzelheiten zur API finden Sie unter [GetBucketMetricsConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetBucketNotification** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketNotification` verwendet wird.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Benachrichtigungskonfiguration für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-notification --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "TopicConfiguration": {
    "Topic": "arn:aws:sns:us-west-2:123456789012:my-notification-topic",
    "Id": "YmQzMmEwM2EjZWVlI0NGItNzVtZjI1MC00ZjgyLWZDBiZWNI",
    "Event": "s3:ObjectCreated:*",
    "Events": [
      "s3:ObjectCreated:*"
    ]
  }
}
```

- Einzelheiten zur API finden Sie [GetBucketNotification](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die Benachrichtigungskonfiguration des angegebenen Buckets abgerufen.

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty TopicConfigurations
```

Ausgabe:

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Einzelheiten zur API finden Sie unter [GetBucketNotification AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die Benachrichtigungskonfiguration des angegebenen Buckets abgerufen.

```
Get-S3BucketNotification -BucketName amzn-s3-demo-bucket | select -ExpandProperty
TopicConfigurations
```

Ausgabe:

```
Id      Topic
--      -
mimo    arn:aws:sns:eu-west-1:123456789012:topic-1
```

- Einzelheiten zur API finden Sie unter [GetBucketNotification AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetBucketPolicy** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketPolicy verwendet wird.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie [GetBucketPolicy](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Bucket-Richtlinie für einen Bucket mit dem Namen `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "Policy": "{\n\"Version\": \"2008-10-17\", \"Statement\": [\n{\n\"Sid\": \"\", \"Effect\": \"Allow\", \"Principal\": \"*\", \"Action\": \"s3:GetObject\", \"Resource\": \"arn:aws:s3:::amzn-s3-demo-bucket/*\"}, {\n\"Sid\": \"\", \"Effect\": \"Deny\", \"Principal\": \"*\", \"Action\": \"s3:GetObject\", \"Resource\": \"arn:aws:s3:::amzn-s3-demo-bucket/secret/*\"}]\n}"
}
```

So rufen Sie eine Bucket-Richtlinie ab und fügen sie ein. Das folgende Beispiel zeigt, wie Sie eine Amazon-S3-Bucket-Richtlinie herunterladen, Änderungen an der Datei vornehmen, und die geänderte Bucket-Richtlinie mithilfe von `put-bucket-policy` anwenden können. Führen Sie folgenden Befehl aus, um die Bucket-Richtlinie in eine Datei herunterzuladen:

```
aws s3api get-bucket-policy --bucket amzn-s3-demo-bucket --query Policy --output text > policy.json
```

Anschließend können Sie die `policy.json`-Datei nach Bedarf ändern. Schließlich können Sie diese geänderte Richtlinie wieder auf den S3-Bucket anwenden, indem Sie folgenden Befehl ausführen:

`policy.json`-Datei nach Bedarf. Schließlich können Sie diese geänderte Richtlinie wieder auf den S3-Bucket anwenden, indem Sie folgenden Befehl ausführen:

-Datei nach Bedarf. Schließlich können Sie diese geänderte Richtlinie wieder auf den S3-Bucket anwenden, indem Sie folgenden Befehl ausführen:

```
aws s3api put-bucket-policy --bucket amzn-s3-demo-bucket --policy file://  
policy.json
```

- Einzelheiten zur API finden Sie [GetBucketPolicy](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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 "";
    }
}
```

- Einzelheiten zur API finden Sie [GetBucketPolicy](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie die Bucket-Richtlinie ab.

```
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;
    }
}
};
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [GetBucketPolicy](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}
}
```

- Einzelheiten zur API finden Sie [GetBucketPolicy](#) in der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V4 PowerShell

Beispiel 1: Dieser Befehl gibt die mit dem angegebenen S3-Bucket verknüpfte Bucket-Richtlinie aus.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetBucketPolicy AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die mit dem angegebenen S3-Bucket verknüpfte Bucket-Richtlinie aus.

```
Get-S3BucketPolicy -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetBucketPolicy AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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)
```

- Einzelheiten zur API finden Sie [GetBucketPolicy](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
# 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
```

- Einzelheiten zur API finden Sie [GetBucketPolicy](#) in der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [GetBucketPolicy](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetBucketPolicyStatus** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketPolicyStatus verwendet wird.

CLI

AWS CLI

So rufen Sie den Richtlinienstatus eines Buckets ab, der angibt, ob es sich um einen öffentlichen Bucket handelt

Im folgenden Beispiel für `get-bucket-policy-status` wird der Richtlinienstatus für den Bucket `amzn-s3-demo-bucket` abgerufen.

```
aws s3api get-bucket-policy-status \
  --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "PolicyStatus": {
    "IsPublic": false
  }
}
```

- Einzelheiten zur API finden Sie [GetBucketPolicyStatus](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt den Richtlinienstatus für den angegebenen S3-Bucket zurück und gibt an, ob der Bucket öffentlich ist.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetBucketPolicyStatus AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt den Richtlinienstatus für den angegebenen S3-Bucket zurück und gibt an, ob der Bucket öffentlich ist.

```
Get-S3BucketPolicyStatus -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetBucketPolicyStatus AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetBucketReplication** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketReplication verwendet wird.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Replikationskonfiguration für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-replication --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "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"
  }
}
```

- Einzelheiten zur API finden Sie [GetBucketReplication](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

```
}
```

- Einzelheiten zur API finden Sie [GetBucketReplication](#) in der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Informationen der Replikationskonfiguration zurück, die für den Bucket mit dem Namen „amzn-s3-demo-bucket“ festgelegt wurden.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter [GetBucketReplication AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Informationen der Replikationskonfiguration zurück, die für den Bucket mit dem Namen „amzn-s3-demo-bucket“ festgelegt wurden.

```
Get-S3BucketReplication -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter [GetBucketReplication AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetBucketRequestPayment** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetBucketRequestPayment` verwendet wird.

CLI

AWS CLI

So rufen Sie die Zahlungsanforderungskonfiguration eines Buckets auf

Im folgenden Beispiel für `get-bucket-request-payment` wird die Zahlung-durch-den-Anforderer-Konfiguration für den angegebenen Bucket abgerufen.

```
aws s3api get-bucket-request-payment \
  --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "Payer": "BucketOwner"
}
```

- Einzelheiten zur API finden Sie [GetBucketRequestPayment](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Gibt die Zahlungsanforderungskonfiguration für den Bucket mit dem Namen „amzn-s3-demo-bucket“ zurück. Standardmäßig zahlt der Bucket-Eigentümer für die Downloads aus dem Bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter [GetBucketRequestPayment AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Gibt die Zahlungsanforderungskonfiguration für den Bucket mit dem Namen „amzn-s3-demo-bucket“ zurück. Standardmäßig zahlt der Bucket-Eigentümer für die Downloads aus dem Bucket.

```
Get-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter [GetBucketRequestPayment AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetBucketTagging** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketTagging verwendet wird.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Tag-Konfiguration für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-tagging --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "TagSet": [
    {
      "Value": "marketing",
      "Key": "organization"
    }
  ]
}
```

- Einzelheiten zur API finden Sie [GetBucketTagging](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt alle Tags zurück, die mit dem angegebenen Bucket verknüpft sind.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetBucketTagging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt alle Tags zurück, die mit dem angegebenen Bucket verknüpft sind.

```
Get-S3BucketTagging -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetBucketTagging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetBucketVersioning** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketVersioning verwendet wird.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Versionierungskonfiguration für einen Bucket namens amzn-s3-demo-bucket abgerufen:

```
aws s3api get-bucket-versioning --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "Status": "Enabled"
}
```

- Einzelheiten zur API finden Sie [GetBucketVersioning](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt den Status der Versionierung in Bezug auf den angegebenen Bucket zurück.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetBucketVersioning AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt den Status der Versionierung in Bezug auf den angegebenen Bucket zurück.

```
Get-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetBucketVersioning AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetBucketWebsite** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie GetBucketWebsite verwendet wird.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// 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}");

```

- Einzelheiten zur API finden Sie [GetBucketWebsite](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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();
}

```

- Einzelheiten zur API finden Sie [GetBucketWebsite](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die statische Webseitenkonfiguration für einen Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-bucket-website --bucket amzn-s3-demo-bucket
```

Ausgabe:

```

{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}

```

- Einzelheiten zur API finden Sie [GetBucketWebsite](#) in der AWS CLI Befehlsreferenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie die Website-Konfiguration ab.

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

/**
 * Log the website configuration for a bucket.
 * @param {{ bucketName }}
 */
export const main = async ({ bucketName }) => {
  const client = new S3Client({});

  try {
    const response = await client.send(
      new GetBucketWebsiteCommand({
        Bucket: bucketName,
      }),
    );
    console.log(
      `Your bucket is set up to host a website with the following configuration:
\n${JSON.stringify(response, null, 2)}`,
    );
  } catch (caught) {
    if (
      caught instanceof S3ServiceException &&
      caught.name === "NoSuchWebsiteConfiguration"
    ) {
```

```
        console.error(
            `Error from S3 while getting website configuration for ${bucketName}. The
            bucket isn't configured as a website.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while getting website configuration for ${bucketName}.
            ${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};
```

- Einzelheiten zur API finden Sie [GetBucketWebsite](#) in der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die Details der statischen Website-Konfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetBucketWebsite AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die Details der statischen Website-Konfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetBucketWebsite AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetObject** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObject` verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Abrufen eines Objekts aus einem Bucket abrufen, wenn es geändert wurde](#)
- [Holen Sie sich ein Objekt von einem Multi-Region Access Point](#)
- [Erste Schritte mit der Verschlüsselung](#)
- [Erste Schritte mit Objektspeicher](#)
- [Senden von bedingten Anforderungen](#)
- [Verfolgen von Uploads und Downloads](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [GetObject](#) in der AWS SDK für .NET API-Referenz.

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie ein Objekt mithilfe einer bedingten Anforderung ab.

```

    /// <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;
}
}

```

- Einzelheiten zur API finden Sie [GetObject](#) in der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
#####
# 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
}

```

- Einzelheiten zur API finden Sie [GetObject](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie [GetObject](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Das folgende Beispiel veranschaulicht, wie Sie den `get-object`-Befehl verwenden können, um ein Objekt aus Amazon S3 herunterzuladen:

```
aws s3api get-object --bucket text-content --key dir/  
my_images.tar.bz2 my_images.tar.bz2
```

Beachten Sie, dass der Outfile-Parameter ohne einen Optionsnamen wie „`--outfile`“ angegeben wird. Der Name der Ausgabedatei muss der letzte Parameter im Befehl sein.

Das folgende Beispiel veranschaulicht die Verwendung von `--range`, um einen bestimmten Bytebereich aus einem Objekt herunterzuladen. Beachten Sie, dass den Bytebereichen das Präfix „`bytes=`“ vorangestellt werden muss:

```
aws s3api get-object --bucket text-content --key dir/my_data --  
range bytes=8888-9999 my_data_range
```

Weitere Informationen über das Abrufen von Objekten finden Sie unter „Objekte abrufen“ im Entwicklerhandbuch für Amazon S3.

- Einzelheiten zur API finden Sie [GetObject](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}
```

- Einzelheiten zur API finden Sie [GetObject](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Lesen Sie Daten als Byte-Array mit einem [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);
            }
        });
    }
}

```

Verwenden Sie an [S3TransferManager](#), um [ein Objekt in einem S3-Bucket in eine lokale Datei herunterzuladen](#). Sehen Sie sich die [vollständige Datei](#) an und [testen](#) Sie sie.

```

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import software.amazon.awssdk.core.exception.SdkException;
import software.amazon.awssdk.core.sync.RequestBody;
import software.amazon.awssdk.transfer.s3.S3TransferManager;
import software.amazon.awssdk.transfer.s3.model.CompletedFileDownload;
import software.amazon.awssdk.transfer.s3.model.DownloadFileRequest;
import software.amazon.awssdk.transfer.s3.model.FileDownload;
import software.amazon.awssdk.transfer.s3.progress.LoggingTransferListener;

import java.io.IOException;

```

```

import java.net.URISyntaxException;
import java.net.URL;
import java.nio.file.Files;
import java.nio.file.NoSuchFileException;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.UUID;

    public Long downloadFile(S3TransferManager transferManager, String
bucketName,

                                String key, String downloadedFilePath) {
        DownloadFileRequest downloadFileRequest = DownloadFileRequest.builder()
            .getObjectRequest(b -> b.bucket(bucketName).key(key))
            .destination(Paths.get(downloadedFilePath))
            .build();

        FileDownload downloadFile =
transferManager.downloadFile(downloadFileRequest);

        CompletedFileDownload downloadResult =
downloadFile.completionFuture().join();
        logger.info("Content length [{}]",
downloadResult.response().contentLength());
        return downloadResult.response().contentLength();
    }

```

Lesen Sie Tags, die zu einem Objekt gehören, mit einem [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);
    }
}
}

```

Rufen Sie eine URL für ein Objekt mit einem [S3Client](#) ab.

```

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);
    }
}
}

```

Rufen Sie ein Objekt mithilfe des S3Presigner-Client-Objekts mit einem [S3Client](#) ab.

```

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.HttpURLConnection;
import java.time.Duration;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.model.GetObjectRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import
    software.amazon.awssdk.services.s3.presigner.model.GetObjectPresignRequest;
import
    software.amazon.awssdk.services.s3.presigner.model.PresignedGetObjectRequest;
import software.amazon.awssdk.services.s3.presigner.S3Presigner;
import software.amazon.awssdk.utils.IoUtils;

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

```

Usage:

```

        <bucketName> <keyName>\s

Where:
    bucketName - The Amazon S3 bucket name.\s
    keyName - A key name that represents a text file.\s
    """;

if (args.length != 2) {
    System.out.println(USAGE);
    System.exit(1);
}

String bucketName = args[0];
String keyName = args[1];
Region region = Region.US_EAST_1;
S3Presigner presigner = S3Presigner.builder()
    .region(region)
    .build();

getPresignedUrl(presigner, bucketName, keyName);
presigner.close();
}

/**
 * Generates a pre-signed URL for an Amazon S3 object.
 *
 * @param presigner The {@link S3Presigner} instance to use for generating
the pre-signed URL.
 * @param bucketName The name of the Amazon S3 bucket where the object is
stored.
 * @param keyName The key name (file name) of the object in the Amazon S3
bucket.
 *
 * @throws S3Exception If there is an error interacting with the Amazon S3
service.
 * @throws IOException If there is an error opening the HTTP connection or
reading/writing the request/response.
 */
public static void getPresignedUrl(S3Presigner presigner, String bucketName,
String keyName) {
    try {
        GetObjectRequest getObjectRequest = GetObjectRequest.builder()
            .bucket(bucketName)
            .key(keyName)

```



```
        .build();

        GetObjectPresignRequest getObjectPresignRequest =
GetObjectPresignRequest.builder()
        .signatureDuration(Duration.ofMinutes(60))
        .getObjectRequest(getObjectRequest)
        .build();

        PresignedGetObjectRequest presignedGetObjectRequest =
presigner.presignGetObject(getObjectPresignRequest);
        String theUrl = presignedGetObjectRequest.url().toString();
        System.out.println("Presigned URL: " + theUrl);
        HttpURLConnection connection = (HttpURLConnection)
presignedGetObjectRequest.url().openConnection();
        presignedGetObjectRequest.httpRequest().headers().forEach((header,
values) -> {
            values.forEach(value -> {
                connection.addRequestProperty(header, value);
            });
        });

        // Send any request payload that the service needs (not needed when
// isBrowserExecutable is true).
        if (presignedGetObjectRequest.signedPayload().isPresent()) {
            connection.setDoOutput(true);

            try (InputStream signedPayload =
presignedGetObjectRequest.signedPayload().get().asInputStream();
                OutputStream httpOutputStream =
connection.getOutputStream()) {
                IoUtils.copy(signedPayload, httpOutputStream);
            }
        }

        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            System.out.println("Service returned response: ");
            IoUtils.copy(content, System.out);
        }
    } catch (S3Exception | IOException e) {
        e.printStackTrace();
    }
}
```

```
}
```

Rufen Sie ein Objekt mithilfe eines ResponseTransformer Objekts und [S3Client](#) ab.

```
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);
    }
}
}

```

- Einzelheiten zur API finden Sie unter [GetObject AWS SDK for Java 2.x](#)API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#)einrichten und ausführen.

Laden Sie das Objekt herunter.

```

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;
  }
}
};
```

Laden Sie das Objekt unter der Bedingung herunter, dass sein ETag mit dem angegebenen übereinstimmt.

```
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/utills/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"));
  }
}

```

Laden Sie das Objekt unter der Bedingung herunter, dass sein ETag nicht mit dem angegebenen übereinstimmt.

```

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"));
  }
}

```

Laden das Objekt unter der Bedingung herunter, dass es in einem bestimmten Zeitraum erstellt oder geändert wurde.

```

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"));
    }
}

```

Laden das Objekt unter der Bedingung herunter, dass es in einem bestimmten Zeitraum nicht erstellt oder geändert wurde.

```

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,
            })
        );
    } catch (error) {
        if (error instanceof NoSuchKey) {
            console.log("Object not found");
        } else {
            console.error("Error:", error);
        }
    }
};

```

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

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [GetObject](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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")
    }
}
}
```

- Einzelheiten zur API finden Sie [GetObject](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie ein Objekt ab.

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

- Einzelheiten zur API finden Sie [GetObject](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl ruft das Objekt "sample.txt" aus dem Bucket „amzn-s3-demo-bucket“ ab und speichert es in einer Datei mit dem Namen "local-sample.txt" am aktuellen Speicherort. Die Datei „local-sample.txt“ muss vor dem Aufruf dieses Befehls nicht vorhanden sein.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Beispiel 2: Mit diesem Befehl wird das virtuelle Verzeichnis „DIR“ aus dem Bucket „amzn-s3-demo-bucket“ abgerufen und in einem Ordner mit dem Namen "" am aktuellen Speicherort gespeichert. Local-DIR Der Ordner "Local-DIR" muss nicht existieren, bevor dieser Befehl aufgerufen wird.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Beispiel 3: Lädt alle Objekte mit Schlüsseln, die auf „.json“ enden, aus Buckets, deren Name „config“ enthält, in Dateien im angegebenen Ordner herunter. Die Objektschlüssel werden verwendet, um die Dateinamen festzulegen.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

- Einzelheiten zur API finden Sie unter [GetObject AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl ruft das Objekt "sample.txt" aus dem Bucket „amzn-s3-demo-bucket“ ab und speichert es in einer Datei mit dem Namen "local-sample.txt" am aktuellen Speicherort. Die Datei „local-sample.txt“ muss vor dem Aufruf dieses Befehls nicht vorhanden sein.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt
```

Beispiel 2: Mit diesem Befehl wird das virtuelle Verzeichnis „DIR“ aus dem Bucket „amzn-s3-demo-bucket“ abgerufen und in einem Ordner mit dem Namen "" am aktuellen Speicherort

gespeichert. Local-DIR Der Ordner "Local-DIR" muss nicht existieren, bevor dieser Befehl aufgerufen wird.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix DIR -Folder Local-DIR
```

Beispiel 3: Lädt alle Objekte mit Schlüsseln, die auf „.json“ enden, aus Buckets, deren Name „config“ enthält, in Dateien im angegebenen Ordner herunter. Die Objektschlüssel werden verwendet, um die Dateinamen festzulegen.

```
Get-S3Bucket | ? { $_.BucketName -like '*config*' } | Get-S3Object | ? { $_.Key -like '*.json' } | Read-S3Object -Folder C:\ConfigObjects
```

Beispiel 4: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen parallel Downloads herunter, um die Leistung bei großen Dateien zu verbessern.

```
Read-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt -File local-sample.txt -UseMultipartDownload
```

Beispiel 5: Lädt ein großes Objekt im RANGE-Modus mit einer benutzerdefinierten Teilgröße von 64 MB herunter. Der RANGE-Modus verwendet HTTP-Bytebereichsanforderungen und funktioniert für alle Objekte, unabhängig davon, wie sie hochgeladen wurden.

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

Beispiel 6: Lädt alle Objekte mit dem key prefix „data/“ mithilfe eines mehrteiligen Downloads mit gleichzeitigen Dateidownloads in einen lokalen Ordner herunter.

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

Beispiel 7: Lädt ein einzelnes Objekt mithilfe eines mehrteiligen Downloads mit 20 gleichzeitigen Verbindungen herunter, um den Durchsatz in Netzwerken mit hoher Bandbreite zu maximieren.


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

Beispiel 8: Lädt alle Objekte mit dem key prefix „data/“ mithilfe eines mehrteiligen Downloads in einen lokalen Ordner herunter, wobei einzelne Dateifehler überwunden werden, anstatt den Vorgang abubrechen.

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

- Einzelheiten zur API finden Sie unter [GetObject AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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

```

Rufen Sie ein Objekt mithilfe einer bedingten Anforderung ab.

```

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

```

- Einzelheiten zur API finden Sie [GetObject](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie ein Objekt ab.

```
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__

```

Rufen Sie ein Objekt ab und melden Sie seinen serverseitigen Verschlüsselungsstatus.

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
  # successful; otherwise nil.
  def object
    @object.get
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
    end
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object.txt"

  wrapper = ObjectGetEncryptionWrapper.new(Aws::S3::Object.new(bucket_name,
    object_key))
  obj_data = wrapper.get_object
  return unless obj_data
end

```

```

    encryption = obj_data.server_side_encryption.nil? ? 'no' :
    obj_data.server_side_encryption
    puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__

```

- Einzelheiten zur API finden Sie [GetObject](#) in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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)
}

```

- Einzelheiten zur API finden Sie [GetObject](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel](#) einrichten und ausführen.

```

TRY.
    oo_result = lo_s3->getobject(
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.

```

- Einzelheiten zur API finden Sie [GetObject](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
    }
}
```

- Einzelheiten zur API finden Sie [GetObject](#) in der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetObjectAcl** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObjectAcl` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Verwalten von Zugriffssteuerungslisten \(ACL\)](#)

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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";
    }
}

```

- Einzelheiten zur API finden Sie [GetObjectAcl](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird die Zugriffskontrollliste für ein Objekt in einem Bucket namens `amzn-s3-demo-bucket` abgerufen:

```
aws s3api get-object-acl --bucket amzn-s3-demo-bucket --key index.html
```

Ausgabe:

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

- Einzelheiten zur API finden Sie [GetObjectAcl](#) in der AWS CLI Befehlsreferenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [GetObjectAcl](#) in der API-Referenz zum AWS SDK für Kotlin.

PowerShell

Tools für V5 PowerShell

Beispiel 1: Der Befehl ruft die Details des Objektbesitzers des S3-Objekts ab.

```
(Get-S3ObjectACL -BucketName 'amzn-s3-demo-bucket' -key 'initialize.ps1' -Select
*).Owner
```

Ausgabe:

```

DisplayName Id
----- --
testusername
9988776a6554433d22f1100112e334acb45566778899009e9887bd7f66c5f544

```

- Einzelheiten zur API finden Sie unter [GetObjectAcl AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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

```

- Einzelheiten zur API finden Sie [GetObjectAcl](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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.

```

- Einzelheiten zur API finden Sie [GetObjectAcl](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetObjectAttributes** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObjectAttributes` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Arbeiten mit Amazon-S3-Objektintegrität](#)

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// ! 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;
}

```

- Einzelheiten zur API finden Sie [GetObjectAttributes](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie Metadaten von einem Objekt ab, ohne das Objekt selbst zurückzugeben

Im folgenden Beispiel für `get-object-attributes` werden Metadaten aus dem Objekt `doc1.rtf` abgerufen.

```

aws s3api get-object-attributes \
  --bucket amzn-s3-demo-bucket \
  --key doc1.rtf \
  --object-attributes "StorageClass" "ETag" "ObjectSize"

```

Ausgabe:

```

{
  "LastModified": "2022-03-15T19:37:31+00:00",
  "VersionId": "IuCPjXTDzHNfldAuitVBIKJpF2p1fg4P",
  "ETag": "b662d79adeb7c8d787ea7eafb9ef6207",

```

```
"StorageClass": "STANDARD",  
"ObjectSize": 405  
}
```

Weitere Informationen finden Sie [GetObjectAttributes](#) in der Amazon S3 S3-API-Referenz.

- Einzelheiten zur API finden Sie [GetObjectAttributes](#) unter AWS CLI Befehlsreferenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetObjectLegalHold** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObjectLegalHold` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Sperren von Amazon-S3-Objekten](#)

.NET

SDK für .NET

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

```

- Einzelheiten zur API finden Sie [GetObjectLegalHold](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie den Status „Rechtliche Aufbewahrungsfrist“ eines Objekts ab

Im folgenden Beispiel für `get-object-legal-hold` wird der Status Rechtliche Aufbewahrungsfrist für das angegebene Objekt abgerufen.

```

aws s3api get-object-legal-hold \
    --bucket amzn-s3-demo-bucket-with-object-lock \
    --key doc1.rtf

```

Ausgabe:

```

{
  "LegalHold": {
    "Status": "ON"
  }
}

```

```
}
```

- Einzelheiten zur API finden Sie [GetObjectLegalHold](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}

```

- Einzelheiten zur API finden Sie [GetObjectLegalHold](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.


```
// 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;
}
```

- Einzelheiten zur API finden Sie [GetObjectLegalHold](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [GetObjectLegalHold](#) in der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Legen Sie eine rechtliche Aufbewahrungspflicht für ein Objekt fest.

```
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,
        )
```

- Einzelheiten zur API finden Sie [GetObjectLegalHold](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [GetObjectLegalHold](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetObjectLockConfiguration** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie GetObjectLockConfiguration verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Sperren von Amazon-S3-Objekten](#)

.NET

SDK für .NET

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [GetObjectLockConfiguration](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie eine Objektsperrkonfiguration für einen Bucket ab

Im folgenden Beispiel für `get-object-lock-configuration` wird die Object Lock-Konfiguration für den angegebenen Bucket abgerufen.

```
aws s3api get-object-lock-configuration \
  --bucket amzn-s3-demo-bucket-with-object-lock
```

Ausgabe:

```
{
  "ObjectLockConfiguration": {
    "ObjectLockEnabled": "Enabled",
    "Rule": {
      "DefaultRetention": {
        "Mode": "COMPLIANCE",
        "Days": 50
      }
    }
  }
}
```

- Einzelheiten zur API finden Sie [GetObjectLockConfiguration](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}
```

- Einzelheiten zur API finden Sie [GetObjectLockConfiguration](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// 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());
    }

```

- Einzelheiten zur API finden Sie [GetObjectLockConfiguration](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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"));
    }
}
```

- Einzelheiten zur API finden Sie [GetObjectLockConfiguration](#) in der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt den Wert „Enabled“ zurück, wenn die Objektsperrkonfiguration für den angegebenen S3-Bucket aktiviert ist.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select
ObjectLockConfiguration.ObjectLockEnabled
```

Ausgabe:

```
Value
-----
Enabled
```

- Einzelheiten zur API finden Sie unter [GetObjectLockConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt den Wert „Enabled“ zurück, wenn die Objektsperrkonfiguration für den angegebenen S3-Bucket aktiviert ist.

```
Get-S3ObjectLockConfiguration -BucketName 'amzn-s3-demo-bucket' -Select  
ObjectLockConfiguration.ObjectLockEnabled
```

Ausgabe:

```
Value  
-----  
Enabled
```

- Einzelheiten zur API finden Sie unter [GetObjectLockConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie die Object-Lock-Konfiguration ab.

```
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

```

- Einzelheiten zur API finden Sie [GetObjectLockConfiguration](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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.

```

- Einzelheiten zur API finden Sie [GetObjectLockConfiguration](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetObjectRetention** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `GetObjectRetention` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Sperren von Amazon-S3-Objekten](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [GetObjectRetention](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So rufen Sie die Konfiguration für die Aufbewahrung eines Objekts ab

Im folgenden Beispiel für `get-object-retention` wird die Konfiguration für die Aufbewahrung des angegebenen Objekts abgerufen.

```
aws s3api get-object-retention \
  --bucket amzn-s3-demo-bucket-with-object-lock \
  --key doc1.rtf
```

Ausgabe:

```
{
  "Retention": {
    "Mode": "GOVERNANCE",
    "RetainUntilDate": "2025-01-01T00:00:00.000Z"
  }
}
```

- Einzelheiten zur API finden Sie [GetObjectRetention](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}

```

- Einzelheiten zur API finden Sie [GetObjectRetention](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

// 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;
    }
}

```

- Einzelheiten zur API finden Sie [GetObjectRetention](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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"));
  }
}
```

- Einzelheiten zur API finden Sie [GetObjectRetention](#) in der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl gibt den Modus und das Datum zurück, bis das Objekt aufbewahrt wird.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Einzelheiten zur API finden Sie unter [GetObjectRetention AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl gibt den Modus und das Datum zurück, bis das Objekt aufbewahrt wird.

```
Get-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt'
```

- Einzelheiten zur API finden Sie unter [GetObjectRetention AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetObjectTagging** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie GetObjectTagging verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erste Schritte mit Tags](#)

CLI

AWS CLI

So rufen Sie die an ein Objekt angehängten Tags ab

Im folgenden Beispiel für get-object-tagging werden die Werte für den angegebenen Schlüssel aus dem angegebenen Objekt abgerufen.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc1.rtf
```

Ausgabe:

```
{  
  "TagSet": [  
    {  
      "Value": "confidential",  
      "Key": "designation"  
    }  
  ]  
}
```

```
}
```

Im folgenden Beispiel für `get-object-tagging` wird versucht, die Tag-Sätze des Objekts `doc2.rtf` abzurufen, das jedoch keine Tags hat.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc2.rtf
```

Ausgabe:

```
{  
  "TagSet": []  
}
```

Im folgenden Beispiel für `get-object-tagging` werden die Tag-Sätze des Objekts `doc3.rtf` abgerufen, das über mehrere Tags verfügt.

```
aws s3api get-object-tagging \  
  --bucket amzn-s3-demo-bucket \  
  --key doc3.rtf
```

Ausgabe:

```
{  
  "TagSet": [  
    {  
      "Value": "confidential",  
      "Key": "designation"  
    },  
    {  
      "Value": "finance",  
      "Key": "department"  
    },  
    {  
      "Value": "payroll",  
      "Key": "team"  
    }  
  ]  
}
```

- Einzelheiten zur API finden Sie [GetObjectTagging](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel werden die Tags zurückgegeben, die mit dem Objekt im angegebenen S3-Bucket verknüpft sind.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Key  Value
---  -
test value
```

- Einzelheiten zur API finden Sie unter [GetObjectTagging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel werden die Tags zurückgegeben, die mit dem Objekt im angegebenen S3-Bucket verknüpft sind.

```
Get-S3ObjectTagSet -Key 'testfile.txt' -BucketName 'amzn-s3-demo-bucket'
```

Ausgabe:

```
Key  Value
---  -
test value
```

- Einzelheiten zur API finden Sie unter [GetObjectTagging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **GetPublicAccessBlock** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `GetPublicAccessBlock` verwendet wird.

CLI

AWS CLI

So richten Sie die Konfiguration zum Blockieren des öffentlichen Zugriffs für einen Bucket ein oder ändern sie

Im folgenden Beispiel für `get-public-access-block` wird die Konfiguration zum Blockieren des öffentlichen Zugriff für den angegebenen Bucket angezeigt.

```
aws s3api get-public-access-block \
  --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "PublicAccessBlockConfiguration": {
    "IgnorePublicAcls": true,
    "BlockPublicPolicy": true,
    "BlockPublicAcls": true,
    "RestrictPublicBuckets": true
  }
}
```

- Einzelheiten zur API finden Sie [GetPublicAccessBlock](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl gibt die Konfiguration der öffentlichen Zugriffssperre des angegebenen S3-Buckets zurück.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetPublicAccessBlock AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl gibt die Konfiguration der öffentlichen Zugriffssperre des angegebenen S3-Buckets zurück.

```
Get-S3PublicAccessBlock -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [GetPublicAccessBlock AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **HeadBucket** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie HeadBucket verwendet wird.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
#####
# 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
}
```

- Einzelheiten zur API finden Sie [HeadBucket](#) in der AWS CLI Befehlsreferenz.

CLI

AWS CLI

Der folgende Befehl überprüft den Zugriff auf einen Bucket namens `amzn-s3-demo-bucket`:

```
aws s3api head-bucket --bucket amzn-s3-demo-bucket
```

Wenn der Bucket vorhanden ist und Sie Zugriff darauf haben, wird keine Ausgabe zurückgegeben. Andernfalls wird eine Fehlermeldung ausgegeben. Beispiel:

```
A client error (404) occurred when calling the HeadBucket operation: Not Found
```

- Einzelheiten zur API finden Sie [HeadBucket](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}
```

- Einzelheiten zur API finden Sie [HeadBucket](#) in der AWS SDK für Go API-Referenz.

PowerShell

Tools für PowerShell V5

Beispiel 1: Dieser Befehl gibt die Ausgabe mit dem HTTP-Statuscode 200 OK für einen vorhandenen Bucket zurück, wenn der Benutzer die Erlaubnis hat, darauf zuzugreifen. BucketArn Der Parameter wird nur für S3-Verzeichnis-Buckets unterstützt.

```
Get-S3HeadBucket -BucketName amzn-s3-demo-bucket
```

Ausgabe:

```
AccessPointAlias    : False
BucketArn           :
BucketLocationName  :
BucketLocationType  :
BucketRegion        : us-east-2
ResponseMetadata    : Amazon.Runtime.ResponseMetadata
ContentLength       : 0
HttpStatusCode       : OK
```

Beispiel 2: Dieser Befehl gibt einen Fehler mit HTTP-Statuscode `NotFound` für einen nicht existierenden Bucket aus.

```
Get-S3HeadBucket -BucketName amzn-s3-non-existing-bucket
```

Ausgabe:

```
Get-S3HeadBucket: Error making request with Error Code NotFound and Http Status
Code NotFound. No further error information was returned by the service.
```

Beispiel 3: Dieser Befehl löst einen Fehler mit dem HTTP-Statuscode „Forbidden“ für einen vorhandenen Bucket aus, auf den der Benutzer keine Zugriffsberechtigung hat.

```
Get-S3HeadBucket -BucketName amzn-s3-no-access-bucket
```

Ausgabe:

```
Get-S3HeadBucket: Error making request with Error Code Forbidden and Http Status
Code Forbidden. No further error information was returned by the service.
```

- Einzelheiten zur API finden Sie unter [HeadBucket AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie [HeadBucket](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [HeadBucket](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **HeadObject** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie HeadObject verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Prüfen Sie, ob ein Objekt existiert](#)
- [Erste Schritte mit Objektspeicher](#)

CLI

AWS CLI

Der folgende Befehl ruft Metadaten für ein Objekt in einem Bucket mit dem Namen `amzn-s3-demo-bucket` ab:

```
aws s3api head-object --bucket amzn-s3-demo-bucket --key index.html
```

Ausgabe:

```
{
  "AcceptRanges": "bytes",
  "ContentType": "text/html",
  "LastModified": "Thu, 16 Apr 2015 18:19:14 GMT",
  "ContentLength": 77,
  "VersionId": "null",
  "ETag": "\"30a6ec7e1a9ad79c203d05a589c8b400\"",
  "Metadata": {}
}
```

- Einzelheiten zur API finden Sie [HeadObject](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Bestimmen Sie den Inhaltstyp eines Objekts.

```
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);
        }
    }
}

```

Rufen Sie den Wiederherstellungsstatus eines Objekts ab.

```

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);
        }
    }
}
```

```
}
```

- Einzelheiten zur API finden Sie [HeadObject](#) in der AWS SDK for Java 2.x API-Referenz.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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__
```

- Einzelheiten zur API finden Sie [HeadObject](#) in der AWS SDK für Ruby API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **ListBucketAnalyticsConfigurations** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie ListBucketAnalyticsConfigurations verwendet wird.

CLI

AWS CLI

So rufen Sie eine Liste der Analytics-Konfigurationen für einen Bucket ab

Im folgenden Beispiel für list-bucket-analytics-configurations wird eine Liste von Analytics-Konfigurationen für den angegebenen Bucket abgerufen.

```
aws s3api list-bucket-analytics-configurations \
  --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "AnalyticsConfigurationList": [
    {
      "StorageClassAnalysis": {},
      "Id": "1"
    }
  ]
}
```

```
    ],  
    "IsTruncated": false  
}
```

- Einzelheiten zur API finden Sie [ListBucketAnalyticsConfigurations](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die ersten 100 Analysekonfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [ListBucketAnalyticsConfigurations AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die ersten 100 Analysekonfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketAnalyticsConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [ListBucketAnalyticsConfigurations AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **ListBucketInventoryConfigurations** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie ListBucketInventoryConfigurations verwendet wird.

CLI

AWS CLI

So rufen Sie eine Liste der Bestandskonfigurationen für einen Bucket ab

Im folgenden Beispiel für `list-bucket-inventory-configurations` werden die Listen der Bestandskonfigurationen für den angegebenen Bucket aufgelistet.

```
aws s3api list-bucket-inventory-configurations \
  --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "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
  }
```

- Einzelheiten zur API finden Sie [ListBucketInventoryConfigurations](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt die ersten 100 Inventarkonfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [ListBucketInventoryConfigurations AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt die ersten 100 Inventarkonfigurationen des angegebenen S3-Buckets zurück.

```
Get-S3BucketInventoryConfigurationList -BucketName 'amzn-s3-demo-bucket'
```

- Einzelheiten zur API finden Sie unter [ListBucketInventoryConfigurations AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ListBuckets** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie ListBuckets verwendet wird.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);
    }
}

```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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;
}
```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Der folgende Befehl verwendet den `list-buckets`-Befehl, um die Namen aller Ihrer Amazon-S3-Buckets (in allen Regionen) anzuzeigen:

```
aws s3api list-buckets --query "Buckets[].Name"
```

Die Abfrageoption filtert die Ausgabe von `list-buckets` nach unten, sodass nur die Bucket-Namen angezeigt werden.

Weitere Informationen über Buckets finden Sie unter „Arbeiten mit Amazon-S3-Buckets“ im Entwicklerhandbuch für Amazon S3.

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}

```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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()));
    }
}
```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Listen Sie die Buckets auf.

```
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;
  }
}
};
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl gibt alle S3-Buckets zurück.

```
Get-S3Bucket
```

Beispiel 2: Dieser Befehl gibt einen Bucket mit dem Namen „test-files“ zurück

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter [ListBuckets AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl gibt alle S3-Buckets zurück.

```
Get-S3Bucket
```

Beispiel 2: Dieser Befehl gibt einen Bucket mit dem Namen „test-files“ zurück

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket
```

- Einzelheiten zur API finden Sie unter [ListBuckets AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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

```

- Einzelheiten zur API finden Sie [ListBuckets](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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
  end
end

```

```

    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__

```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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(())
}

```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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.

```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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())
}

```

- Einzelheiten zur API finden Sie [ListBuckets](#) in der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ListMultipartUploads** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie ListMultipartUploads verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Löschen von unvollständigen mehrteiligen Uploads](#)

CLI

AWS CLI

Der folgende Befehl listet alle aktiven mehrteiligen Uploads für einen Bucket mit dem Namen `amzn-s3-demo-bucket` auf:

```
aws s3api list-multipart-uploads --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{
  "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":
        "100719349fc3b6dcd7c820a124bf7aec408092c3d7b51b38494939801fc248b"
      }
    }
  ]
}
```

```
    ],  
    "CommonPrefixes": []  
}
```

Bei laufenden mehrteiligen Uploads fallen Speicherkosten in Amazon S3 an. Schließen Sie einen aktiven mehrteiligen Upload ab oder brechen Sie ihn ab, um seine Teile aus Ihrem Konto zu entfernen.

- Einzelheiten zur API finden Sie [ListMultipartUploads](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);  
        }  
    }  
}
```

- Einzelheiten zur API finden Sie [ListMultipartUploads](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ListObjectVersions** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie ListObjectVersions verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Erste Schritte mit Objektspeicher](#)
- [Mit versionierten Objekten arbeiten](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

```

    /// <summary>
    /// This example lists the versions of the objects in a version enabled
    /// Amazon Simple Storage Service (Amazon S3) bucket.
    /// </summary>
    public class ListObjectVersions
    {
        public static async Task Main()
        {
            string bucketName = "amzn-s3-demo-bucket";

            // If the AWS Region where your bucket is defined is different from
            // the AWS Region where the Amazon S3 bucket is defined, pass the
constant
            // for the AWS Region to the client constructor like this:
            //     var client = new AmazonS3Client(RegionEndpoint.USWest2);
            IAmazonS3 client = new AmazonS3Client();
            await GetObjectListWithAllVersionsAsync(client, bucketName);
        }

        /// <summary>
        /// This method lists all versions of the objects within an Amazon S3
        /// version enabled bucket.
        /// </summary>
        /// <param name="client">The initialized client object used to call
        /// ListVersionsAsync.</param>
        /// <param name="bucketName">The name of the version enabled Amazon S3
bucket
        /// for which you want to list the versions of the contained objects.</
param>
        public static async Task GetObjectListWithAllVersionsAsync(IAmazonS3
client, string bucketName)
        {
            try
            {
                // When you instantiate the ListVersionRequest, you can
                // optionally specify a key name prefix in the request
                // if you want a list of object versions of a specific object.

                // For this example we set a small limit in MaxKeys to return
                // a small list of versions.
                ListVersionsRequest request = new ListVersionsRequest()
                {
                    BucketName = bucketName,

```

```
        MaxKeys = 2,
    };

    do
    {
        ListVersionsResponse response = await
client.ListVersionsAsync(request);

        // Process response.
        foreach (S3ObjectVersion entry in response.Versions)
        {
            Console.WriteLine($"key: {entry.Key} size:
{entry.Size}");
        }

        // If response is truncated, set the marker to get the next
        // set of keys.
        if (response.IsTruncated)
        {
            request.KeyMarker = response.NextKeyMarker;
            request.VersionIdMarker = response.NextVersionIdMarker;
        }
        else
        {
            request = null;
        }
    }
    while (request != null);
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error: '{ex.Message}'");
}
}
```

- Einzelheiten zur API finden Sie [ListObjectVersions](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl werden Versionsinformationen für ein Objekt in einem Bucket mit dem Namen `amzn-s3-demo-bucket` abgerufen:

```
aws s3api list-object-versions --bucket amzn-s3-demo-bucket --prefix index.html
```

Ausgabe:

```
{
  "DeleteMarkers": [
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": true,
      "VersionId": "B2VsEK5saUNNHKc0AJj7hIE86RozToyq",
      "Key": "index.html",
      "LastModified": "2015-11-10T00:57:03.000Z"
    },
    {
      "Owner": {
        "DisplayName": "my-username",
        "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
      },
      "IsLatest": false,
      "VersionId": ".FLQEZscLIcfxSq.jsFJ.szUkmng2Yw6",
      "Key": "index.html",
      "LastModified": "2015-11-09T23:32:20.000Z"
    }
  ],
  "Versions": [
    {
      "LastModified": "2015-11-10T00:20:11.000Z",
      "VersionId": "Rb_l2T8UHDkFEwCgJjhlGPOZC0qJ.vpD",
      "ETag": "\"0622528de826c0df5db1258a23b80be5\"",
      "StorageClass": "STANDARD",
      "Key": "index.html",
    }
  ]
}
```

```

        "Owner": {
            "DisplayName": "my-username",
            "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
        },
        "IsLatest": false,
        "Size": 38
    },
    {
        "LastModified": "2015-11-09T23:26:41.000Z",
        "VersionId": "rasWWGpgk9E4s0LyTJgusGeRQKLVIAff",
        "ETag": "\"06225825b8028de826c0df5db1a23be5\"",
        "StorageClass": "STANDARD",
        "Key": "index.html",
        "Owner": {
            "DisplayName": "my-username",
            "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
        },
        "IsLatest": false,
        "Size": 38
    },
    {
        "LastModified": "2015-11-09T22:50:50.000Z",
        "VersionId": "null",
        "ETag": "\"d1f45267a863c8392e07d24dd592f1b9\"",
        "StorageClass": "STANDARD",
        "Key": "index.html",
        "Owner": {
            "DisplayName": "my-username",
            "ID":
"7009a8971cd660687538875e7c86c5b672fe116bd438f46db45460ddcd036c32"
        },
        "IsLatest": false,
        "Size": 533823
    }
}
]
}

```

- Einzelheiten zur API finden Sie [ListObjectVersions](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}

```

- Einzelheiten zur API finden Sie [ListObjectVersions](#) in der AWS SDK für Go API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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(())
}

```


- Einzelheiten zur API finden Sie [ListObjectVersions](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [ListObjectVersions](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **ListObjects** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie ListObjects verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Eine Webseite erstellen, die Amazon-S3-Objekte auflistet](#)

CLI

AWS CLI

Im folgenden Beispiel wird der Befehl `list-objects` verwendet, um die Namen aller Objekte im angegebenen Bucket anzuzeigen:

```
aws s3api list-objects --bucket text-content --query 'Contents[].{Key: Key, Size: Size}'
```

Im Beispiel wird das Argument `--query` verwendet, um die Ausgabe von `list-objects` nach Schlüsselwert und Größe für jedes Objekt zu filtern.

Weitere Informationen über Objekte finden Sie unter „Arbeiten mit Amazon-S3-Objekten“ im Entwicklerhandbuch zu Amazon S3.

- Einzelheiten zur API finden Sie [ListObjects](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl ruft die Informationen über alle Elemente im Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Beispiel 2: Dieser Befehl ruft die Informationen über das Objekt „sample.txt“ aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Beispiel 3: Dieser Befehl ruft die Informationen über alle Elemente mit dem Präfix „sample“ aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Einzelheiten zur API finden Sie unter [ListObjects AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl ruft die Informationen über alle Elemente im Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket
```

Beispiel 2: Dieser Befehl ruft die Informationen über das Objekt „sample.txt“ aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -Key sample.txt
```

Beispiel 3: Dieser Befehl ruft die Informationen über alle Elemente mit dem Präfix „sample“ aus dem Bucket „test-files“ ab.

```
Get-S3Object -BucketName amzn-s3-demo-bucket -KeyPrefix sample
```

- Einzelheiten zur API finden Sie unter [ListObjects AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ListObjectsV2** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie ListObjectsV2 verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Löschen Sie alle Objekte im Bucket.](#)
- [Erste Schritte mit Objektspeicher](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS SDK für .NET API-Referenz. SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Listen Sie Objekte mit einem Paginator auf.

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

```
}
}
```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
#####
# 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
}
```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

So rufen Sie eine Liste von Objekten in einem Bucket ab

Im folgenden Beispiel für `list-objects-v2` werden die Objekte im angegebenen Bucket aufgelistet.

```
aws s3api list-objects-v2 \  
  --bucket amzn-s3-demo-bucket
```

Ausgabe:

```
{  
  "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  
    }  
  ]  
}
```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}

```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

/**
 * 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);
        });
    }
}

```

Listet Objekte mithilfe der Paginierung auf.

```

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);
    }
}
}
```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Listen Sie alle Objekte in Ihrem Bucket auf. Wenn es mehr als ein Objekt gibt, `IsTruncated` `NextContinuationToken` wird es verwendet, um über die gesamte Liste zu iterieren.

```
import {
  S3Client,
  S3ServiceException,
  // This command supersedes the ListObjectsCommand and is the recommended way to
  list objects.
  paginateListObjectsV2,
} from "@aws-sdk/client-s3";

/**
 * Log all of the object keys in a bucket.
 * @param {{ bucketName: string, pageSize: string }}
 */
export const main = async ({ bucketName, pageSize }) => {
  const client = new S3Client({});
  /** @type {string[][]} */
  const objects = [];
  try {
    const paginator = paginateListObjectsV2(
      { client, /* Max items per page */ pageSize: Number.parseInt(pageSize) },
      { Bucket: bucketName },
    );

    for await (const page of paginator) {
```

```

        objects.push(page.Contents.map((o) => o.Key));
    }
    objects.forEach((objectList, pageNum) => {
        console.log(
            `Page ${pageNum + 1}\n-----\n${objectList.map((o) => `•
${o}`)}.join("\n")}\n`,
        );
    });
} catch (caught) {
    if (
        caught instanceof S3ServiceException &&
        caught.name === "NoSuchBucket"
    ) {
        console.error(
            `Error from S3 while listing objects for "${bucketName}". The bucket
doesn't exist.`,
        );
    } else if (caught instanceof S3ServiceException) {
        console.error(
            `Error from S3 while listing objects for "${bucketName}".
${caught.name}: ${caught.message}`,
        );
    } else {
        throw caught;
    }
}
};

```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.


```
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
```

- API-Details finden Sie unter [ListObjectsV2](#) im AWS SDK für die Kotlin-API-Referenz.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Listen Sie Objekte in einem Bucket auf.

```
$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.");
}

```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS SDK für PHP API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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

```

- API-Details finden Sie unter [ListObjectsV2](#) in der AWS API-Referenz zum SDK for Python (Boto3).

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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__

```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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(())
}
```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}
}

```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS SDK-Referenz zur Swift-API.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketAccelerateConfiguration** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketAccelerateConfiguration verwendet wird.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

/// <summary>
/// Amazon Simple Storage Service (Amazon S3) Transfer Acceleration is a
/// bucket-level feature that enables you to perform faster data transfers
/// to Amazon S3. This example shows how to configure Transfer
/// Acceleration.
/// </summary>
public class TransferAcceleration
{
    /// <summary>
    /// The main method initializes the client object and sets the
    /// Amazon Simple Storage Service (Amazon S3) bucket name before
    /// calling EnableAccelerationAsync.
    /// </summary>
    public static async Task Main()
    {
        var s3Client = new AmazonS3Client();
        const string bucketName = "amzn-s3-demo-bucket";

        await EnableAccelerationAsync(s3Client, bucketName);
    }

    /// <summary>
    /// This method sets the configuration to enable transfer acceleration
    /// for the bucket referred to in the bucketName parameter.
    /// </summary>
    /// <param name="client">An Amazon S3 client used to enable the
    /// acceleration on an Amazon S3 bucket.</param>
```



```

    the
    /// <param name="bucketName">The name of the Amazon S3 bucket for which
    /// method will be enabling acceleration.</param>
    private static async Task EnableAccelerationAsync(AmazonS3Client client,
string bucketName)
    {
        try
        {
            var putRequest = new PutBucketAccelerateConfigurationRequest
            {
                BucketName = bucketName,
                AccelerateConfiguration = new AccelerateConfiguration
                {
                    Status = BucketAccelerateStatus.Enabled,
                },
            };
            await client.PutBucketAccelerateConfigurationAsync(putRequest);

            var getRequest = new GetBucketAccelerateConfigurationRequest
            {
                BucketName = bucketName,
            };
            var response = await
client.GetBucketAccelerateConfigurationAsync(getRequest);

            Console.WriteLine($"Acceleration state = '{response.Status}' ");
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"Error occurred. Message:'{ex.Message}' when
setting transfer acceleration");
        }
    }
}

```

- Einzelheiten zur API finden Sie [PutBucketAccelerateConfiguration](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So legen Sie die Beschleunigungskonfiguration eines vorhandenen Buckets fest

Im folgenden Beispiel für `put-bucket-accelerate-configuration` wird die Accelerate-Konfiguration für den angegebenen Bucket aktiviert.

```
aws s3api put-bucket-accelerate-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --accelerate-configuration Status=Enabled
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [PutBucketAccelerateConfiguration](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl aktiviert die Übertragungsbeschleunigung für den angegebenen S3-Bucket.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Einzelheiten zur API finden Sie unter [PutBucketAccelerateConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl aktiviert die Übertragungsbeschleunigung für den angegebenen S3-Bucket.

```
$statusVal = New-Object Amazon.S3.BucketAccelerateStatus('Enabled')  
Write-S3BucketAccelerateConfiguration -BucketName 'amzn-s3-demo-bucket' -  
AccelerateConfiguration_Status $statusVal
```

- Einzelheiten zur API finden Sie unter [PutBucketAccelerateConfiguration AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketAc1** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketAc1 verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Verwalten von Zugriffssteuerungslisten \(ACL\)](#)

.NET

SDK für .NET

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <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;
    }

```

- Einzelheiten zur API finden Sie [PutBucketAcl](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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;
}

```

- Einzelheiten zur API finden Sie [PutBucketAcl](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

In diesem Beispiel werden zwei `full control` AWS Benutzern (`user1@example.com` und `user2@example.com`) und allen Benutzern die `read` folgenden Berechtigungen erteilt:

```

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

```

<http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> Einzelheiten zu benutzerdefinierten ACLs finden Sie unter (die `s3api`-ACL-Befehle verwenden z. `put-bucket-acl` B. dieselbe Kurzschreibweise für Argumente).

- Einzelheiten zur API finden Sie in der Befehlsreferenz. [PutBucketAcl](#) AWS CLI

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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);
        }
    }
}
```

- Einzelheiten zur API finden Sie [PutBucketAcl](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Legen Sie die Bucket-ACL fest.

```
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;
    }
  }
}

```

```
}  
};
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [PutBucketAcl](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
suspend fun setBucketAcl(  
    bucketName: String,  
    idVal: String,  
) {  
    val myGrant =  
        Grantee {  
            id = idVal  
            type = Type.CanonicalUser  
        }  
  
    val ownerGrant =  
        Grant {  
            grantee = myGrant  
            permission = Permission.FullControl  
        }  
  
    val grantList = mutableListOf<Grant>()  
    grantList.add(ownerGrant)  
  
    val ownerOb =  
        Owner {  
            id = idVal  
        }  
}
```

```

val acl =
    AccessControlPolicy {
        owner = ownerObj
        grants = grantList
    }

val request =
    PutBucketAclRequest {
        bucket = bucketName
        accessControlPolicy = acl
    }

S3Client.fromEnvironment { region = "us-east-1" }.use { s3 ->
    s3.putBucketAcl(request)
    println("An ACL was successfully set on $bucketName")
}
}

```

- Einzelheiten zur API finden Sie [PutBucketAcl](#) in der API-Referenz zum AWS SDK für Kotlin.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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

```

- Einzelheiten zur API finden Sie [PutBucketAcl](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [PutBucketAcl](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketCors** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketCors verwendet wird.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [PutBucketCors](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Im folgenden Beispiel werden PUT-, POST- und DELETE-Anfragen von www.example.com aktiviert und es werden GET-Anfragen von jeder beliebigen Domain aktiviert:


```
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
    }
  ]
}
```

- Einzelheiten zur API finden Sie [PutBucketCors](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);
}
}
}

```

- Einzelheiten zur API finden Sie [PutBucketCors](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Fügen Sie eine CORS-Regel hinzu.

```
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;
    }
}
};

```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [PutBucketCors](#) in der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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
```

- Einzelheiten zur API finden Sie [PutBucketCors](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie [PutBucketCors](#) in der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [PutBucketCors](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketEncryption** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketEncryption verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erste Schritte mit Objektspeicher](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <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;
}

```

- Einzelheiten zur API finden Sie [PutBucketEncryption](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So konfigurieren Sie die serverseitige Verschlüsselung für einen Bucket

Im folgenden Beispiel für `put-bucket-encryption` wird die AES256-Verschlüsselung als Standard für den angegebenen Bucket festgelegt.

```

aws s3api put-bucket-encryption \
    --bucket amzn-s3-demo-bucket \
    --server-side-encryption-configuration '{"Rules":
[{"ApplyServerSideEncryptionByDefault": {"SSEAlgorithm": "AES256"}}]}'

```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [PutBucketEncryption](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl aktiviert die serverseitige AES256-Standardverschlüsselung mit Amazon S3 Managed Keys (SSE-S3) für den angegebenen Bucket.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Einzelheiten zur API finden Sie unter [PutBucketEncryption AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl aktiviert die serverseitige AES256-Standardverschlüsselung mit Amazon S3 Managed Keys (SSE-S3) für den angegebenen Bucket.

```
$Encryptionconfig = @{ServerSideEncryptionByDefault =  
    @{ServerSideEncryptionAlgorithm = "AES256"}}  
Set-S3BucketEncryption -BucketName 'amzn-s3-demo-bucket' -  
ServerSideEncryptionConfiguration_ServerSideEncryptionRule $Encryptionconfig
```

- Einzelheiten zur API finden Sie [PutBucketEncryption](#) in der AWS -Tools für PowerShell Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketLifecycleConfiguration** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketLifecycleConfiguration verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Löschen von unvollständigen mehrteiligen Uploads](#)

- [Verwalten großer Nachrichten mit S3](#)
- [Mit versionierten Objekten arbeiten](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [PutBucketLifecycleConfiguration](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Der folgende Befehl wendet eine Lebenszykluskonfiguration auf einen Bucket mit dem Namen `amzn-s3-demo-bucket` an:

```
aws s3api put-bucket-lifecycle-configuration --bucket amzn-s3-demo-bucket --  
lifecycle-configuration file://lifecycle.json
```

Die Datei `lifecycle.json` ist ein JSON-Dokument im aktuellen Ordner, das zwei Regeln angibt:

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

Die erste Regel verschiebt Dateien mit dem Präfix `rotated` am angegebenen Datum nach Glacier. Die zweite Regel verschiebt alte Objektversionen nach Glacier, wenn sie nicht mehr

aktuell sind. Informationen zu akzeptablen Zeitstempelformaten finden Sie unter „Angaben von Parameterwerten“ im Benutzerhandbuch für AWS CLI.

- Einzelheiten zur API finden Sie [PutBucketLifecycleConfiguration](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);
    }
}
}

```

- Einzelheiten zur API finden Sie [PutBucketLifecycleConfiguration](#) in der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie [PutBucketLifecycleConfiguration](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [PutBucketLifecycleConfiguration](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketLogging** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketLogging verwendet wird.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
    }
}

```

- Einzelheiten zur API finden Sie [PutBucketLogging](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

Beispiel 1: So legen Sie die Protokollierung für eine Bucket-Richtlinie fest

Im folgenden Beispiel für `put-bucket-logging` wird die Protokollierungsrichtlinie für `amzn-s3-demo-bucket` festgelegt. Erteilen Sie zunächst dem Prinzipal des Protokollservices mit dem `put-bucket-policy`-Befehl die Berechtigung in Ihrer Bucket-Richtlinie.

```

aws s3api put-bucket-policy \
  --bucket amzn-s3-demo-bucket \
  --policy file://policy.json

```

Inhalt von `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"}
      }
    }
  ]
}

```

Wenden Sie dann die Protokollierungsrichtlinie mit `put-bucket-logging` an.

```
aws s3api put-bucket-logging \  
  --bucket amzn-s3-demo-bucket \  
  --bucket-logging-status file://logging.json
```

Inhalt von `logging.json`:

```
{  
  "LoggingEnabled": {  
    "TargetBucket": "amzn-s3-demo-bucket",  
    "TargetPrefix": "Logs/"  
  }  
}
```

Der `put-bucket-policy`-Befehl ist erforderlich, um dem Protokollierungsserviceprinzipal `s3:PutObject`-Berechtigungen zu erteilen.

Weitere Informationen finden Sie unter [Amazon-S3-Server-Zugriffsprotokoll](#) im Benutzerhandbuch für Amazon S3.

Beispiel 2: So legen Sie eine Bucket-Richtlinie für die Zugriffsprotokollierung auf einen einzelnen Benutzer fest

Im folgenden Beispiel für `put-bucket-logging` wird die Protokollierungsrichtlinie für `amzn-s3-demo-bucket` festgelegt. Der AWS Benutzer `bob@example.com` hat die volle Kontrolle über die Protokolldateien, und niemand sonst hat Zugriff. Erteilen Sie zunächst S3 die Berechtigung mit `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
```

Wenden Sie dann die Protokollierungsrichtlinie mit `put-bucket-logging` an.

```
aws s3api put-bucket-logging \  
  --bucket amzn-s3-demo-bucket \  
  --bucket-logging-status file://logging.json
```

Inhalt von `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"
      }
    ]
  }
}
```

Der Befehl `put-bucket-acl` wird benötigt, um dem System zur Bereitstellung der Amazon-S3-Protokolle die erforderlichen Berechtigungen („write-acp“ und „read-acp“) zu gewähren.

Weitere Informationen finden Sie unter [Amazon-S3-Server-Zugriffsprotokoll](#) im Entwicklerhandbuch für Amazon S3.

- Einzelheiten zur API finden Sie [PutBucketLogging](#) in der AWS CLI Befehlsreferenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **PutBucketNotification** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketNotification` verwendet wird.

CLI

AWS CLI

Im Beispiel wird eine Benachrichtigungskonfiguration auf einen Bucket namens `amzn-s3-demo-bucket` angewendet:

```
aws s3api put-bucket-notification --bucket amzn-s3-demo-bucket --notification-configuration file://notification.json
```

Die Datei `notification.json` ist eine JSON-Datei im aktuellen Ordner, die ein SNS-Thema und einen zu überwachenden Ereignistyp angibt:

```
{
  "TopicConfiguration": {
    "Event": "s3:ObjectCreated:*",
    "Topic": "arn:aws:sns:us-west-2:123456789012:s3-notification-topic"
  }
}
```

Dem SNS-Thema muss eine IAM-Richtlinie angehängt sein, die es Amazon S3 ermöglicht, darin zu veröffentlichen.

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

- Einzelheiten zur API finden Sie [PutBucketNotification](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: In diesem Beispiel wird die SNS-Themenkonfiguration für das S3-Ereignis `ObjectRemovedDelete` und die Benachrichtigung für den angegebenen S3-Bucket aktiviert

```
$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
```

Beispiel 2: Dieses Beispiel aktiviert Benachrichtigungen `ObjectCreatedAll` für den angegebenen Bucket und sendet ihn an die Lambda-Funktion.

```
$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
```

Beispiel 3: In diesem Beispiel werden zwei verschiedene Lambda-Konfigurationen auf der Grundlage unterschiedlicher Schlüsselsuffixe erstellt und beide in einem einzigen Befehl konfiguriert.

```
#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

```

- Einzelheiten zur API finden Sie unter [PutBucketNotification AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: In diesem Beispiel wird die SNS-Themenkonfiguration für das S3-Ereignis konfiguriert ObjectRemovedDelete und die Benachrichtigung für den angegebenen S3-Bucket aktiviert

```

$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

```

Beispiel 2: Dieses Beispiel aktiviert Benachrichtigungen ObjectCreatedAll für den angegebenen Bucket und sendet ihn an die Lambda-Funktion.

```

$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

```

Beispiel 3: In diesem Beispiel werden zwei verschiedene Lambda-Konfigurationen auf der Grundlage unterschiedlicher Schlüsselsuffixe erstellt und beide in einem einzigen Befehl konfiguriert.

```

#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

```

- Einzelheiten zur API finden Sie unter [PutBucketNotification AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketNotificationConfiguration** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketNotificationConfiguration verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Process-S3-Ereignisbenachrichtigungen](#)
- [Senden Sie Ereignisbenachrichtigungen an EventBridge](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [PutBucketNotificationConfiguration](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So aktivieren Sie die angegebenen Benachrichtigungen für einen Bucket

Im folgenden Beispiel für `put-bucket-notification-configuration` wird veranschaulicht, wie eine Benachrichtigungskonfiguration auf einen Bucket namens `amzn-s3-demo-bucket` angewendet wird. Die Datei `notification.json` ist eine JSON-Datei im aktuellen Ordner, die ein SNS-Thema und einen zu überwachenden Ereignistyp angibt:

```
aws s3api put-bucket-notification-configuration \  
  --bucket amzn-s3-demo-bucket \  
  --notification-configuration file://notification.json
```

Inhalt von `notification.json`:

```
{  
  "TopicConfigurations": [  
    {  
      "TopicArn": "arn:aws:sns:us-west-2:123456789012:s3-notification-  
topic",  
      "Events": [  
        "s3:ObjectCreated:*"  
      ]  
    }  
  ]  
}
```

Dem SNS-Thema muss eine IAM-Richtlinie angehängt sein, die es Amazon S3 ermöglicht, darin zu veröffentlichen.

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

- Einzelheiten zur API finden Sie [PutBucketNotificationConfiguration](#) in der AWS CLI Befehlsreferenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketPolicy** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketPolicy verwendet wird.

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                  const Aws::String &policyBody,
                                  const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
        Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
        s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}

//! Build a policy JSON string.
/*!
```



```

\param userArn: Aws user Amazon Resource Name (ARN).
    For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
\param bucketName: Name of a bucket.
\return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                           const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\", \n"
        "  \"Statement\": [\n"
        "    {\n"
        "      \"Sid\": \"1\", \n"
        "      \"Effect\": \"Allow\", \n"
        "      \"Principal\": {\n"
        "        \"AWS\": \"\"
        + userArn +
        "\"\"\"      }, \n"
        "      \"Action\": [ \"s3:getObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3::\"
        + bucketName +
        \"/*\" ] \n"
        "    } \n"
        "  ] \n"
        "}";
}

```

- Einzelheiten zur API finden Sie [PutBucketPolicy](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

In diesem Beispiel können alle Benutzer jedes Objekt im `amzn-s3-demo-bucket` abrufen, mit Ausnahme der Objekte im `MySecretFolder`. Es gewährt dem Root-Benutzer des `put` Kontos auch `delete` die folgenden Berechtigungen: `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/*"
    }
  ]
}

```

- Einzelheiten zur API finden Sie [PutBucketPolicy](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
}
}

```

- Einzelheiten zur API finden Sie [PutBucketPolicy](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Fügen Sie die Richtlinie hinzu.

```
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;
  }
}
};
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [PutBucketPolicy](#) in der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
```


- Einzelheiten zur API finden Sie [PutBucketPolicy](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
# 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
```

- Einzelheiten zur API finden Sie [PutBucketPolicy](#) in der AWS SDK für Ruby API-Referenz.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [PutBucketPolicy](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketReplication** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketReplication verwendet wird.

CLI

AWS CLI

So konfigurieren Sie die Replikation für einen S3-Bucket

Im folgenden Beispiel für `put-bucket-replication` wird eine Replikationskonfiguration auf den angegebenen S3-Bucket angewendet.

```
aws s3api put-bucket-replication \  
  --bucket amzn-s3-demo-bucket1 \  
  --replication-configuration file://replication.json
```

Inhalt von `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"  
      }  
    }  
  ]  
}
```

Für den Ziel-Bucket muss die Versionsverwaltung aktiviert sein. Die angegebene Rolle muss über Schreibberechtigungen für den Ziel-Bucket verfügen und eine Vertrauensbeziehung besitzen, die es Amazon S3 ermöglicht, die Rolle zu übernehmen.

Beispiel einer Rollenberechtigungsrichtlinie:

```
{  
  "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/*"
    }
  ]
}

```

Beispiel einer Vertrauensstellungsrichtlinie:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "s3.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Weitere Informationen finden Sie unter [Titel des Themas](#) im Benutzerhandbuch für Amazon Simple Storage Service Console.

- Einzelheiten zur API finden Sie [PutBucketReplication](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [PutBucketReplication](#) in der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieses Beispiel legt eine Replikationskonfiguration mit einer einzigen Regel fest, die die Replikation aller neuen Objekte, die mit dem Schlüsselnamenpräfix "" im Bucket 'amzn-s3-demo-bucket' erstellt wurden, im Bucket 'amzn-s3-demo-bucket' ermöglicht. TaxDocs

```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Beispiel 2: Dieses Beispiel legt eine Replikationskonfiguration mit mehreren Regeln fest, die die Replikation aller neuen Objekte, die entweder mit dem Schlüsselnamenpräfix "" oder "TaxDocs" erstellt wurden, in den Bucket 'amzn-s3-demo-bucket' ermöglichen. OtherDocs Die Schlüsselpräfixe dürfen sich nicht überschneiden.

```
$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

```

Beispiel 3: In diesem Beispiel wird die Replikationskonfiguration für den angegebenen Bucket aktualisiert, um die Regel zu deaktivieren, die die Replikation von Objekten mit dem Schlüsselnamenpräfix "" in den Bucket 'amzn-s3-demo-bucket' steuert. TaxDocs

```

$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

```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V4). [PutBucketReplication](#)AWS - Tools für PowerShell

Tools für V5 PowerShell

Beispiel 1: Dieses Beispiel legt eine Replikationskonfiguration mit einer einzigen Regel fest, die die Replikation aller neuen Objekte, die mit dem Schlüsselnamenpräfix "" im Bucket 'amzn-s3-demo-bucket' erstellt wurden, im Bucket 'amzn-s3-demo-bucket' ermöglicht. TaxDocs


```
$rule1 = New-Object Amazon.S3.Model.ReplicationRule
$rule1.ID = "Rule-1"
$rule1.Status = "Enabled"
$rule1.Prefix = "TaxDocs"
$rule1.Destination = @{ BucketArn = "arn:aws:s3:::amzn-s3-demo-destination-
bucket" }

$params = @{
    BucketName = "amzn-s3-demo-bucket"
    Configuration_Role = "arn:aws:iam::35667example:role/
CrossRegionReplicationRoleForS3"
    Configuration_Rule = $rule1
}

Write-S3BucketReplication @params
```

Beispiel 2: Dieses Beispiel legt eine Replikationskonfiguration mit mehreren Regeln fest, die die Replikation aller neuen Objekte, die entweder mit dem Schlüsselnamenpräfix "" oder "TaxDocs" erstellt wurden, in den Bucket 'amzn-s3-demo-bucket' ermöglichen. OtherDocs Die Schlüsselpräfixe dürfen sich nicht überschneiden.

```
$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

Beispiel 3: In diesem Beispiel wird die Replikationskonfiguration für den angegebenen Bucket aktualisiert, um die Regel zu deaktivieren, die die Replikation von Objekten mit dem Schlüsselnamenpräfix "" in den Bucket 'amzn-s3-demo-bucket' steuert. TaxDocs

```
$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
```

- Einzelheiten zur API finden Sie unter Cmdlet-Referenz (V5). [PutBucketReplication](#)AWS - Tools für PowerShell

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **PutBucketRequestPayment** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketRequestPayment verwendet wird.

CLI**AWS CLI**

Beispiel 1: So aktivieren Sie die „requester pays“-Konfiguration für einen Bucket

Im folgenden Beispiel für put-bucket-request-payment wird requester pays für den angegebenen Bucket aktiviert.

```
aws s3api put-bucket-request-payment \  
  --bucket amzn-s3-demo-bucket \  
  --request-payment-configuration '{"Payer":"Requester"}'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

Beispiel 1: So deaktivieren Sie die „requester pays“-Konfiguration für einen Bucket

Im folgenden Beispiel für `put-bucket-request-payment` wird `requester pays` für den angegebenen Bucket deaktiviert.

```
aws s3api put-bucket-request-payment \  
  --bucket amzn-s3-demo-bucket \  
  --request-payment-configuration '{"Payer":"BucketOwner"}'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [PutBucketRequestPayment](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Aktualisiert die Konfiguration für Zahlungsanforderungen für den Bucket mit dem Namen „amzn-s3-demo-bucket“, sodass der Person, die Downloads aus dem Bucket anfordert, die Kosten für den Download in Rechnung gestellt werden. Standardmäßig zahlt der Bucket-Eigentümer für Downloads. Verwenden Sie 'BucketOwner' für den Parameter `RequestPaymentConfiguration_Payer`, um die Zahlung für die Anforderung wieder auf die Standardwerte zurückzusetzen.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- Einzelheiten zur API finden Sie unter [PutBucketRequestPayment AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Aktualisiert die Konfiguration für Zahlungsanforderungen für den Bucket mit dem Namen „amzn-s3-demo-bucket“, sodass der Person, die Downloads aus dem Bucket anfordert, die Kosten für den Download in Rechnung gestellt werden. Standardmäßig zahlt

der Bucket-Eigentümer für Downloads. Verwenden Sie 'BucketOwner' für den Parameter `RequestPaymentConfiguration_Payer`, um die Zahlung für die Anforderung wieder auf die Standardwerte zurückzusetzen.

```
Write-S3BucketRequestPayment -BucketName amzn-s3-demo-bucket -  
RequestPaymentConfiguration_Payer Requester
```

- Einzelheiten zur API finden Sie unter [PutBucketRequestPayment AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwendung von **PutBucketTagging** mit einer CLI

Die folgenden Code-Beispiele zeigen, wie `PutBucketTagging` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erste Schritte mit Objektspeicher](#)

CLI

AWS CLI

Der folgende Befehl wendet eine Tagging-Konfiguration auf einen Bucket mit dem Namen `amzn-s3-demo-bucket` an:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging file://  
tagging.json
```

Die Datei `tagging.json` ist ein JSON-Dokument im aktuellen Ordner, das Tags angibt:

```
{  
  "TagSet": [  
    {  
      "Key": "organization",  
      "Value": "marketing"
```

```
}
]
}
```

Alternativ können Sie eine Tagging-Konfiguration für `amzn-s3-demo-bucket` direkt von der Befehlszeile aus anwenden:

```
aws s3api put-bucket-tagging --bucket amzn-s3-demo-bucket --tagging
'TagSet=[{Key=organization,Value=marketing}]'
```

- Einzelheiten zur API finden Sie [PutBucketTagging](#) in der AWS CLI Befehlsreferenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl wendet zwei Tags auf einen Bucket namens „**cloudtrail-test-2018**“ an: ein Tag mit dem Schlüssel „Stage“ und dem Wert „Test“ sowie ein Tag mit dem Schlüssel „Environment“ und dem Wert „Alpha“. Um zu überprüfen, ob die Tags zum Bucket hinzugefügt wurden, führen Sie **Get-S3BucketTagging -BucketName bucket_name** aus. Die Ergebnisse sollten die Tags anzeigen, die Sie im ersten Befehl auf den Bucket angewendet haben. Beachten Sie, dass **Write-S3BucketTagging** den gesamten vorhandenen Tag-Satz eines Buckets überschreibt. Um einzelne Tags hinzuzufügen oder zu löschen, führen Sie die Cmdlets Resource Groups und Tagging API aus, **Add-RGTResourceTag** und **Remove-RGTResourceTag**. Verwenden Sie alternativ den Tag-Editor in der AWS Management Console, um S3-Bucket-Tags zu verwalten.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage";
Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Beispiel 2: Dieser Befehl übergibt einen Bucket namens **cloudtrail-test-2018** in das **Write-S3BucketTagging**-Cmdlet. Es wendet Tags Stage:Production und auf Department:Finance den Bucket an. Beachten Sie, dass **Write-S3BucketTagging** den gesamten vorhandenen Tag-Satz eines Buckets überschreibt.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging
-TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department";
Value="Finance" } )
```

- Einzelheiten zur API finden Sie unter [PutBucketTagging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl wendet zwei Tags auf einen Bucket namens „**cloudtrail-test-2018**“ an: ein Tag mit dem Schlüssel „Stage“ und dem Wert „Test“ sowie ein Tag mit dem Schlüssel „Environment“ und dem Wert „Alpha“. Um zu überprüfen, ob die Tags zum Bucket hinzugefügt wurden, führen Sie **Get-S3BucketTagging -BucketName bucket_name** aus. Die Ergebnisse sollten die Tags anzeigen, die Sie im ersten Befehl auf den Bucket angewendet haben. Beachten Sie, dass **Write-S3BucketTagging** den gesamten vorhandenen Tag-Satz eines Buckets überschreibt. Um einzelne Tags hinzuzufügen oder zu löschen, führen Sie die Cmdlets Resource Groups und Tagging API aus, **Add-RGResourceTag** und **Remove-RGResourceTag**. Verwenden Sie alternativ den Tag-Editor in der AWS Management Console, um S3-Bucket-Tags zu verwalten.

```
Write-S3BucketTagging -BucketName amzn-s3-demo-bucket -TagSet @( @{ Key="Stage"; Value="Test" }, @{ Key="Environment"; Value="Alpha" } )
```

Beispiel 2: Dieser Befehl übergibt einen Bucket namens **cloudtrail-test-2018** in das **Write-S3BucketTagging**-Cmdlet. Es wendet Tags Stage:Production und auf Department:Finance den Bucket an. Beachten Sie, dass **Write-S3BucketTagging** den gesamten vorhandenen Tag-Satz eines Buckets überschreibt.

```
Get-S3Bucket -BucketName amzn-s3-demo-bucket | Write-S3BucketTagging -TagSet @( @{ Key="Stage"; Value="Production" }, @{ Key="Department"; Value="Finance" } )
```

- Einzelheiten zur API finden Sie unter [PutBucketTagging AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketVersioning** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketVersioning verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erste Schritte mit Objektspeicher](#)

CLI

AWS CLI

Mit dem folgenden Befehl wird die Versionsverwaltung auf dem `amzn-s3-demo-bucket`-Bucket aktiviert:

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled
```

Der folgende Befehl aktiviert die Versionierung und verwendet einen MFA-Code

```
aws s3api put-bucket-versioning --bucket amzn-s3-demo-bucket --versioning-configuration Status=Enabled --mfa "SERIAL 123456"
```

- Einzelheiten zur API finden Sie [PutBucketVersioning](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [PutBucketVersioning](#) in der AWS SDK for Java 2.x API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl aktiviert die Versionierung für den angegebenen S3-Bucket.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- Einzelheiten zur API finden Sie unter [PutBucketVersioning AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl aktiviert die Versionierung für den angegebenen S3-Bucket.

```
Write-S3BucketVersioning -BucketName 'amzn-s3-demo-bucket' -  
VersioningConfig_Status Enabled
```

- Einzelheiten zur API finden Sie unter [PutBucketVersioning AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [PutBucketVersioning](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketWebsite** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutBucketWebsite verwendet wird.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// Put the website configuration.
PutBucketWebsiteRequest putRequest = new
PutBucketWebsiteRequest()
{
    BucketName = bucketName,
    WebsiteConfiguration = new WebsiteConfiguration()
    {
        IndexDocumentSuffix = indexDocumentSuffix,
        ErrorDocument = errorDocument,
    },
};
PutBucketWebsiteResponse response = await
client.PutBucketWebsiteAsync(putRequest);
```

- Einzelheiten zur API finden Sie [PutBucketWebsite](#) in der AWS SDK für .NET API-Referenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
}

```

- Einzelheiten zur API finden Sie [PutBucketWebsite](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Es wird eine statische Webseitenkonfiguration auf einen Bucket mit dem Namen `amzn-s3-demo-bucket` angewendet:

```
aws s3api put-bucket-website --bucket amzn-s3-demo-bucket --website-configuration file://website.json
```

Die Datei `website.json` ist ein JSON-Dokument im aktuellen Ordner, das Index- und Fehlerseiten für die Webseite angibt:

```
{
  "IndexDocument": {
    "Suffix": "index.html"
  },
  "ErrorDocument": {
    "Key": "error.html"
  }
}
```

- Einzelheiten zur API finden Sie [PutBucketWebsite](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);
}
}
```

- Einzelheiten zur API finden Sie [PutBucketWebsite](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Legen Sie die Website-Konfiguration fest.

```
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;
  }
}
};
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [PutBucketWebsite](#) in der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl aktiviert das Website-Hosting für den angegebenen Bucket mit dem Indextdokument „index.html“ und dem Fehlerdokument „error.html“.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Einzelheiten zur API finden Sie unter [PutBucketWebsite AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl aktiviert das Website-Hosting für den angegebenen Bucket mit dem Indextdokument „index.html“ und dem Fehlerdokument „error.html“.

```
Write-S3BucketWebsite -BucketName 'amzn-s3-demo-bucket'
-WebsiteConfiguration_IndexDocumentSuffix 'index.html' -
WebsiteConfiguration_ErrorDocument 'error.html'
```

- Einzelheiten zur API finden Sie unter [PutBucketWebsite AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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__
```

- Einzelheiten zur API finden Sie [PutBucketWebsite](#) in der AWS SDK für Ruby API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutObject** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutObject verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Sie können diese Aktion in den folgenden Codebeispielen im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Erste Schritte mit Objektspeicher](#)
- [Senden von bedingten Anforderungen](#)
- [Verfolgen von Uploads und Downloads](#)
- [Arbeiten mit Amazon-S3-Objektintegrität](#)

.NET

SDK für .NET (v4)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [PutObject](#) in der AWS SDK für .NET API-Referenz.

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Laden Sie ein Objekt mit serverseitiger Verschlüsselung hoch.

```
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");
    }
}
```

```

    }
  }
}

```

Fügen Sie ein Objekt mithilfe einer bedingten Anforderung ein.

```

/// <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;
    }
}

```

- Einzelheiten zur API finden Sie [PutObject](#) in der AWS SDK für .NET API-Referenz.

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

#####
# 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
}
```

- Einzelheiten zur API finden Sie [PutObject](#) in der AWS CLI Befehlsreferenz.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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();
}

```

- Einzelheiten zur API finden Sie [PutObject](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Beispiel 1: So laden Sie ein Objekt auf Amazon S3 hoch

Der folgende put-object-Befehl lädt ein Objekt in Amazon S3 hoch.

```
aws s3api put-object \
```

```
--bucket amzn-s3-demo-bucket \  
--key my-dir/MySampleImage.png \  
--body MySampleImage.png
```

Weitere Informationen zum Hochladen von Objekten finden Sie unter Hochladen von Objekten < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html> > im Amazon S3 Developer Guide.

Beispiel 2: So laden Sie eine Videodatei auf Amazon S3 hoch

Der folgende `put-object`-Befehl lädt eine Videodatei hoch.

```
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
```

Weitere Informationen zum Hochladen von Objekten finden Sie unter Hochladen von Objekten < <http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html> > im Amazon S3 Developer Guide.

- Einzelheiten zur API finden Sie [PutObject](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Platzieren Sie ein Objekt in einem Bucket, indem Sie die Low-Level-API verwenden.

```
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
}
```

Laden Sie ein Objekt mithilfe eines Transfer-Managers in einen Bucket hoch.

```
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
}
```

- Einzelheiten zur API finden Sie [PutObject](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Laden Sie eine Datei mit einem [S3Client](#) in einen Bucket hoch.

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

Verwenden Sie an [S3TransferManager](#), um [eine Datei in einen Bucket hochzuladen](#). Sehen Sie sich die [vollständige Datei](#) an und [testen](#) Sie sie.

```
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();
    }
```

Laden Sie ein Objekt mit einem [S3Client](#) in einen Bucket hoch und legen Sie Tags fest.

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

```
}  
}
```

Laden Sie ein Objekt mit einem [S3Client](#) in einen Bucket hoch und legen Sie Metadaten fest.

```
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);
    }
}
}

```

Laden Sie ein Objekt mit einem [S3Client](#) in einen Bucket hoch und legen Sie einen Wert für die Objektaufbewahrung fest.

```

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);
    }
}
}

```

- Einzelheiten zur API finden Sie [PutObject](#) unter AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Laden Sie das Objekt hoch.

```

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;
    }
  }
};

```

Laden Sie das Objekt unter der Bedingung hoch, dass sein ETag mit dem angegebenen übereinstimmt.

```

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"));
  }
}
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).
- Einzelheiten zur API finden Sie [PutObject](#) in der AWS SDK für JavaScript API-Referenz.

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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}")
    }
}
```

- Einzelheiten zur API finden Sie [PutObject](#) in der API-Referenz zum AWS SDK für Kotlin.

PHP

SDK für PHP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Laden Sie ein Objekt in einen Bucket hoch.

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

- Einzelheiten zur API finden Sie [PutObject](#) in der AWS SDK für PHP API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Dieser Befehl lädt die einzelne Datei „local-sample.txt“ auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel „sample.txt“ im Bucket „test-files“.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Beispiel 2: Dieser Befehl lädt die einzelne Datei „sample.txt“ auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel „sample.txt“ im Bucket „test-files“. Wenn der Parameter -Key nicht angegeben wird, wird der Dateiname als S3-Objektschlüssel verwendet.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Beispiel 3: Dieser Befehl lädt die einzelne Datei "local-sample.txt" auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel "prefix/to/sample.txt" im Bucket „test-files“.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -  
File .\local-sample.txt
```

Beispiel 4: Dieser Befehl lädt alle Dateien im Unterverzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame key prefix "SampleScripts" auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename", wobei 'Dateiname' variiert.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\
```

Beispiel 5: Dieser Befehl lädt alle *.ps1-Dateien im lokalen Verzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame key prefix "" SampleScripts auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename.ps1", wobei 'Dateiname' variiert.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Beispiel 6: Dieser Befehl erstellt ein neues S3-Objekt, das die angegebene Inhaltszeichenfolge mit dem Schlüssel „sample.txt“ enthält.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Beispiel 7: Dieser Befehl lädt die angegebene Datei hoch (der Dateiname wird als Schlüssel verwendet) und wendet die angegebenen Tags auf das neue Objekt an.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
{Key="key1";Value="value1"},{Key="key2";Value="value2"}
```

Beispiel 8: Dieser Befehl lädt den angegebenen Ordner rekursiv hoch und wendet die angegebenen Tags auf alle neuen Objekte an.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"
               -Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Einzelheiten zur API finden Sie unter [PutObject AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Dieser Befehl lädt die einzelne Datei „local-sample.txt“ auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel „sample.txt“ im Bucket „test-files“.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -File .\local-
sample.txt
```

Beispiel 2: Dieser Befehl lädt die einzelne Datei „sample.txt“ auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel „sample.txt“ im Bucket „test-files“. Wenn der Parameter -Key nicht angegeben wird, wird der Dateiname als S3-Objektschlüssel verwendet.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File .\sample.txt
```

Beispiel 3: Dieser Befehl lädt die einzelne Datei "local-sample.txt" auf Amazon S3 hoch und erstellt ein Objekt mit dem Schlüssel "prefix/to/sample.txt" im Bucket „test-files“.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "prefix/to/sample.txt" -
File .\local-sample.txt
```

Beispiel 4: Dieser Befehl lädt alle Dateien im Unterverzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame key prefix "SampleScripts" auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename", wobei 'Dateiname' variiert.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix
SampleScripts\
```

Beispiel 5: Dieser Befehl lädt alle *.ps1-Dateien im lokalen Verzeichnis „Scripts“ in den Bucket „test-files“ hoch und wendet das gemeinsame key prefix "" SampleScripts auf jedes Objekt an. Jede hochgeladene Datei hat den Schlüssel "SampleScripts/filename.ps1", wobei 'Dateiname' variiert.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder .\Scripts -KeyPrefix  
SampleScripts\ -SearchPattern *.ps1
```

Beispiel 6: Dieser Befehl erstellt ein neues S3-Objekt, das die angegebene Inhaltszeichenfolge mit dem Schlüssel „sample.txt“ enthält.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Key "sample.txt" -Content "object  
contents"
```

Beispiel 7: Dieser Befehl lädt die angegebene Datei hoch (der Dateiname wird als Schlüssel verwendet) und wendet die angegebenen Tags auf das neue Objekt an.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -File "sample.txt" -TagSet  
@{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

Beispiel 8: Dieser Befehl lädt den angegebenen Ordner rekursiv hoch und wendet die angegebenen Tags auf alle neuen Objekte an.

```
Write-S3Object -BucketName amzn-s3-demo-bucket -Folder . -KeyPrefix "TaggedFiles"  
-Recurse -TagSet @{Key="key1";Value="value1"},@{Key="key2";Value="value2"}
```

- Einzelheiten zur API finden Sie unter [PutObject AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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()
```

Laden Sie ein Objekt mithilfe einer bedingten Anforderung hoch.

```
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

```

- Einzelheiten zur API finden Sie [PutObject](#) in AWS SDK for Python (Boto3) API Reference.

Ruby

SDK für Ruby

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Laden Sie eine Datei mit einem verwalteten Uploader (`Object.upload_file`) hoch.

```

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__

```

Laden Sie eine Datei hoch mit. `Object.put`

```

require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{@object.key}. Here's why:
    #{e.message}"
  end
end

```

```

    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

  wrapper = ObjectPutWrapper.new(Aws::S3::Object.new(bucket_name, object_key))
  success = wrapper.put_object(file_path)
  return unless success

  puts "Put file #{file_path} into #{object_key} in #{bucket_name}."
end

run_demo if $PROGRAM_NAME == __FILE__

```

Laden Sie eine Datei mit serverseitiger Verschlüsselung hoch `Object.put` und fügen Sie sie hinzu.

```

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__
```

- Einzelheiten zur API finden Sie [PutObject](#) unter AWS SDK für Ruby API-Referenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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)  
}
```

- Einzelheiten zur API finden Sie [PutObject](#) in der API-Referenz zum AWS SDK für Rust.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
"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.
```

- Einzelheiten zur API finden Sie [PutObject](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Swift

SDK für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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  
}  
}
```

- Einzelheiten zur API finden Sie [PutObject](#) in der API-Referenz zum AWS SDK für Swift.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutObjectAc1** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutObjectAc1 verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Verwalten von Zugriffssteuerungslisten \(ACL\)](#)

C++

SDK für C++

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.


```
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;
}

```

- Einzelheiten zur API finden Sie [PutObjectAcl](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl erhalten `full control` zwei AWS Benutzer (`user1@example.com` und `user2@example.com`) Zugriff `read` auf alle Benutzer:

```

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

```

<http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTacl.html> Einzelheiten zu benutzerdefinierten ACLs finden Sie unter (die `s3api-ACL`-Befehle verwenden z. `put-object-acl` B. dieselbe Kurzschreibweise für Argumente).

- Einzelheiten zur API finden Sie in der Befehlsreferenz. [PutObjectAcl](#) AWS CLI

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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

```

- Einzelheiten zur API finden Sie [PutObjectAcl](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [PutObjectAcl](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutObjectLegalHold** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutObjectLegalHold verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Sperren von Amazon-S3-Objekten](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

```
}
```

- Einzelheiten zur API finden Sie [PutObjectLegalHold](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So wenden Sie eine gesetzliche Aufbewahrungsfrist auf ein Objekt an

Im folgenden Beispiel für `put-object-legal-hold` wird für das Objekt `doc1.rtf` eine gesetzliche Aufbewahrungsfrist eingerichtet.

```
aws s3api put-object-legal-hold \  
  --bucket amzn-s3-demo-bucket-with-object-lock \  
  --key doc1.rtf \  
  --legal-hold Status=ON
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [PutObjectLegalHold](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}
```


- Einzelheiten zur API finden Sie [PutObjectLegalHold](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// 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 +".");
}
```

- Einzelheiten zur API finden Sie [PutObjectLegalHold](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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"));
  }
}

```

- Einzelheiten zur API finden Sie [PutObjectLegalHold](#) in der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Legen Sie eine rechtliche Aufbewahrungspflicht für ein Objekt fest.

```

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
        )

```

- Einzelheiten zur API finden Sie [PutObjectLegalHold](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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.

```

- Einzelheiten zur API finden Sie [PutObjectLegalHold](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutObjectLockConfiguration** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutObjectLockConfiguration verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Sperren von Amazon-S3-Objekten](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

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

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```

/// <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;
}
}
```

- Einzelheiten zur API finden Sie [PutObjectLockConfiguration](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So legen Sie die Object-Lock-Konfiguration eines Buckets fest

Im folgenden Beispiel für `put-object-lock-configuration` wird ein 50-tägiges Object Lock für den angegebenen Bucket eingerichtet.

```
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 }}}'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [PutObjectLockConfiguration](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

```
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
}

```

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```

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
}
```

- Einzelheiten zur API finden Sie [PutObjectLockConfiguration](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

```
// 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()
+ "'");
    }
}
```

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```
// 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
+".");
    }

```

- Einzelheiten zur API finden Sie [PutObjectLockConfiguration](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Legen Sie die Object-Lock-Konfiguration eines Buckets fest.

```

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"));
    }
}

```

Legen Sie die Standardaufbewahrungsfrist für einen Bucket fest.

```

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"));
    }
}
```

- Einzelheiten zur API finden Sie [PutObjectLockConfiguration](#) in der AWS SDK für JavaScript API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

PUT-Object Lock-Konfiguration.

```
s3_client.put_object_lock_configuration(  
    Bucket=bucket,  
    ObjectLockConfiguration={"ObjectLockEnabled": "Disabled", "Rule":  
{}},  
)
```

- Einzelheiten zur API finden Sie [PutObjectLockConfiguration](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [PutObjectLockConfiguration](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutObjectRetention** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie PutObjectRetention verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Sperren von Amazon-S3-Objekten](#)

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/// <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;
        }
    }
}

```

- Einzelheiten zur API finden Sie [PutObjectRetention](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So legen Sie die Konfiguration für die Aufbewahrung eines Objekts fest

Im folgenden Beispiel für `put-object-retention` wird eine Konfiguration für die Aufbewahrung des angegebenen Objekts bis zum 01.01.2025 festgelegt.

```

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" }'
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [PutObjectRetention](#) in der AWS CLI Befehlsreferenz.

Go

SDK für Go V2

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
}
```

- Einzelheiten zur API finden Sie [PutObjectRetention](#) in der AWS SDK für Go API-Referenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
// 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 +".");
}
```

- Einzelheiten zur API finden Sie [PutObjectRetention](#) in der AWS SDK for Java 2.x API-Referenz.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [PutObjectRetention](#) in der AWS SDK für JavaScript API-Referenz.

PowerShell

Tools für PowerShell V4

Beispiel 1: Der Befehl aktiviert den Governance-Aufbewahrungsmodus bis zum Datum „31st Dec 2019 00:00:00“ für das Objekt „testfile.txt“ im angegebenen S3-Bucket.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Einzelheiten zur API finden Sie unter [PutObjectRetention AWS -Tools für PowerShell](#) Cmdlet-Referenz (V4).

Tools für V5 PowerShell

Beispiel 1: Der Befehl aktiviert den Governance-Aufbewahrungsmodus bis zum Datum „31st Dec 2019 00:00:00“ für das Objekt „testfile.txt“ im angegebenen S3-Bucket.

```
Write-S3ObjectRetention -BucketName 'amzn-s3-demo-bucket' -Key 'testfile.txt' -
Retention_Mode GOVERNANCE -Retention_RetainUntilDate "2019-12-31T00:00:00"
```

- Einzelheiten zur API finden Sie unter [PutObjectRetention AWS -Tools für PowerShell](#) Cmdlet-Referenz (V5).

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Legen Sie eine Objektaufbewahrung fest.

```
s3_client.put_object_retention(  
    Bucket=bucket,  
    Key=key,  
    VersionId=version_id,  
    Retention={"Mode": "GOVERNANCE", "RetainUntilDate":  
far_future_date},  
    BypassGovernanceRetention=True,  
)
```

- Einzelheiten zur API finden Sie [PutObjectRetention](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [PutObjectRetention](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **RestoreObject** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie RestoreObject verwendet wird.

.NET

SDK für .NET

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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}");
}
}

```

- Einzelheiten zur API finden Sie [RestoreObject](#) in der AWS SDK für .NET API-Referenz.

CLI

AWS CLI

So erstellen Sie eine Wiederherstellungsanforderung für ein Objekt

Im folgenden Beispiel für `restore-object` wird das angegebene Objekt von Amazon S3 Glacier für den Bucket `my-glacier-bucket` für 10 Tage wiederhergestellt.

```
aws s3api restore-object \  
  --bucket my-glacier-bucket \  
  --key doc1.rtf \  
  --restore-request Days=10
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [RestoreObject](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);
        }
    }
}

```

- Einzelheiten zur API finden Sie [RestoreObject](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **SelectObjectContent** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie SelectObjectContent verwendet wird.

CLI

AWS CLI

So filtern Sie den Inhalt eines Amazon-S3-Objekts auf der Grundlage einer SQL-Anweisung

Im folgenden Beispiel für select-object-content wird das Objekt my-data-file.csv mit der angegebenen SQL-Anweisung gefiltert und die Ausgabe wird an eine Datei gesendet.

```
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"
```

Mit diesem Befehl wird keine Ausgabe zurückgegeben.

- Einzelheiten zur API finden Sie [SelectObjectContent](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Das folgende Beispiel zeigt eine Abfrage mit einem JSON-Objekt. Das [vollständige Beispiel](#) zeigt auch die Verwendung eines CSV-Objekts.

```
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:
{}\nbytesProcessed: {}\nbytesReturned:{}",
                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;  
    }  
}
```

- Einzelheiten zur API finden Sie [SelectObjectContent](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **UploadPart** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie UploadPart verwendet wird.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- [Verwenden der Prüfsummen](#)
- [Arbeiten mit Amazon-S3-Objektintegrität](#)

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
//! 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);
}

```

- Einzelheiten zur API finden Sie [UploadPart](#) in der AWS SDK für C++ API-Referenz.

CLI

AWS CLI

Mit dem folgenden Befehl wird der erste Teil eines mehrteiligen Uploads hochgeladen, der mit dem Befehl `create-multipart-upload` initiiert wurde:

```

aws s3api upload-part --bucket amzn-s3-demo-bucket --key
'multipart/01' --part-number 1 --body part01 --upload-id
'dfRtDYU0WMCCcH43C3WFbkR0NycyCpTJJvxu2i5GYkZLjF.Yxwh6XG7WfS2vC4to6HiV6Yjlx.cph0gtNBtJ8P'

```

Die `body`-Option verwendet den Namen oder Pfad einer lokalen Datei für den Upload (verwenden Sie nicht das Präfix `file://`). Die Mindestgröße von Teilen beträgt 5 MB. Die Upload-ID wird von `create-multipart-upload` zurückgegeben und kann auch mit `list-multipart-uploads` abgerufen werden. Bucket und Schlüssel werden angegeben, wenn Sie den mehrteiligen Upload erstellen.

Ausgabe:

```

{
  "ETag": "\"e868e0f4719e394144ef36531ee6824c\""
}

```

Speichern Sie den ETag-Wert der einzelnen Teile für später. Diese Werte werden benötigt, um den mehrteiligen Upload abzuschließen.

- Einzelheiten zur API finden Sie [UploadPart](#) in der AWS CLI Befehlsreferenz.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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?;
```

- Einzelheiten zur API finden Sie [UploadPart](#) in der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **UploadPartCopy** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie UploadPartCopy verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Erstellen einer mehrteiligen Kopie](#)

CLI

AWS CLI

So laden Sie einen Teil eines Objekts durch das Kopieren von Daten aus einem vorhandenen Objekt als Datenquelle hoch

Im folgenden Beispiel für upload-part-copy wird ein Teil durch Kopieren von Daten aus einem vorhandenen Objekt hochgeladen, das als Datenquelle dient.

```
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"
```

Ausgabe:

```
{  
  "CopyPartResult": {  
    "LastModified": "2019-12-13T23:16:03.000Z",  
    "ETag": "\"711470fc377698c393d94aed6305e245\""  
  }  
}
```

- Einzelheiten zur API finden Sie [UploadPartCopy](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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;  
}
```

- Einzelheiten zur API finden Sie [UploadPartCopy](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Szenarien für die Verwendung von Amazon S3 AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie gängige Szenarien in Amazon S3 mit AWS SDKs implementieren. Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von Amazon S3 oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- [Prüfen, ob ein Bucket vorhanden ist](#)
- [Prüfen Sie, ob ein Objekt existiert](#)
- [Konvertieren Sie Text in Sprache und zurück in Text mithilfe eines AWS SDK](#)
- [Erstellen Sie eine vorsignierte URL für Amazon S3 mit einem AWS SDK](#)
- [Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können](#)
- [Eine Webseite, die Amazon S3 S3-Objekte auflistet, und zwar mithilfe eines AWS SDK](#)
- [Erstellen Sie eine Amazon-Textextract-Explorer-Anwendung](#)
- [Löschen Sie alle Objekte in einem bestimmten Amazon S3 S3-Bucket mit einem AWS SDK](#)

- [Löschen Sie unvollständige mehrteilige Uploads auf Amazon S3 mit einem AWS SDK](#)
- [Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK](#)
- [Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mithilfe eines AWS SDK](#)
- [Erkennen Sie Gesichter in einem Bild mit einem AWS SDK](#)
- [Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK](#)
- [Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK](#)
- [Herunterladen von S3-Verzeichnissen aus einem Bucket in Amazon Simple Storage Service \(Amazon S3\)](#)
- [Herunterladen aller Objekte aus einem Amazon Simple Storage Service \(Amazon S3\)-Bucket in ein lokales Verzeichnis](#)
- [Laden Sie einen Stream unbekannter Größe von einem Amazon S3 S3-Objekt herunter, indem Sie eine AWS SDK](#)
- [Rufen Sie ein Amazon S3 S3-Objekt von einem Multi-Region Access Point ab, indem Sie eine AWS SDK](#)
- [Holen Sie sich ein Objekt aus einem Amazon S3 S3-Bucket mit einem AWS SDK, mit Angabe eines If-Modified-Since Headers](#)
- [Erste Schritte mit der Verschlüsselung von Amazon S3 S3-Objekten mithilfe eines AWS SDK](#)
- [Erste Schritte mit Tags für Amazon S3 S3-Objekte mithilfe eines AWS SDK](#)
- [Erste Schritte mit Clustern zur Verarbeitung großer Datenmengen](#)
- [Erste Schritte mit der Konfigurationsverwaltung](#)
- [Erste Schritte mit der Textextraktion von Dokumenten](#)
- [Erste Schritte mit Feature-Stores für maschinelles Lernen](#)
- [Erste Schritte mit Objektspeicher](#)
- [Erste Schritte mit Query Analytics](#)
- [Arbeiten Sie mit den Objektsperreffunktionen von Amazon S3 mithilfe eines AWS SDK](#)
- [Stellen Sie bedingte Amazon S3 S3-Anfragen mit einem AWS SDK](#)
- [Verwalten Sie Zugriffskontrolllisten \(ACLs\) für Amazon S3 S3-Buckets mithilfe eines AWS SDK](#)
- [Verwalten Sie große Amazon SQS SQS-Nachrichten mithilfe von Amazon S3 mit einem AWS SDK](#)

- [Verwalten Sie versionierte Amazon S3 S3-Objekte stapelweise mit einer Lambda-Funktion mithilfe eines AWS SDK](#)
- [Analysieren Sie Amazon S3 S3-URLs mit einem AWS SDK](#)
- [Erstellen Sie eine mehrteilige Kopie eines Amazon S3 S3-Objekts mit einem AWS SDK](#)
- [Empfangen und verarbeiten Sie Amazon S3 S3-Ereignisbenachrichtigungen mithilfe eines AWS SDK](#)
- [Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK](#)
- [Senden Sie S3-Ereignisbenachrichtigungen an Amazon EventBridge mit einem AWS SDK](#)
- [Verfolgen Sie den Upload oder Download eines Amazon S3 S3-Objekts mithilfe eines AWS SDK](#)
- [Transformieren von Daten für Ihre Anwendung mit S3 Object Lambda](#)
- [Beispielansätze für Einheiten- und Integrationstests mit einem AWS SDK](#)
- [Rekursives Hochladen eines lokalen Verzeichnisses in einen Amazon Simple Storage Service \(Amazon S3\)-Bucket](#)
- [Laden Sie große Dateien zu und von Amazon S3 hoch oder laden Sie sie herunter mit einem AWS SDK](#)
- [Laden Sie einen Stream unbekannter Größe in ein Amazon S3 S3-Objekt hoch, indem Sie eine AWS SDK](#)
- [Verwenden Sie Prüfsummen, um mit einem Amazon S3 S3-Objekt zu arbeiten, indem Sie AWS SDK](#)
- [Arbeiten Sie mit den Objektintegritätsfunktionen von Amazon S3 mithilfe eines AWS SDK](#)
- [Arbeiten Sie mit Objekten mit Amazon S3 S3-Version unter Verwendung eines AWS SDK](#)

Prüfen, ob ein Bucket vorhanden ist

Das folgende Codebeispiel zeigt, wie Sie überprüfen, ob ein Bucket vorhanden ist.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Sie können die folgende `doesBucketExists` Methode als Ersatz für die SDK for Java V1 [AmazonS3Client#doesBucketExist V2 \(String\)](#) -Methode verwenden.

```
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;
        }
    }
}

```

- Einzelheiten zur API finden Sie [GetBucketAcl](#) in AWS SDK for Java 2.x der API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Prüfen Sie, ob ein Objekt existiert

Das folgende Codebeispiel zeigt, wie überprüft wird, ob ein Objekt existiert.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Sie können die folgende `doesObjectExist` Methode als Ersatz für die SDK for Java V1 [AmazonS3Client #does ObjectExist \(String, String\)](#) -Methode verwenden.

```
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;
    }
}
}

```

- Einzelheiten zur API finden Sie [HeadObject](#) in AWS SDK for Java 2.x der API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Konvertieren Sie Text in Sprache und zurück in Text mithilfe eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Verwenden Sie Amazon Polly, um eine Klartext-Eingabedatei (UTF-8) zu einer Audiodatei zu synthetisieren.
- Laden Sie die Audiodatei in einen Amazon-S3-Bucket hoch.
- Konvertieren Sie die Audiodatei mit Amazon Transcribe in Text.
- Zeigen Sie den Text an.

Rust

SDK für Rust

Verwenden Sie Amazon Polly, um eine Nur-Text-Eingabedatei (UTF-8) in eine Audiodatei zu synthetisieren, die Audiodatei in einen Amazon S3 S3-Bucket hochzuladen, Amazon Transcribe zu verwenden, um diese Audiodatei in Text zu konvertieren und den Text anzuzeigen.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. [GitHub](#)

In diesem Beispiel verwendete Dienste

- Amazon Polly
- Amazon S3
- Amazon Transcribe

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erstellen Sie eine vorsignierte URL für Amazon S3 mit einem AWS SDK

Die folgenden Codebeispiele veranschaulichen, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Generieren Sie eine vorsignierte URL, die für einen begrenzten Zeitraum eine Amazon-S3-Aktion ausführen kann.

```
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/AmazonTAWSConfigsS3.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;
    }
}
```

Generieren Sie eine vorsignierte URL und führen Sie ein Upload mit dieser URL durch.

```
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 für .NET (v4)

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen und verwenden Sie vorsignierte POST URLs für direkte Browser-Uploads.

```
/// <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 für C++

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Generieren Sie eine vorsignierte URL, um ein Objekt herunterzuladen.

```
//! 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);
}

```

Führen Sie einen Download unter Verwendung von libcurl durch.

```

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;
}
```

Generieren Sie eine vorsignierte URL, um ein Objekt hochzuladen.

```
//! 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);
}

```

Führen Sie einen Upload unter Verwendung von libcurl durch.

```

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 für Go V2

 Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie Funktionen, die S3-Vorsignierungsaktionen umschließen.

```
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
}
```

Führen Sie ein interaktives Beispiel aus, das vorsignierte URLs generiert und verwendet, um ein S3-Objekt hochzuladen, herunterzuladen und zu löschen.

```
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))
}
```


Definieren Sie einen HTTP-Request-Wrapper, der im Beispiel verwendet wird, um HTTP-Anfragen zu stellen.

```
// 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 für Java 2.x

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Im Folgenden werden drei Beispiele gezeigt, wie vorsignierte URLs erstellt und die URLs mit HTTP-Clientbibliotheken verwendet werden:

- Eine HTTP-GET-Anfrage, die die URL mit drei HTTP-Clientbibliotheken verwendet
- Eine HTTP-GET-Anfrage mit Metadaten in den Headern, die die URL mit drei HTTP-Clientbibliotheken verwendet
- Eine HTTP-PUT-Anfrage mit Abfrageparametern, die die URL mit einer HTTP-Clientbibliothek verwendet

Generieren Sie eine vorsignierte URL für ein Objekt und laden Sie sie dann herunter (GET-Anfrage).

Importe.

```
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;
```

Generieren Sie die 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.presignedGetObject(presignRequest);
        logger.info("Presigned URL: [{}]",
presignedRequest.url().toString());
        logger.info("HTTP method: [{}]",
presignedRequest.httpRequest().method());

        return presignedRequest.url().toExternalForm();
    }
}

```

Laden Sie das Objekt herunter, indem Sie einen der folgenden drei Ansätze verwenden.

Verwenden Sie die JDK-Klasse `URLConnection` (seit v1.1), um den Download durchzuführen.

```

/* Use the JDK HttpURLConnection (since v1.1) class to do the download. */
public byte[] useHttpURLConnectionToGet(String presignedUrlString) {
    ByteArrayOutputStream byteArrayOutputStream = new
ByteArrayOutputStream(); // Capture the response body to a byte array.

    try {
        URL presignedUrl = new URL(presignedUrlString);
        HttpURLConnection connection = (HttpURLConnection)
presignedUrl.openConnection();
        connection.setRequestMethod("GET");
        // Download the result of executing the request.
        try (InputStream content = connection.getInputStream()) {
            IoUtils.copy(content, byteArrayOutputStream);
        }
        logger.info("HTTP response code is " + connection.getResponseCode());

    } catch (S3Exception | IOException e) {
        logger.error(e.getMessage(), e);
    }
    return byteArrayOutputStream.toByteArray();
}

```

Verwenden Sie die JDK-Klasse `HttpClient` (seit v11), um den Download durchzuführen.

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

Verwenden Sie die `SdkHttpClient` Klasse AWS SDK for Java, um den Download durchzuführen.

```
/* 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();
}

```

Generieren Sie eine vorsignierte URL mit Metadaten in den Headern für einen Upload und laden Sie dann eine Datei hoch (PUT-Anfrage).

Importe.

```

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;
```

Generieren Sie die 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();
    }
}

```

Laden Sie ein Dateiojekt herunter, indem Sie einen der folgenden drei Ansätze verwenden.

Verwenden Sie die JDK-Klasse `URLConnection` (seit v1.1), um den Upload durchzuführen.

```

/* 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);
}
}

```

Verwenden Sie die JDK-Klasse `HttpClient` (seit v11), um den Upload durchzuführen.

```

/* 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);
    }
}

```

Verwenden Sie die `SdkHttpClient` Klasse AWS for Java V2, um den Upload durchzuführen.

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

Generieren Sie eine vorsignierte URL mit Abfrageparametern für einen Upload und laden Sie dann eine Datei hoch (PUT-Anfrage).

Importe.

```
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;

```

Generieren Sie die 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();
        }
    }
}

```

Verwenden Sie die `SdkHttpClient` Klasse AWS for Java V2, um den Upload durchzuführen.

```

/**
 * Use the AWS SDK for Java V2 SdkHttpClient class to execute the PUT
 * request. Since the

```

```
* URL contains the query parameters, no headers are needed for metadata, SSE
settings, or ACL settings.
*
* @param presignedUrlString The URL for the PUT request.
* @param fileToPut File to upload
*/
public void useSdkHttpClientToPut(String presignedUrlString, File fileToPut)
{
    logger.info("Begin [{}] upload", fileToPut.toString());

    try {
        URL presignedUrl = new URL(presignedUrlString);

        SdkHttpRequest.Builder requestBuilder = SdkHttpRequest.builder()
            .method(SdkHttpMethod.PUT)
            .uri(presignedUrl.toURI());

        SdkHttpRequest request = requestBuilder.build();

        HttpExecuteRequest executeRequest = HttpExecuteRequest.builder()
            .request(request)
            .contentStreamProvider(new
FileContentStreamProvider(fileToPut.toPath()))
            .build();

        try (SdkHttpClient sdkHttpClient = ApacheHttpClient.create()) {
            HttpExecuteResponse response =
sdkHttpClient.prepareRequest(executeRequest).call();
            logger.info("Response code: {}",
response.httpResponse().statusCode());
        }
    } catch (URISyntaxException | IOException e) {
        logger.error(e.getMessage(), e);
    }
}
```

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie eine vorsignierte URL, um ein Objekt in einen Bucket hochzuladen.

```
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;
    }
  }
};
```

Erstellen Sie eine vorsignierte URL, um ein Objekt aus einem Bucket herunterzuladen.

```
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;
  }
}
};
```

- Weitere Informationen finden Sie im [AWS SDK für JavaScript -Entwicklerhandbuch](#).

Kotlin

SDK für Kotlin

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie eine vorsignierte GetObject-Anfrage und verwenden Sie die URL, um ein Objekt herunterzuladen.

```
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
}
```

Erstellen Sie eine GetObject-vorsignierte Anfrage mit erweiterten Optionen.

```
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
}
```

Erstellen Sie eine vorsignierte PutObject-Anfrage und verwenden Sie sie, um ein Objekt hochzuladen.

```
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)
}
```

- Weitere Informationen finden Sie im [Entwicklerhandbuch zum AWS SDK für Kotlin](#).

PHP

SDK für PHP

 Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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 für Python (Boto3)

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Generieren Sie eine vorsignierte URL, die für einen begrenzten Zeitraum eine S3-Aktion ausführen kann. Verwenden Sie das Anforderungspaket, um eine Anforderung mit der URL zu stellen.

```
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()

```

Generieren Sie eine vorsignierte POST-Anforderung zum Hochladen einer Datei.

```

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 für Ruby

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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 für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie Vorsignieranforderungen für GET-S3-Objekte.

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

Erstellen Sie Vorsignieranforderungen für GET- und PUT-S3-Objekte.

```
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 für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie vorsignierte Anforderungen für GET-S3-Objekte.

```

" 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( ).

```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Eine Anwendung für Foto-Asset-Management erstellen, mit der Benutzer Fotos mithilfe von Labels verwalten können

Die folgenden Codebeispiele zeigen, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels verwalten können.

.NET

SDK für .NET

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK für Java 2.x

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDK für JavaScript (v3)

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK für Kotlin

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK für PHP

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK für Rust

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der [AWS - Community](#).

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Eine Webseite, die Amazon S3 S3-Objekte auflistet, und zwar mithilfe eines AWS SDK

Im folgenden Codebeispiel wird veranschaulicht, wie Sie Amazon-S3-Objekte auf einer Webseite auflisten.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Der folgende Code ist die relevante React-Komponente, die das AWS SDK aufruft. Eine lauffähige Version der Anwendung, die diese Komponente enthält, finden Sie unter dem vorherigen GitHub Link.

```
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;

```

- Einzelheiten zur API finden Sie unter [ListObjects AWS SDK für JavaScript](#) API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erstellen Sie eine Amazon-Textract-Explorer-Anwendung

Die folgenden Code-Beispiele zeigen, wie man die Amazon-Textract-Ausgabe in einer interaktiven Anwendung untersuchen kann.

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Sie mit AWS SDK für JavaScript dem eine React-Anwendung erstellen, die Amazon Textract verwendet, um Daten aus einem Dokumentbild zu extrahieren und auf einer interaktiven Webseite anzuzeigen. Dieses Beispiel wird in einem Webbrowser ausgeführt und erfordert eine authentifizierte Amazon-Cognito-Identität für Anmeldeinformationen. Es verwendet Amazon Simple Storage Service (Amazon S3) zur Speicherung und fragt für Benachrichtigungen eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, die ein Amazon Simple Notification Service (Amazon SNS)-Thema abonniert hat.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK für Python (Boto3)

Zeigt, wie Sie AWS SDK für Python (Boto3) mit Amazon Textract Text-, Formular- und Tabellenelemente in einem Dokumentbild erkennen können. Das Eingabe-Image und die Amazon-Textract-Ausgabe werden in einer Tkinter-Anwendung angezeigt, mit der Sie die erkannten Elemente untersuchen können.

- Senden Sie ein Dokument-Image an Amazon Textract und untersuchen Sie die Ausgabe erkannter Elemente.

- Senden Sie Images direkt an Amazon Textract oder über einen Amazon Simple Storage Service (Amazon S3)-Bucket.
- Verwenden Sie asynchrone APIs, um einen Auftrag zu starten, der eine Benachrichtigung an ein Amazon Simple Notification Service (Amazon SNS)-Thema veröffentlicht.
- Stellen Sie eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange ab, um eine Meldung zum Abschluss des Auftrags zu erhalten.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Löschen Sie alle Objekte in einem bestimmten Amazon S3 S3-Bucket mit einem AWS SDK

Im folgenden Codebeispiel wird gezeigt, wie Sie alle Objekte aus einem Amazon-S3-Bucket löschen.

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu [GitHub](#). Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie alle Objekte für einen gegebenen Amazon-S3-Bucket.

```
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);
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
 - [DeleteObjects](#)
 - [ListObjectsV2](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Löschen Sie unvollständige mehrteilige Uploads auf Amazon S3 mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie unvollständige mehrteilige Amazon-S3-Uploads löschen oder beenden.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Um mehrteilige Uploads zu beenden, die gerade bearbeitet werden oder aus irgendeinem Grund unvollständig sind, können Sie eine Liste mit Uploads abrufen und sie dann löschen, wie im folgenden Beispiel gezeigt.

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

Um unvollständige mehrteilige Uploads zu löschen, die vor oder nach einem Datum initiiert wurden, können Sie mehrteilige Uploads anhand eines bestimmten Zeitpunkts selektiv löschen, wie im folgenden Beispiel gezeigt.


```

/**
 * 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);
            }
        }
    }
}

```

Wenn Sie nach dem Start eines mehrteiligen Uploads Zugriff auf die Upload-ID haben, können Sie den laufenden Upload mithilfe der ID löschen.

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

Um unvollständige mehrteilige Uploads, die älter als eine bestimmte Anzahl von Tagen sind, dauerhaft zu löschen, richten Sie eine Bucket-Lebenszykluskonfiguration für den Bucket ein. Das folgende Beispiel zeigt, wie Sie eine Regel erstellen, um unvollständige Uploads zu löschen, die älter als 7 Tage sind.

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

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [AbortMultipartUpload](#)
 - [ListMultipartUploads](#)
 - [PutBucketLifecycleConfiguration](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Ermitteln Sie persönliche Schutzausrüstung in Bildern mit Amazon Rekognition mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie eine App erstellt wird, die Amazon Rekognition verwendet, um persönliche Schutzausrüstung (PSA) auf Bildern zu erkennen.

Java

SDK für Java 2.x

Zeigt, wie eine AWS Lambda Funktion erstellt wird, die Bilder mit persönlicher Schutzausrüstung erkennt.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3
- Amazon SES

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erkennen Sie Entitäten in Text, der aus einem Bild extrahiert wurde, mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Amazon Comprehend verwendet wird, um Entitäten in Text zu erkennen, der von Amazon Textract aus einem in Amazon S3 gespeicherten Bild extrahiert wurde.

Python

SDK für Python (Boto3)

Zeigt, wie das AWS SDK für Python (Boto3) in einem Jupyter-Notizbuch verwendet wird, um Entitäten in Text zu erkennen, der aus einem Bild extrahiert wurde. In diesem Beispiel extrahiert Amazon Textract Text aus einem Bild, das in Amazon Simple Storage Service (Amazon S3) und Amazon Comprehend gespeichert ist, um Entitäten im extrahierten Text zu erkennen.

Dieses Beispiel ist ein Jupyter Notebook und muss in einer Umgebung ausgeführt werden, die Notebooks hosten kann. Anweisungen zur Ausführung des Beispiels mit Amazon SageMaker AI finden Sie in den Anweisungen unter [TextextractAndComprehendNotebook.ipynb](#).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Comprehend

- Amazon S3
- Amazon Textract

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erkennen Sie Gesichter in einem Bild mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Speichern Sie ein Bild in einem Amazon-S3-Bucket.
- Verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (z B. Lächeln) zu erkennen.
- Zeigen Sie diese Details an.

Rust

SDK für Rust

Speichern Sie das Bild in einem Amazon-S3-Bucket mit einem uploads-Präfix, verwenden Sie Amazon Rekognition, um Gesichtsdetails wie Altersgruppe, Geschlecht und Emotionen (Lächeln usw.) zu erkennen, und zeigen Sie diese Details an.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erkennen Sie Objekte in Bildern mit Amazon Rekognition mithilfe eines AWS SDK

Die folgenden Code-Beispiele zeigen, wie man eine App erstellt, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu erkennen.

.NET

SDK für .NET

Zeigt, wie Sie die Amazon-Rekognition-.NET-API verwenden, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Bucket von Amazon Simple Storage Service (Amazon S3) befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Rekognition

- Amazon S3
- Amazon SES

JavaScript

SDK für JavaScript (v3)

Zeigt, wie Amazon Rekognition zusammen mit dem verwendet wird, um eine App AWS SDK für JavaScript zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Lernen Sie Folgendes:

- Erstellen Sie mit Amazon Cognito einen nicht authentifizierten Benutzer.
- Analysieren Sie mit Amazon Rekognition Bilder für Objekte.
- Verifizieren Sie eine E-Mail-Adresse für Amazon SES.
- Senden Sie eine E-Mail-Benachrichtigung mit Amazon SES.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter. [GitHub](#)

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Kotlin

SDK für Kotlin

Zeigt, wie man die Amazon-Rekognition-Kotlin-API verwendet, um eine App zu erstellen, die Amazon Rekognition verwendet, um Objekte nach Kategorien in Bildern zu identifizieren, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Python

SDK für Python (Boto3)

Zeigt Ihnen, wie Sie mit AWS SDK für Python (Boto3) eine Webanwendung erstellen, mit der Sie Folgendes tun können:

- Laden Sie Fotos in einen Bucket von Amazon Simple Storage Service (Amazon S3) hoch.
- Verwenden Sie Amazon Rekognition, um die Fotos zu analysieren und zu markieren.
- Verwenden Sie Amazon Simple Email Service (Amazon SES), um E-Mail-Berichte von Bildanalysen zu senden.

Dieses Beispiel enthält zwei Hauptkomponenten: eine eingeschriebene Webseite JavaScript, die mit React erstellt wurde, und einen in Python geschriebenen REST-Dienst, der mit Flask-RESTful erstellt wurde.

Sie können die React-Webseite verwenden, um Folgendes auszuführen:

- Zeigen Sie eine Liste der Bilder an, die in Ihrem S3-Bucket gespeichert sind.
- Laden Sie Bilder von Ihrem Computer in Ihren S3-Bucket hoch.
- Zeigen Sie Bilder und Markierungen an, die Elemente identifizieren, welche im Bild erkannt werden.
- Rufen Sie einen Bericht über alle Bilder in Ihrem S3-Bucket ab und senden Sie eine E-Mail mit dem Bericht.

Die Webseite ruft den REST-Service auf. Der Service sendet Anforderungen an AWS, um die folgenden Aktionen durchzuführen:

- Die Liste der Bilder abrufen und in Ihrem S3-Bucket filtern.
- Fotos in Ihren S3-Bucket hochladen.

- Verwenden Sie Amazon Rekognition, um einzelne Fotos zu analysieren und eine Liste von Markierungen zu erhalten, die die auf dem Foto erkannten Elemente identifizieren.
- Analysieren Sie alle Fotos in Ihrem S3-Bucket und verwenden Sie Amazon SES, um einen Bericht per E-Mail zu senden.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erkennen Sie Personen und Objekte in einem Video mit Amazon Rekognition mithilfe eines AWS SDK

Die folgenden Code-Beispiele zeigen, wie man Personen und Objekte in einem Video mit Amazon Rekognition erkennt.

Java

SDK für Java 2.x

Zeigt, wie man die Amazon-Rekognition-Java-API verwendet, um eine App zu erstellen, die Gesichter und Objekte in Videos erkennt, die sich in einem Amazon Simple Storage Service (Amazon S3)-Bucket befinden. Die App sendet dem Administrator eine E-Mail-Benachrichtigung mit den Ergebnissen über Amazon Simple Email Service (Amazon SES).

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3

- Amazon SES
- Amazon SNS
- Amazon SQS

Python

SDK für Python (Boto3)

Verwenden Sie Amazon Rekognition, um Gesichter, Objekte und Personen in Videos zu erkennen, indem Sie asynchrone Erkennungsaufträge starten. In diesem Beispiel wird Amazon Rekognition auch so konfiguriert, dass es ein Amazon Simple Notification Service (Amazon SNS)-Thema benachrichtigt, wenn Aufträge abgeschlossen sind, und eine Amazon Simple Queue Service (Amazon SQS)-Warteschlange bei dem Thema abonniert. Wenn die Warteschlange eine Meldung über einen Job erhält, wird der Job abgerufen und die Ergebnisse werden ausgegeben.

Dieses Beispiel lässt sich am besten auf [ansehen GitHub](#). Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon Rekognition
- Amazon S3
- Amazon SES
- Amazon SNS
- Amazon SQS

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Herunterladen von S3-Verzeichnissen aus einem Bucket in Amazon Simple Storage Service (Amazon S3)

Das folgende Codebeispiel zeigt, wie Sie den Inhalt der Verzeichnisse von Amazon-S3-Buckets herunterladen und filtern.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Dieses Beispiel zeigt, wie Sie die [S3TransferManager](#) in verwenden AWS SDK for Java 2.x , um „Verzeichnisse“ aus einem Amazon S3 S3-Bucket herunterzuladen. Es zeigt auch, wie man es [DownloadFilters](#) in der Anfrage verwendet.

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

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Herunterladen aller Objekte aus einem Amazon Simple Storage Service (Amazon S3)-Bucket in ein lokales Verzeichnis

Das folgende Codebeispiel zeigt, wie Sie alle Objekte aus einem Amazon Simple Storage Service (Amazon S3)-Bucket in ein lokales Verzeichnis herunterladen.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Verwenden Sie an [S3TransferManager](#), um [alle S3-Objekte im selben S3-Bucket herunterzuladen](#). Sehen Sie sich die [vollständige Datei](#) an und [testen](#) Sie sie.

```
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();
    }
```

- Einzelheiten zur API finden Sie [DownloadDirectory](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Laden Sie einen Stream unbekannter Größe von einem Amazon S3 S3-Objekt herunter, indem Sie eine AWS SDK

Im folgenden Codebeispiel wird veranschaulicht, wie Sie einen Stream unbekannter Größe aus einem Amazon S3-Objekt herunterladen.

Swift

SDK für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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).")
}

```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Rufen Sie ein Amazon S3 S3-Objekt von einem Multi-Region Access Point ab, indem Sie eine AWS SDK

Das folgende Codebeispiel zeigt, wie ein Objekt von einem Multi-Region Access Point abgerufen wird.

Kotlin

SDK für Kotlin

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Konfigurieren Sie den S3-Client so, dass er den Signaturalgorithmus Asymmetric Sigv4 (SigV4a) verwendet.

```
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  
}
```

Verwenden Sie den Multi-Region Access Point-ARN anstelle eines Bucket-Namens, um das Objekt abzurufen.

```
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
    }
}
```

- Weitere Informationen finden Sie im [Entwicklerhandbuch zum AWS SDK für Kotlin](#).
- Einzelheiten zur API finden Sie [GetObject](#) in der API-Referenz zum AWS SDK für Kotlin.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Holen Sie sich ein Objekt aus einem Amazon S3 S3-Bucket mit einem AWS SDK, mit Angabe eines If-Modified-Since Headers

Das folgende Codebeispiel zeigt, wie Sie Daten aus einem Objekt in einem S3 Bucket lesen, jedoch nur, wenn dieser Bucket seit dem letzten Abruf nicht geändert wurde.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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(())
}

```

- Einzelheiten zur API finden Sie [GetObject](#) in der API-Referenz zum AWS SDK für Rust.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte mit der Verschlüsselung von Amazon S3 S3-Objekten mithilfe eines AWS SDK

Das folgende Codebeispiel veranschaulicht die ersten Schritte mit der Verschlüsselung von Amazon-S3-Objekten.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
using System;
using System.IO;
using System.Security.Cryptography;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to apply client encryption to an object in an
/// Amazon Simple Storage Service (Amazon S3) bucket.
/// </summary>
public class SSEClientEncryption
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket";
        string keyName = "exampleobject.txt";
        string copyTargetKeyName = "examplecopy.txt";
```



```
// If the AWS Region defined for your default user is different
// from the Region where your Amazon S3 bucket is located,
// pass the Region name to the Amazon S3 client object's constructor.
// For example: RegionEndpoint.USWest2.
IAmazonS3 client = new AmazonS3Client();

try
{
    // Create an encryption key.
    Aes aesEncryption = Aes.Create();
    aesEncryption.KeySize = 256;
    aesEncryption.GenerateKey();
    string base64Key = Convert.ToBase64String(aesEncryption.Key);

    // Upload the object.
    PutObjectRequest putObjectRequest = await
UploadObjectAsync(client, bucketName, keyName, base64Key);

    // Download the object and verify that its contents match what
you uploaded.
    await DownloadObjectAsync(client, bucketName, keyName, base64Key,
putObjectRequest);

    // Get object metadata and verify that the object uses AES-256
encryption.
    await GetObjectMetadataAsync(client, bucketName, keyName,
base64Key);

    // Copy both the source and target objects using server-side
encryption with
    // an encryption key.
    await CopyObjectAsync(client, bucketName, keyName,
copyTargetKeyName, aesEncryption, base64Key);
}
catch (AmazonS3Exception ex)
{
    Console.WriteLine($"Error: {ex.Message}");
}

}

/// <summary>
/// Uploads an object to an Amazon S3 bucket.
/// </summary>
```

```

    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// PutObjectAsync.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket to which
the
    /// object will be uploaded.</param>
    /// <param name="keyName">The name of the object to upload to the Amazon
S3
    /// bucket.</param>
    /// <param name="base64Key">The encryption key.</param>
    /// <returns>The PutObjectRequest object for use by
DownloadObjectAsync.</returns>
    public static async Task<PutObjectRequest> UploadObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string base64Key)
    {
        PutObjectRequest putObjectRequest = new PutObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,
            ContentBody = "sample text",
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };
        PutObjectResponse putObjectResponse = await
client.PutObjectAsync(putObjectRequest);
        return putObjectRequest;
    }

    /// <summary>
    /// Downloads an encrypted object from an Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// GetObjectAsync.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket where the
object
    /// is located.</param>
    /// <param name="keyName">The name of the Amazon S3 object to download.</
param>
    /// <param name="base64Key">The encryption key used to encrypt the

```

```

    /// object.</param>
    /// <param name="putObjectRequest">The PutObjectRequest used to upload
    /// the object.</param>
    public static async Task DownloadObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string base64Key,
        PutObjectRequest putObjectRequest)
    {
        GetObjectRequest getObjectRequest = new GetObjectRequest
        {
            BucketName = bucketName,
            Key = keyName,

            // Provide encryption information for the object stored in Amazon
S3.
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };

        using (GetObjectResponse getResponse = await
client.GetObjectAsync(getObjectRequest))
            using (StreamReader reader = new
StreamReader(getResponse.ResponseStream))
            {
                string content = reader.ReadToEnd();
                if (string.Compare(putObjectRequest.ContentBody, content) == 0)
                {
                    Console.WriteLine("Object content is same as we uploaded");
                }
                else
                {
                    Console.WriteLine("Error...Object content is not same.");
                }

                if (getResponse.ServerSideEncryptionCustomerMethod ==
ServerSideEncryptionCustomerMethod.AES256)
                {
                    Console.WriteLine("Object encryption method is AES256, same
as we set");
                }
                else

```

```

        {
            Console.WriteLine("Error...Object encryption method is not
the same as AES256 we set");
        }
    }

    /// <summary>
    /// Retrieves the metadata associated with an Amazon S3 object.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used
    /// to call GetObjectMetadataAsync.</param>
    /// <param name="bucketName">The name of the Amazon S3 bucket containing
the
    /// object for which we want to retrieve metadata.</param>
    /// <param name="keyName">The name of the object for which we wish to
    /// retrieve the metadata.</param>
    /// <param name="base64Key">The encryption key associated with the
    /// object.</param>
    public static async Task GetObjectMetadataAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string base64Key)
    {
        GetObjectMetadataRequest getObjectMetadataRequest = new
GetObjectMetadataRequest
        {
            BucketName = bucketName,
            Key = keyName,

            // The object stored in Amazon S3 is encrypted, so provide the
necessary encryption information.
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = base64Key,
        };

        GetObjectMetadataResponse getObjectMetadataResponse = await
client.GetObjectMetadataAsync(getObjectMetadataRequest);
        Console.WriteLine("The object metadata show encryption method used
is: {0}", getObjectMetadataResponse.ServerSideEncryptionCustomerMethod);
    }

```

```

    /// <summary>
    /// Copies an encrypted object from one Amazon S3 bucket to another.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// CopyObjectAsync.</param>
    /// <param name="bucketName">The Amazon S3 bucket containing the object
    /// to copy.</param>
    /// <param name="keyName">The name of the object to copy.</param>
    /// <param name="copyTargetKeyName">The Amazon S3 bucket to which the
object
    /// will be copied.</param>
    /// <param name="aesEncryption">The encryption type to use.</param>
    /// <param name="base64Key">The encryption key to use.</param>
    public static async Task CopyObjectAsync(
        IAmazonS3 client,
        string bucketName,
        string keyName,
        string copyTargetKeyName,
        Aes aesEncryption,
        string base64Key)
    {
        aesEncryption.GenerateKey();
        string copyBase64Key = Convert.ToBase64String(aesEncryption.Key);

        CopyObjectRequest copyRequest = new CopyObjectRequest
        {
            SourceBucket = bucketName,
            SourceKey = keyName,
            DestinationBucket = bucketName,
            DestinationKey = copyTargetKeyName,

            // Information about the source object's encryption.
            CopySourceServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            CopySourceServerSideEncryptionCustomerProvidedKey = base64Key,

            // Information about the target object's encryption.
            ServerSideEncryptionCustomerMethod =
ServerSideEncryptionCustomerMethod.AES256,
            ServerSideEncryptionCustomerProvidedKey = copyBase64Key,
        };
        await client.CopyObjectAsync(copyRequest);
    }

```

```
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
 - [CopyObject](#)
 - [GetObject](#)
 - [GetObjectMetadata](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte mit Tags für Amazon S3 S3-Objekte mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt die ersten Schritte mit Tags für Amazon-S3-Objekte.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [GetObjectTagging](#) in der AWS SDK für .NET API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte mit Clustern zur Verarbeitung großer Datenmengen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein EC2-Schlüsselpaar
- Richten Sie den Speicher ein und bereiten Sie Ihre Anwendung vor
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im [Tutorials und Beispiele](#)-Repository für Entwickler einrichten und ausführen.

```
#!/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."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
 - [AddSteps](#)
 - [Cp](#)
 - [CreateCluster](#)
 - [CreateDefaultRoles](#)
 - [CreateKeyPair](#)
 - [DescribeCluster](#)
 - [DescribeKeyPairs](#)
 - [DescribeStep](#)
 - [Ls](#)
 - [Mb](#)

- [Rb](#)
- [Rm](#)
- [SSH](#)
- [TerminateClusters](#)
- [Wait](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte mit der Konfigurationsverwaltung

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Amazon S3 S3-Bucket
- Erstellen Sie ein Amazon SNS-Thema.
- Erstellen Sie eine IAM-Rolle für Config
- Richten Sie den Config Configuration Recorder ein
- Richten Sie den Config-Lieferkanal ein
- Starten Sie den Konfigurationsrekorder
- Überprüfen Sie das Config-Setup

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im [Tutorials und Beispiele](#)-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# AWS Config Setup Script (v2)
```

```
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
}
```

```
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
"true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
"$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
"true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
"$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
"$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        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!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
 - [AttachRolePolicy](#)
 - [CreateBucket](#)
 - [CreateRole](#)

- [CreateTopic](#)
- [DeleteBucket](#)
- [DeleteConfigurationRecorder](#)
- [DeleteDeliveryChannel](#)
- [DeleteRole](#)
- [DeleteRolePolicy](#)
- [DeleteTopic](#)
- [DescribeConfigurationRecorderStatus](#)
- [DescribeConfigurationRecorders](#)
- [DescribeDeliveryChannels](#)
- [DetachRolePolicy](#)
- [GetCallerIdentity](#)
- [PutConfigurationRecorder](#)
- [PutDeliveryChannel](#)
- [PutPublicAccessBlock](#)
- [PutRolePolicy](#)
- [Rm](#)
- [StartConfigurationRecorder](#)
- [StopConfigurationRecorder](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte mit der Textextraktion von Dokumenten

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen eines S3-Buckets
- Laden Sie ein Dokument auf S3 hoch
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im [Tutorials und Beispiele](#)-Repository für Entwickler einrichten und ausführen.

```
#!/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
    fi
}
```



```
        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 $AWS_REGION..."
TEXTRACT_CHECK=$(aws textract help 2>&1)
TEXTRACT_CHECK_STATUS=$?
if [ $TEXTRACT_CHECK_STATUS -ne 0 ]; then
    echo "ERROR: Amazon Textract may not be available in region $AWS_REGION."
    exit 1
fi

# Generate a random identifier for S3 bucket
RANDOM_ID=$(openssl rand -hex 6)
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null || echo "")
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    BUCKET_NAME="textract-${RANDOM_ID}"
fi
DOCUMENT_NAME="document.png"
RESOURCES_CREATED=()

# Step 1: Create S3 bucket
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Creating S3 bucket: $BUCKET_NAME"
```

```

    CREATE_BUCKET_OUTPUT=$(aws s3 mb "s3://$BUCKET_NAME" --region "$AWS_REGION"
2>&1)
    CREATE_BUCKET_STATUS=$?
    echo "$CREATE_BUCKET_OUTPUT"
    check_error $CREATE_BUCKET_STATUS "$CREATE_BUCKET_OUTPUT" "aws s3 mb s3://
$BUCKET_NAME"

    aws s3api put-bucket-tagging \
        --bucket "$BUCKET_NAME" \
        --tagging 'TagSet=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-textract-gs}]'

    # Apply security settings to bucket
    aws s3api put-bucket-versioning --bucket "$BUCKET_NAME" --versioning-
configuration Status=Enabled 2>&1 || true
    aws s3api put-bucket-encryption --bucket "$BUCKET_NAME" --server-side-
encryption-configuration '{"Rules": [{"ApplyServerSideEncryptionByDefault":
{"SSEAlgorithm": "AES256"}]}]' 2>&1 || true
    aws s3api put-bucket-acl --bucket "$BUCKET_NAME" --acl private 2>&1 || true

    RESOURCES_CREATED+=("S3 Bucket: $BUCKET_NAME")
fi

# Step 2: Check if sample document exists, if not create a simple one
if [ ! -f "$DOCUMENT_NAME" ]; then
    echo "Sample document not found. Generating a sample document..."

    # Create a simple PNG document using ImageMagick or convert
    if command -v convert &> /dev/null; then
        convert -size 400x300 xc:white -pointsize 20 -fill black -draw "text
50,50 'Sample Document'" "$DOCUMENT_NAME"
        chmod 600 "$DOCUMENT_NAME"
        echo "Generated sample document: $DOCUMENT_NAME"
    else
        # Fallback: create a minimal valid PNG using base64
        echo "iVBORw0KGgoAAAANSUgAAAAEAAAABCAyAAAAFfcSJAAAADULEQVR42mNk
+M9QDwADhgGAWjR9awAAAABJRu5ErkJggg==" | base64 -d > "$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
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
 - [AnalyzeDocument](#)
 - [Cp](#)
 - [Hilfe](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte mit Feature-Stores für maschinelles Lernen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Einrichten von IAM-Berechtigungen
- Erstellen Sie eine SageMaker Ausführungsrolle
- Erstellt eine Funktionsgruppe.
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im [Tutorials und Beispiele](#)-Repository für Entwickler einrichten und ausführen.

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
```



```

        --output text)
    echo "Current status: ${status}"

    if [ "$status" = "Failed" ]; then
        handle_error "Feature group ${feature_group_name} creation failed"
    fi
done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
"$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)

```

```
        echo "Unknown resource type: $resource_type"
        ;;
    esac
done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
    Key=tutorial,Value=sagemaker-featurestore 2>&1
```

```
# Attach necessary policies
echo "Attaching policies to role..." >&2

# SageMaker execution policy
local policy1_result=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

if echo "$policy1_result" | grep -i "error" > /dev/null; then
    handle_error "Failed to attach SageMaker policy: $policy1_result"
fi

# S3 access policy
local policy2_result=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

if echo "$policy2_result" | grep -i "error" > /dev/null; then
    handle_error "Failed to attach S3 policy: $policy2_result"
fi

# Get account ID for role ARN
local account_id=$(aws sts get-caller-identity --query Account --output text)
local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

echo "Role ARN: $role_arn" >&2
echo "Waiting 10 seconds for role to propagate..." >&2
sleep 10

# Return only the role ARN to stdout
echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"
fi
```

```
# Validate the role ARN
ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\\///')
ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
    echo "Creating a new role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi

# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
```

```
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
            --region "$REGION" 2>&1)
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
            --region "$REGION" \
            --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
    fi
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
    'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
    "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
```

```

        exit 1
    fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-`${RANDOM_ID}`"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://${S3_BUCKET_NAME}/${PREFIX}"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

```

```

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-`${RANDOM_ID}`"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
  --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
  --record-identifier-feature-name "customer_id" \
  --event-time-feature-name "EventTime" \
  --feature-definitions '[
    {"FeatureName": "customer_id", "FeatureType": "Integral"},
    {"FeatureName": "order_id", "FeatureType": "String"},
    {"FeatureName": "order_date", "FeatureType": "String"},
    {"FeatureName": "product", "FeatureType": "String"},
    {"FeatureName": "quantity", "FeatureType": "Integral"},
    {"FeatureName": "amount", "FeatureType": "Fractional"},
    {"FeatureName": "EventTime", "FeatureType": "Fractional"}
  ]' \
  --online-store-config '{"EnableOnlineStore": true}' \
  --offline-store-config '{
    "S3StorageConfig": {
      "S3Uri": "s3://`${S3_BUCKET_NAME}`/`${PREFIX}`"
    },
    "DisableGlueTableCreation": false
  }' \
  --role-arn "$ROLE_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to create orders feature group: $ORDERS_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

```

```
# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "573291"},
    {"FeatureName": "name", "ValueAsString": "John Doe"},
    {"FeatureName": "age", "ValueAsString": "35"},
    {"FeatureName": "address", "ValueAsString": "123 Main St"},
    {"FeatureName": "membership_type", "ValueAsString": "premium"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
  --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
  --record '[
    {"FeatureName": "customer_id", "ValueAsString": "109382"},
    {"FeatureName": "name", "ValueAsString": "Jane Smith"},
    {"FeatureName": "age", "ValueAsString": "28"},
    {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
    {"FeatureName": "membership_type", "ValueAsString": "standard"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
  ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
  echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
  cleanup_resources
  exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
```



```

--feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
--record '[
    {"FeatureName": "customer_id", "ValueAsString": "573291"},
    {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
    {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
    {"FeatureName": "product", "ValueAsString": "Laptop"},
    {"FeatureName": "quantity", "ValueAsString": "1"},
    {"FeatureName": "amount", "ValueAsString": "1299.99"},
    {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"

```

```
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
| grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi
```

```

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "======"
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "======"
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "======"
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "======"
    echo "CLEANUP CONFIRMATION"
    echo "======"
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else

```

```
    echo "Skipping cleanup. Resources will remain in your account."
    echo "To clean up later, delete the following resources:"
    echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
$ORDERS_FEATURE_GROUP_NAME"
    echo "- S3 Bucket: $S3_BUCKET_NAME"
    if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
        echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
'IAMRole:[^[:space:]]*' | cut -d: -f2)"
    fi
    echo ""
    echo "Estimated ongoing cost: ~$0.01 per month for online store"
fi
fi

echo "Script completed at $(date)"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
 - [AttachRolePolicy](#)
 - [CreateBucket](#)
 - [CreateFeatureGroup](#)
 - [CreateRole](#)
 - [DeleteBucket](#)
 - [DeleteFeatureGroup](#)
 - [DeleteRole](#)
 - [DescribeFeatureGroup](#)
 - [DetachRolePolicy](#)
 - [GetCallerIdentity](#)
 - [GetRole](#)
 - [ListFeatureGroups](#)
 - [PutPublicAccessBlock](#)
 - [Rm](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte mit Objektspeicher

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie Ihren ersten S3-Bucket
- Hochladen eines Objekts
- Aktivieren von Versioning
- Konfigurieren Sie die Standardverschlüsselung
- Fügen Sie Ihrem Bucket Tags hinzu
- Listet Objekte und Versionen auf
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im [Tutorials und Beispiele](#)-Repository für Entwickler einrichten und ausführen.

```
#!/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
    fi
}

```



```

    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"
                }
            ]
        }' >/dev/null 2>&1;
    then
        echo "ERROR: Failed to tag bucket $BUCKET_NAME"
        exit 1
    fi
fi

```

```

        },
        {
            "Key": "tutorial",
            "Value": "s3-gettingstarted"
        }
    ]
}' >/dev/null 2>&1; then
    echo "WARNING: Failed to tag bucket"
fi
fi
echo ""

# =====
# Step 2: Upload a sample text file
# =====

echo "Step 2: Uploading a sample text file..."

SAMPLE_FILE="${TEMP_DIR}/sample.txt"
cat > "$SAMPLE_FILE" << 'EOF'
Hello, Amazon S3! This is a sample file for the getting started tutorial.
EOF

if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload sample.txt"
    exit 1
fi
echo "File uploaded."
echo ""

# =====
# Step 3: Download the object
# =====

echo "Step 3: Downloading the object..."

DOWNLOAD_FILE="${TEMP_DIR}/downloaded-sample.txt"
if ! aws s3api get-object \
    --bucket "$BUCKET_NAME" \

```

```

        --key "sample.txt" \
        "$DOWNLOAD_FILE" >/dev/null 2>&1; then
        echo "ERROR: Failed to download sample.txt"
        exit 1
    fi
    echo "Downloaded to: ${DOWNLOAD_FILE}"
    echo "Contents:"
    cat "$DOWNLOAD_FILE"
    echo ""

    # =====
    # Step 4: Copy the object to a folder prefix
    # =====

    echo "Step 4: Copying object to a folder prefix..."

    if ! aws s3api copy-object \
        --bucket "$BUCKET_NAME" \
        --copy-source "${BUCKET_NAME}/sample.txt" \
        --key "backup/sample.txt" \
        --server-side-encryption AES256 \
        --metadata-directive COPY >/dev/null 2>&1; then
        echo "ERROR: Failed to copy object to backup/sample.txt"
        exit 1
    fi
    echo "Object copied to backup/sample.txt."
    echo ""

    # =====
    # Step 5: Enable versioning and upload a second version
    # =====

    echo "Step 5: Enabling versioning..."

    if ! aws s3api put-bucket-versioning \
        --bucket "$BUCKET_NAME" \
        --versioning-configuration Status=Enabled >/dev/null 2>&1; then
        echo "ERROR: Failed to enable versioning"
        exit 1
    fi
    echo "Versioning enabled."

    echo "Uploading a second version of sample.txt..."
    cat > "$SAMPLE_FILE" << 'EOF'

```

```
Hello, Amazon S3! This is version 2 of the sample file.
```

```
EOF
```

```
if ! aws s3api put-object \
    --bucket "$BUCKET_NAME" \
    --key "sample.txt" \
    --body "$SAMPLE_FILE" \
    --server-side-encryption AES256 \
    --metadata "tutorial=s3-gettingstarted,version=2" >/dev/null 2>&1; then
    echo "ERROR: Failed to upload second version of sample.txt"
    exit 1
fi
```

```
echo "Second version uploaded."
```

```
echo ""
```

```
# =====
# Step 6: Configure SSE-S3 encryption
# =====
```

```
echo "Step 6: Configuring SSE-S3 default encryption..."
```

```
if ! aws s3api put-bucket-encryption \
    --bucket "$BUCKET_NAME" \
    --server-side-encryption-configuration '{
        "Rules": [
            {
                "ApplyServerSideEncryptionByDefault": {
                    "SSEAlgorithm": "AES256"
                },
                "BucketKeyEnabled": true
            }
        ]
    }' >/dev/null 2>&1; then
    echo "ERROR: Failed to configure SSE-S3 encryption"
    exit 1
fi
```

```
echo "SSE-S3 encryption configured."
echo ""
```

```
# =====
# Step 7: Block all public access
# =====
```

```
echo "Step 7: Blocking all public access..."
```

```

if ! aws s3api put-public-access-block \
  --bucket "$BUCKET_NAME" \
  --public-access-block-configuration '{
    "BlockPublicAcls": true,
    "IgnorePublicAcls": true,
    "BlockPublicPolicy": true,
    "RestrictPublicBuckets": true
  }' >/dev/null 2>&1; then
  echo "ERROR: Failed to block public access"
  exit 1
fi
echo "Public access blocked."
echo ""

# =====
# Step 8: Configure bucket logging
# =====

echo "Step 8: Configuring bucket logging..."

LOG_TARGET_BUCKET="${BUCKET_NAME}-logs"
if [ "$BUCKET_IS_SHARED" = "false" ]; then
  REGION=$(get_region)
  if [ "$REGION" = "us-east-1" ]; then
    aws s3api create-bucket --bucket "$LOG_TARGET_BUCKET" >/dev/null 2>&1 ||
true
  else
    aws s3api create-bucket \
      --bucket "$LOG_TARGET_BUCKET" \
      --region "$REGION" \
      --create-bucket-configuration LocationConstraint="$REGION" >/dev/null
2>&1 || true
  fi

  if ! aws s3api put-bucket-tagging \
    --bucket "$LOG_TARGET_BUCKET" \
    --tagging '{
      "TagSet": [
        {
          "Key": "project",
          "Value": "doc-smith"
        },
        {

```

```

        "Key": "tutorial",
        "Value": "s3-gettingstarted"
    }
]
}' >/dev/null 2>&1; then
echo "WARNING: Failed to tag log bucket"
fi

aws s3api put-bucket-acl --bucket "$LOG_TARGET_BUCKET" --acl log-delivery-
write 2>/dev/null || true

if ! aws s3api put-bucket-logging \
    --bucket "$BUCKET_NAME" \
    --bucket-logging-status '{
        "LoggingEnabled": {
            "TargetBucket": "'$LOG_TARGET_BUCKET'",
            "TargetPrefix": "logs/"
        }
    }' >/dev/null 2>&1; then
echo "WARNING: Failed to configure bucket logging"
else
echo "Bucket logging configured."
fi
else
echo "Skipping logging configuration for shared bucket."
fi
echo ""

# =====
# Step 9: Tag the bucket
# =====

echo "Step 9: Tagging the bucket..."

if ! aws s3api put-bucket-tagging \
    --bucket "$BUCKET_NAME" \
    --tagging '{
        "TagSet": [
            {
                "Key": "project",
                "Value": "doc-smith"
            },
            {
                "Key": "tutorial",

```

```

        "Value": "s3-gettingstarted"
    },
    {
        "Key": "Environment",
        "Value": "Tutorial"
    },
    {
        "Key": "Project",
        "Value": "S3-GettingStarted"
    },
    {
        "Key": "ManagedBy",
        "Value": "Bash-Tutorial"
    }
]
}' >/dev/null 2>&1; then
echo "ERROR: Failed to tag bucket"
exit 1
fi
echo "Bucket tagged."

echo "Verifying tags..."
if ! aws s3api get-bucket-tagging --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to retrieve bucket tags"
fi
echo ""

# =====
# Step 10: List objects and versions
# =====

echo "Step 10: Listing objects..."

if ! aws s3api list-objects-v2 --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list objects"
fi
echo ""

echo "Listing object versions..."

if ! aws s3api list-object-versions --bucket "$BUCKET_NAME" 2>&1; then
    echo "WARNING: Failed to list object versions"
fi
echo ""

```

```
# =====
# Step 11: Cleanup
# =====

echo ""
echo "=====
echo "TUTORIAL COMPLETE"
echo "=====
echo ""
echo "Resources created:"
if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
    for RESOURCE in "${CREATED_RESOURCES[@}"; do
        echo "  - ${RESOURCE}"
    done
else
    echo "  (none)"
fi
echo ""
echo "=====
echo "CLEANUP"
echo "=====
echo "Cleaning up all created resources..."
cleanup

echo ""
echo "Done."
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [HeadObject](#)
 - [ListObjectVersions](#)
 - [ListObjectsV2](#)
 - [PutBucketEncryption](#)

- [PutBucketTagging](#)
- [PutBucketVersioning](#)
- [PutObject](#)
- [PutPublicAccessBlock](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erste Schritte mit Query Analytics

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen S3-Bucket für Abfrageergebnisse
- Erstellen einer Datenbank.
- Erstellen einer Tabelle
- Ausführen einer -Abfrage
- Benannte Abfragen erstellen und verwenden
- Bereinigen von Ressourcen

Bash

AWS CLI mit Bash-Skript

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie es im [Tutorials und Beispiele](#)-Repository für Entwickler einrichten und ausführen.

```
#!/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.
$TABLE_NAME (
    \\`Date\\` DATE,
    Time STRING,
    Location STRING,
    Bytes INT,
    RequestIP STRING,
    Method STRING,
    Host STRING,
    Uri STRING,
    Status INT,
    Referrer STRING,
    os STRING,
    Browser STRING,
    BrowserVersion STRING
)
ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.RegexSerDe'
WITH SERDEPROPERTIES (
    \"input.regex\" = \"^(?!#)([ ^ ]+)\\\\\\\\s+([ ^ ]+)\\\\\\\\s+([ ^ ]+)\\\\\\\\s+([ ^ ]+)\\\\\\\\s+
+([ ^ ]+)\\\\\\\\s+([ ^ ]+)\\\\\\\\s+([ ^ ]+)\\\\\\\\s+([ ^ ]+)\\\\\\\\s+([ ^ ]+)\\\\\\\\s+([ ^ ]+)\\\\\\\\s+([ ^ ]+)\\\\\\\\s+[^
\\\\\\\\(]+[\\\\\\\\(]([ ^\\\\\\\\;]+).*\\\\\\\\%20([ ^\\\\\\\\\\/]+)[\\\\\\\\\\/](.*)$\"
) 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 [ "$QUERY_STATUS" = "SUCCEEDED" ]; then
        echo "Named query execution completed successfully."
        break
    elif [ "$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
            fi
        done
    fi
fi

```

```

        elif [ "$QUERY_STATUS" = "FAILED" ] || [ "$QUERY_STATUS" =
"CANCELLED" ]; then
            echo "Warning: Database deletion failed with status:
$QUERY_STATUS"
            break
        fi
        echo "Database deletion in progress, status: $QUERY_STATUS"
        sleep 2
        ((ELAPSED+=2))
    done
fi
fi

# Empty and delete S3 bucket (only if not shared)
if [ "$BUCKET_IS_SHARED" = false ]; then
    echo "Emptying S3 bucket: $S3_BUCKET"
    EMPTY_BUCKET_RESULT=$(aws s3 rm "s3://$S3_BUCKET" --recursive 2>&1)
    if echo "$EMPTY_BUCKET_RESULT" | grep -qi "error\|failed"; then
        echo "Warning: Failed to empty S3 bucket: $EMPTY_BUCKET_RESULT"
    else
        echo "S3 bucket emptied successfully."
    fi

    echo "Deleting S3 bucket: $S3_BUCKET"
    DELETE_BUCKET_RESULT=$(aws s3 rb "s3://$S3_BUCKET" 2>&1)
    if echo "$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!"
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS CLI - Befehlsreferenz.
 - [Cp](#)
 - [CreateNamedQuery](#)
 - [DeleteNamedQuery](#)
 - [GetNamedQuery](#)
 - [GetQueryExecution](#)
 - [GetQueryResults](#)
 - [ListDatabases](#)
 - [ListNamedQueries](#)
 - [ListTableMetadata](#)
 - [Mb](#)
 - [Rb](#)
 - [Rm](#)
 - [StartQueryExecution](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Arbeiten Sie mit den Objektsperreffunktionen von Amazon S3 mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie mit S3-Object-Lock-Features arbeiten.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Object-Lock zeigt.

```
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 =
                    $"\\nWould you like to add a legal hold to
{exampleFileName} in {bucketName}? (y/n)";
                if (GetYesNoResponse(question))
                {
                    // Set a legal hold.
                    await
                _s3ActionsWrapper.ModifyObjectLegalHold(bucketName, exampleFileName,
                ObjectLockLegalHoldStatus.On);

                }
                break;
            }
            case 1:
            {
                var question =
                    $"\\nWould you like to add a 1 day Governance
retention period to {exampleFileName} in {bucketName}? (y/n)" +
                    "\\nReminder: Only a user with the
s3:BypassGovernanceRetention permission will be able to delete this file or its
bucket until the retention period has expired.";
                if (GetYesNoResponse(question))
                {
                    // Set a Governance mode retention period for
                1 day.
                    await
                _s3ActionsWrapper.ModifyObjectRetentionPeriod(
                    bucketName, exampleFileName,
                    ObjectLockRetentionMode.Governance,
                    DateTime.UtcNow.AddDays(1));
                }
                break;
            }
        }
    }
}
}
}
Console.WriteLine(new string('-', 80));
return true;
```

```

    }

    /// <summary>
    /// List all of the current buckets and objects.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>The list of buckets and objects.</returns>
    public static async Task<List<S3ObjectVersion>> ListBucketsAndObjects(bool
interactive)
    {
        var allObjects = new List<S3ObjectVersion>();
        foreach (var bucketName in bucketNames)
        {
            var objectsInBucket = await
_s3ActionsWrapper.ListBucketObjectsAndVersions(bucketName);
            foreach (var objectKey in objectsInBucket.Versions)
            {
                allObjects.Add(objectKey);
            }
        }

        if (interactive)
        {
            Console.WriteLine("\nCurrent buckets and objects:\n");
            int i = 0;
            foreach (var bucketObject in allObjects)
            {
                i++;
                Console.WriteLine(
                    $"{i}: {bucketObject.Key} \n\tBucket:
{bucketObject.BucketName}\n\tVersion: {bucketObject.VersionId}");
            }
        }

        return allObjects;
    }

    /// <summary>
    /// Present the user with the demo action choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<bool> DemoActionChoices()
    {
        var choices = new string[]{

```

```
        "List all files in buckets.",
        "Attempt to delete a file.",
        "Attempt to delete a file with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the object and bucket retention settings for a file.",
        "View the legal hold settings for a file.",
        "Finish the scenario."};

var choice = 0;
// Keep asking the user until they choose to move on.
while (choice != 6)
{
    Console.WriteLine(new string('-', 80));
    choice = GetChoiceResponse(
        "\nExplore the S3 locking features by selecting one of the
following choices:"
        , choices);
    Console.WriteLine(new string('-', 80));
    switch (choice)
    {
        case 0:
        {
            await ListBucketsAndObjects(true);
            break;
        }
        case 1:
        {
            Console.WriteLine("\nEnter the number of the object to
delete:");

            var allFiles = await ListBucketsAndObjects(true);
            var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
            await
_s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, false, allFiles[fileChoice].VersionId);
            break;
        }
        case 2:
        {
            Console.WriteLine("\nEnter the number of the object to
delete:");

            var allFiles = await ListBucketsAndObjects(true);
            var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
```



```
        await
        _s3ActionsWrapper.DeleteObjectFromBucket(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, true, allFiles[fileChoice].VersionId);
        break;
    }
    case 3:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object to
overwrite:");
        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        // Create the file if it does not already exist.
        if (!File.Exists(allFiles[fileChoice].Key))
        {
            await using StreamWriter sw =
File.CreateText(allFiles[fileChoice].Key);
            await sw.WriteLineAsync(
                "This is a sample file for uploading to a
bucket.");
        }
        await
        _s3ActionsWrapper.UploadFileAsync(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key, allFiles[fileChoice].Key);
        break;
    }
    case 4:
    {
        var allFiles = await ListBucketsAndObjects(true);
        Console.WriteLine("\nEnter the number of the object and
bucket to view:");
        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
        _s3ActionsWrapper.GetObjectRetention(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
        await
        _s3ActionsWrapper.GetBucketObjectLockConfiguration(allFiles[fileChoice].BucketName);
        break;
    }
    case 5:
    {
        var allFiles = await ListBucketsAndObjects(true);
```

```

        Console.WriteLine("\nEnter the number of the object to
view:");

        var fileChoice = GetChoiceResponse(null,
allFiles.Select(f => f.Key).ToArray());
        await
_s3ActionsWrapper.GetObjectLegalHold(allFiles[fileChoice].BucketName,
allFiles[fileChoice].Key);
        break;
    }

    }
    }
    return true;
}

// <summary>
/// Clean up the resources from the scenario.
/// </summary>
/// <param name="interactive">True to run as interactive.</param>
/// <returns>True if successful.</returns>
public static async Task<bool> Cleanup(bool interactive)
{
    Console.WriteLine(new string('-', 80));

    if (!interactive || GetYesNoResponse("Do you want to clean up all files
and buckets? (y/n) "))
    {
        // Remove all locks and delete all buckets and objects.
        var allFiles = await ListBucketsAndObjects(false);
        foreach (var fileInfo in allFiles)
        {
            // Check for a legal hold.
            var legalHold = await
_s3ActionsWrapper.GetObjectLegalHold(fileInfo.BucketName, fileInfo.Key);
            if (legalHold?.Status?.Value == ObjectLockLegalHoldStatus.On)
            {
                await
_s3ActionsWrapper.ModifyObjectLegalHold(fileInfo.BucketName, fileInfo.Key,
ObjectLockLegalHoldStatus.Off);
            }

            // Check for a retention period.
            var retention = await
_s3ActionsWrapper.GetObjectRetention(fileInfo.BucketName, fileInfo.Key);

```

```

        var hasRetentionPeriod = retention?.Mode ==
ObjectLockRetentionMode.Governance && retention.RetainUntilDate >
DateTime.UtcNow.Date;
        await
_s3ActionsWrapper.DeleteObjectFromBucket(fileInfo.BucketName, fileInfo.Key,
hasRetentionPeriod, fileInfo.VersionId);
    }

    foreach (var bucketName in bucketNames)
    {
        await _s3ActionsWrapper.DeleteBucketByName(bucketName);
    }

}
else
{
    Console.WriteLine(
        "Ok, we'll leave the resources intact.\n" +
        "Don't forget to delete them when you're done with them or you
might incur unexpected charges."
    );
}

Console.WriteLine(new string('-', 80));
return true;
}

/// <summary>
/// Helper method to get a yes or no response from the user.
/// </summary>
/// <param name="question">The question string to print on the console.</
param>
/// <returns>True if the user responds with a yes.</returns>
private static bool GetYesNoResponse(string question)
{
    Console.WriteLine(question);
    var ynResponse = Console.ReadLine();
    var response = ynResponse != null && ynResponse.Equals("y",
StringComparison.InvariantCultureIgnoreCase);
    return response;
}

/// <summary>
/// Helper method to get a choice response from the user.

```

```
    /// </summary>
    /// <param name="question">The question string to print on the console.</param>
    /// <param name="choices">The choices to print on the console.</param>
    /// <returns>The index of the selected choice</returns>
    private static int GetChoiceResponse(string? question, string[] choices)
    {
        if (question != null)
        {
            Console.WriteLine(question);

            for (int i = 0; i < choices.Length; i++)
            {
                Console.WriteLine($"{i + 1}. {choices[i]}");
            }
        }

        var choiceNumber = 0;
        while (choiceNumber < 1 || choiceNumber > choices.Length)
        {
            var choice = Console.ReadLine();
            Int32.TryParse(choice, out choiceNumber);
        }

        return choiceNumber - 1;
    }
}
```

Eine Wrapper-Klasse für S3-Funktionen.

```
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($"{bucketName}\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($"\\tUnable to delete object {objectKey} in bucket
{bucketName}: " + ex.Message);
            return false;
        }
    }

    /// <summary>
    /// Delete a specific bucket.
    /// </summary>
    /// <param name="bucketName">The Amazon S3 bucket to use.</param>
    /// <param name="objectKey">The key of the object to delete.</param>
    /// <param name="versionId">Optional versionId.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteBucketByName(string bucketName)
    {
        try
        {
            var request = new DeleteBucketRequest() { BucketName = bucketName, };
            var response = await _amazonS3.DeleteBucketAsync(request);
            Console.WriteLine($"\\tDelete for {bucketName} complete.");
            return response.HttpStatusCode == HttpStatusCode.OK;
        }
        catch (AmazonS3Exception ex)
        {
            Console.WriteLine($"\\tUnable to delete bucket {bucketName}: " +
ex.Message);
            return false;
        }
    }
}

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)

- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

Go

SDK für Go V2

 Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Object-Lock zeigt.

```
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))
}

```

Definieren Sie eine Struktur, die in diesem Beispiel verwendete S3-Aktionen umschließt.

```
import (
```



```
"bytes"
"context"
"errors"
"fmt"
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking
// enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    }

    if enableObjectLock {
        input.ObjectLockEnabledForBucket = aws.Bool(true)
    }

    _, err := actor.S3Client.CreateBucket(ctx, input)
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", bucket)
        }
    }
}
```

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

return bucket, err
}

// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
    var status *types.ObjectLockLegalHoldStatus
    input := &s3.GetObjectLegalHoldInput{
        Bucket:    aws.String(bucket),
        Key:       aws.String(key),
        VersionId: aws.String(versionId),
    }

    output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
    if err != nil {
        var noSuchKeyErr *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noSuchKeyErr) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noSuchKeyErr
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "NoSuchObjectLockConfiguration":
                log.Printf("Object %s does not have an object lock configuration.\n", key)
                err = nil
            case "InvalidRequest":
                log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
                err = nil
            }
        }
    }
}
```

```
    }
  } else {
    status = &output.LegalHold.Status
  }

  return status, err
}

// GetObjectLockConfiguration retrieves the object lock configuration for an S3
// bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
  string) (*types.ObjectLockConfiguration, error) {
  var lockConfig *types.ObjectLockConfiguration
  input := &s3.GetObjectLockConfigurationInput{
    Bucket: aws.String(bucket),
  }

  output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
  if err != nil {
    var noBucket *types.NoSuchBucket
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noBucket) {
      log.Printf("Bucket %s does not exist.\n", bucket)
      err = noBucket
    } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
      "ObjectLockConfigurationNotFoundError" {
      log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
      err = nil
    }
  } else {
    lockConfig = output.ObjectLockConfiguration
  }

  return lockConfig, err
}

// GetObjectRetention retrieves the object retention configuration for an S3
// object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
  string) (*types.ObjectLockRetention, error) {
```

```

var retention *types.ObjectLockRetention
input := &s3.GetObjectRetentionInput{
    Bucket: aws.String(bucket),
    Key:    aws.String(key),
}

output, err := actor.S3Client.GetObjectRetention(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have locking enabled.", bucket)
            err = nil
        }
    }
} else {
    retention = output.Retention
}

return retention, err
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error
{
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:    aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
}

```

```
}

_, err := actor.S3Client.PutObjectLegalHold(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    }
}

return err
}

// ModifyDefaultBucketRetention modifies the default retention period of an
// existing bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }

    _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}
```

```
}

// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket
string) error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
            MFADelete: types.MFADeleteDisabled,
            Status:    types.BucketVersioningStatusEnabled,
        },
    }
    _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
        return err
    }

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: types.ObjectLockEnabledEnabled,
        },
    }
    _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}
```

```
// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
    string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
    error {
    input := &s3.PutObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        Retention: &types.ObjectLockRetention{
            Mode:          retentionMode,
            RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
        },
        BypassGovernanceRetention: aws.Bool(true),
    }

    _, err := actor.S3Client.PutObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key
    string, contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:      aws.String(bucket),
        Key:         aws.String(key),
        Body:        bytes.NewReader([]byte(contents)),
        ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
    }
    output, err := actor.S3Manager.Upload(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
        }
    }
}
```

```
    err = noBucket
}
} else {
    err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key,
            bucket)
    } else {
        outKey = *output.Key
    }
}
return outKey, err
}

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
    var versions []types.ObjectVersion
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
    for versionPaginator.HasMorePages() {
        output, err = versionPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
            break
        } else {
            versions = append(versions, output.Versions...)
        }
    }
    return versions, err
}
```



```
// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key
    string, versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "AccessDenied":
                log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
                err = nil
            case "InvalidArgument":
                if bypassGovernance {
                    log.Printf("You cannot specify bypass governance on a bucket without lock
                    enabled.")
                    err = nil
                }
            }
        }
    } else {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.
            \n", key, bucket)
        } else {
            deleted = true
        }
    }
}
```

```

    }
}
return deleted, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &types.Delete{
            Objects: objects,
            Quiet:   aws.Bool(true),
        },
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
    if err != nil || len(delOut.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucket)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
        }
        } else if len(delOut.Errors) > 0 {
            for _, outErr := range delOut.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *delOut.Errors[0].Message)
        }
    } else {
        for _, delObjs := range delOut.Deleted {
            err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
                time.Minute)
        }
    }
}

```

```
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to be deleted.\n",
*del0bjs.Key)
    } else {
        log.Printf("Deleted %s.\n", *del0bjs.Key)
    }
}
}
return err
}
```

Bereinigen von Ressourcen.

```
import (
    "context"
    "log"
    "s3_object_lock/actions"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
    name            string
    retentionEnabled bool
    objectKeys      []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario
and handles
// cleanup when the scenario finishes.
type Resources struct {
    demoBuckets map[string]*DemoBucket

    s3Actions *actions.S3Actions
}
```

```

    questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
    demotools.IQuestioner) {
    resources.s3Actions = s3Actions
    resources.questioner = questioner
    resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
            log.Println("Use the AWS Management Console to remove any remaining resources"
                " +
                "that were created for this scenario.")
        }
    }()

    wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
        resources that were created "+
        "during this demo (y/n)?", "y")
    if !wantDelete {
        log.Println("Be sure to remove resources when you're done with them to avoid
            unexpected charges!")
        return
    }

    log.Println("Removing objects from S3 buckets and deleting buckets...")
    resources.deleteBuckets(ctx)
    //resources.deleteRetentionObjects(resources.retentionBucket,
    resources.retentionObjects)

    log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
ObjectLockScenario.
func (resources *Resources) deleteBuckets(ctx context.Context) {
    for _, info := range createInfo {
        bucket := resources.demoBuckets[info.name]

```

```

resources.deleteObjects(ctx, bucket)
_, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
    Bucket: aws.String(bucket.name),
})
if err != nil {
    panic(err)
}
}
for _, info := range createInfo {
    bucket := resources.demoBuckets[info.name]
    err := s3.NewBucketNotExistsWaiter(resources.s3Actions.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket.name)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to be deleted.\n",
            bucket.name)
    } else {
        log.Printf("Deleted %s.\n", bucket.name)
    }
}
resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket
    *DemoBucket) {
    lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx,
        bucket.name)
    if err != nil {
        panic(err)
    }
    versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("No objects to get from %s.\n", bucket.name)
        default:
            panic(err)
        }
    }
    delObjects := make([]types.ObjectIdentifier, len(versions))
    for i, version := range versions {
        if lockConfig != nil && lockConfig.ObjectLockEnabled ==
            types.ObjectLockEnabledEnabled {

```

```

    status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
*version.Key, *version.VersionId)
    if err != nil {
        switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Couldn't determine legal hold status for %s in %s.\n",
*version.Key, bucket.name)
            default:
                panic(err)
        }
    } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
        err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
*version.VersionId, types.ObjectLockLegalHoldStatusOff)
        if err != nil {
            switch err.(type) {
                case *types.NoSuchKey:
                    log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
bucket.name)
                default:
                    panic(err)
            }
        }
    }
    }
    del0bjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.DeleteObjects(ctx, bucket.name, del0bjects,
bucket.retentionEnabled)
if err != nil {
    switch err.(type) {
        case *types.NoSuchBucket:
            log.Println("Nothing to delete.")
        default:
            panic(err)
    }
}
}

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für Go -API-Referenz.

- [GetObjectLegalHold](#)
- [GetObjectLockConfiguration](#)
- [GetObjectRetention](#)
- [PutObjectLegalHold](#)
- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Object-Lock zeigt.

```
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;
    }
}

```

Eine Wrapper-Klasse für S3-Funktionen.

```

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.s3.S3Client;
import software.amazon.awssdk.services.s3.model.BucketVersioningStatus;
import software.amazon.awssdk.services.s3.model.ChecksumAlgorithm;
import software.amazon.awssdk.services.s3.model.CreateBucketRequest;
import software.amazon.awssdk.services.s3.model.DefaultRetention;

```



```
import software.amazon.awssdk.services.s3.model.DeleteBucketRequest;
import software.amazon.awssdk.services.s3.model.DeleteObjectRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldRequest;
import software.amazon.awssdk.services.s3.model.GetObjectLegalHoldResponse;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationRequest;
import
    software.amazon.awssdk.services.s3.model.GetObjectLockConfigurationResponse;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.GetObjectRetentionResponse;
import software.amazon.awssdk.services.s3.model.HeadBucketRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsRequest;
import software.amazon.awssdk.services.s3.model.ListObjectVersionsResponse;
import software.amazon.awssdk.services.s3.model.MFADelete;
import software.amazon.awssdk.services.s3.model.ObjectLockConfiguration;
import software.amazon.awssdk.services.s3.model.ObjectLockEnabled;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHold;
import software.amazon.awssdk.services.s3.model.ObjectLockLegalHoldStatus;
import software.amazon.awssdk.services.s3.model.ObjectLockRetention;
import software.amazon.awssdk.services.s3.model.ObjectLockRetentionMode;
import software.amazon.awssdk.services.s3.model.ObjectLockRule;
import software.amazon.awssdk.services.s3.model.PutBucketVersioningRequest;
import software.amazon.awssdk.services.s3.model.PutObjectLegalHoldRequest;
import
    software.amazon.awssdk.services.s3.model.PutObjectLockConfigurationRequest;
import software.amazon.awssdk.services.s3.model.PutObjectRequest;
import software.amazon.awssdk.services.s3.model.PutObjectResponse;
import software.amazon.awssdk.services.s3.model.PutObjectRetentionRequest;
import software.amazon.awssdk.services.s3.model.S3Exception;
import software.amazon.awssdk.services.s3.model.VersioningConfiguration;
import software.amazon.awssdk.services.s3.waiters.S3Waiter;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.time.Instant;
import java.time.ZoneId;
import java.time.ZonedDateTime;
import java.time.format.DateTimeFormatter;
import java.time.temporal.ChronoUnit;
import java.util.List;
import java.util.concurrent.atomic.AtomicInteger;
import java.util.stream.Collectors;

// Contains application logic for the Amazon S3 operations used in this workflow.
public class S3LockActions {
```

```
private static S3Client getClient() {
    return S3Client.builder()
        .region(Region.US_EAST_1)
        .build();
}

// Set or modify a retention period on an object in an S3 bucket.
public void modifyObjectRetentionPeriod(String bucketName, String objectKey)
{
    // Calculate the instant one day from now.
    Instant futureInstant = Instant.now().plus(1, ChronoUnit.DAYS);

    // Convert the Instant to a ZonedDateTime object with a specific time
    zone.
    ZonedDateTime zonedDateTime =
futureInstant.atZone(ZoneId.systemDefault());

    // Define a formatter for human-readable output.
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");

    // Format the ZonedDateTime object to a human-readable date string.
    String humanReadableDate = formatter.format(zonedDateTime);

    // Print the formatted date string.
    System.out.println("Formatted Date: " + humanReadableDate);
    ObjectLockRetention retention = ObjectLockRetention.builder()
        .mode(ObjectLockRetentionMode.GOVERNANCE)
        .retainUntilDate(futureInstant)
        .build();

    PutObjectRetentionRequest retentionRequest =
PutObjectRetentionRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .retention(retention)
        .build();

    getClient().putObjectRetention(retentionRequest);
    System.out.println("Set retention for "+objectKey +" in " +bucketName +"
until "+ humanReadableDate +".");
}
```

```
// Get the legal hold details for an S3 object.
public ObjectLockLegalHold getObjectLegalHold(String bucketName, String
objectKey) {
    try {
        GetObjectLegalHoldRequest legalHoldRequest =
GetObjectLegalHoldRequest.builder()
            .bucket(bucketName)
            .key(objectKey)
            .build();

        GetObjectLegalHoldResponse response =
getClient().getObjectLegalHold(legalHoldRequest);
        System.out.println("Object legal hold for " + objectKey + " in " +
bucketName +
            ":\n\tStatus: " + response.legalHold().status());
        return response.legalHold();

    } catch (S3Exception ex) {
        System.out.println("\tUnable to fetch legal hold: '" +
ex.getMessage() + "'");
    }

    return null;
}

// Create a new Amazon S3 bucket with object lock options.
public void createBucketWithLockOptions(boolean enableObjectLock, String
bucketName) {
    S3Waiter s3Waiter = getClient().waiter();
    CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
        .bucket(bucketName)
        .objectLockEnabledForBucket(enableObjectLock)
        .build();

    getClient().createBucket(bucketRequest);
    HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
        .bucket(bucketName)
        .build();

    // Wait until the bucket is created and print out the response.
    s3Waiter.waitUntilBucketExists(bucketRequestWait);
    System.out.println(bucketName + " is ready");
}
```

```

    public List<S3InfoObject> listBucketsAndObjects(List<String> bucketNames,
Boolean interactive) {
        AtomicInteger counter = new AtomicInteger(0); // Initialize counter.
        return bucketNames.stream()
            .flatMap(bucketName ->
listBucketObjectsAndVersions(bucketName).versions().stream()
                .map(version -> {
                    S3InfoObject s3InfoObject = new S3InfoObject();
                    s3InfoObject.setBucketName(bucketName);
                    s3InfoObject.setVersion(version.versionId());
                    s3InfoObject.setKeyName(version.key());
                    return s3InfoObject;
                })))
            .peek(s3InfoObject -> {
                int i = counter.incrementAndGet(); // Increment and get the
updated value.
                if (interactive) {
                    System.out.println(i + ": " + s3InfoObject.getKeyName());
                    System.out.printf("%5s Bucket name: %s\n", "",
s3InfoObject.getBucketName());
                    System.out.printf("%5s Version: %s\n", "",
s3InfoObject.getVersion());
                }
            })
            .collect(Collectors.toList());
    }

    public ListObjectVersionsResponse listBucketObjectsAndVersions(String
bucketName) {
        ListObjectVersionsRequest versionsRequest =
ListObjectVersionsRequest.builder()
            .bucket(bucketName)
            .build();

        return getClient().listObjectVersions(versionsRequest);
    }

    // Set or modify a retention period on an S3 bucket.
    public void modifyBucketDefaultRetention(String bucketName) {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .mfaDelete(MFADelete.DISABLED)
            .status(BucketVersioningStatus.ENABLED)
            .build();
    }

```

```
        PutBucketVersioningRequest versioningRequest =
PutBucketVersioningRequest.builder()
    .bucket(bucketName)
    .versioningConfiguration(versioningConfiguration)
    .build();

getClient().putBucketVersioning(versioningRequest);
DefaultRetention rention = DefaultRetention.builder()
    .days(1)
    .mode(ObjectLockRetentionMode.GOVERNANCE)
    .build();

ObjectLockRule lockRule = ObjectLockRule.builder()
    .defaultRetention(rention)
    .build();

ObjectLockConfiguration objectLockConfiguration =
ObjectLockConfiguration.builder()
    .objectLockEnabled(ObjectLockEnabled.ENABLED)
    .rule(lockRule)
    .build();

PutObjectLockConfigurationRequest putObjectLockConfigurationRequest =
PutObjectLockConfigurationRequest.builder()
    .bucket(bucketName)
    .objectLockConfiguration(objectLockConfiguration)
    .build();

getClient().putObjectLockConfiguration(putObjectLockConfigurationRequest) ;
    System.out.println("Added a default retention to bucket "+bucketName
+ ".");
}

// Enable object lock on an existing bucket.
public void enableObjectLockOnBucket(String bucketName) {
    try {
        VersioningConfiguration versioningConfiguration =
VersioningConfiguration.builder()
            .status(BucketVersioningStatus.ENABLED)
            .build();
```

```
        PutBucketVersioningRequest putBucketVersioningRequest =
PutBucketVersioningRequest.builder()
            .bucket(bucketName)
            .versioningConfiguration(versioningConfiguration)
            .build();

        // Enable versioning on the bucket.
        getClient().putBucketVersioning(putBucketVersioningRequest);
        PutObjectLockConfigurationRequest request =
PutObjectLockConfigurationRequest.builder()
            .bucket(bucketName)
            .objectLockConfiguration(ObjectLockConfiguration.builder()
                .objectLockEnabled(ObjectLockEnabled.ENABLED)
                .build())
            .build();

        getClient().putObjectLockConfiguration(request);
        System.out.println("Successfully enabled object lock on
"+bucketName);

    } catch (S3Exception ex) {
        System.out.println("Error modifying object lock: '" + ex.getMessage()
+ "'");
    }
}

public void uploadFile(String bucketName, String objectName, String filePath)
{
    Path file = Paths.get(filePath);
    PutObjectRequest request = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(objectName)
        .checksumAlgorithm(ChecksumAlgorithm.SHA256)
        .build();

    PutObjectResponse response = getClient().putObject(request, file);
    if (response != null) {
        System.out.println("\tSuccessfully uploaded " + objectName + " to " +
bucketName + ".");
    } else {
        System.out.println("\tCould not upload " + objectName + " to " +
bucketName + ".");
    }
}
```

```
// Set or modify a legal hold on an object in an S3 bucket.
public void modifyObjectLegalHold(String bucketName, String objectKey,
boolean legalHoldOn) {
    ObjectLockLegalHold legalHold ;
    if (legalHoldOn) {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.ON)
            .build();
    } else {
        legalHold = ObjectLockLegalHold.builder()
            .status(ObjectLockLegalHoldStatus.OFF)
            .build();
    }

    PutObjectLegalHoldRequest legalHoldRequest =
PutObjectLegalHoldRequest.builder()
        .bucket(bucketName)
        .key(objectKey)
        .legalHold(legalHold)
        .build();

    getClient().putObjectLegalHold(legalHoldRequest) ;
    System.out.println("Modified legal hold for "+ objectKey +" in
"+bucketName +".");
}

// Delete an object from a specific bucket.
public void deleteObjectFromBucket(String bucketName, String objectKey,
boolean hasRetention, String versionId) {
    try {
        DeleteObjectRequest objectRequest;
        if (hasRetention) {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
                .bypassGovernanceRetention(true)
                .build();
        } else {
            objectRequest = DeleteObjectRequest.builder()
                .bucket(bucketName)
                .key(objectKey)
                .versionId(versionId)
```

```
        .build();
    }

    getClient().deleteObject(objectRequest) ;
    System.out.println("The object was successfully deleted");

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

}

// Get the retention period for an S3 object.
public ObjectLockRetention getObjectRetention(String bucketName, String key){
    try {
        GetObjectRetentionRequest retentionRequest =
GetObjectRetentionRequest.builder()
            .bucket(bucketName)
            .key(key)
            .build();

        GetObjectRetentionResponse response =
getClient().getObjectRetention(retentionRequest);
        System.out.println("tObject retention for "+key +"
in "+ bucketName +": " + response.retention().mode() +" until "+
response.retention().retainUntilDate() +".");
        return response.retention();

    } catch (S3Exception e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        return null;
    }
}

public void deleteBucketByName(String bucketName) {
    try {
        DeleteBucketRequest request = DeleteBucketRequest.builder()
            .bucket(bucketName)
            .build();

        getClient().deleteBucket(request);
        System.out.println(bucketName +" was deleted.");

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



```
    }  
}  
  
// Get the object lock configuration details for an S3 bucket.  
public void getBucketObjectLockConfiguration(String bucketName) {  
    GetObjectLockConfigurationRequest objectLockConfigurationRequest =  
    GetObjectLockConfigurationRequest.builder()  
        .bucket(bucketName)  
        .build();  
  
    GetObjectLockConfigurationResponse response =  
    getClient().getObjectLockConfiguration(objectLockConfigurationRequest);  
    System.out.println("Bucket object lock config for "+bucketName+": ");  
    System.out.println("\tEnabled:  
"+response.getObjectLockConfiguration().objectLockEnabled());  
    System.out.println("\tRule: "+  
    response.getObjectLockConfiguration().rule().defaultRetention());  
}  
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Einstiegspunkt für das Szenario (index.js). Dadurch werden alle Schritte orchestriert. Die Implementierungsdetails für Szenario, ScenarioInput ScenarioOutput, und finden GitHub Sie unter 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]",
    });
}

```

Geben Sie Willkommensnachrichten an die Konsole aus (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 };
```

Stellen Sie Buckets, Objekte und Dateieinstellungen bereit (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,
};

```

Zeigen Sie Dateien in den Buckets an und löschen Sie sie (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 };

```

Vernichten Sie alle erstellten Ressourcen (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 };
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Stellen Sie bedingte Amazon S3 S3-Anfragen mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie Vorbedingungen zu Amazon-S3-Anforderungen hinzufügen.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der bedingten Amazon-S3-Anforderung zeigt.

```
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;
    }
}
```

Eine Wrapper-Klasse für S3-Funktionen.

```
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;
    }
}
}

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
 - [CopyObject](#)

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

JavaScript

SDK für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Einstiegspunkt für den Workflow (index.js). Dadurch werden alle Schritte orchestriert. Die Implementierungsdetails für Szenario, ScenarioInput ScenarioOutput, und finden GitHub Sie unter 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]",
  });
}

```

Geben Sie Willkommensnachrichten an die Konsole aus (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 };
```

Stellen Sie Buckets und Objekte bereit (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,
};
```

Rufen Sie Objekte mithilfe von bedingten S3-Anfragen (`repl.steps.js`) ab, kopieren Sie sie und fügen Sie sie ein.

```
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 };

```

Vernichten Sie alle erstellten Ressourcen (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 (error) {
            // Ignore errors for deleted objects
          }
        }
      } catch (error) {
        // Ignore errors for buckets that don't exist
      }
    }
  })

```

```
        })),
    );
    } 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 };
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für JavaScript -API-Referenz.
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die bedingten Amazon-S3-Anforderungen zeigt.


```
"""
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()
```

Eine Wrapper-Klasse, die die Operationen der bedingten Anfrage enthält.

```
import boto3  
import logging  
  
from botocore.exceptions import ClientError  
  
# Configure logging  
logger = logging.getLogger(__name__)  
  
class S3ConditionalRequests:  
    """Encapsulates S3 conditional request operations."""  
  
    def __init__(self, s3_client):  
        self.s3 = s3_client  
  
    @classmethod  
    def from_client(cls):  
        """  
        Instantiates this class from a Boto3 client.  
        """  
        s3_client = boto3.client("s3")  
        return cls(s3_client)  
  
    def get_object_conditional(  
        self,  
        object_key: str,  
        source_bucket: str,  
        condition_type: str,  
        condition_value: str,  
    ):  
        """  
        Retrieves an object from Amazon S3 with a conditional request.  
  
        :param object_key: The key of the object to retrieve.
```



```

        :param source_bucket: The source bucket of the object.
        :param condition_type: The type of condition: 'IfMatch', 'IfNoneMatch',
        'IfModifiedSince', 'IfUnmodifiedSince'.
        :param condition_value: The value to use for the condition.
        """
        try:
            response = self.s3.get_object(
                Bucket=source_bucket,
                Key=object_key,
                **{condition_type: condition_value},
            )
            sample_bytes = response["Body"].read(20)
            print(
                f"\tConditional read successful. Here are the first 20 bytes of
the object:\n"
            )
            print(f"\t{sample_bytes}")
        except ClientError as e:
            error_code = e.response["Error"]["Code"]
            if error_code == "PreconditionFailed":
                print("\tConditional read failed: Precondition failed")
            elif error_code == "304": # Not modified error code.
                print("\tConditional read failed: Object not modified")
            else:
                logger.error(f"Unexpected error: {error_code}")
                raise

    def put_object_conditional(self, object_key: str, source_bucket: str, data:
bytes):
        """
        Uploads an object to Amazon S3 with a conditional request. Prevents
        overwrite
        using an IfNoneMatch condition for the object key.

        :param object_key: The key of the object to upload.
        :param source_bucket: The source bucket of the object.
        :param data: The data to upload.
        """
        try:
            self.s3.put_object(
                Bucket=source_bucket, Key=object_key, Body=data, IfNoneMatch="*"
            )

```

```

        print(
            f"\tConditional write successful for key {object_key} in bucket
{source_bucket}."
        )
    except ClientError as e:
        error_code = e.response["Error"]["Code"]
        if error_code == "PreconditionFailed":
            print("\tConditional write failed: Precondition failed")
        else:
            logger.error(f"Unexpected error: {error_code}")
            raise

def copy_object_conditional(
    self,
    source_key: str,
    dest_key: str,
    source_bucket: str,
    dest_bucket: str,
    condition_type: str,
    condition_value: str,
):
    """
    Copies an object from one Amazon S3 bucket to another with a conditional
    request.

    :param source_key: The key of the source object to copy.
    :param dest_key: The key of the destination object.
    :param source_bucket: The source bucket of the object.
    :param dest_bucket: The destination bucket of the object.
    :param condition_type: The type of condition to apply, e.g.
        'CopySourceIfMatch', 'CopySourceIfNoneMatch',
        'CopySourceIfModifiedSince', 'CopySourceIfUnmodifiedSince'.
    :param condition_value: The value to use for the condition.
    """
    try:
        self.s3.copy_object(
            Bucket=dest_bucket,
            Key=dest_key,
            CopySource={"Bucket": source_bucket, "Key": source_key},
            **{condition_type: condition_value},
        )
        print(

```

```
        f"\tConditional copy successful for key {dest_key} in bucket  
{dest_bucket}."  
    )  
    except ClientError as e:  
        error_code = e.response["Error"]["Code"]  
        if error_code == "PreconditionFailed":  
            print("\tConditional copy failed: Precondition failed")  
        elif error_code == "304": # Not modified error code.  
            print("\tConditional copy failed: Object not modified")  
        else:  
            logger.error(f"Unexpected error: {error_code}")  
            raise
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
 - [CopyObject](#)
 - [GetObject](#)
 - [PutObject](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwalten Sie Zugriffskontrolllisten (ACLs) für Amazon S3 S3-Buckets mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie eine neue Zugriffssteuerungsliste (ACL) für Amazon-S3-Buckets verwalten.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

using System;
using System.Collections.Generic;
using System.Threading.Tasks;
using Amazon.S3;
using Amazon.S3.Model;

/// <summary>
/// This example shows how to manage Amazon Simple Storage Service
/// (Amazon S3) access control lists (ACLs) to control Amazon S3 bucket
/// access.
/// </summary>
public class ManageACLs
{
    public static async Task Main()
    {
        string bucketName = "amzn-s3-demo-bucket1";
        string newBucketName = "amzn-s3-demo-bucket2";
        string keyName = "sample-object.txt";
        string emailAddress = "someone@example.com";

        // If the AWS Region where your bucket is located is different from
        // the Region defined for the default user, pass the Amazon S3
bucket's
        // name to the client constructor. It should look like this:
        // RegionEndpoint bucketRegion = RegionEndpoint.USEast1;
        IAmazonS3 client = new AmazonS3Client();

        await TestBucketObjectACLsAsync(client, bucketName, newBucketName,
keyName, emailAddress);
    }

    /// <summary>
    /// Creates a new Amazon S3 bucket with a canned ACL, then retrieves the
ACL
    /// information and then adds a new ACL to one of the objects in the
    /// Amazon S3 bucket.
    /// </summary>
    /// <param name="client">The initialized Amazon S3 client object used to
call
    /// methods to create a bucket, get an ACL, and add a different ACL to
    /// one of the objects.</param>
    /// <param name="bucketName">A string representing the original Amazon S3
    /// bucket name.</param>

```

S3

be

```

    /// <param name="newBucketName">A string representing the name of the
    /// new bucket that will be created.</param>
    /// <param name="keyName">A string representing the key name of an Amazon
    /// object for which we will change the ACL.</param>
    /// <param name="emailAddress">A string representing the email address
    /// belonging to the person to whom access to the Amazon S3 bucket will
    /// be granted.</param>
    public static async Task TestBucketObjectACLsAsync(
        IAmazonS3 client,
        string bucketName,
        string newBucketName,
        string keyName,
        string emailAddress)
    {
        try
        {
            // Create a new Amazon S3 bucket and specify canned ACL.
            var success = await CreateBucketWithCannedACLAsync(client,
newBucketName);

            // Get the ACL on a bucket.
            await GetBucketACLAsync(client, bucketName);

            // Add (replace) the ACL on an object in a bucket.
            await AddACLToExistingObjectAsync(client, bucketName, keyName,
emailAddress);
        }
        catch (AmazonS3Exception amazonS3Exception)
        {
            Console.WriteLine($"Exception: {amazonS3Exception.Message}");
        }
    }

    /// <summary>
    /// Creates a new Amazon S3 bucket with a canned ACL attached.
    /// </summary>
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="newBucketName">A string representing the name of the
    /// new Amazon S3 bucket.</param>
    /// <returns>Returns a boolean value indicating success or failure.</
returns>

```

```
public static async Task<bool> CreateBucketWithCannedACLAsync(IAmazonS3
client, string newBucketName)
{
    var request = new PutBucketRequest()
    {
        BucketName = newBucketName,
        BucketRegion = S3Region.EUWest1,

        // Add a canned ACL.
        CannedACL = S3CannedACL.LogDeliveryWrite,
    };

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

/// <summary>
/// Retrieves the ACL associated with the Amazon S3 bucket name in the
/// bucketName parameter.
/// </summary>
/// <param name="client">The initialized client object used to call
/// PutBucketAsync.</param>
/// <param name="bucketName">The Amazon S3 bucket for which we want to
get the
/// ACL list.</param>
/// <returns>Returns an S3AccessControlList returned from the call to
/// GetACLAsync.</returns>
public static async Task<S3AccessControlList> GetBucketACLAsync(IAmazonS3
client, string bucketName)
{
    GetACLResponse response = await client.GetACLAsync(new GetACLRequest
    {
        BucketName = bucketName,
    });

    return response.AccessControlList;
}

/// <summary>
/// Adds a new ACL to an existing object in the Amazon S3 bucket.
/// </summary>
```

```
    /// <param name="client">The initialized client object used to call
    /// PutBucketAsync.</param>
    /// <param name="bucketName">A string representing the name of the Amazon
S3    /// bucket containing the object to which we want to apply a new ACL.</
param>    /// <param name="keyName">A string representing the name of the object
    /// to which we want to apply the new ACL.</param>
    /// <param name="emailAddress">The email address of the person to whom
    /// we will be applying to whom access will be granted.</param>
    public static async Task AddACLToExistingObjectAsync(IAmazonS3 client,
string    string bucketName, string keyName, string emailAddress)
    {
        // Retrieve the ACL for an object.
        GetACLResponse aclResponse = await client.GetACLAsync(new
GetACLRequest    {
            BucketName = bucketName,
            Key = keyName,
        });

        S3AccessControlList acl = aclResponse.AccessControlList;

        // Retrieve the owner.
        Owner owner = acl.Owner;

        // Clear existing grants.
        acl.Grants.Clear();

        // Add a grant to reset the owner's full permission
        // (the previous clear statement removed all permissions).
        var fullControlGrant = new S3Grant
        {
            Grantee = new S3Grantee { CanonicalUser = acl.Owner.Id },
        };
        acl.AddGrant(fullControlGrant.Grantee, S3Permission.FULL_CONTROL);

        // Specify email to identify grantee for granting permissions.
        var grantUsingEmail = new S3Grant
        {
            Grantee = new S3Grantee { EmailAddress = emailAddress },
            Permission = S3Permission.WRITE_ACP,
        };
    }
}
```

```
// Specify log delivery group as grantee.
var grantLogDeliveryGroup = new S3Grant
{
    Grantee = new S3Grantee { URI = "http://acs.amazonaws.com/groups/
s3/LogDelivery" },
    Permission = S3Permission.WRITE,
};

// Create a new ACL.
var newAcl = new S3AccessControlList
{
    Grants = new List<S3Grant> { grantUsingEmail,
grantLogDeliveryGroup },
    Owner = owner,
};

// Set the new ACL. We're throwing away the response here.
_ = await client.PutACLAsync(new PutACLRequest
{
    BucketName = bucketName,
    Key = keyName,
    AccessControlList = newAcl,
});
}

}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
 - [GetBucketAcl](#)
 - [GetObjectAcl](#)
 - [PutBucketAcl](#)
 - [PutObjectAcl](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwalten Sie große Amazon SQS SQS-Nachrichten mithilfe von Amazon S3 mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie die erweiterte Clientbibliothek von Amazon SQS verwenden, um mit großen Amazon-SQS-Nachrichten zu arbeiten.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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;
}
}
```

- Weitere Informationen finden Sie im [AWS SDK for Java 2.x -Entwicklerhandbuch](#).
- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [CreateBucket](#)
 - [PutBucketLifecycleConfiguration](#)
 - [ReceiveMessage](#)
 - [SendMessage](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwalten Sie versionierte Amazon S3 S3-Objekte stapelweise mit einer Lambda-Funktion mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie versionierte S3-Objekte in Batches mit einer Lambda-Funktion verwaltet werden.

Python

SDK für Python (Boto3)

Zeigt, wie versionierte Objekte von Amazon Simple Storage Service (Amazon S3) stapelweise bearbeitet werden, indem Jobs erstellt werden, die AWS Lambda Funktionen aufrufen, um die Verarbeitung durchzuführen. In diesem Beispiel wird ein versionsfähiger Bucket erstellt, der die Strophen des Gedichts You Are Old, Father William von Lewis Carroll hochlädt und Amazon-S3-Batch-Aufträge verwendet, um das Gedicht auf verschiedene Arten zu verdrehen.

Lernen Sie Folgendes:

- Erstellen Sie Lambda-Funktionen, die mit versionierten Objekten arbeiten.
- Erstellen Sie ein Manifest von zu aktualisierenden Objekten.
- Erstellen Sie Batch-Aufträge, die Lambda-Funktionen zum Aktualisieren von Objekten aufrufen.
- Löschen Sie Lambda-Funktionen.
- Leeren und löschen Sie einen versionierten Bucket.

Dieses Beispiel lässt sich am besten auf ansehen. [GitHub](#) Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Amazon S3

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Analysieren Sie Amazon S3 S3-URLs mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie Amazon S3-URLs analysieren, um wichtige Komponenten wie den Bucket-Namen und Objektschlüssel zu extrahieren.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Analysieren Sie eine Amazon S3-URL mithilfe der [S3Uri-Klasse](#).

```
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);
        }
    }
}

```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Erstellen Sie eine mehrteilige Kopie eines Amazon S3 S3-Objekts mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie Sie eine mehrteilige Kopie eines Amazon-S3-Objekts erstellen.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für .NET - API-Referenz.
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [GetObjectMetadata](#)
 - [UploadPartCopy](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Empfangen und verarbeiten Sie Amazon S3 S3-Ereignisbenachrichtigungen mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie mit S3-Ereignisbenachrichtigungen auf objektorientierte Weise arbeiten.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Dieses Beispiel zeigt, wie Sie ein S3-Benachrichtigungsereignis mithilfe von Amazon SQS verarbeiten.

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

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [DeleteMessageBatch](#)
 - [GetQueueAttributes](#)
 - [PutBucketNotificationConfiguration](#)
 - [ReceiveMessage](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Speichern Sie EXIF und andere Bildinformationen mit einem AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab.
- Laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch.
- Verwenden Sie Amazon Rekognition, um die drei wichtigsten Attribute (Labels) in der Datei zu identifizieren.
- Fügen Sie die EXIF- und Label-Informationen einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Rust

SDK für Rust

Rufen Sie EXIF-Informationen aus einer JPG-, JPEG- oder PNG-Datei ab, laden Sie die Bilddatei in einen Amazon-S3-Bucket hoch und identifizieren Sie mit Amazon Rekognition die drei wichtigsten Attribute (Labels in Amazon Rekognition) in der Datei. Fügen Sie die EXIF- und Labelinformationen dann einer Amazon-DynamoDB-Tabelle in der Region hinzu.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- DynamoDB
- Amazon Rekognition
- Amazon S3

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Senden Sie S3-Ereignisbenachrichtigungen an Amazon EventBridge mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie ein Bucket aktiviert wird, um S3-Ereignisbenachrichtigungen an ein Amazon SNS-Thema EventBridge und eine Amazon SQS SQS-Warteschlange zu senden und Benachrichtigungen an diese weiterzuleiten.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/** This method configures a bucket to send events to AWS EventBridge and
    creates a rule
    * to route the S3 object created events to a topic and a queue.
    *
    * @param bucketName Name of existing bucket
    * @param topicArn ARN of existing topic to receive S3 event notifications
    * @param queueArn ARN of existing queue to receive S3 event notifications
    *
    * An AWS CloudFormation stack sets up the bucket, queue, topic before the
    method runs.
    */
public static String setBucketNotificationToEventBridge(String bucketName,
String topicArn, String queueArn) {
    try {
        // Enable bucket to emit S3 Event notifications to EventBridge.
        s3Client.putBucketNotificationConfiguration(b -> b
            .bucket(bucketName)
            .notificationConfiguration(b1 -> b1
                .eventBridgeConfiguration(
                    SdkBuilder::build)
            ).build()).join();

        // Create an EventBridge rule to route Object Created notifications.
        PutRuleRequest putRuleRequest = PutRuleRequest.builder()
            .name(RULE_NAME)
            .eventPattern("""
                {
                    "source": ["aws.s3"],
                    "detail-type": ["Object Created"],
                    "detail": {
                        "bucket": {
                            "name": ["%s"]
                        }
                    }
                }
            """)
            .build();
    } catch (Exception e) {
        // Handle exception
    }
}
```

```

        }
    }
}
"".formatted(bucketName))
.build();

// Add the rule to the default event bus.
PutRuleResponse putRuleResponse =
eventBridgeClient.putRule(putRuleRequest)
    .whenComplete((r, t) -> {
        if (t != null) {
            logger.error("Error creating event bus rule: " +
t.getMessage(), t);
                throw new RuntimeException(t.getCause().getMessage(),
t);
        }
        logger.info("Event bus rule creation request sent
successfully. ARN is: {}", r.ruleArn());
    }).join();

// Add the existing SNS topic and SQS queue as targets to the rule.
eventBridgeClient.putTargets(b -> b
    .eventBusName("default")
    .rule(RULE_NAME)
    .targets(List.of (
        Target.builder()
            .arn(queueArn)
            .id("Queue")
            .build(),
        Target.builder()
            .arn(topicArn)
            .id("Topic")
            .build()
    )
    ).join();
    return putRuleResponse.ruleArn();
} catch (S3Exception e) {
    System.err.println(e.awsErrorDetails().errorMessage());
    System.exit(1);
}
return null;
}

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [PutBucketNotificationConfiguration](#)
 - [PutRule](#)
 - [PutTargets](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verfolgen Sie den Upload oder Download eines Amazon S3 S3-Objekts mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie einen Upload oder Download in ein bzw. von einem Amazon-S3-Objekt verfolgen.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Verfolgen Sie den Fortschritt eines Datei-Uploads.

```
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!

    */
}

```

Verfolgen Sie den Fortschritt eines Datei-Downloads.

```

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!

```

```

*/
}

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [GetObject](#)
 - [PutObject](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Transformieren von Daten für Ihre Anwendung mit S3 Object Lambda

Das folgende Codebeispiel zeigt, wie Sie Daten für Ihre Anwendung mit S3 Object Lambda transformieren können.

.NET

SDK für .NET

Zeigt, wie Sie benutzerdefinierten Code zu standardmäßigen S3-GET-Anforderungen hinzufügen, um das angeforderte Objekt, das von S3 abgerufen wurde, so zu ändern, dass das Objekt den Anforderungen des anfordernden Clients oder der anfordernden Anwendung entspricht.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter [GitHub](#).

In diesem Beispiel verwendete Dienste

- Lambda
- Amazon S3

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Beispielansätze für Einheiten- und Integrationstests mit einem AWS SDK

Das folgende Codebeispiel zeigt Beispiele für bewährte Techniken beim Schreiben von Einheiten- und Integrationstests mit einem AWS SDK.

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Cargo.toml zum Testen von Beispielen.

```
[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"
```

Beispiel für Modultests mit automock und einem Service-Wrapper

```
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);
}
}

```

Beispiel für Integrationstests mit 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(&[]);
}
}

```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Rekursives Hochladen eines lokalen Verzeichnisses in einen Amazon Simple Storage Service (Amazon S3)-Bucket

Das folgende Codebeispiel zeigt, wie Sie ein lokales Verzeichnis rekursiv in einen Amazon Simple Storage Service (Amazon S3)-Bucket hochladen.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Verwenden Sie an [S3TransferManager](#), um [ein lokales Verzeichnis hochzuladen](#). Sehen Sie sich die [vollständige Datei](#) an und [testen](#) Sie sie.

```
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();
    }
```

- Einzelheiten zur API finden Sie [UploadDirectory](#) unter AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Laden Sie große Dateien zu und von Amazon S3 hoch oder laden Sie sie herunter mit einem AWS SDK

Die folgenden Codebeispiele veranschaulichen, wie Sie große Dateien zu S3 hochladen bzw. von S3 herunterladen.

Weitere Informationen finden Sie unter [Hochladen eines Objekts mit Multipart-Upload](#).

.NET

SDK für .NET

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie Funktionen auf, die Dateien mithilfe von Amazon S3 zu und von einem S3-Bucket übertragen `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 S3 bucket.");
    Console.WriteLine("\t3. Download a single object from an S3 bucket.");
    Console.WriteLine("\t4. Download the objects in an S3 bucket to a local
directory.");
    Console.WriteLine($"{sepBar}");
}
```

```
}

// Pauses the scenario.
static void PressEnter()
{
    Console.WriteLine("Press <Enter> to continue.");
    _ = Console.ReadLine();
    Console.WriteLine("\n");
}

// Returns the string textToCenter, padded on the left with spaces
// that center the text on the console display.
static string CenterText(string textToCenter)
{
    var centeredText = new StringBuilder();
    var screenWidth = Console.WindowWidth;
    centeredText.Append(new string(' ', (int)(screenWidth -
textToCenter.Length) / 2));
    centeredText.Append(textToCenter);
    return centeredText.ToString();
}

// Displays a list of file names included in the specified path.
static void DisplayLocalFiles(string localPath)
{
    var fileList = Directory.GetFiles(localPath);
    if (fileList.Length > 0)
    {
        foreach (var fileName in fileList)
        {
            Console.WriteLine(fileName);
        }
    }
}

// Displays a list of the files in the specified S3 bucket and prefix.
static async Task DisplayBucketFiles(IAmazonS3 client, string bucketName, string
s3Path)
{
    ListObjectsV2Request request = new()
    {
        BucketName = bucketName,
        Prefix = s3Path,
        MaxKeys = 5,
```

```
};

var response = new ListObjectsV2Response();

do
{
    response = await client.ListObjectsV2Async(request);

    response.S3Objects
        .ForEach(obj => Console.WriteLine($"{obj.Key}"));

    // If the response is truncated, set the request ContinuationToken
    // from the NextContinuationToken property of the response.
    request.ContinuationToken = response.NextContinuationToken;
} while (response.IsTruncated);
}
```

Laden Sie eine einzelne Datei hoch.

```
/// <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;
    }
}

```

Laden Sie ein ganzes lokales Verzeichnis hoch.

```

/// <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;
        }
    }
}
```

Laden Sie eine einzelne Datei herunter.

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

```

Laden Sie den Inhalt eines S3 Buckets herunter.

```

    /// <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;
    }
}
```

Verfolgen Sie den Fortschritt eines Uploads mit dem 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}");  
    }  
}
```

Laden Sie ein Objekt mit Verschlüsselung hoch.

```
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);  
    }  
}
```

```

        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 für Go V2

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie Funktionen, die Upload- und Download-Manager verwenden, um die Daten in Teile zu zerlegen und sie gleichzeitig zu übertragen.

```

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
}

```

```
}
```

Führen Sie ein interaktives Szenario aus, das Ihnen zeigt, wie Sie die Upload- und Download-Manager im Kontext verwenden.

```
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 für Java 2.x

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie Funktionen auf, die Dateien mit dem zu und von einem S3-Bucket übertragen `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();
}
```

Laden Sie ein ganzes lokales Verzeichnis hoch.

```
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();
    }
}
```

Laden Sie eine einzelne Datei hoch.

```
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();
}
```

In den Codebeispielen werden folgende Importe verwendet.

```
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;
```

Verwenden Sie den [S3 Transfer Manager](#) auf dem [AWS CRT-based S3-Client](#), um einen mehrteiligen Upload transparent durchzuführen, wenn die Größe des Inhalts einen Schwellenwert überschreitet. Der Standardschwellenwert beträgt 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();
}
```

Verwenden Sie die [S3Client-API](#), um einen mehrteiligen Upload durchzuführen.

```
/**
```

```
* 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()));
}

```

Verwenden Sie die [S3AsyncClient API](#) mit aktivierter mehrteiliger Unterstützung, um einen mehrteiligen Upload durchzuführen.

```

/**
 * 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 für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Laden Sie eine große Datei hoch.

```
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;
  }
}
};

```

Laden Sie eine große Datei herunter.

```

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 für Python (Boto3)

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie Funktionen, die Dateien mit mehreren der verfügbaren Transfer-Manager-Einstellungen übertragen. Verwenden Sie eine Callback-Klasse, um während der Dateiübertragung Callback-Fortschritte zu schreiben.

```
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

```

Demonstrieren Sie die Funktionen des Transfer-Managers und melden Sie Ergebnisse.

```

import hashlib
import os
import platform
import shutil
import time

import boto3
from boto3.s3.transfer import TransferConfig
from botocore.exceptions import ClientError
from botocore.exceptions import ParamValidationError

```

```
from botocore.exceptions import NoCredentialsError

import file_transfer

MB = 1024 * 1024
# These configuration attributes affect both uploads and downloads.
CONFIG_ATTRS = (
    "multipart_threshold",
    "multipart_chunksize",
    "max_concurrency",
    "use_threads",
)
# These configuration attributes affect only downloads.
DOWNLOAD_CONFIG_ATTRS = ("max_io_queue", "io_chunksize", "num_download_attempts")

class TransferDemoManager:
    """
    Manages the demonstration. Collects user input from a command line, reports
    transfer results, maintains a list of artifacts created during the
    demonstration, and cleans them up after the demonstration is completed.
    """

    def __init__(self):
        self._s3 = boto3.resource("s3")
        self._chore_list = []
        self._create_file_cmd = None
        self._size_multiplier = 0
        self.file_size_mb = 30
        self.demo_folder = None
        self.demo_bucket = None
        self._setup_platform_specific()
        self._terminal_width = shutil.get_terminal_size(fallback=(80, 80))[0]

    def collect_user_info(self):
        """
        Collect local folder and Amazon S3 bucket name from the user. These
        locations are used to store files during the demonstration.
        """
        while not self.demo_folder:
            self.demo_folder = input(
                "Which file folder do you want to use to store " "demonstration
files? "
            )
```

```

        if not os.path.isdir(self.demo_folder):
            print(f"{self.demo_folder} isn't a folder!")
            self.demo_folder = None

    while not self.demo_bucket:
        self.demo_bucket = input(
            "Which Amazon S3 bucket do you want to use to store "
            "demonstration files? "
        )
    try:
        self._s3.meta.client.head_bucket(Bucket=self.demo_bucket)
    except ParamValidationError as err:
        print(err)
        self.demo_bucket = None
    except ClientError as err:
        print(err)
        print(
            f"Either {self.demo_bucket} doesn't exist or you don't "
            f"have access to it."
        )
        self.demo_bucket = None

    def demo(
        self, question, upload_func, download_func, upload_args=None,
download_args=None
    ):
        """Run a demonstration.

        Ask the user if they want to run this specific demonstration.
        If they say yes, create a file on the local path, upload it
        using the specified upload function, then download it using the
        specified download function.
        """
        if download_args is None:
            download_args = {}
        if upload_args is None:
            upload_args = {}
        question = question.format(self.file_size_mb)
        answer = input(f"{question} (y/n)")
        if answer.lower() == "y":
            local_file_path, object_key, download_file_path =
self._create_demo_file()

            file_transfer.TransferConfig = self._config_wrapper(

```

```

        TransferConfig, CONFIG_ATTRS
    )
    self._report_transfer_params(
        "Uploading", local_file_path, object_key, **upload_args
    )
    start_time = time.perf_counter()
    thread_info = upload_func(
        local_file_path,
        self.demo_bucket,
        object_key,
        self.file_size_mb,
        **upload_args,
    )
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

    file_transfer.TransferConfig = self._config_wrapper(
        TransferConfig, CONFIG_ATTRS + DOWNLOAD_CONFIG_ATTRS
    )
    self._report_transfer_params(
        "Downloading", object_key, download_file_path, **download_args
    )
    start_time = time.perf_counter()
    thread_info = download_func(
        self.demo_bucket,
        object_key,
        download_file_path,
        self.file_size_mb,
        **download_args,
    )
    end_time = time.perf_counter()
    self._report_transfer_result(thread_info, end_time - start_time)

def last_name_set(self):
    """Get the name set used for the last demo."""
    return self._chore_list[-1]

def cleanup(self):
    """
    Remove files from the demo folder, and uploaded objects from the
    Amazon S3 bucket.
    """
    print("-" * self._terminal_width)

```

```

        for local_file_path, s3_object_key, downloaded_file_path in
self._chore_list:
            print(f"Removing {local_file_path}")
            try:
                os.remove(local_file_path)
            except FileNotFoundError as err:
                print(err)

            print(f"Removing {downloaded_file_path}")
            try:
                os.remove(downloaded_file_path)
            except FileNotFoundError as err:
                print(err)

            if self.demo_bucket:
                print(f"Removing {self.demo_bucket}:{s3_object_key}")
                try:
                    self._s3.Bucket(self.demo_bucket).Object(s3_object_key).delete()
                except ClientError as err:
                    print(err)

def _setup_platform_specific(self):
    """Set up platform-specific command used to create a large file."""
    if platform.system() == "Windows":
        self._create_file_cmd = "fsutil file createnew {} {}".format(
            self._size_multiplier, MB
        )
    elif platform.system() == "Linux" or platform.system() == "Darwin":
        self._create_file_cmd = f"dd if=/dev/urandom of={{}} " f"bs={{MB}}
count={{}}"
        self._size_multiplier = 1
    else:
        raise EnvironmentError(
            f"Demo of platform {platform.system()} isn't supported."
        )

def _create_demo_file(self):
    """
    Create a file in the demo folder specified by the user. Store the local
    path, object name, and download path for later cleanup.

    Only the local file is created by this method. The Amazon S3 object and
    download file are created later during the demonstration.

```

```

Returns:
A tuple that contains the local file path, object name, and download
file path.
"""
file_name_template = "TestFile{}-{}.demo"
local_suffix = "local"
object_suffix = "s3object"
download_suffix = "downloaded"
file_tag = len(self._chore_list) + 1

local_file_path = os.path.join(
    self.demo_folder, file_name_template.format(file_tag, local_suffix)
)

s3_object_key = file_name_template.format(file_tag, object_suffix)

downloaded_file_path = os.path.join(
    self.demo_folder, file_name_template.format(file_tag,
download_suffix)
)

filled_cmd = self._create_file_cmd.format(
    local_file_path, self.file_size_mb * self._size_multiplier
)

print(
    f"Creating file of size {self.file_size_mb} MB "
    f"in {self.demo_folder} by running:"
)
print(f"{'':4}{filled_cmd}")
os.system(filled_cmd)

chore = (local_file_path, s3_object_key, downloaded_file_path)
self._chore_list.append(chore)
return chore

def _report_transfer_params(self, verb, source_name, dest_name, **kwargs):
    """Report configuration and extra arguments used for a file transfer."""
    print("-" * self._terminal_width)
    print(f"{verb} {source_name} ({self.file_size_mb} MB) to {dest_name}")
    if kwargs:
        print("With extra args:")
        for arg, value in kwargs.items():
            print(f"{'':4}{arg:<20}: {value}')

```

```

@staticmethod
def ask_user(question):
    """
    Ask the user a yes or no question.

    Returns:
    True when the user answers 'y' or 'Y'; otherwise, False.
    """
    answer = input(f"{question} (y/n) ")
    return answer.lower() == "y"

@staticmethod
def _config_wrapper(func, config_attrs):
    def wrapper(*args, **kwargs):
        config = func(*args, **kwargs)
        print("With configuration:")
        for attr in config_attrs:
            print(f'{"":4}{attr:<20}: {getattr(config, attr)}')
        return config

    return wrapper

@staticmethod
def _report_transfer_result(thread_info, elapsed):
    """Report the result of a transfer, including per-thread data."""
    print(f"\nUsed {len(thread_info)} threads.")
    for ident, byte_count in thread_info.items():
        print(f'{"":4}Thread {ident} copied {byte_count} bytes.')
    print(f"Your transfer took {elapsed:.2f} seconds.")

def main():
    """
    Run the demonstration script for s3_file_transfer.
    """
    demo_manager = TransferDemoManager()
    demo_manager.collect_user_info()

    # Upload and download with default configuration. Because the file is 30 MB
    # and the default multipart_threshold is 8 MB, both upload and download are
    # multipart transfers.
    demo_manager.demo(
        "Do you want to upload and download a {} MB file "

```



```
        "using the default configuration?",
        file_transfer.upload_with_default_configuration,
        file_transfer.download_with_default_configuration,
    )

# Upload and download with multipart_threshold set higher than the size of
# the file. This causes the transfer manager to use standard transfers
# instead of multipart transfers.
demo_manager.demo(
    "Do you want to upload and download a {} MB file "
    "as a standard (not multipart) transfer?",
    file_transfer.upload_with_high_threshold,
    file_transfer.download_with_high_threshold,
)

# Upload with specific chunk size and additional metadata.
# Download with a single thread.
demo_manager.demo(
    "Do you want to upload a {} MB file with a smaller chunk size and "
    "then download the same file using a single thread?",
    file_transfer.upload_with_chunksize_and_meta,
    file_transfer.download_with_single_thread,
    upload_args={
        "metadata": {
            "upload_type": "chunky",
            "favorite_color": "aqua",
            "size": "medium",
        }
    },
)

# Upload using server-side encryption with customer-provided
# encryption keys.
# Generate a 256-bit key from a passphrase.
sse_key = hashlib.sha256("demo_passphrase".encode("utf-8")).digest()
demo_manager.demo(
    "Do you want to upload and download a {} MB file using "
    "server-side encryption?",
    file_transfer.upload_with_sse,
    file_transfer.download_with_sse,
    upload_args={"sse_key": sse_key},
    download_args={"sse_key": sse_key},
)
```

```
# Download without specifying an encryption key to show that the
# encryption key must be included to download an encrypted object.
if demo_manager.ask_user(
    "Do you want to try to download the encrypted "
    "object without sending the required key?"
):
    try:
        _, object_key, download_file_path = demo_manager.last_name_set()
        file_transfer.download_with_default_configuration(
            demo_manager.demo_bucket,
            object_key,
            download_file_path,
            demo_manager.file_size_mb,
        )
    except ClientError as err:
        print(
            "Got expected error when trying to download an encrypted "
            "object without specifying encryption info:"
        )
        print(f"{'':4}{err}")

# Remove all created and downloaded files, remove all objects from
# S3 storage.
if demo_manager.ask_user(
    "Demonstration complete. Do you want to remove local files " "and S3
objects?"
):
    demo_manager.cleanup()

if __name__ == "__main__":
    try:
        main()
    except NoCredentialsError as error:
        print(error)
        print(
            "To run this example, you must have valid credentials in "
            "a shared credential file or set in environment variables."
        )
```

Rust

SDK für Rust

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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(()  
}
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Laden Sie einen Stream unbekannter Größe in ein Amazon S3 S3-Objekt hoch, indem Sie eine AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie einen Stream unbekannter Größe in ein Amazon S3-Objekt hochladen.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Verwenden Sie den [AWS CRT-based S3-Client](#).

```
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;
    }
}
```

Verwenden Sie den standardmäßigen [asynchronen S3-Client mit aktiviertem mehrteiligem Upload](#).

```
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;
    }
}
```

Verwenden Sie den [Amazon-S3-Transfer-Manager](#).

```
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 für Swift

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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).")
}
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie Prüfsummen, um mit einem Amazon S3 S3-Objekt zu arbeiten, indem Sie AWS SDK

Das folgende Codebeispiel zeigt, wie Sie Prüfsummen verwenden, um mit einem Amazon-S3-Objekt zu arbeiten.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

In den Codebeispielen wird eine Teilmenge der folgenden Importe verwendet.

```
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;
```

Geben Sie einen Prüfsummenalgorithmus für die `putObject`-Methode an, wenn Sie [PutObjectRequest erstellen](#).

```
public void putObjectWithChecksum(String bucketName, String key) {
    s3Client.putObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumAlgorithm(ChecksumAlgorithm.CRC32),
        RequestBody.fromString("This is a test"));
}
```

Überprüfen Sie die Prüfsumme für die `getObject` Methode, wenn Sie [die GetObjectRequest erstellen](#).


```
public GetObjectResponse getObjectWithChecksum(String bucketName, String key)
{
    return s3Client.getObject(b -> b
        .bucket(bucketName)
        .key(key)
        .checksumMode(ChecksumMode.ENABLED))
        .response();
}
```

Pre-calculate eine Prüfsumme für die `putObject` Methode, wenn Sie die [erstellen](#).
`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)));
}
```

Verwenden Sie den [S3 Transfer Manager](#) zusätzlich zum [AWS CRT-based S3-Client](#), um einen mehrteiligen Upload transparent durchzuführen, wenn die Größe des Inhalts einen Schwellenwert überschreitet. Der Standardschwellenwert beträgt 8 MB.

Sie können einen Prüfsummenalgorithmus angeben, den das SDK verwenden soll. Standardmäßig verwendet das SDK den CRC32-Algorithmus.

```
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();
}
```

Verwenden Sie die [S3Client-API oder \(S3AsyncClient API\)](#), um einen mehrteiligen Upload durchzuführen. Wenn Sie eine zusätzliche Prüfsumme angeben, müssen Sie den Algorithmus angeben, der bei der Initiierung des Uploads verwendet werden soll. Sie müssen auch den Algorithmus für jede Teilanforderung angeben und die für jedes Teil nach dem Hochladen berechnete Prüfsumme bereitstellen.

```
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()));
}

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [UploadPart](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter. [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#) Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Arbeiten Sie mit den Objektintegritätsfunktionen von Amazon S3 mithilfe eines AWS SDK

Das folgende Codebeispiel zeigt, wie Sie mit Features der S3-Objektintegrität arbeiten.

C++

SDK für C++

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Objektintegrität zeigt.

```
//! 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;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
 - [AbortMultipartUpload](#)
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [DeleteObject](#)
 - [GetObjectAttributes](#)
 - [PutObject](#)
 - [UploadPart](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Arbeiten Sie mit Objekten mit Amazon S3 S3-Version unter Verwendung eines AWS SDK

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen versionierten S3 Bucket.
- Rufen Sie alle Versionen eines Objekts ab.
- Setzen Sie eine Richtlinie auf eine frühere Version zurück.
- Löschen Sie ein versioniertes Objekt und stellen Sie es wieder her.
- Löschen Sie alle Versionen eines Objekts dauerhaft.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie Funktionen, die S3-Aktionen umschließen.

```
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
```

Laden Sie die Strophe eines Gedichts in ein versioniertes Objekt hoch und führen Sie eine Reihe von Aktionen dafür aus.

```
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!")
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
 - [CreateBucket](#)
 - [DeleteObject](#)
 - [ListObjectVersions](#)
 - [PutBucketLifecycleConfiguration](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Serverless-Beispiele für Amazon S3

Die folgenden Codebeispiele zeigen, wie Amazon S3 mit AWS SDKs verwendet wird.

Beispiele

- [Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser](#)

Aufrufen einer Lambda-Funktion über einen Amazon-S3-Auslöser

In den folgenden Codebeispiele wird die Implementierung einer Lambda-Funktion gezeigt, die ein Ereignis empfängt, das durch Hochladen eines Objekts in einen S3-Bucket ausgelöst wird. Die Funktion ruft den Namen des S3-Buckets sowie den Objektschlüssel aus dem Ereignisparameter ab und ruft die Amazon-S3-API auf, um den Inhaltstyp des Objekts abzurufen und zu protokollieren.

.NET

SDK für .NET

Note

Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von .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 für Go V2

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von 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 für Java 2.x

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von 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 für JavaScript (v3)

Note

Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Konsumieren eines S3-Ereignisses mit Lambda unter Verwendung 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);
  }
};
```

Konsumieren eines S3-Ereignisses mit Lambda unter Verwendung 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 für PHP

Note

Es gibt noch mehr dazu. GitHub Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von 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 für Python (Boto3)

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von 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 für Ruby

Note

Es gibt noch mehr GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von 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 für Rust

Note

Es gibt noch mehr dazu GitHub. Das vollständige Beispiel sowie eine Anleitung zum Einrichten und Ausführen finden Sie im Repository mit [Serverless-Beispielen](#).

Nutzen eines S3-Ereignisses mit Lambda unter Verwendung von 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(())
}
```

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Codebeispiele für Amazon S3 Control mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Amazon S3 Control mit einem AWS Software Development Kit (SDK) verwendet wird.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Amazon S3 Control

- [Grundlegende Beispiele für die Verwendung von Amazon S3 Control AWS SDKs](#)
 - [Hello Amazon S3 Control](#)
 - [Lernen Sie die Grundlagen von Amazon S3 Control mit einem AWS SDK](#)
 - [Aktionen für Amazon S3 Control mit AWS SDKs](#)
 - [Verwenden Sie CreateJob mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie DeleteJobTagging mit einem AWS SDK](#)
 - [Verwenden Sie DescribeJob mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie GetJobTagging mit einem AWS SDK](#)
 - [Verwenden Sie PutJobTagging mit einem AWS SDK](#)
 - [Verwenden Sie UpdateJobPriority mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie UpdateJobStatus mit einem AWS SDK oder CLI](#)

Grundlegende Beispiele für die Verwendung von Amazon S3 Control AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon S3 Control mit AWS SDKs verwendet werden.

Beispiele

- [Hello Amazon S3 Control](#)
- [Lernen Sie die Grundlagen von Amazon S3 Control mit einem AWS SDK](#)
- [Aktionen für Amazon S3 Control mit AWS SDKs](#)
 - [Verwenden Sie CreateJob mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie DeleteJobTagging mit einem AWS SDK](#)
 - [Verwenden Sie DescribeJob mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie GetJobTagging mit einem AWS SDK](#)
 - [Verwenden Sie PutJobTagging mit einem AWS SDK](#)
 - [Verwenden Sie UpdateJobPriority mit einem AWS SDK oder CLI](#)
 - [Verwenden Sie UpdateJobStatus mit einem AWS SDK oder CLI](#)

Hello Amazon S3 Control

Die folgenden Codebeispiele zeigen, wie Sie mit Amazon S3 Control beginnen können.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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);
        });
    }
}

```

- Einzelheiten zur API finden Sie [ListJobs](#) in der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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
```

- Einzelheiten zur API finden Sie [ListJobs](#) in AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Lernen Sie die Grundlagen von Amazon S3 Control mit einem AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie die Kernoperationen von Amazon S3 Control erlernen.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erlernen Sie die Kernoperationen.

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

```
}
```

Erstellen Sie eine Klasse, die Operationen umschließt.

```
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();
}
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [CreateJob](#)

- [DeleteJobTagging](#)
- [DescribeJob](#)
- [GetJobTagging](#)
- [ListJobs](#)
- [PutJobTagging](#)
- [UpdateJobPriority](#)
- [UpdateJobStatus](#)

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Lernen Sie das Szenario S3 Batch Basics kennen.

```
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

```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
 - [CreateJob](#)
 - [DeleteJobTagging](#)
 - [DescribeJob](#)
 - [GetJobTagging](#)
 - [ListJobs](#)
 - [PutJobTagging](#)
 - [UpdateJobPriority](#)

- [UpdateJobStatus](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Aktionen für Amazon S3 Control mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne Amazon S3 Control-Aktionen mit AWS SDKs durchgeführt werden. Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der [API-Referenz für Amazon S3](#).

Beispiele

- [Verwenden Sie CreateJob mit einem AWS SDK oder CLI](#)
- [Verwenden Sie DeleteJobTagging mit einem AWS SDK](#)
- [Verwenden Sie DescribeJob mit einem AWS SDK oder CLI](#)
- [Verwenden Sie GetJobTagging mit einem AWS SDK](#)
- [Verwenden Sie PutJobTagging mit einem AWS SDK](#)
- [Verwenden Sie UpdateJobPriority mit einem AWS SDK oder CLI](#)
- [Verwenden Sie UpdateJobStatus mit einem AWS SDK oder CLI](#)

Verwenden Sie **CreateJob** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie CreateJob verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

CLI

AWS CLI

So erstellen Sie einen neuen Auftrag für Amazon S3 Batch Operations

Im folgenden Beispiel für `create-job` wird ein Auftrag von Amazon S3 Batch Operations erstellt, um Objekte als `confidential` in the bucket `employee-records` zu kennzeichnen.

```
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
```

Ausgabe:

```
{
  "JobId": "93735294-df46-44d5-8638-6356f335324e"
}
```

- Einzelheiten zur API finden Sie [CreateJob](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie einen asynchronen S3-Auftrag.

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

```

Erstellen Sie einen Auftrag zur Compliance-Aufbewahrung.

```

/**
 * 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();
}

```

Erstellen Sie einen Auftrag für eine gesetzliche Aufbewahrungsfrist.

```

/**
 * 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();
}

```

Erstellen Sie einen neuen Auftrag zur Governance-Aufbewahrung.

```

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

- Einzelheiten zur API finden Sie [CreateJob](#) in der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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

```

- Einzelheiten zur API finden Sie [CreateJob](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

 Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.

```

- Einzelheiten zur API finden Sie [CreateJob](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteJobTagging** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie DeleteJobTagging verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

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

- Einzelheiten zur API finden Sie [DeleteJobTagging](#) in der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie [DeleteJobTagging](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [DeleteJobTagging](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DescribeJob** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie `DescribeJob` verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

CLI

AWS CLI

So beschreiben Sie einen Auftrag von Amazon S3 Batch Operations

Im folgenden `describe-job` werden die Konfigurationsparameter und der Status für den angegebenen Batchoperationsauftrag bereitgestellt.

```
aws s3control describe-job \  
  --account-id 123456789012 \  
  --job-id 93735294-df46-44d5-8638-6356f335324e
```

Ausgabe:

```
{  
  "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"
}
}

```

- Einzelheiten zur API finden Sie [DescribeJob](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

/**
 * 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);
    });
}

```

- Einzelheiten zur API finden Sie [DescribeJob](#) in der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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

```

- Einzelheiten zur API finden Sie [DescribeJob](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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.
```

- Einzelheiten zur API finden Sie [DescribeJobin](#) der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetJobTagging** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie GetJobTagging verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**  
 * 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
        });
}

```

- Einzelheiten zur API finden Sie [GetJobTagging](#) in der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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
```

- Einzelheiten zur API finden Sie [GetJobTagging](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [GetJobTagging](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutJobTagging** mit einem AWS SDK

Die folgenden Code-Beispiele zeigen, wie PutJobTagging verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * 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
    });
}

```

- Einzelheiten zur API finden Sie [PutJobTagging](#) in der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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

```

- Einzelheiten zur API finden Sie [PutJobTagging](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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.

```

- Einzelheiten zur API finden Sie [PutJobTagging](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **UpdateJobPriority** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie UpdateJobPriority verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

CLI

AWS CLI

So aktualisieren Sie die Auftragspriorität eines Auftrags von Amazon S3 Batch Operations

Im folgenden Beispiel für update-job-priority wird der angegebene Auftrag auf eine neue Priorität aktualisiert.

```

aws s3control update-job-priority \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \

```

--priority 52

Ausgabe:

```
{
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386",
  "Priority": 52
}
```

- Einzelheiten zur API finden Sie [UpdateJobPriority](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
/**
 * 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;
}

```

- Einzelheiten zur API finden Sie [UpdateJobPriority](#) in der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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
```


- Einzelheiten zur API finden Sie [UpdateJobPriority](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

Note

Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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.
```

- Einzelheiten zur API finden Sie [UpdateJobPriority](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **UpdateJobStatus** mit einem AWS SDK oder CLI

Die folgenden Code-Beispiele zeigen, wie UpdateJobStatus verwendet wird.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

CLI

AWS CLI

So aktualisieren Sie den Status eines Auftrags von Amazon S3 Batch Operations

Im folgenden Beispiel für `update-job-status` wird der angegebene Auftrag gelöscht, der auf Genehmigung wartet.

```
aws s3control update-job-status \
  --account-id 123456789012 \
  --job-id 8d9a18fe-c303-4d39-8ccc-860d372da386 \
  --requested-job-status Cancelled
```

Ausgabe:

```
{
  "Status": "Cancelled",
  "JobId": "8d9a18fe-c303-4d39-8ccc-860d372da386"
}
```

Im folgenden Beispiel für `update-job-status` wird der angegebene Auftrag bestätigt und ausgeführt, der auf Genehmigung wartet.

```
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"  
}
```

Im folgenden Beispiel für `update-job-status` wird der angegebene Auftrag gelöscht, der bereits ausgeführt wird.

```
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"  
}
```

- Einzelheiten zur API finden Sie [UpdateJobStatus](#) in der AWS CLI Befehlsreferenz.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

`/**`

```

    * 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
            });
    }
}

```

- Einzelheiten zur API finden Sie [UpdateJobStatus](#) in der AWS SDK for Java 2.x API-Referenz.

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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

```

- Einzelheiten zur API finden Sie [UpdateJobStatus](#) in AWS SDK for Python (Boto3) API Reference.

SAP ABAP

SDK für SAP ABAP

 Note

Es gibt noch mehr dazu. [GitHub](#) Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```

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.

```

- Einzelheiten zur API finden Sie [UpdateJobStatus](#) in der API-Referenz zum AWS SDK für SAP ABAP.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Codebeispiele für S3-Verzeichnis-Buckets mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie Sie S3 Directory Buckets mit einem AWS Software Development Kit (SDK) verwenden.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

S3-Verzeichnis-Buckets

- [Grundlegende Beispiele für S3-Verzeichnis-Buckets mit AWS SDKs](#)
 - [Hello Amazon-S3-Verzeichnis-Buckets](#)
 - [Lernen Sie die Grundlagen von S3 Directory Buckets mit einem AWS SDK](#)
 - [Aktionen für S3-Verzeichnis-Buckets mit AWS SDKs](#)
 - [Verwenden Sie AbortMultipartUploadmit einem AWS SDK](#)
 - [Verwenden Sie CompleteMultipartUploadmit einem AWS SDK](#)
 - [Verwenden Sie CopyObjectmit einem AWS SDK](#)
 - [Verwenden Sie CreateBucketmit einem AWS SDK](#)
 - [Verwenden Sie CreateMultipartUploadmit einem AWS SDK](#)
 - [Verwenden Sie CreateSessionmit einem AWS SDK](#)
 - [Verwenden Sie DeleteBucketmit einem AWS SDK](#)
 - [Verwenden Sie DeleteBucketEncryptionmit einem AWS SDK](#)

- [Verwenden Sie DeleteBucketPolicy mit einem AWS SDK](#)
- [Verwenden Sie DeleteObject mit einem AWS SDK](#)
- [Verwenden Sie DeleteObjects mit einem AWS SDK](#)
- [Verwenden Sie GetBucketEncryption mit einem AWS SDK](#)
- [Verwenden Sie GetBucketPolicy mit einem AWS SDK](#)
- [Verwenden Sie GetObject mit einem AWS SDK](#)
- [Verwenden Sie GetObjectAttributes mit einem AWS SDK](#)
- [Verwenden Sie HeadBucket mit einem AWS SDK](#)
- [Verwenden Sie HeadObject mit einem AWS SDK](#)
- [Verwenden Sie ListDirectoryBuckets mit einem AWS SDK](#)
- [Verwenden Sie ListMultipartUploads mit einem AWS SDK](#)
- [Verwenden Sie ListObjectsV2 mit einem AWS SDK](#)
- [Verwenden Sie ListParts mit einem AWS SDK](#)
- [Verwenden Sie PutBucketEncryption mit einem AWS SDK](#)
- [Verwenden Sie PutBucketPolicy mit einem AWS SDK](#)
- [Verwenden Sie PutObject mit einem AWS SDK](#)
- [Verwenden Sie UploadPart mit einem AWS SDK](#)
- [Verwenden Sie UploadPartCopy mit einem AWS SDK](#)
- [Szenarien für S3-Verzeichnis-Buckets mit AWS SDKs](#)
 - [Erstellen Sie eine vorsignierte URL für Amazon S3 S3-Verzeichnis-Buckets, um ein Objekt mithilfe eines AWS SDK](#)

Grundlegende Beispiele für S3-Verzeichnis-Buckets mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie die Grundlagen von Amazon-S3-Verzeichnis-Buckets mit AWS SDKs verwendet werden.

Beispiele

- [Hello Amazon-S3-Verzeichnis-Buckets](#)
- [Lernen Sie die Grundlagen von S3 Directory Buckets mit einem AWS SDK](#)
- [Aktionen für S3-Verzeichnis-Buckets mit AWS SDKs](#)
 - [Verwenden Sie AbortMultipartUpload mit einem AWS SDK](#)

- [Verwenden Sie CompleteMultipartUploadmit einem AWS SDK](#)
- [Verwenden Sie CopyObjectmit einem AWS SDK](#)
- [Verwenden Sie CreateBucketmit einem AWS SDK](#)
- [Verwenden Sie CreateMultipartUploadmit einem AWS SDK](#)
- [Verwenden Sie CreateSessionmit einem AWS SDK](#)
- [Verwenden Sie DeleteBucketmit einem AWS SDK](#)
- [Verwenden Sie DeleteBucketEncryptionmit einem AWS SDK](#)
- [Verwenden Sie DeleteBucketPolycymit einem AWS SDK](#)
- [Verwenden Sie DeleteObjectmit einem AWS SDK](#)
- [Verwenden Sie DeleteObjectsmitem einem AWS SDK](#)
- [Verwenden Sie GetBucketEncryptionmit einem AWS SDK](#)
- [Verwenden Sie GetBucketPolycymit einem AWS SDK](#)
- [Verwenden Sie GetObjectmit einem AWS SDK](#)
- [Verwenden Sie GetObjectAttributesmit einem AWS SDK](#)
- [Verwenden Sie HeadBucketmit einem AWS SDK](#)
- [Verwenden Sie HeadObjectmit einem AWS SDK](#)
- [Verwenden Sie ListDirectoryBucketsmit einem AWS SDK](#)
- [Verwenden Sie ListMultipartUploadsmitem einem AWS SDK](#)
- [Verwenden Sie ListObjectsV2 mit einem AWS SDK](#)
- [Verwenden Sie ListPartsmitem einem AWS SDK](#)
- [Verwenden Sie PutBucketEncryptionmit einem AWS SDK](#)
- [Verwenden Sie PutBucketPolycymit einem AWS SDK](#)
- [Verwenden Sie PutObjectmit einem AWS SDK](#)
- [Verwenden Sie UploadPartmit einem AWS SDK](#)
- [Verwenden Sie UploadPartCopymitem einem AWS SDK](#)

Hello Amazon-S3-Verzeichnis-Buckets

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Verzeichnis-Buckets von

Amazon S3.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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;  
    }  
}  
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [CreateBucket](#)
 - [ListDirectoryBuckets](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Lernen Sie die Grundlagen von S3 Directory Buckets mit einem AWS SDK

Die folgenden Code-Beispiele veranschaulichen Folgendes:

- Richten Sie eine VPC und einen VPC-Endpunkt ein.
- Richten Sie die Richtlinien, Rollen und Benutzer so ein, dass sie mit S3-Verzeichnis-Buckets und der Speicherklasse S3 Express One Zone arbeiten.
- Erstellen Sie zwei S3-Clients.
- Erstellen Sie zwei Buckets.
- Erstellen Sie ein Objekt und kopieren Sie es.
- Demonstrieren Sie den Leistungsunterschied.
- Füllen Sie die Buckets aus, um den lexikografischen Unterschied aufzuzeigen.
- Fragen Sie den Benutzer, ob er die Ressourcen bereinigen möchte.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein interaktives Szenario durch, in dem die Amazon-S3-Features gezeigt werden.

```
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.deleteBucketAnd0bjectsAsync(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
    objects in a different manner to regular buckets. The key is in the
    \"Directory\". Where regular buckets store their key/value pairs in a
    flat manner, directory buckets use actual directories/folders.
    This allows for more rapid indexing, traversing, and therefore
    retrieval times!

    The more segmented your bucket is, with lots of
    directories, sub-directories, and objects, the more efficient it
    becomes.
    This structural difference also causes `ListObject` operations to
    behave
    differently, which can cause unexpected results. Let's add a few
    more
    objects in sub-directories to see how the output of
    ListObjects changes.
    \"\");

    waitForInputToContinue(scanner);

    // Populate a few more files in each bucket so that we can use
    // ListObjects and show the difference.
    String otherObject = "other/" + bucketObject;
    String altObject = "alt/" + bucketObject;

```

```
String otherAltObject = "other/alt/" + bucketObject;

try {
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, altObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, altObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, otherAltObject, "").join();
    s3DirectoriesActions.putObjectAsync(mS3ExpressClient,
mdirectoryBucketName, otherAltObject, "").join();

    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchBucketException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    try {
        // List objects in both S3 buckets.
        List<String> dirBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3ExpressClient,
mdirectoryBucketName).join();
        List<String> regBucketObjects =
s3DirectoriesActions.listObjectsAsync(mS3RegularClient,
mregularBucketName).join();

        logger.info("Directory bucket content");
        for (String obj : dirBucketObjects) {
            logger.info(obj);
        }

        logger.info("Regular bucket content");
        for (String obj : regBucketObjects) {
            logger.info(obj);
        }
    }
}
```

```

    }
    } catch (CompletionException e) {
        logger.error("Async operation failed: {} ",
e.getCause().getMessage());
        return;
    }

    logger.info("""
        Notice how the regular bucket lists objects in lexicographical order,
while the directory bucket does not. This is
        because the regular bucket considers the whole "key" to be the object
identifier, while the directory bucket actually
        creates directories and uses the object "key" as a path to the
object.
        """);
    waitForInputToContinue(scanner);
}

/**
 * Demonstrates the performance difference between downloading an object from
a directory bucket and a regular bucket.
 *
 * <p>This method:
 * <ul>
 *     <li>Prompts the user to choose the number of downloads (default is
1,000).</li>
 *     <li>Downloads the specified object from the directory bucket and
measures the total time.</li>
 *     <li>Downloads the same object from the regular bucket and measures the
total time.</li>
 *     <li>Compares the time differences and prints the results.</li>
 * </ul>
 *
 * <p>Note: The performance difference will be more pronounced if this
example is run on an EC2 instance
 * in the same Availability Zone as the buckets.
 *
 * @param bucketObject the name of the object to download
 */
private static void demonstratePerformance(String bucketObject) {
    logger.info(DASHES);
    logger.info("6. Demonstrate the performance difference.");
    logger.info("""

```

```

        Now, let's do a performance test. We'll download the same object from
each
        bucket repeatedly and compare the total time needed.

        Note: the performance difference will be much more pronounced if this
        example is run in an EC2 instance in the same Availability Zone as
        the bucket.
        """);
    waitForInputToContinue(scanner);

    int downloads = 1000; // Default value.
    logger.info("The default number of downloads of the same object for this
example is set at " + downloads + ".");

    // Ask if the user wants to download a different number.
    logger.info("Would you like to download the file a different number of
times? (y/n): ");
    String response = scanner.next().trim().toLowerCase();
    if (response.equals("y")) {
        int maxDownloads = 1_000_000;

        // Ask for a valid number of downloads.
        while (true) {
            logger.info("Enter a number between 1 and " + maxDownloads + "
for the number of downloads: ");
            if (scanner.hasNextInt()) {
                downloads = scanner.nextInt();
                if (downloads >= 1 && downloads <= maxDownloads) {
                    break;
                } else {
                    logger.info("Please enter a number between 1 and " +
maxDownloads + ".");
                }
            } else {
                logger.info("Invalid input. Please enter a valid integer.");
                scanner.next();
            }
        }

        logger.info("You have chosen to download {} items.", downloads);
    } else {
        logger.info("No changes made. Using default downloads: {}",
downloads);
    }
}

```

```
// Simulating the download process for the directory bucket.
logger.info("Downloading from the directory bucket.");
long directoryTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        // Get the object from the directory bucket.
        s3DirectoriesActions.getObjectAsync(mS3ExpressClient,
mdirectoryBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
}
long directoryTimeDifference = System.nanoTime() - directoryTimeStart;

// Download from the regular bucket.
logger.info("Downloading from the regular bucket.");
long normalTimeStart = System.nanoTime();
for (int index = 0; index < downloads; index++) {
    if (index % 50 == 0) {
        logger.info("Download " + index + " of " + downloads);
    }

    try {
        s3DirectoriesActions.getObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof NoSuchKeyException) {
            logger.error("S3Exception occurred: {}", cause.getMessage(),
ce);
        } else {
```

```

        logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
    }
    return;
}
}

    long normalTimeDifference = System.nanoTime() - normalTimeStart;
    logger.info("The directory bucket took " + directoryTimeDifference
+ " nanoseconds, while the regular bucket took " + normalTimeDifference + "
nanoseconds.");
    long difference = normalTimeDifference - directoryTimeDifference;
    logger.info("That's a difference of " + difference + " nanoseconds, or");
    logger.info(difference / 1_000_000_000.0 + " seconds.");

    if (difference < 0) {
        logger.info("The directory buckets were slower. This can happen if
you are not running on the cloud within a VPC.");
    }
    waitForInputToContinue(scanner);
}

private static String createSessionAddObjects() {
    logger.info(DASHES);
    logger.info("""
        5. Create an object and copy it.
        We'll create an object consisting of some text and upload it to the
        regular bucket.
        """);
    waitForInputToContinue(scanner);

    String bucketObject = "basic-text-object.txt";
    try {
        s3DirectoriesActions.putObjectAsync(mS3RegularClient,
mregularBucketName, bucketObject, "Look Ma, I'm a bucket!").join();
        s3DirectoriesActions.createSessionAsync(mS3ExpressClient,
mdirectoryBucketName).join();

        // Copy the object to the destination S3 bucket.
        s3DirectoriesActions.copyObjectAsync(mS3ExpressClient,
mregularBucketName, bucketObject, mdirectoryBucketName, bucketObject).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof S3Exception) {

```

```

        logger.error("S3Exception occurred: {}", cause.getMessage(), ce);
    } else {
        logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
    }
}
logger.info("""
    It worked! This is because the S3Client that performed the copy
operation
    is the expressClient using the credentials for the user with
permission to
    work with directory buckets.

    It's important to remember the user permissions when interacting
with
    directory buckets. Instead of validating permissions on every call
as
    regular buckets do, directory buckets utilize the user credentials
and session
    token to validate. This allows for much faster connection speeds on
every call.
    For single calls, this is low, but for many concurrent calls
    this adds up to a lot of time saved.
    """);
    waitForInputToContinue(scanner);
    return bucketObject;
}

/**
 * Creates VPC users for the S3 Express One Zone scenario.
 * <p>
 * This method performs the following steps:
 * <ol>
 *     <li>Optionally creates a new VPC and VPC Endpoint if the application
is running in an EC2 instance in the same Availability Zone as the directory
buckets.</li>
 *     <li>Creates two IAM users: one with S3 Express One Zone permissions
and one without.</li>
 * </ol>
 *
 * @return a {@link UserNames} object containing the names of the created IAM
users
 */
public static UserNames createVpcUsers() {

```



```

    /*
    Optionally create a VPC.
    Create two IAM users, one with S3 Express One Zone permissions and one
without.
    */
    logger.info(DASHES);
    logger.info("""
        1. First, we'll set up a new VPC and VPC Endpoint if this program is
running in an EC2 instance in the same AZ as your\s
        directory buckets will be. Are you running this in an EC2 instance
located in the same AZ as your intended directory buckets?
        """);

    logger.info("Do you want to setup a VPC Endpoint? (y/n)");
    String endpointAns = scanner.nextLine().trim();
    if (endpointAns.equalsIgnoreCase("y")) {
        logger.info("""
            Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to.
            """);
        try {
            s3DirectoriesActions.setupVPCAsync().join();
        } catch (CompletionException ce) {
            Throwable cause = ce.getCause();
            if (cause instanceof Ec2Exception) {
                logger.error("IamException occurred: {}", cause.getMessage(),
ce);
            } else {
                logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            }
        }
        waitForInputToContinue(scanner);
    } else {
        logger.info("Skipping the VPC setup. Don't forget to use this in
production!");
    }
    logger.info(DASHES);
    logger.info("""
        2. Create a RegularUser and ExpressUser by using the AWS CDK.
One IAM User, named RegularUser, will have permissions to work only
with regular buckets and one IAM user, named ExpressUser, will have
permissions to work only with directory buckets.
        """);

```

```

        waitForInputToContinue(scanner);

        // Create two users required for this scenario.
        Map<String, String> stackOutputs = createUsersUsingCDK(stackName);
        regularUser = stackOutputs.get("RegularUser");
        expressUser = stackOutputs.get("ExpressUser");

        UserNames names = new UserNames();
        names.setRegularUserName(regularUser);
        names.setExpressUserName(expressUser);
        return names;
    }

    /**
     * Creates users using AWS CloudFormation.
     *
     * @return a {@link Map} of String keys and String values representing the
     stack outputs,
     * which may include user-related information such as user names and IDs.
     */
    public static Map<String, String> createUsersUsingCDK(String stackName) {
        logger.info("We'll use an AWS CloudFormation template to create the IAM
        users and policies.");
        CloudFormationHelper.deployCloudFormationStack(stackName);
        return CloudFormationHelper.getStackOutputsAsync(stackName).join();
    }

    /**
     * Sets up the necessary clients and buckets for the S3 Express service.
     *
     * @param expressUserName the username for the user with S3 Express
        permissions
     * @param regularUserName the username for the user with regular S3
        permissions
     */
    public static void setupClientsAndBuckets(String expressUserName, String
        regularUserName) {
        Scanner locscanner = new Scanner(System.in);
        String accessKeyIdforRegUser;
        String secretAccessforRegUser;
        try {
            CreateAccessKeyResponse keyResponse =
            s3DirectoriesActions.createAccessKeyAsync(regularUserName).join();
            accessKeyIdforRegUser = keyResponse.accessKey().accessKeyId();

```

```

        secretAccessforRegUser = keyResponse.accessKey().secretAccessKey();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof IamException) {
            logger.error("IamException occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    String accessKeyIdforExpressUser;
    String secretAccessforExpressUser;
    try {
        CreateAccessKeyResponse keyResponseExpress =
s3DirectoriesActions.createAccessKeyAsync(expressUserName).join();
        accessKeyIdforExpressUser =
keyResponseExpress.accessKey().accessKeyId();
        secretAccessforExpressUser =
keyResponseExpress.accessKey().secretAccessKey();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof IamException) {
            logger.error("IamException occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    logger.info(DASHES);
    logger.info("""
        3. Create two S3Clients; one uses the ExpressUser's credentials and
one uses the RegularUser's credentials.
        The 2 S3Clients will use different credentials.
        """);
    waitForInputToContinue(locscanner);
    try {

```

```

        mS3RegularClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforRegUser,
secretAccessforRegUser).join();
        mS3ExpressClient =
createS3ClientWithAccessKeyAsync(accessKeyIdforExpressUser,
secretAccessforExpressUser).join();
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof IllegalArgumentException) {
            logger.error("An invalid argument exception occurred: {}",
cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    logger.info("""
        We can now use the ExpressUser client to make calls to S3 Express
operations.
        """);
    waitForInputToContinue(locscanner);
    logger.info(DASHES);
    logger.info("""
        4. Create two buckets.
        Now we will create a directory bucket which is the linchpin of the S3
Express One Zone service. Directory buckets
        behave differently from regular S3 buckets which we will explore
here. We'll also create a regular bucket, put
        an object into the regular bucket, and copy it to the directory
bucket.
        """);

    logger.info("""
        Now, let's choose an availability zone (AZ) for the directory
bucket.
        We'll choose one that is supported.
        """);
    String zoneId;
    String regularBucketName;
    try {
        zoneId = s3DirectoriesActions.selectAvailabilityZoneIdAsync().join();

```

```

        regularBucketName = "amzn-s3-demo-bucket"; // Replace with your
        bucket name.
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof Ec2Exception) {
            logger.error("EC2Exception occurred: {}", cause.getMessage(),
ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }
    logger.info("""
        Now, let's create the actual directory bucket, as well as a regular
        bucket."
        """);

    String directoryBucketName = "amzn-s3-demo-bucket--" + zoneId + "--x-
s3"; // Replace with your directory bucket name.
    try {
        s3DirectoriesActions.createDirectoryBucketAsync(mS3ExpressClient,
directoryBucketName, zoneId).join();
        logger.info("Created directory bucket {}", directoryBucketName);
    } catch (CompletionException ce) {
        Throwable cause = ce.getCause();
        if (cause instanceof BucketAlreadyExistsException) {
            logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
        }
        return;
    }

    // Assign to the data member.
    mdirectoryBucketName = directoryBucketName;
    try {
        s3DirectoriesActions.createBucketAsync(mS3RegularClient,
regularBucketName).join();
        logger.info("Created regular bucket {} ", regularBucketName);
        mregularBucketName = regularBucketName;
    } catch (CompletionException ce) {

```

```

        Throwable cause = ce.getCause();
        if (cause instanceof BucketAlreadyExistsException) {
            logger.error("The bucket already exists. Moving on: {}",
cause.getMessage(), ce);
        } else {
            logger.error("An unexpected error occurred: {}",
cause.getMessage(), ce);
            return;
        }
    }
    logger.info("Great! Both buckets were created.");
    waitForInputToContinue(locscanner);
}

/**
 * Creates an asynchronous S3 client with the specified access key and secret
access key.
 *
 * @param accessKeyId      the AWS access key ID
 * @param secretAccessKey the AWS secret access key
 * @return a {@link CompletableFuture} that asynchronously creates the S3
client
 * @throws IllegalArgumentException if the access key ID or secret access key
is null
 */
public static CompletableFuture<S3AsyncClient>
createS3ClientWithAccessKeyAsync(String accessKeyId, String secretAccessKey) {
    return CompletableFuture.supplyAsync(() -> {
        // Validate input parameters
        if (accessKeyId == null || accessKeyId.isBlank() || secretAccessKey
== null || secretAccessKey.isBlank()) {
            throw new IllegalArgumentException("Access Key ID and Secret
Access Key must not be null or empty");
        }

        AwsBasicCredentials awsCredentials =
AwsBasicCredentials.create(accessKeyId, secretAccessKey);
        return S3AsyncClient.builder()

.credentialsProvider(StaticCredentialsProvider.create(awsCredentials))
        .region(Region.US_WEST_2)
        .build();
    });
}

```

```
private static void waitForInputToContinue(Scanner scanner) {
    while (true) {
        logger.info("");
        logger.info("Enter 'c' followed by <ENTER> to continue:");
        String input = scanner.nextLine();

        if (input.trim().equalsIgnoreCase("c")) {
            logger.info("Continuing with the program...");
            logger.info("");
            break;
        } else {
            logger.info("Invalid input. Please try again.");
        }
    }
}
```

Eine Wrapper-Klasse für Methoden des Amazon S3 SDK.

```
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()));
                        }
                    });
            }
        })
        .thenCompose(ignored -> {
            DeleteBucketRequest deleteBucketRequest =
DeleteBucketRequest.builder()
                .bucket(bucketName)
                .build();
            return s3AsyncClient.deleteBucket(deleteBucketRequest);
        });
}

```

```

    })
    .thenCompose(ignored -> {
        S3AsyncWaiter waiter = s3AsyncClient.waiter();
        HeadBucketRequest headBucketRequest =
HeadBucketRequest.builder().bucket(bucketName).build();
        return waiter.waitUntilBucketNotExists(headBucketRequest);
    })
    .whenComplete((ignored, exception) -> {
        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof S3Exception) {
                throw new CompletionException("Error deleting bucket: " +
bucketName, cause);
            }
            throw new CompletionException("Failed to delete bucket and
objects: " + bucketName, exception);
        }
        logger.info("Bucket deleted successfully: " + bucketName);
    });
}

/**
 * Lists the objects in an S3 bucket asynchronously.
 *
 * @param s3Client the S3 async client to use for the operation
 * @param bucketName the name of the S3 bucket containing the objects to list
 * @return a {@link CompletableFuture} that contains the list of object keys
in the specified bucket
 */
public CompletableFuture<List<String>> listObjectsAsync(S3AsyncClient
s3Client, String bucketName) {
    ListObjectsV2Request request = ListObjectsV2Request.builder()
        .bucket(bucketName)
        .build();

    return s3Client.listObjectsV2(request)
        .thenApply(response -> response.contents().stream()
            .map(S3Object::key)
            .toList())
        .whenComplete((result, exception) -> {
            if (exception != null) {
                throw new CompletionException("Couldn't list objects in
bucket: " + bucketName, exception);
            }
        })
}

```

```

        });
    }

    /**
     * Retrieves an object from an Amazon S3 bucket asynchronously.
     *
     * @param s3Client    the S3 async client to use for the operation
     * @param bucketName the name of the S3 bucket containing the object
     * @param keyName     the unique identifier (key) of the object to retrieve
     * @return a {@link CompletableFuture} that, when completed, contains the
     *         object's content as a {@link ResponseBytes} of {@link GetObjectResponse}
     */
    public CompletableFuture<ResponseBytes<GetObjectResponse>>
    getObjectAsync(S3AsyncClient s3Client, String bucketName, String keyName) {
        GetObjectRequest objectRequest = GetObjectRequest.builder()
            .key(keyName)
            .bucket(bucketName)
            .build();

        // Get the object asynchronously and transform it into a byte array
        return s3Client.getObject(objectRequest,
            AsyncResponseTransformer.toBytes())
            .exceptionally(exception -> {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchKeyException) {
                    throw new CompletionException("Failed to get the object.
Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to get the object",
exception);
            });
    }

    /**
     * Asynchronously copies an object from one S3 bucket to another.
     *
     * @param s3Client    the S3 async client to use for the copy
operation
     * @param sourceBucket the name of the source bucket
     * @param sourceKey    the key of the object to be copied in the source
bucket
     * @param destinationBucket the name of the destination bucket
     * @param destinationKey  the key of the copied object in the destination
bucket

```

```

    * @return a {@link CompletableFuture} that completes when the copy operation
    is finished
    */
    public CompletableFuture<Void> copyObjectAsync(S3AsyncClient s3Client, String
sourceBucket, String sourceKey, String destinationBucket, String destinationKey)
{
    CopyObjectRequest copyRequest = CopyObjectRequest.builder()
        .sourceBucket(sourceBucket)
        .sourceKey(sourceKey)
        .destinationBucket(destinationBucket)
        .destinationKey(destinationKey)
        .build();

    return s3Client.copyObject(copyRequest)
        .thenRun(() -> logger.info("Copied object '" + sourceKey + "' from
bucket '" + sourceBucket + "' to bucket '" + destinationBucket + "'"))
        .whenComplete((ignored, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof S3Exception) {
                    throw new CompletionException("Couldn't
copy object '" + sourceKey + "' from bucket '" + sourceBucket + "'
to bucket '" + destinationBucket + "'. Reason: " + ((S3Exception)
cause).awsErrorDetails().errorMessage(), cause);
                }
                throw new CompletionException("Failed to copy object",
exception);
            }
        });
}

/**
 * Asynchronously creates a session for the specified S3 bucket.
 *
 * @param s3Client the S3 asynchronous client to use for creating the
session
 * @param bucketName the name of the S3 bucket for which to create the
session
 * @return a {@link CompletableFuture} that completes when the session is
created, or throws a {@link CompletionException} if an error occurs
 */
    public CompletableFuture<CreateSessionResponse>
createSessionAsync(S3AsyncClient s3Client, String bucketName) {
        CreateSessionRequest request = CreateSessionRequest.builder()

```

```

        .bucket(bucketName)
        .build();

    return s3Client.createSession(request)
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof S3Exception) {
                    throw new CompletionException("Couldn't create the
session. Reason: " + ((S3Exception) cause).awsErrorDetails().errorMessage(),
cause);
                }
                throw new CompletionException("Unexpected error occurred
while creating session", exception);
            }
            logger.info("Created session for bucket: " + bucketName);
        });
    }

/**
 * Creates a new S3 directory bucket in a specified Zone (For example, a
 * specified Availability Zone in this code example).
 *
 * @param s3Client    The asynchronous S3 client used to create the bucket
 * @param bucketName The name of the bucket to be created
 * @param zone        The Availability Zone where the bucket will be created
 * @throws CompletionException if there's an error creating the bucket
 */
    public CompletableFuture<CreateBucketResponse>
createDirectoryBucketAsync(S3AsyncClient s3Client, String bucketName, String
zone) {
        logger.info("Creating bucket: " + bucketName);

        CreateBucketConfiguration bucketConfiguration =
CreateBucketConfiguration.builder()
            .location(LocationInfo.builder()
                .type(LocationType.AVAILABILITY_ZONE)
                .name(zone)
                .build())
            .bucket(BucketInfo.builder()
                .type(BucketType.DIRECTORY)
                .dataRedundancy(DataRedundancy.SINGLE_AVAILABILITY_ZONE)
                .build())

```

```

        .build();

        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .createBucketConfiguration(bucketConfiguration)
            .build();

        return s3Client.createBucket(bucketRequest)
            .whenComplete((response, exception) -> {
                if (exception != null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof BucketAlreadyExistsException) {
                        throw new CompletionException("The bucket already exists:
" + ((S3Exception) cause).awsErrorDetails().errorMessage(), cause);
                    }
                    throw new CompletionException("Unexpected error occurred
while creating bucket", exception);
                }
                logger.info("Bucket created successfully with location: " +
response.location());
            });
    }

    /**
     * Creates an S3 bucket asynchronously.
     *
     * @param s3Client    the S3 async client to use for the bucket creation
     * @param bucketName  the name of the S3 bucket to create
     * @return a {@link CompletableFuture} that completes with the {@link
WaiterResponse} containing the {@link HeadBucketResponse}
     *         when the bucket is successfully created
     * @throws CompletionException if there's an error creating the bucket
     */
    public CompletableFuture<WaiterResponse<HeadBucketResponse>>
createBucketAsync(S3AsyncClient s3Client, String bucketName) {
        CreateBucketRequest bucketRequest = CreateBucketRequest.builder()
            .bucket(bucketName)
            .build();

        return s3Client.createBucket(bucketRequest)
            .thenCompose(response -> {
                S3AsyncWaiter s3Waiter = s3Client.waiter();
                HeadBucketRequest bucketRequestWait = HeadBucketRequest.builder()
                    .bucket(bucketName)

```

```

        .build();
        return s3Waiter.waitUntilBucketExists(bucketRequestWait);
    })
    .whenComplete((response, exception) -> {
        if (exception != null) {
            Throwable cause = exception.getCause();
            if (cause instanceof BucketAlreadyExistsException) {
                throw new CompletionException("The S3 bucket exists: " +
cause.getMessage(), cause);
            } else {
                throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);
            }
        }
        logger.info(bucketName + " is ready");
    });
}

/**
 * Uploads an object to an Amazon S3 bucket asynchronously.
 *
 * @param s3Client      the S3 async client to use for the upload
 * @param bucketName    the destination S3 bucket name
 * @param bucketObject  the name of the object to be uploaded
 * @param text          the content to be uploaded as the object
 */
public CompletableFuture<PutObjectResponse> putObjectAsync(S3AsyncClient
s3Client, String bucketName, String bucketObject, String text) {
    PutObjectRequest objectRequest = PutObjectRequest.builder()
        .bucket(bucketName)
        .key(bucketObject)
        .build();

    return s3Client.putObject(objectRequest,
AsyncRequestBody.fromString(text))
        .whenComplete((response, exception) -> {
            if (exception != null) {
                Throwable cause = exception.getCause();
                if (cause instanceof NoSuchBucketException) {
                    throw new CompletionException("The S3 bucket does not
exist: " + cause.getMessage(), cause);
                } else {
                    throw new CompletionException("Failed to create access
key: " + exception.getMessage(), exception);

```

```

        }
    }
    });
}

/**
 * Creates an AWS IAM access key asynchronously for the specified user name.
 *
 * @param userName the name of the IAM user for whom to create the access key
 * @return a {@link CompletableFuture} that completes with the {@link
CreateAccessKeyResponse} containing the created access key
 */
public CompletableFuture<CreateAccessKeyResponse> createAccessKeyAsync(String
userName) {
    CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
        .userName(userName)
        .build();

    return getIAMAsyncClient().createAccessKey(request)
        .whenComplete((response, exception) -> {
            if (response != null) {
                logger.info("Access Key Created.");
            } else {
                if (exception == null) {
                    Throwable cause = exception.getCause();
                    if (cause instanceof IamException) {
                        throw new CompletionException("IAM error while
creating access key: " + cause.getMessage(), cause);
                    } else {
                        throw new CompletionException("Failed to create
access key: " + exception.getMessage(), exception);
                    }
                }
            }
        });
}

/**
 * Asynchronously selects an Availability Zone ID from the available EC2
zones.
 *
 * @return A {@link CompletableFuture} that resolves to the selected
Availability Zone ID.

```



```

    * @throws CompletionException if an error occurs during the request or
    processing.
    */
    public CompletableFuture<String> selectAvailabilityZoneIdAsync() {
        DescribeAvailabilityZonesRequest zonesRequest =
DescribeAvailabilityZonesRequest.builder()
            .build();

        return getEc2AsyncClient().describeAvailabilityZones(zonesRequest)
            .thenCompose(response -> {
                List<AvailabilityZone> zonesList = response.availabilityZones();
                if (zonesList.isEmpty()) {
                    logger.info("No availability zones found.");
                    return CompletableFuture.completedFuture(null); // Return
null if no zones are found
                }

                List<String> zoneIds = zonesList.stream()
                    .map(AvailabilityZone::zoneId) // Get the zoneId (e.g.,
"usw2-az1")
                    .toList();

                return CompletableFuture.supplyAsync(() ->
promptUserForZoneSelection(zonesList, zoneIds))
                    .thenApply(selectedZone -> {
                        // Return only the selected Zone ID (e.g., "usw2-az1").
                        return selectedZone.zoneId();
                    });
            })
            .whenComplete((result, exception) -> {
                if (exception == null) {
                    if (result != null) {
                        logger.info("Selected Availability Zone ID: " + result);
                    } else {
                        logger.info("No availability zone selected.");
                    }
                } else {
                    Throwable cause = exception.getCause();
                    if (cause instanceof Ec2Exception) {
                        throw new CompletionException("EC2 error while selecting
availability zone: " + cause.getMessage(), cause);
                    }
                    throw new CompletionException("Failed to select availability
zone: " + exception.getMessage(), exception);
                }
            });
    }

```

```

        }
    });
}

/**
 * Prompts the user to select an Availability Zone from the given list.
 *
 * @param zonesList the list of Availability Zones
 * @param zoneIds the list of zone IDs
 * @return the selected Availability Zone
 */
private static AvailabilityZone
promptUserForZoneSelection(List<AvailabilityZone> zonesList, List<String>
zoneIds) {
    Scanner scanner = new Scanner(System.in);
    int index = -1;

    while (index < 0 || index >= zoneIds.size()) {
        logger.info("Select an availability zone:");
        IntStream.range(0, zoneIds.size()).forEach(i ->
            logger.info(i + ": " + zoneIds.get(i))
        );

        logger.info("Enter the number corresponding to your choice: ");
        if (scanner.hasNextInt()) {
            index = scanner.nextInt();
        } else {
            scanner.next();
        }
    }

    AvailabilityZone selectedZone = zonesList.get(index);
    logger.info("You selected: " + selectedZone.zoneId());
    return selectedZone;
}

/**
 * Asynchronously sets up a new VPC, including creating the VPC, finding the
 * associated route table, and
 * creating a VPC endpoint for the S3 service.
 *
 * @return a {@link CompletableFuture} that, when completed, contains a
 * AbstractMap with the
 * VPC ID and VPC endpoint ID.

```

```

    */
    public CompletableFuture<AbstractMap.SimpleEntry<String, String>>
    setupVPCAsync() {
        String cidr = "10.0.0.0/16";
        CreateVpcRequest vpcRequest = CreateVpcRequest.builder()
            .cidrBlock(cidr)
            .build();

        return getEc2AsyncClient().createVpc(vpcRequest)
            .thenCompose(vpcResponse -> {
                String vpcId = vpcResponse.vpc().vpcId();
                logger.info("VPC Created: {}", vpcId);

                Ec2AsyncWaiter waiter = getEc2AsyncClient().waiter();
                DescribeVpcsRequest request = DescribeVpcsRequest.builder()
                    .vpcIds(vpcId)
                    .build();

                return waiter.waitUntilVpcAvailable(request)
                    .thenApply(waiterResponse -> vpcId);
            })
            .thenCompose(vpcId -> {
                Filter filter = Filter.builder()
                    .name("vpc-id")
                    .values(vpcId)
                    .build();

                DescribeRouteTablesRequest describeRouteTablesRequest =
                DescribeRouteTablesRequest.builder()
                    .filters(filter)
                    .build();

                return
                getEc2AsyncClient().describeRouteTables(describeRouteTablesRequest)
                    .thenApply(routeTablesResponse -> {
                        if (routeTablesResponse.routeTables().isEmpty()) {
                            throw new CompletionException("No route tables found
for VPC: " + vpcId, null);
                        }
                        String routeTableId =
routeTablesResponse.routeTables().get(0).routeTableId();
                        logger.info("Route table found: {}", routeTableId);
                        return new AbstractMap.SimpleEntry<>(vpcId,
routeTableId);
                    });
            });
    }

```

```

        });
    })
    .thenCompose(vpcAndRouteTable -> {
        String vpcId = vpcAndRouteTable.getKey();
        String routeTableId = vpcAndRouteTable.getValue();
        Region region =
getEc2AsyncClient().serviceClientConfiguration().region();
        String serviceName = String.format("com.amazonaws.%s.s3express",
region.id());

        CreateVpcEndpointRequest endpointRequest =
CreateVpcEndpointRequest.builder()
            .vpcId(vpcId)
            .routeTableIds(routeTableId)
            .serviceName(serviceName)
            .build();

        return getEc2AsyncClient().createVpcEndpoint(endpointRequest)
            .thenApply(vpcEndpointResponse -> {
                String vpcEndpointId =
vpcEndpointResponse.vpcEndpoint().vpcEndpointId();
                logger.info("VPC Endpoint created: {}", vpcEndpointId);
                return new AbstractMap.SimpleEntry<>(vpcId,
vpcEndpointId);
            });
    })
    .exceptionally(exception -> {
        Throwable cause = exception.getCause() != null ?
exception.getCause() : exception;
        if (cause instanceof Ec2Exception) {
            logger.error("EC2 error during VPC setup: {}",
cause.getMessage(), cause);
            throw new CompletionException("EC2 error during VPC setup: "
+ cause.getMessage(), cause);
        }

        logger.error("VPC setup failed: {}", cause.getMessage(), cause);
        throw new CompletionException("VPC setup failed: " +
cause.getMessage(), cause);
    });
}
}

```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK for Java 2.x - API-Referenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

PHP

SDK für PHP

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein Szenario aus, in dem die Grundlagen des Verzeichnis-Buckets von Amazon S3 und S3 Express One Zone gezeigt werden.

```
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 = uuid();

    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;  
    }  
}  
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für PHP - API-Referenz.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

Python

SDK für Python (Boto3)

Note

Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Führen Sie ein Szenario aus, in dem die Grundlagen des Verzeichnis-Buckets von Amazon S3 und S3 Express One Zone gezeigt werden.

```
class S3ExpressScenario:
    """Runs an interactive scenario that shows how to get started with S3
    Express."""

    def __init__(
        self,
        cloud_formation_resource: ServiceResource,
        ec2_client: client,
        iam_client: client,
    ):
        self.cloud_formation_resource = cloud_formation_resource
        self.ec2_client = ec2_client
        self.iam_client = iam_client
        self.region = ec2_client.meta.region_name
        self.stack = None
        self.vpc_id = None
        self.vpc_endpoint_id = None
        self.regular_bucket_name = None
        self.directory_bucket_name = None
        self.s3_express_wrapper = None
        self.s3_regular_wrapper = None

    def s3_express_scenario(self):
        """
        Runs the scenario.
        """
        print("")
        print_dashes()
```

```

    print("Welcome to the Amazon S3 Express Basics demo using Python (Boto
3)!")
    print_dashes()
    print(
        """

```

Let's get started! First, please note that S3 Express One Zone works best when working within the AWS infrastructure, specifically when working in the same Availability Zone. To see the best results in this example and when you implement Directory buckets into your infrastructure, it is best to put your compute resources in the same AZ as your Directory bucket.

```

        """
    )
    press_enter_to_continue()

    # Create an optional VPC and create 2 IAM users.
    express_user_name, regular_user_name = self.create_vpc_and_users()

    # Set up two S3 clients, one regular and one express, and two buckets,
    one regular and one express.
    self.setup_clients_and_buckets(express_user_name, regular_user_name)

    # Create an S3 session for the express S3 client and add objects to the
    buckets.
    bucket_object = self.create_session_and_add_objects()

    # Demonstrate performance differences between regular and express
    buckets.
    self.demonstrate_performance(bucket_object)

    # Populate the buckets to show the lexicographical difference between
    regular and express buckets.
    self.show_lexicographical_differences(bucket_object)

    print("")
    print("That's it for our tour of the basic operations for S3 Express One
Zone.")

    if q.ask(
        "Would you like to delete all the resources created during this demo
(y/n)? ",
        q.is_yesno,
    ):

```

```

        self.cleanup()

    def create_vpc_and_users(self) -> None:
        """
        Optionally create a VPC.
        Create two IAM users, one with S3 Express One Zone permissions and one
        without.
        """
        # Configure a gateway VPC endpoint. This is the recommended method to
        allow S3 Express One Zone traffic without
        # the need to pass through an internet gateway or NAT device.
        print(
            """
1. First, we'll set up a new VPC and VPC Endpoint if this program is running in
an EC2 instance in the same AZ as your
Directory buckets will be. Are you running this in an EC2 instance located in the
same AZ as your intended Directory buckets?
"""
        )
        if q.ask("Do you want to setup a VPC Endpoint? (y/n) ", q.is_yesno):
            print(
                "Great! Let's set up a VPC, retrieve the Route Table from it, and
create a VPC Endpoint to connect the S3 Client to."
            )
            self.setup_vpc()
            press_enter_to_continue()
        else:
            print("Skipping the VPC setup. Don't forget to use this in
production!")
            print(
                """
2. Policies, users, and roles with CDK.
Now, we'll set up some policies, roles, and a user. This user will only have
permissions to do S3 Express One Zone actions.
"""
            )
            press_enter_to_continue()
            stack_name = f"cfn-stack-s3-express-basics--{uuid.uuid4()}"
            template_as_string = S3ExpressScenario.get_template_as_string()
            self.stack = self.deploy_cloudformation_stack(stack_name,
template_as_string)
            regular_user_name = None
            express_user_name = None
            outputs = self.stack.outputs

```

```

    for output in outputs:
        if output.get("OutputKey") == "RegularUser":
            regular_user_name = output.get("OutputValue")
        elif output.get("OutputKey") == "ExpressUser":
            express_user_name = output.get("OutputValue")
    if not regular_user_name or not express_user_name:
        error_string = f"""
        Failed to retrieve required outputs from CloudFormation stack.
        'regular_user_name'={regular_user_name},
'express_user_name'={express_user_name}
        """
        logger.error(error_string)
        raise ValueError(error_string)
    return express_user_name, regular_user_name

def setup_clients_and_buckets(
    self, express_user_name: str, regular_user_name: str
) -> None:
    """
    Set up two S3 clients, one regular and one express, and two buckets, one
    regular and one express.
    :param express_user_name: The name of the user with S3 Express
    permissions.
    :param regular_user_name: The name of the user with regular S3
    permissions.
    """
    regular_credentials = self.create_access_key(regular_user_name)
    express_credentials = self.create_access_key(express_user_name)
    # 3. Create an additional client using the credentials with S3 Express
    permissions.
    print(
        """
3. Create an additional client using the credentials with S3 Express permissions.
This client is created with the
credentials associated with the user account with the S3 Express policy attached,
so it can perform S3 Express operations.
"""
    )
    press_enter_to_continue()
    s3_regular_client = self.create_s3_client_with_access_key_credentials(
        regular_credentials
    )
    self.s3_regular_wrapper = S3ExpressWrapper(s3_regular_client)
    s3_express_client = self.create_s3_client_with_access_key_credentials(

```



```

        express_credentials
    )
    self.s3_express_wrapper = S3ExpressWrapper(s3_express_client)
    print(
        """

```

All the roles and policies were created and attached to the user. Then a new S3 Client were created using that user's credentials. We can now use this client to make calls to S3 Express operations. Keeping permissions in mind (and adhering to least-privilege) is crucial to S3 Express.

```

    """

```

```

    )
    press_enter_to_continue()
    # 4. Create two buckets.
    print(
        """

```

3. Create two buckets.

Now we will create a Directory bucket which is the linchpin of the S3 Express One Zone service. Directory buckets

behave in different ways from regular S3 buckets which we will explore here.

We'll also create a normal bucket, put an object into the normal bucket, and copy it over to the Directory bucket.

```

    """

```

```

    )

```

```

        # Create a directory bucket. These are different from normal S3 buckets
        in subtle ways.

```

```

        bucket_prefix = q.ask(
            "Enter a bucket name prefix that will be used for both buckets: ",
            q.re_match(r"[a-z0-9](?:[a-z0-9-\.]*)[a-z0-9]$",),
        )

```

```

        # Some availability zones are not supported for Directory buckets. We'll
        choose one that is supported.

```

```

        print(
            "Now, let's choose an availability zone for the Directory bucket.

```

```

        We'll choose one that is supported."

```

```

        )

```

```

        while True:

```

```

            availability_zone = self.select_availability_zone_id(self.region)

```

```

            # Construct the parts of a directory bucket name that is made unique
            with a UUID string.

```

```

            directory_bucket_suffix = f"--{availability_zone['ZoneId']}--x-s3"

```

```

        max_uuid_length = 63 - len(bucket_prefix) -
len(directory_bucket_suffix) - 1
        bucket_uuid = str(uuid.uuid4()).replace("-", "")[:max_uuid_length]
        directory_bucket_name = (
            f"{bucket_prefix}-{bucket_uuid}{directory_bucket_suffix}"
        )
        regular_bucket_name = f"{bucket_prefix}-regular-{bucket_uuid}"
        configuration = {
            "Bucket": {
                "Type": "Directory",
                "DataRedundancy": "SingleAvailabilityZone",
            },
            "Location": {
                "Name": availability_zone["ZoneId"],
                "Type": "AvailabilityZone",
            },
        }
        press_enter_to_continue()
        print(
            "Now, let's create the actual Directory bucket, as well as a
regular bucket."
        )
        press_enter_to_continue()
        try:
            self.s3_express_wrapper.create_bucket(
                directory_bucket_name, configuration
            )
            break
        except ClientError as client_error:
            if client_error.response["Error"]["Code"] == "InvalidBucketName":
                print(
                    f"Bucket '{directory_bucket_name}' is invalid. This may
be because of selected availability zone."
                )
                if q.ask(
                    "Would you like to select a different availability zone?"
                ),
                    q.is_yesno,
                ):
                    continue
            else:
                raise
        else:
            raise

```

```

        print(f"Created directory bucket, '{directory_bucket_name}'")
        self.directory_bucket_name = directory_bucket_name

        self.s3_regular_wrapper.create_bucket(regular_bucket_name)
        print(f"Created regular bucket, '{regular_bucket_name}'")
        self.regular_bucket_name = regular_bucket_name
        print("Great! Both buckets were created.")
        press_enter_to_continue()

    def create_session_and_add_objects(self) -> None:
        """
        Create a session for the express S3 client and add objects to the
        buckets.
        """
        print(
            """
5. Create an object and copy it over.
We'll create a basic object consisting of some text and upload it to the normal
bucket. Next we'll copy the object
into the Directory bucket using the regular client. This works fine because copy
operations are not restricted for
Directory buckets.
            """
        )
        press_enter_to_continue()
        bucket_object = "basic-text-object"
        self.s3_regular_wrapper.put_object(
            self.regular_bucket_name, bucket_object, "Look Ma, I'm a bucket!"
        )
        self.s3_express_wrapper.create_session(self.directory_bucket_name)
        self.s3_express_wrapper.copy_object(
            self.regular_bucket_name,
            bucket_object,
            self.directory_bucket_name,
            bucket_object,
        )
        print(
            """
It worked! It's important to remember the user permissions when interacting with
Directory buckets. Instead of validating
permissions on every call as normal buckets do, Directory buckets utilize the
user credentials and session token to validate.
This allows for much faster connection speeds on every call. For single calls,
this is low, but for many concurrent calls
            """
        )

```

```
this adds up to a lot of time saved.
```

```
"""
```

```
    )
    press_enter_to_continue()
    return bucket_object
```

```
def demonstrate_performance(self, bucket_object: str) -> None:
```

```
    """
```

```
    Demonstrate performance differences between regular and Directory
    buckets.
```

```
    :param bucket_object: The name of the object to download from each
    bucket.
```

```
    """
```

```
    print("")
    print("6. Demonstrate performance difference.")
    print(
        """
```

Now, let's do a performance test. We'll download the same object from each bucket
'downloads' times

and compare the total time needed. Note: the performance difference will be much
more pronounced if this

example is run in an EC2 instance in the same Availability Zone as the bucket.

```
"""
```

```
    )
    downloads = 1000
    print(
```

```
        f"The number of downloads of the same object for this example is set
    at {downloads}."
    )
```

```
    if q.ask("Would you like to download a different number? (y/n) ",
    q.is_yesno):
```

```
        max_downloads = 1000000
```

```
        downloads = q.ask(
```

```
            f"Enter a number between 1 and {max_downloads} for the number of
    downloads: ",
```

```
            q.is_int,
            q.in_range(1, max_downloads),
        )
```

```
    # Download the object 'downloads' times from each bucket and time it to
    demonstrate the speed difference.
```

```
    print("Downloading from the Directory bucket.")
```

```
    directory_time_start = time.time_ns()
```

```
    for index in range(downloads):
```

```

        if index % 10 == 0:
            print(f"Download {index} of {downloads}")

        self.s3_express_wrapper.get_object(
            self.directory_bucket_name, bucket_object
        )

    directory_time_difference = time.time_ns() - directory_time_start
    print("Downloading from the normal bucket.")
    normal_time_start = time.time_ns()

    for index in range(downloads):
        if index % 10 == 0:
            print(f"Download {index} of {downloads}")
            self.s3_regular_wrapper.get_object(self.regular_bucket_name,
            bucket_object)

    normal_time_difference = time.time_ns() - normal_time_start
    print(
        f"The directory bucket took {directory_time_difference} nanoseconds,
while the normal bucket took {normal_time_difference}."
    )
    difference = normal_time_difference - directory_time_difference
    print(f"That's a difference of {difference} nanoseconds, or")
    print(f"{(difference) / 1000000000} seconds.")
    if difference < 0:
        print(
            "The directory buckets were slower. This can happen if you are
not running on the cloud within a vpc."
        )
        press_enter_to_continue()

    def show_lexicographical_differences(self, bucket_object: str) -> None:
        """
        Show the lexicographical difference between Directory buckets and regular
        buckets.

        This is done by creating a few objects in each bucket and listing them to
        show the difference.

        :param bucket_object: The object to use for the listing operations.
        """
        print(
            """
7. Populate the buckets to show the lexicographical difference.

```

Now let's explore how Directory buckets store objects in a different manner to regular buckets. The key is in the name "Directory". Where regular buckets store their key/value pairs in a flat manner, Directory buckets use actual directories/folders. This allows for more rapid indexing, traversing, and therefore retrieval times! The more segmented your bucket is, with lots of directories, sub-directories, and objects, the more efficient it becomes. This structural difference also causes ListObjects to behave differently, which can cause unexpected results. Let's add a few more objects with layered directories to see how the output of ListObjects changes.

```

"""
)
press_enter_to_continue()
# Populate a few more files in each bucket so that we can use ListObjects
and show the difference.
other_object = f"other/{bucket_object}"
alt_object = f"alt/{bucket_object}"
other_alt_object = f"other/alt/{bucket_object}"
self.s3_regular_wrapper.put_object(self.regular_bucket_name,
other_object, "")
self.s3_express_wrapper.put_object(self.directory_bucket_name,
other_object, "")
self.s3_regular_wrapper.put_object(self.regular_bucket_name, alt_object,
"")
self.s3_express_wrapper.put_object(self.directory_bucket_name,
alt_object, "")
self.s3_regular_wrapper.put_object(
    self.regular_bucket_name, other_alt_object, ""
)
self.s3_express_wrapper.put_object(
    self.directory_bucket_name, other_alt_object, ""
)
)
directory_bucket_objects = self.s3_express_wrapper.list_objects(
    self.directory_bucket_name
)

regular_bucket_objects = self.s3_regular_wrapper.list_objects(
    self.regular_bucket_name
)

print("Directory bucket content")
for bucket_object in directory_bucket_objects:
    print(f"    {bucket_object['Key']}")

```

```

print("Normal bucket content")
for bucket_object in regular_bucket_objects:
    print(f"    {bucket_object['Key']}")
print(
    """

```

Notice how the normal bucket lists objects in lexicographical order, while the directory bucket does not. This is because the normal bucket considers the whole "key" to be the object identifier, while the directory bucket actually creates directories and uses the object "key" as a path to the object.

```

    """
)
press_enter_to_continue()

def cleanup(self) -> None:
    """
    Delete resources created by this scenario.
    """
    if self.directory_bucket_name is not None:
        self.s3_express_wrapper.delete_bucket_and_objects(
            self.directory_bucket_name
        )
        print(f"Deleted directory bucket, '{self.directory_bucket_name}'")
        self.directory_bucket_name = None

    if self.regular_bucket_name is not None:

self.s3_regular_wrapper.delete_bucket_and_objects(self.regular_bucket_name)
        print(f"Deleted regular bucket, '{self.regular_bucket_name}'")
        self.regular_bucket_name = None

    if self.stack is not None:
        self.destroy_cloudformation_stack(self.stack)
        self.stack = None

    self.tear_down_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_done_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"],
        )

```

Eine Wrapper-Klasse für SDK-Funktionen von 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
```

- Weitere API-Informationen finden Sie in den folgenden Themen der API-Referenz zum AWS SDK für Python (Boto3).
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObject](#)
 - [GetObject](#)
 - [ListObjects](#)
 - [PutObject](#)

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Aktionen für S3-Verzeichnis-Buckets mit AWS SDKs

Die folgenden Codebeispiele zeigen, wie einzelne S3 Directory Buckets-Aktionen mit SDKs ausgeführt werden. AWS Jedes Beispiel enthält einen Link zu GitHub, wo Sie Anweisungen zum Einrichten und Ausführen des Codes finden.

Diese Auszüge rufen die S3-Verzeichnis-Buckets-API auf und sind Codeauszüge aus größeren Programmen, die im Kontext ausgeführt werden müssen. Sie können Aktionen im Kontext unter [Szenarien für S3-Verzeichnis-Buckets mit AWS SDKs](#) anzeigen.

Die folgenden Beispiele enthalten nur die am häufigsten verwendeten Aktionen. Eine vollständige Liste finden Sie in der [API-Referenz für Amazon-S3-Verzeichnis-Buckets](#).

Beispiele

- [Verwenden Sie AbortMultipartUploadmit einem AWS SDK](#)
- [Verwenden Sie CompleteMultipartUploadmit einem AWS SDK](#)
- [Verwenden Sie CopyObjectmit einem AWS SDK](#)
- [Verwenden Sie CreateBucketmit einem AWS SDK](#)
- [Verwenden Sie CreateMultipartUploadmit einem AWS SDK](#)
- [Verwenden Sie CreateSessionmit einem AWS SDK](#)
- [Verwenden Sie DeleteBucketmit einem AWS SDK](#)
- [Verwenden Sie DeleteBucketEncryptionmit einem AWS SDK](#)
- [Verwenden Sie DeleteBucketPolycymit einem AWS SDK](#)
- [Verwenden Sie DeleteObjectmit einem AWS SDK](#)
- [Verwenden Sie DeleteObjectsmitem einem AWS SDK](#)
- [Verwenden Sie GetBucketEncryptionmit einem AWS SDK](#)
- [Verwenden Sie GetBucketPolycymit einem AWS SDK](#)
- [Verwenden Sie GetObjectmit einem AWS SDK](#)
- [Verwenden Sie GetObjectAttributemit einem AWS SDK](#)
- [Verwenden Sie HeadBucketmit einem AWS SDK](#)

- [Verwenden Sie HeadObject mit einem AWS SDK](#)
- [Verwenden Sie ListDirectoryBuckets mit einem AWS SDK](#)
- [Verwenden Sie ListMultipartUploads mit einem AWS SDK](#)
- [Verwenden Sie ListObjectsV2 mit einem AWS SDK](#)
- [Verwenden Sie ListParts mit einem AWS SDK](#)
- [Verwenden Sie PutBucketEncryption mit einem AWS SDK](#)
- [Verwenden Sie PutBucketPolicy mit einem AWS SDK](#)
- [Verwenden Sie PutObject mit einem AWS SDK](#)
- [Verwenden Sie UploadPart mit einem AWS SDK](#)
- [Verwenden Sie UploadPartCopy mit einem AWS SDK](#)

Verwenden Sie **AbortMultipartUpload** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `AbortMultipartUpload`.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Brechen Sie einen mehrteiligen Upload in einen Verzeichnis-Bucket ab.

```
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;
    }
}

```

- Einzelheiten zur API finden Sie [AbortMultipartUpload](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CompleteMultipartUpload** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `CompleteMultipartUpload`.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Schließen Sie einen mehrteiligen Upload in einen Verzeichnis-Bucket ab.

```
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;
    }
}

```

- Einzelheiten zur API finden Sie [CompleteMultipartUpload](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CopyObject** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `CopyObject`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Kopieren Sie ein Objekt aus einem Verzeichnis-Bucket in einen Verzeichnis-Bucket.

```
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;
    }
}
```

- Einzelheiten zur API finden Sie [CopyObject](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreateBucket** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `CreateBucket`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie einen S3-Verzeichnis-Bucket.

```
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;
    }
}
```

```
}  
}
```

- Einzelheiten zur API finden Sie [CreateBucket](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreateMultipartUpload** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird **CreateMultipartUpload**.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie einen mehrteiligen Upload in einen Verzeichnis-Bucket.

```
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;
        }
    }
}

```

- Einzelheiten zur API finden Sie [CreateMultipartUpload](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **CreateSession** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird **CreateSession**.

Python

SDK für Python (Boto3)

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

```
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

```

- Einzelheiten zur API finden Sie [CreateSession](#) in AWS SDK for Python (Boto3) API Reference.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteBucket** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `DeleteBucket`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie einen S3-Verzeichnis-Bucket.

```
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;
        }
    }
}
```

- Einzelheiten zur API finden Sie [DeleteBucket](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteBucketEncryption** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird **DeleteBucketEncryption**.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie die Verschlüsselungskonfiguration für einen Verzeichnis-Bucket.

```
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;
    }
}
```

- Einzelheiten zur API finden Sie [DeleteBucketEncryption](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteBucketPolicy** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird **DeleteBucketPolicy**.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie eine Bucket-Richtlinie für einen Verzeichnis-Bucket.

```
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;
    }
}
```

- Einzelheiten zur API finden Sie [DeleteBucketPolicy](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteObject** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `DeleteObject`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen eines Objekts in einem Verzeichnis-Bucket

```
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;
    }
}
```

- Einzelheiten zur API finden Sie [DeleteObject](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **DeleteObjects** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `DeleteObjects`.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie mehrere Objekte in einem Verzeichnis-Bucket

```
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;
    }
}
```

- Einzelheiten zur API finden Sie [DeleteObjects](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetBucketEncryption** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `GetBucketEncryption`.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Löschen Sie die Verschlüsselungskonfiguration eines Verzeichnis-Buckets.

```
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;
    }
}
```

```
}  
}
```

- Einzelheiten zur API finden Sie [GetBucketEncryption](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetBucketPolicy** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `GetBucketPolicy`.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie die Richtlinie eines Verzeichnis-Buckets ab.

```
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;
    }
}
```

- Einzelheiten zur API finden Sie [GetBucketPolicy](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetObject** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `GetObject`.

Aktionsbeispiele sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. In den folgenden Codebeispielen können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)
- [Erstellen einer vorsignierten URL, um ein Objekt abzurufen](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie ein Objekt aus einem Bucket ab.

```
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;
    }
}

```

- Einzelheiten zur API finden Sie [GetObject](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **GetObjectAttributes** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `GetObjectAttributes`.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie die Objektattribute aus einem eines Verzeichnis-Bucket ab.

```
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;
    }
}
```

- Einzelheiten zur API finden Sie [GetObjectAttributes](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **HeadBucket** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird **HeadBucket**.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Prüfen Sie, ob das angegebene S3-Verzeichnis-Bucket existiert und zugänglich ist.

```
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;
        }
    }
}
```

- Einzelheiten zur API finden Sie [HeadBucket](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **HeadObject** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `HeadObject`.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Rufen Sie Metadaten eines Objekts in einem Verzeichnis-Bucket ab.

```
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;
        }
    }
}

```

- Einzelheiten zur API finden Sie [HeadObject](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ListDirectoryBuckets** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `ListDirectoryBuckets`.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Listen Sie alle Verzeichnis-Buckets auf.

```
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;
}
}
```

- Einzelheiten zur API finden Sie [ListDirectoryBuckets](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ListMultipartUploads** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `ListMultipartUploads`.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Listen Sie mehrteilige Uploads in einen Verzeichnis-Bucket auf.

```
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
    }
}

```

- Einzelheiten zur API finden Sie [ListMultipartUploads](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ListObjectsV2** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird **ListObjectsV2**.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Listen Sie Objekte in einem Verzeichnis-Bucket auf.

```
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;
    }
}
```

- Einzelheiten zur API finden Sie unter [ListObjectsV2](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **ListParts** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird **ListParts**.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Listen Sie Teile eines mehrteiligen Uploads in einen Verzeichnis-Bucket auf.

```
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
        }
    }
}

```

- Einzelheiten zur API finden Sie [ListParts](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketEncryption** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird **PutBucketEncryption**.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Legen Sie die Bucket-Verschlüsselung auf einen Verzeichnis-Bucket fest.

```
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;
    }
}
```

- Einzelheiten zur API finden Sie [PutBucketEncryption](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutBucketPolicy** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird **PutBucketPolicy**.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Wenden Sie eine Bucket-Richtlinie auf einen Verzeichnis-Bucket an.

```
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;
    }
}

```

```
}  
}
```

- Einzelheiten zur API finden Sie [PutBucketPolicy](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **PutObject** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `PutObject`.

Beispiele für Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Im folgenden Codebeispiel können Sie diese Aktion im Kontext sehen:

- [Kennenlernen der Grundlagen](#)

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Legen Sie ein Objekts in einen Verzeichnis-Bucket.

```
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;
    }
}
```

```
}  
}
```

- Einzelheiten zur API finden Sie [PutObject](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **UploadPart** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `UploadPart`.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Laden Sie einen Teil eines mehrteiligen Uploads in einen Verzeichnis-Bucket hoch.

```
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;
}
```

- Einzelheiten zur API finden Sie [UploadPart](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Verwenden Sie **UploadPartCopy** mit einem AWS SDK

Das folgende Codebeispiel zeigt, wie es verwendet wird `UploadPartCopy`.

Java

SDK für Java 2.x

 Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Erstellen Sie Kopierteile basierend auf der Größe des Quellobjekts und kopieren Sie einzelne Teile in einen Verzeichnis-Bucket.

```
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;
    }
}
```

- Einzelheiten zur API finden Sie [UploadPartCopy](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Szenarien für S3-Verzeichnis-Buckets mit AWS SDKs

Die folgenden Codebeispiele zeigen Ihnen, wie Sie allgemeine Szenarien in S3 Directory Buckets mit SDKs implementieren. AWS Diese Szenarien zeigen Ihnen, wie Sie bestimmte Aufgaben durch den Aufruf mehrerer Funktionen innerhalb von S3-Verzeichnis-Buckets oder in Kombination mit anderen AWS-Services ausführen können. Jedes Szenario enthält einen Link zum vollständigen Quell-Code, wo Sie Anleitungen zum Einrichten und Ausführen des Codes finden.

Szenarien zielen auf eine mittlere Erfahrungsebene ab, um Ihnen zu helfen, Service-Aktionen im Kontext zu verstehen.

Beispiele

- [Erstellen Sie eine vorsignierte URL für Amazon S3 S3-Verzeichnis-Buckets, um ein Objekt mithilfe eines AWS SDK](#)

Erstellen Sie eine vorsignierte URL für Amazon S3 S3-Verzeichnis-Buckets, um ein Objekt mithilfe eines AWS SDK

Die folgenden Codebeispiele zeigen, wie Sie eine vorsignierte URL für S3-Verzeichnis-Buckets erstellen und ein Objekt hochladen.

Java

SDK für Java 2.x

Note

Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das [AWS -Code-Beispiel-](#) einrichten und ausführen.

Generieren Sie eine vorsignierte GET-URL für den Zugriff auf ein Objekt in einem S3-Verzeichnis-Bucket.

```
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;
        }
    }
}
```

- Einzelheiten zur API finden Sie [GetObject](#) in der AWS SDK for Java 2.x API-Referenz.

Eine vollständige Liste der AWS SDK-Entwicklerhandbücher und Codebeispiele finden Sie unter [Entwickeln mit Amazon S3 unter Verwendung der AWS SDKs](#). Dieses Thema enthält auch Informationen zu den ersten Schritten und Details zu früheren SDK-Versionen.

Anfragen authentifizieren (AWS Signatur (Version 4))

Themen

- [Authentifizierungsmethoden](#)
- [Einführung in das Signieren von Anfragen](#)
- [Authentifizieren von Anfragen: Verwenden des Autorisierungs-Headers \(AWS Signatur \(Version 4\)\)](#)
- [Anfragen authentifizieren: Verwenden von Abfrageparametern \(AWS Signatur \(Version 4\)\)](#)
- [Beispiele: Signaturberechnungen in AWS Signaturversion 4](#)
- [Anfragen authentifizieren: Browser-Based Uploads mit POST \(AWS Signatur \(Version 4\)\)](#)
- [Amazon S3 Signature Version 4: Spezifische Richtlinien](#)

Jede Interaktion mit Amazon S3 erfolgt entweder authentifiziert oder anonym. In diesem Abschnitt wird die Anforderungsauthentifizierung mit dem AWS Signature Version 4-Algorithmus erklärt.

Note

Wenn Sie die AWS SDKs (siehe [Beispielcode und Bibliotheken](#)) zum Senden Ihrer Anfragen verwenden, müssen Sie diesen Abschnitt nicht lesen, da die SDK-Clients Ihre Anfragen mithilfe der von Ihnen bereitgestellten Zugriffsschlüssel authentifizieren. Sofern Sie keinen guten Grund haben, dies nicht zu tun, sollten Sie immer die AWS SDKs verwenden. In Regionen, die beide Signaturversionen unterstützen, können Sie AWS SDKs auffordern, eine bestimmte Signaturversion zu verwenden. Weitere Informationen finden Sie unter [Senden authentifizierter Anfragen mit dem AWS SDKs](#). Sie müssen diesen Abschnitt nur lesen, wenn Sie den AWS Signature Version 4-Algorithmus in Ihrem benutzerdefinierten Client implementieren.

Die Authentifizierung mit AWS Signature Version 4 bietet einige oder alle der folgenden Funktionen, je nachdem, wie Sie Ihre Anfrage signieren:

- **Überprüfung der Identität des Anforderers** — Für authentifizierte Anfragen ist eine Signatur erforderlich, die Sie mithilfe Ihrer Zugangsschlüssel (Zugriffsschlüssel-ID, geheimer Zugriffsschlüssel) erstellen. Informationen zum Abrufen von Zugriffsschlüsseln finden Sie

unter [Grundlegendes und Abrufen Ihrer Sicherheitsanmeldeinformationen](#) im. Allgemeine AWS-Referenz Wenn Sie temporäre Sicherheitsanmeldeinformationen verwenden, ist für die Signaturberechnungen auch ein Sicherheitstoken erforderlich. Weitere Informationen finden Sie unter [Anfordern von temporären Sicherheitsanmeldeinformationen](#) im -IAM-Benutzerhandbuch.

- In-transit Datenschutz — Um zu verhindern, dass eine Anfrage während der Übertragung manipuliert wird, verwenden Sie einige der Anforderungselemente, um die Anforderungssignatur zu berechnen. Nach Erhalt der Anfrage berechnet Amazon S3 die Signatur mithilfe derselben Anforderungselemente. Wenn eine von Amazon S3 empfangene Anforderungskomponente nicht mit der Komponente übereinstimmt, die zur Berechnung der Signatur verwendet wurde, lehnt Amazon S3 die Anfrage ab.
- Vor Wiederverwendung der signierten Teile der Anfrage schützen — Die signierten Teile (mithilfe von AWS Signaturen) von Anfragen sind innerhalb von 15 Minuten ab dem Zeitstempel in der Anfrage gültig. Eine nicht autorisierte Partei, die Zugriff auf eine signierte Anfrage hat, kann die unsignierten Teile der Anfrage ändern, ohne dass die Gültigkeit der Anfrage innerhalb des 15-Minuten-Fensters beeinträchtigt wird. Aus diesem Grund empfehlen wir Ihnen, den Schutz zu maximieren, indem Sie Header und Text der Anfrage signieren, HTTPS-Anfragen an Amazon S3 stellen und den `s3:x-amz-content-sha256` Bedingungsschlüssel (siehe [Amazon S3 Signature Version 4: Spezifische Richtlinien](#)) in den AWS Richtlinien verwenden, um Benutzer dazu zu verpflichten, Amazon S3-Anfragetexte zu signieren.

Note

Amazon S3 unterstützt insgesamt Signature Version 4, ein Protokoll zur Authentifizierung eingehender API-Anfragen an AWS Dienste. AWS-Regionen Derzeit wird das vor dem 30. Januar 2014 AWS-Regionen erstellte Protokoll, Signature Version 2, weiterhin unterstützt. Alle neuen Regionen nach dem 30. Januar 2014 unterstützen nur Signature Version 4, weshalb alle Anfragen an diese Regionen mit Signature Version 4 gestellt werden müssen. Weitere Informationen zu AWS Signature Version 2 finden Sie unter [Signieren und Authentifizieren von REST-Anfragen](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Authentifizierungsmethoden

Sie können Authentifizierungsinformationen mithilfe einer der folgenden Methoden ausdrücken:

- **HTTP-Autorisierungsheader** — Die Verwendung des Authorization HTTP-Headers ist die gängigste Methode zur Authentifizierung einer Amazon S3-Anfrage. Alle Amazon S3-REST-Operationen (mit Ausnahme von browserbasierten Uploads mit POST-Anfragen) benötigen diesen Header. Weitere Informationen zum Authorization Header-Wert und zur Berechnung der Signatur und der zugehörigen Optionen finden Sie unter [Authentifizieren von Anfragen: Verwenden des Autorisierungs-Headers \(AWS Signatur \(Version 4\)\)](#)
- **Parameter für Abfragezeichenfolgen** — Sie können eine Abfragezeichenfolge verwenden, um eine Anfrage vollständig in einer URL auszudrücken. In diesem Fall verwenden Sie Abfrageparameter, um Anforderungsinformationen bereitzustellen, einschließlich der Authentifizierungsinformationen. Da die Anforderungssignatur Teil der URL ist, wird diese Art von URL oft als vorsignierte URL bezeichnet. Sie können vorsignierte URLs verwenden, um anklickbare Links, die bis zu sieben Tage gültig sein können, in HTML einzubetten. Weitere Informationen finden Sie unter [Anfragen authentifizieren: Verwenden von Abfrageparametern \(AWS Signatur \(Version 4\)\)](#).

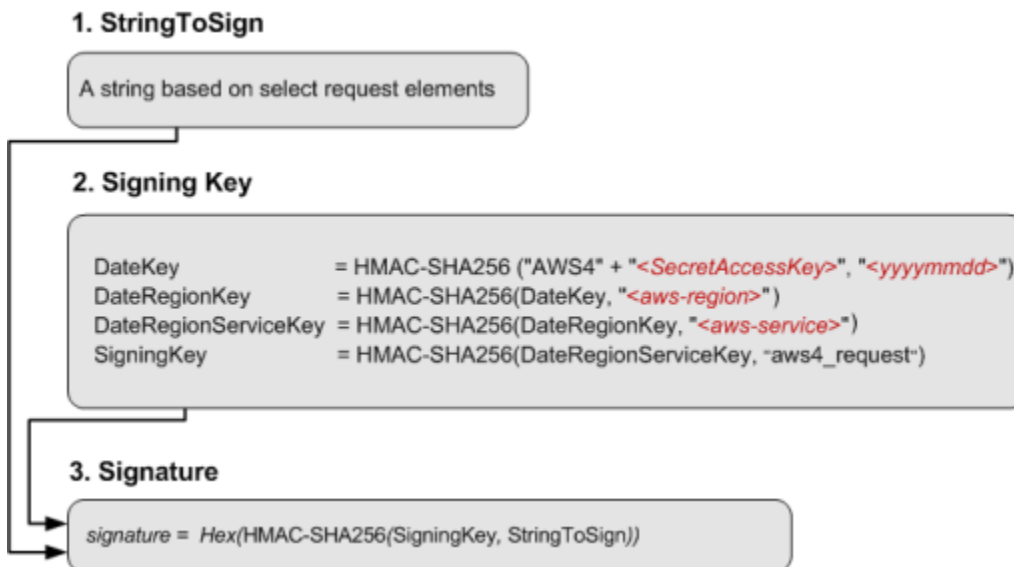
Amazon S3 unterstützt auch browserbasierte Uploads, die HTTP-POST-Anfragen verwenden. Mit einer HTTP-POST-Anfrage können Sie Inhalte direkt aus dem Browser auf Amazon S3 hochladen. Informationen zur Authentifizierung von POST-Anfragen finden Sie unter [Browser-Based Uploads mit POST \(AWS Signatur \(Version 4\)\)](#).

Einführung in das Signieren von Anfragen

Die Authentifizierungsinformationen, die Sie in einer Anfrage senden, müssen eine Signatur enthalten. Um eine Signatur zu berechnen, verketteten Sie zunächst ausgewählte Anforderungselemente zu einer Zeichenfolge, die als zu signierende Zeichenfolge bezeichnet wird. Anschließend verwenden Sie einen Signaturschlüssel, um den Hash-basierten Nachrichtenauthentifizierungscode (HMAC) der zu signierenden Zeichenfolge zu berechnen.

In AWS Signature Version 4 verwenden Sie Ihren geheimen Zugangsschlüssel nicht, um die Anfrage zu signieren. Verwenden Sie stattdessen zunächst Ihren geheimen Zugriffsschlüssel, um einen Signierschlüssel abzuleiten. Der abgeleitete Signaturschlüssel ist spezifisch für Datum, Service und Region. Weitere Informationen zum Ableiten eines Signaturschlüssels in verschiedenen Programmiersprachen finden Sie unter [Beispiele für das Ableiten eines Signaturschlüssels für Signature Version 4](#).

Das folgende Diagramm veranschaulicht den allgemeinen Prozess der Berechnung einer Signatur.



Die zu signierende Zeichenfolge hängt von der Art der Anfrage ab. Wenn Sie beispielsweise den HTTP-Autorisierungsheader oder die Abfrageparameter für die Authentifizierung verwenden, verwenden Sie eine variierende Kombination von Anforderungselementen, um die zu signierende Zeichenfolge zu erstellen. Bei einer HTTP-POST-Anforderung ist die POST-Richtlinie in der Anfrage die Zeichenfolge, die Sie signieren. Weitere Informationen zur Berechnung der zu signierenden Zeichenfolge finden Sie unter den Links am Ende dieses Abschnitts.

Für den Signaturschlüssel zeigt das Diagramm eine Reihe von Berechnungen, wobei das Ergebnis jedes Schritts in den nächsten Schritt einfließt. Der letzte Schritt ist der Signaturschlüssel.

Nach Erhalt einer authentifizierten Anfrage erstellen die Amazon S3-Server die Signatur neu, indem sie die in der Anfrage enthaltenen Authentifizierungsinformationen verwenden. Wenn die Signaturen übereinstimmen, verarbeitet Amazon S3 Ihre Anfrage; andernfalls wird die Anfrage abgelehnt.

Weitere Informationen zur Authentifizierung von Anfragen finden Sie in den folgenden Themen:

- [Authentifizieren von Anfragen: Verwenden des Autorisierungs-Headers \(AWS Signatur \(Version 4\)\)](#)
- [Anfragen authentifizieren: Verwenden von Abfrageparametern \(AWS Signatur \(Version 4\)\)](#)
- [Browser-Based Uploads mit POST \(AWS Signatur \(Version 4\)\)](#)

Authentifizieren von Anfragen: Verwenden des Autorisierungs-Headers (AWS Signatur (Version 4))

Themen

- [-Übersicht](#)
- [Signaturberechnungen für den Autorisierungs-Header: Übertragung der Nutzlast in einem einzigen Block \(AWS Signatur \(Version 4\)\)](#)
- [Signaturberechnungen für den Autorisierungs-Header: Übertragung von Nutzdaten in mehreren Blöcken \(Chunked Upload\) \(AWS Signatur \(Version 4\)\)](#)
- [Signaturberechnungen für nachfolgende Header \(aufgeteilte Uploads\) \(AWS Signatur \(Version 4\)\)](#)

-Übersicht

Die Verwendung des `Authorization` HTTP-Headers ist die gebräuchlichste Methode zur Bereitstellung von Authentifizierungsinformationen. Mit Ausnahme von [POST-Anfragen](#) und Anfragen, die mithilfe von Abfrageparametern signiert werden, verwenden alle Amazon S3-Operationen den `Authorization` Anforderungsheader, um Authentifizierungsinformationen bereitzustellen.

Nachfolgend finden Sie ein Beispiel für den `Authorization`-Header-Wert. Zur besseren Lesbarkeit werden diesem Beispiel Zeilenumbrüche hinzugefügt:

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

In der folgenden Tabelle werden die verschiedenen Komponenten des `Authorization` Header-Werts im vorherigen Beispiel beschrieben:

Komponente	Description
AWS4-HMAC-SHA256	Der Algorithmus, der zur Berechnung der Signatur verwendet wurde. Sie müssen diesen Wert angeben, wenn Sie AWS Signature Version 4 für die Authentifizierung verwenden. Die Zeichenfolge gibt AWS Signature Version 4 (AWS4) und den Signaturalgorithmus (HMAC-SHA256) an.

Komponente	Description
Credential	Ihre Zugriffsschlüssel-ID und die Informationen des Gültigkeitsbereichs, darunter Datum, Region und Service, die zur Berechnung der Signatur verwendet wurden. Diese Zeichenfolge verfügt über das folgende Format: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Wobei Folgendes gilt: • <date> Der Wert wird mithilfe des YYYYMMDD Formats angegeben. • <aws-service> Der Wert wird s3 angegeben, wenn eine Anfrage an Amazon S3 gesendet wird.
SignedHeaders	Eine durch Semikolons getrennte Liste von Anforderungs-Headern, die Sie zur Berechnung von Signature verwendet haben. Die Liste enthält nur Header-Namen und die Header-Namen müssen in Kleinbuchstaben geschrieben sein. Beispiel: <pre>host;range;x-amz-date</pre>

Komponente	Description
Signature	Die 256-Bit-Signatur, ausgedrückt als 64 hexadezimale Kleinbuchstaben. Beispiel: <pre>fe5f80f77d5fa3beca038a248ff027d0445342fe2855d dc963176630326f1024</pre> Beachten Sie, dass die Signaturberechnungen je nach gewählter Option zur Übertragung der Nutzlast variieren.

Die Signaturberechnungen variieren je nach der Methode, die Sie für die Übertragung der Anforderungsnutzlast wählen. S3 unterstützt die folgenden Optionen:

- Nutzdaten in einem einzigen Block übertragen — In diesem Fall haben Sie die folgenden Optionen zur Berechnung der Signatur:
 - Option für signierte Nutzdaten — Sie können optional die gesamte Payload-Prüfsumme berechnen und sie in die Signaturberechnung einbeziehen. Dies bietet zusätzliche Sicherheit, aber Sie müssen Ihre Nutzdaten zweimal lesen oder im Speicher zwischenspeichern.

Um beispielsweise eine Datei hochzuladen, müssen Sie die Datei zuerst lesen, um einen Nutzlast-Hash für die Signaturberechnung und erneut für die Übertragung zu berechnen, wenn Sie die Anfrage erstellen. Für kleinere Nutzlasten könnte dieser Ansatz vorzuziehen sein. Bei großen Dateien kann es jedoch ineffizient sein, die Datei zweimal zu lesen. Daher sollten Sie die Daten möglicherweise stattdessen in Blöcken hochladen.

Wir empfehlen, aus Sicherheitsgründen die Payload-Prüfsumme anzugeben.

- Option „Unsigned Payload“ — Beziehen Sie die Payload-Prüfsumme nicht in die Signaturberechnung ein.

Eine schrittweise Anleitung zur Berechnung der Signatur und zur Berechnung des Authorization-Header-Werts finden Sie unter. [Signaturberechnungen für den Autorisierungs-Header: Übertragung der Nutzlast in einem einzigen Block \(AWS Signatur \(Version 4\)\)](#)

- Nutzlast in mehreren Blöcken übertragen (blockierter Upload) — In diesem Fall übertragen Sie die Nutzlast in Blöcken. Sie können eine Nutzlast unabhängig von der Nutzlastgröße in Blöcken übertragen.

Sie können Ihre Nutzlast in Blöcke aufteilen. Dies können Blöcke mit fester oder variabler Größe sein. Indem Sie Daten in Blöcken hochladen, vermeiden Sie, dass zur Berechnung der Signatur die gesamte Nutzlast gelesen wird. Stattdessen berechnen Sie für den ersten Block eine Startsignatur, die nur die Anforderungsheader verwendet. Der zweite Chunk enthält die Signatur für den ersten Chunk, und jeder nachfolgende Chunk enthält die Signatur für den Chunk, der ihm vorausgeht. Am Ende des Uploads senden Sie einen letzten Block mit 0 Byte an Daten, der die Signatur des letzten Blocks der Payload enthält. Weitere Informationen finden Sie unter [Signaturberechnungen für den Autorisierungs-Header: Übertragung von Nutzdaten in mehreren Blöcken \(Chunked Upload\) \(AWS Signatur \(Version 4\)\)](#).

Beim Signieren Ihrer Anfragen können Sie entweder AWS Signature Version 4 oder AWS Signature Version 4A verwenden. Der Hauptunterschied zwischen den beiden wird dadurch bestimmt, wie die Signatur berechnet wird. Bei AWS Signature Version 4A enthält die Signatur keine Region-specific Informationen und wird mithilfe des AWS4-ECDSA-P256-SHA256 Algorithmus berechnet.

Zusätzlich zu diesen Optionen haben Sie die Möglichkeit, Ihrer Anfrage einen Trailer beizufügen. Um Ihrer Anfrage einen Trailer beizufügen, müssen Sie diesen in der Kopfzeile angeben, indem Sie `x-amz-content-sha256` den entsprechenden Wert festlegen. Wenn Sie einen abschließenden Header verwenden, müssen Sie die Namen der nachfolgenden Header `x-amz-trailer` in den Header aufnehmen und als Zeichenfolge in einer kommasetrennten Liste angeben. Alle nachfolgenden Header werden nach dem letzten Abschnitt geschrieben. Wenn Sie die Daten in mehreren Blöcken hochladen, müssen Sie einen letzten Block mit 0 Byte an Daten senden, bevor Sie den abschließenden Header senden.

Wenn Sie eine Anfrage senden, müssen Sie Amazon S3 mitteilen, welche der vorherigen Optionen Sie bei Ihrer Signaturberechnung gewählt haben, indem Sie den `x-amz-content-sha256` Header mit einem der folgenden Werte hinzufügen:

Header-Wert	Description
Tatsächlicher Payload-Prüfsummenwert	Dieser Wert ist die tatsächliche Prüfsumme Ihres Objekts und nur möglich, wenn Sie die Daten in einem einzigen Block hochladen.
UNSIGNED-PAYLOAD	Verwenden Sie diesen Wert, wenn Sie das Objekt als einzelnen unsignierten Block hochladen.

Header-Wert	Description
STREAMING-UNSIGNED-PAYLOAD-TRAILER	Verwenden Sie dies, wenn Sie eine unsignierte Nutzlast über mehrere Chunks senden. In diesem Fall haben Sie auch einen abschließenden Header, nachdem der Chunk hochgeladen wurde.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD	Verwenden Sie dies, wenn Sie eine Nutzlast über mehrere Chunks senden und die Chunks mit signiert werden. AWS4-HMAC-SHA256 Dadurch wird eine SigV4-Signatur erzeugt.
STREAMING-AWS4-HMAC-SHA256-PAYLOAD-TRAILER	Verwenden Sie dies, wenn Sie eine Nutzlast über mehrere Chunks senden und die Chunks mit signiert werden. AWS4-HMAC-SHA256 Dadurch wird eine SigV4-Signatur erzeugt. Außerdem ist der Digest für die Chunks als abschließender Header enthalten.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD	Verwenden Sie diesen Wert, wenn Sie eine Nutzlast über mehrere Chunks senden und die Chunks mit signiert werden. AWS4-ECDSA-P256-SHA256 Dies erzeugt eine SigV4A-Signatur.
STREAMING-AWS4-ECDSA-P256-SHA256-PAYLOAD-TRAILER	Verwenden Sie dies, wenn Sie eine Nutzlast über mehrere Chunks senden und die Chunks mit signiert werden. AWS4-ECDSA-P256-SHA256 Dies erzeugt eine SigV4A-Signatur. Außerdem ist der Digest für die Chunks als abschließender Header enthalten.

Nach Erhalt der Anfrage erstellt Amazon S3 die zu signierende Zeichenfolge anhand der Informationen im Header und im Authorization Header neu. date Anschließend wird mit dem Authentifizierungsdienst überprüft, ob die Signaturen übereinstimmen. Das Anforderungsdatum kann entweder mithilfe des HTTP Date - oder des x-amz-date Headers angegeben werden. Wenn beide Header vorhanden sind, hat x-amz-date Vorrang.

Wenn die Signaturen übereinstimmen, verarbeitet Amazon S3 Ihre Anfrage. Andernfalls schlägt Ihre Anfrage fehl.

Weitere Informationen finden Sie unter den folgenden Themen:

[Signaturberechnungen für den Autorisierungs-Header: Übertragung der Nutzlast in einem einzigen Block \(AWS Signatur \(Version 4\)\)](#)

[Signaturberechnungen für den Autorisierungs-Header: Übertragung von Nutzdaten in mehreren Blöcken \(Chunked Upload\) \(AWS Signatur \(Version 4\)\)](#)

[Signaturberechnungen für nachfolgende Header \(aufgeteilte Uploads\) \(AWS Signatur \(Version 4\)\)](#)

Signaturberechnungen für den Autorisierungs-Header: Übertragung der Nutzlast in einem einzigen Block (AWS Signatur (Version 4))

Wenn Sie den `Authorization` Header zur Authentifizierung von Anfragen verwenden, enthält der Header-Wert unter anderem eine Signatur. Die Signaturberechnungen variieren je nachdem, welche Wahl Sie für die Übertragung der Nutzlast () [-Übersicht](#) treffen. In diesem Abschnitt werden Signaturberechnungen erläutert, wenn Sie sich dafür entscheiden, die Nutzlast in einem einzigen Block zu übertragen. Der Beispielabschnitt (siehe [Beispiele: Signaturberechnungen](#)) zeigt Signaturberechnungen und daraus resultierende `Authorization` Header, die Sie als Testsuite zur Überprüfung Ihres Codes verwenden können.

Important

Wenn Sie Nutzdaten in einem einzigen Block übertragen, können Sie optional wählen, den Nutzlast-Hash in die Signaturberechnungen einzubeziehen, der als signierte Nutzlast bezeichnet wird (wenn Sie ihn nicht angeben, gilt die Nutzlast als unsigniert). Das im folgenden Abschnitt beschriebene Signaturverfahren gilt für beide, beachten Sie jedoch die folgenden Unterschiede:

- Option für signierte Nutzdaten — Sie beziehen den Payload-Hash bei der Erstellung der kanonischen Anfrage ein (dieser wird dann Teil von `StringToSign`, wie im Abschnitt zur Berechnung der Signatur erklärt). Sie geben auch denselben Wert wie den `x-amz-content-sha256` Header-Wert an, wenn Sie die Anfrage an S3 senden.
- Payload-Option ohne Vorzeichen — Sie geben die Literalzeichenfolge an, `UNSIGNED-PAYLOAD` wenn Sie eine kanonische Anfrage erstellen, und legen denselben Wert wie den `x-amz-content-sha256` Header-Wert fest, wenn Sie die Anfrage an Amazon S3 senden.

Wenn Sie Ihre Anfrage an Amazon S3 senden, informiert Amazon S3 anhand des `x-amz-content-sha256` Header-Werts, ob die Payload signiert ist oder nicht. Amazon S3 kann dann die Signatur entsprechend zur Überprüfung erstellen.

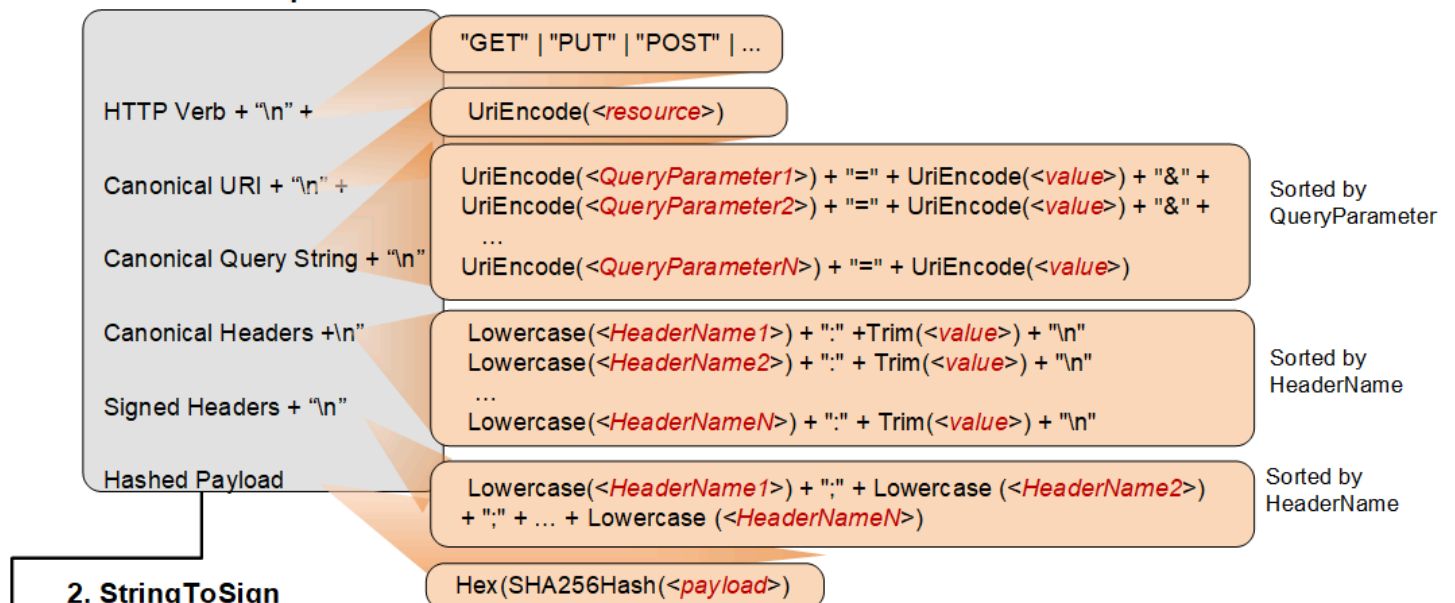
In beiden Fällen müssen Sie den `x-amz-content-sha256` Header nicht als kanonischen Header angeben, da der `x-amz-content-sha256` Header-Wert bereits Teil Ihres ist.

Berechnung einer Signatur

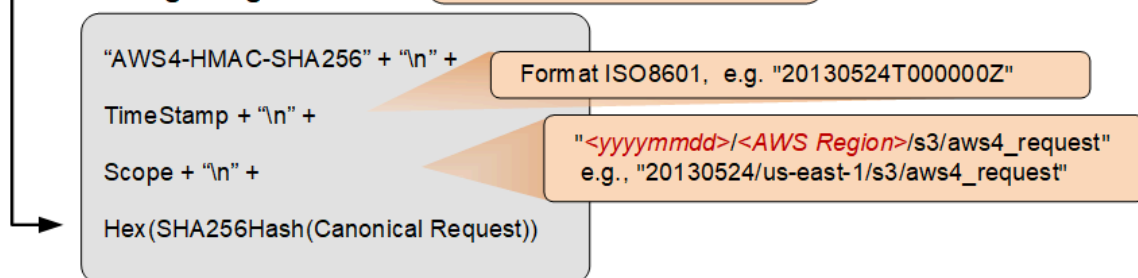
Um eine Signatur zu berechnen, benötigen Sie zunächst eine zu signierende Zeichenfolge. Anschließend berechnen Sie mithilfe eines Signaturschlüssels einen HMAC-SHA256 Hash der zu signierenden Zeichenfolge. Das folgende Diagramm veranschaulicht den Vorgang, einschließlich der verschiedenen Komponenten der Zeichenfolge, die Sie zum Signieren erstellen.

Wenn Amazon S3 eine authentifizierte Anfrage erhält, berechnet es die Signatur und vergleicht sie dann mit der Signatur, die Sie in der Anfrage angegeben haben. Aus diesem Grund müssen Sie die Signatur mit derselben Methode berechnen, die von Amazon S3 verwendet wird. Der Vorgang, bei dem eine Anfrage zur Unterzeichnung in einer vereinbarten Form gestellt wird, wird als Kanonisierung bezeichnet.

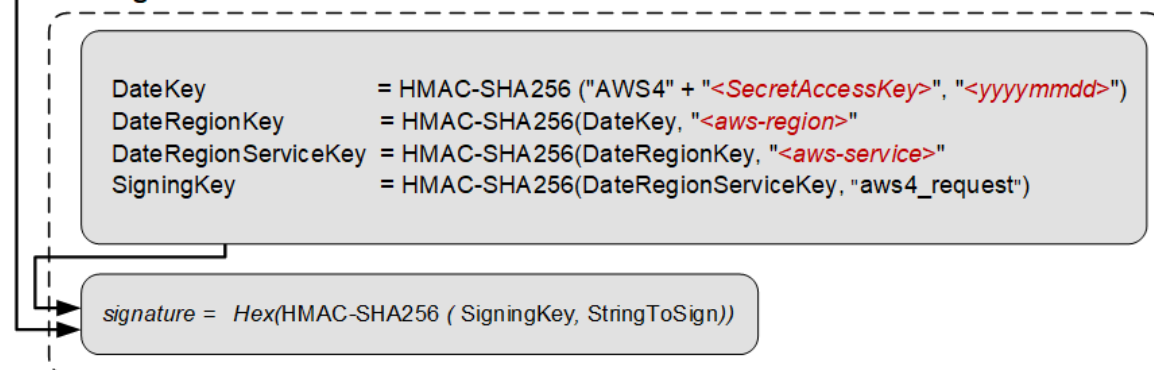
1. Canonical Request



2. StringToSign



3. Signature



In der folgenden Tabelle werden die im Diagramm dargestellten Funktionen beschrieben. Für diese Funktionen muss Code implementiert werden.

Funktion	Description
Lowercase()	Wandeln Sie die Zeichenfolge in Kleinbuchstaben um.

Funktion	Description
Hex()	Kodierung in Kleinbuchstaben im Basis-16-Format.
SHA256Hash()	Kryptografische Hash-Funktion des Secure Hash Algorithm (SHA).
HMAC-SHA256()	Berechnet HMAC unter Verwendung des SHA256-Algorithmus mit dem bereitgestellten Signaturschlüssel. Dies ist die endgültige Signatur.
Trim()	Entfernen Sie alle führenden oder nachgestellten Leerzeichen.

Funktion	Description
<code>UriEncode()</code>	URI-kodiert jedes Byte. <code>UriEncode()</code> muss die folgenden Regeln durchsetzen: • URI kodiert jedes Byte mit Ausnahme der nicht reservierten Zeichen: „A“–„Z“, „a“–„z“, „0“–„9“, „-“, „.“, „_“ und „~“. • Das Leerzeichen ist ein reserviertes Zeichen und muss als „%20“ (und nicht als „+“) codiert werden. • Jedes URI-codierte Byte besteht aus einem „%“ und dem zweistelligen Hexadezimalwert des Bytes. • Buchstaben im Hexadezimalwert müssen in Großbuchstaben geschrieben sein, zum Beispiel „%1A“. • Kodieren Sie den Schrägstrich „/“ überall außer im Objektschlüsselnamen. Wenn der Objektschlüsselname beispielsweise <code>photos/Jan/sample.jpg</code> lautet, wird der Schrägstrich im Schlüsselnamen nicht codiert.  Important Die von Ihrer Entwicklungsplattform bereitgestellten <code>UriEncode</code> Standardfunktionen funktionieren möglicherweise aufgrund von Unterschieden in der Implementierung und den damit verbundenen Mehrdeutigkeiten in den zugrunde liegenden RFCs nicht. Wir empfehlen Ihnen, Ihre eigene benutzerdefinierte <code>UriEncode</code> Funktion zu schreiben, um sicherzustellen, dass Ihre Codierung funktioniert. Das Folgende ist eine Beispielfunktion <code>UriEncode ()</code> in Java. <pre data-bbox="613 1696 1412 1843">public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder();</pre>

Funktion	Description
	<pre> for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Aufgabe 1: Erstellen einer kanonischen Anforderung

Dieser Abschnitt bietet einen Überblick über das Erstellen einer kanonischen Anfrage.

Im Folgenden finden Sie das kanonische Anforderungsformat, das Amazon S3 zur Berechnung einer Signatur verwendet. Damit die Signaturen übereinstimmen, müssen Sie eine kanonische Anfrage in diesem Format erstellen:

```

<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>

```

Wobei Folgendes gilt:

- *HTTPMethod* ist eine der HTTP-Methoden, zum Beispiel GET, PUT, HEAD und DELETE.
- *CanonicalURI* ist die URI-encoded Version der absoluten Pfadkomponente des URI — alles, was mit dem „/“ beginnt und auf den Domainnamen folgt, bis zum Ende der Zeichenfolge oder bis zum Fragezeichen ('?') wenn Sie Abfragezeichenfolgenparameter haben. Beim URI im folgenden

Beispiel, `/examplebucket/myphoto.jpg`, handelt es sich um den absoluten Pfad und Sie codieren das `/` nicht im absoluten Pfad:

```
http://s3.amazonaws.com/examplebucket/myphoto.jpg
```

 Note

Sie normalisieren die URI-Pfade für Anfragen an Amazon S3 nicht. Beispielsweise haben Sie möglicherweise einen Bucket mit einem Objekt namens „my-object//example//photo.user“. Durch die Normalisierung des Pfads wird der Objektname in der Anfrage in „my- /photo.user“ geändert. object/example Dies ist ein falscher Pfad für dieses Objekt.

- **CanonicalQueryString** gibt die Parameter der URI-encoded Abfragezeichenfolge an. Sie geben URI-encode den Namen und die Werte einzeln ein. Sie müssen die Parameter in der kanonischen Abfragezeichenfolge außerdem alphabetisch nach Schlüsselnamen sortieren. Die Sortierung erfolgt nach der Codierung. Die Abfragezeichenfolge im folgenden URI-Beispiel lautet `prefix=somePrefix&marker=someMarker&max-keys=20`:

```
http://s3.amazonaws.com/examplebucket?prefix=somePrefix&marker=someMarker&max-keys=20
```

Die kanonische Abfragezeichenfolge lautet wie folgt (zur besseren Lesbarkeit wurden diesem Beispiel Zeilenumbrüche hinzugefügt):

```
UriEncode("marker")+"="+UriEncode("someMarker")+"&"+  
UriEncode("max-keys")+"="+UriEncode("20") + "&" +  
UriEncode("prefix")+"="+UriEncode("somePrefix")
```

Wenn eine Anfrage auf eine Unterressource abzielt, ist der entsprechende Abfrageparameterwert eine leere Zeichenfolge („“). Der folgende URI identifiziert beispielsweise die ACL-Unterressource im `examplebucket`-Bucket:

```
http://s3.amazonaws.com/examplebucket?acl
```

Das ist CanonicalQueryString in diesem Fall wie folgt:


```
UriEncode("acl") + "=" + ""
```

Wenn der URI kein „?“ enthält, gibt es in der Anfrage keine Abfragezeichenfolge und Sie legen die kanonische Abfragezeichenfolge auf eine leere Zeichenfolge („“) fest. Sie müssen weiterhin das „\n“ einfügen.

- *CanonicalHeaders* ist eine Liste von Anforderungsheadern mit ihren Werten. Einzelne Header-Namens- und Wertpaare werden durch das Zeilenumbruchzeichen („\n“) getrennt. Header-Namen müssen in Kleinbuchstaben geschrieben sein. Sie müssen die Header-Namen alphabetisch sortieren, um die Zeichenfolge zu erstellen, wie im folgenden Beispiel gezeigt:

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"
...
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

Die in diesem Beispiel verwendeten Lowercase()- und Trim()-Funktionen werden im vorangehenden Abschnitt beschrieben.

Die *CanonicalHeaders* Liste muss Folgendes enthalten:

- HTTP-host-Header.
- Wenn der Content-MD5 Header in der Anfrage vorhanden ist, müssen Sie ihn der *CanonicalHeaders* Liste hinzufügen.
- Alle x-amz-* -Header, die Sie in Ihre Anfrage aufnehmen möchten, müssen ebenfalls hinzugefügt werden. Wenn Sie beispielsweise temporäre Sicherheitsanmeldeinformationen verwenden, müssen Sie x-amz-security-token in Ihre Anfrage einschließen. Sie müssen diesen Header zur Liste von hinzufügen *CanonicalHeaders*.

Note

- Der x-amz-content-sha256 Header ist für alle AWS Signature Version 4-Anfragen erforderlich. Es stellt einen Hash der Anforderungsnutzlast bereit. Sie müssen den jedoch nicht x-amz-content-sha256 als kanonischen Header angeben, da S3 seinen Wert automatisch verwendet, um den in der Anfrage gesendeten Nutzlast-Hash zu berechnen.

- Wenn keine Nutzdaten vorhanden sind, müssen Sie den Hash einer leeren Zeichenfolge bereitstellen.
- Wenn Sie nicht möchten, dass S3 den Hash der Anfrage abgleicht, können Sie stattdessen die Literalzeichenfolge verwenden. "UNSIGNED-PAYLOAD"

Im Folgenden finden Sie CanonicalHeaders Beispielzeichenfolgen. Die Header-Namen sind in Kleinbuchstaben geschrieben und sortiert.

Example Beispiel 1

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

Example Beispiel 2

```
host:s3.amazonaws.com
x-amz-content-sha256:UNSIGNED-PAYLOAD
x-amz-date:20130708T220855Z
```

Example Beispiel 3

```
host:s3.amazonaws.com
x-amz-date:20130708T220855Z
```

Note

Für die Berechnung einer Autorisierungssignatur sind der Host-Header und alle x-amz-* Header, ausgenommen x-amz-content-sha256, erforderlich. Um Datenmanipulationen zu verhindern, sollten Sie jedoch in Betracht ziehen, alle Header in die Signaturberechnung einzubeziehen.

Das Signieren des x-amz-content-sha256 Headers ist optional, da S3 seinen Wert bei der Berechnung des Nutzlast-Hashs der empfangenen Anfrage verwendet.

- **SignedHeaders** ist eine alphabetisch sortierte, durch Semikolons getrennte Liste von Anforderungsheadernamen in Kleinbuchstaben. Bei den Anforderungs-

Headern in der Liste handelt es sich um dieselben Header, die Sie in die Zeichenfolge `CanonicalHeaders` eingeschlossen haben. In den vorherigen Beispielen wäre der Wert von wie folgt: *SignedHeaders*

Example Beispiele 1 und 2

```
host;x-amz-content-sha256;x-amz-date
```

Example Beispiel 3

```
host;x-amz-date
```

- *HashedPayload* ist der Hexadezimalwert des SHA256-Hashs der Anforderungsnutzlast.

```
Hex(SHA256Hash(<payload>))
```

Wenn die Anfrage keine Nutzlast enthält, berechnen Sie einen Hash der leeren Zeichenfolge wie folgt:

```
Hex(SHA256Hash(""))
```

Der Hash gibt den folgenden Wert zurück:

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Wenn Sie beispielsweise ein Objekt mithilfe einer PUT-Anfrage hochladen, stellen Sie Objektdaten im Textkörper bereit. Wenn Sie ein Objekt mithilfe einer GET-Anforderung abrufen, berechnen Sie den leeren String-Hash.

Aufgabe 2: Erstellen einer zu signierenden Zeichenfolge

Dieser Abschnitt bietet einen Überblick über das Erstellen einer Zeichenfolge zum Signieren. Eine schrittweise Anleitung finden Sie unter [Aufgabe 2: Erstellen Sie eine Zeichenfolge zum Signieren](#) in Allgemeine AWS-Referenz.

Die zu signierende Zeichenfolge ist eine Verkettung der folgenden Zeichenfolgen:

```
"AWS4-HMAC-SHA256" + "\n" +  
timeStampISO8601Format + "\n" +  
<Scope> + "\n" +  
Hex(SHA256Hash(<CanonicalRequest>))
```

Die konstante Zeichenfolge AWS4-HMAC-SHA256 gibt den Hash-Algorithmus an, den Sie verwenden. HMAC-SHA256 Das timeStamp ist die aktuelle UTC-Zeit im ISO 8601-Format (z. B. 20130524T000000Z).

Scope bindet die resultierende Signatur an ein bestimmtes Datum, AWS-Region, an und einen Dienst. Daher funktioniert Ihre resultierende Signatur nur in der bestimmten Region und für einen bestimmten Dienst. Die Signatur ist nach dem angegebenen Datum sieben Tage lang gültig.

```
date.Format(<YYYYMMDD>) + "/" + <region> + "/" + <service> + "/aws4_request"
```

Für Amazon S3 lautet die Servicezeichenfolge s3. Eine Liste der *region* Zeichenfolgen finden Sie unter [Regionen und Endpunkte](#) im Allgemeinen AWS-Referenz. Die Spalte Region in dieser Tabelle enthält die Liste der gültigen Region-Zeichenfolgen.

Der folgende Bereich beschränkt die resultierende Signatur auf die us-east-1 Region und Amazon S3.

```
20130606/us-east-1/s3/aws4_request
```

Note

Scopemuss dasselbe Datum verwenden, das Sie für die Berechnung des Signaturschlüssels verwenden, wie im folgenden Abschnitt beschrieben.

Aufgabe 3: Signatur berechnen

In AWS Signature Version 4 verwenden Sie Ihre AWS Zugangsschlüssel nicht zum Signieren einer Anfrage, sondern erstellen zunächst einen Signaturschlüssel, der auf eine bestimmte Region und einen bestimmten Dienst beschränkt ist. Weitere Informationen zum Signieren von Schlüsseln finden Sie unter [Einführung in das Signieren von Anfragen](#)

```
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

In einigen Anwendungsfällen können Signaturschlüssel bis zu 7 Tage lang verarbeitet werden. Weitere Informationen finden Sie unter Ein Objekt mit anderen [teilen](#).

Eine Liste der Region-Zeichenfolgen finden Sie unter [Regionen und Endpunkte](#) im Allgemeine AWS-Referenz.

Wenn Sie einen Signaturschlüssel verwenden, können Sie Ihre AWS Anmeldeinformationen an einem sicheren Ort aufbewahren. Wenn Sie beispielsweise mehrere Server haben, die mit Amazon S3 kommunizieren, teilen Sie den Signaturschlüssel mit diesen Servern; Sie müssen nicht auf jedem Server eine Kopie Ihres geheimen Zugriffsschlüssels aufbewahren. Der Signaturschlüssel ist bis zu sieben Tage gültig. Jedes Mal, wenn Sie den Signaturschlüssel berechnen, müssen Sie den Signaturschlüssel also mit Ihren Servern teilen. Weitere Informationen finden Sie unter [Anfragen authentifizieren \(AWS Signatur \(Version 4\)\)](#).

Die endgültige Signatur ist der HMAC-SHA256 Hash der zu signierenden Zeichenfolge, wobei der Signaturschlüssel als Schlüssel verwendet wird.

```
HMAC-SHA256(SigningKey, StringToSign)
```

Eine schrittweise Anleitung zum Erstellen einer Signatur finden Sie unter [Aufgabe 3: Eine Signatur erstellen](#) im Allgemeine AWS-Referenz.

Beispiele: Signaturberechnungen

Sie können die Beispiele in diesem Abschnitt als Referenz verwenden, um die Signaturberechnungen in Ihrem Code zu überprüfen. Die in den Beispielen gezeigten Berechnungen verwenden die folgenden Daten:

- Beispiel für Zugriffsschlüssel.

Parameter	Wert
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- Zeitstempel der Anfrage von 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Name examplebucket des Buckets.
- Es wird davon ausgegangen, dass sich der Bucket in der Region USA Ost (Nord-Virginia) befindet. Die Anmeldeinformationen Scope und die Signing Key Berechnungen werden us-east-1 als Regionsspezifizierer verwendet. Informationen zu anderen Regionen finden Sie unter [Regionen und Endpunkte](#) im Allgemeine AWS-Referenz.
- Sie können Anfragen im Pfadstil oder im virtuellen Hosting-Stil verwenden. Die folgenden Beispiele zeigen, wie Sie beispielsweise eine Anfrage im Stil eines virtuellen Hostings signieren:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Weitere Informationen finden Sie unter [Virtuelles Hosting von Buckets](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Beispiel: GET-Objekt

Im folgenden Beispiel werden die ersten 10 Byte eines Objekts (test.txt) abgerufen examplebucket. Weitere Hinweise zur API-Aktion finden Sie unter [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
```

Da diese GET-Anfrage keinen Textinhalt bereitstellt, kann der x-amz-content-sha256 Wert entweder der Hash des leeren Anforderungstextes oder die Literalzeichenfolge "UNSIGNED-

PAYLOAD" sein. Die folgenden Schritte zeigen Signaturberechnungen und die Konstruktion des Authorization Headers unter Verwendung des Hashs einer leeren Zeichenfolge.

1. StringToSign

a. CanonicalRequest

```
GET
/test.txt

host:examplebucket.s3.amazonaws.com
range:bytes=0-9
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;range;x-amz-content-sha256;x-amz-date
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

In der kanonischen Anforderungszeichenfolge ist die letzte Zeile der Hash des leeren Anforderungstextes. Die dritte Zeile ist leer, da die Anfrage keine Abfrageparameter enthält.

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. Signatur

```
f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

4. Authorization-Header

Der resultierende Authorization Header lautet wie folgt:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/
s3/aws4_request,SignedHeaders=host;range;x-amz-content-sha256;x-amz-
date,Signature=f0e8bdb87c964420e857bd35b5d6ed310bd44f0170aba48dd91039c6036bdb41
```

Beispiel: PUT-Objekt

Diese Beispiel-PUT-Anfrage erstellt ein Objekt (`test$file.text`) in `examplebucket`. In diesem Beispiel wird Folgendes angenommen:

- Sie fordern `REDUCED_REDUNDANCY` als Speicherklasse an, indem Sie den `x-amz-storage-class` Anforderungsheader hinzufügen. Informationen zu Speicherklassen finden Sie unter [Speicherklassen](#) im Amazon Simple Storage Service-Benutzerhandbuch.
- Der Inhalt der hochgeladenen Datei ist eine Zeichenfolge. "Welcome to Amazon S3." Der Wert von `x-amz-content-sha256` in der Anfrage basiert auf dieser Zeichenfolge.

Hinweise zur API-Aktion finden Sie unter [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>
```

Die folgenden Schritte zeigen Signaturberechnungen.

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
```

In der kanonischen Anfrage ist die dritte Zeile leer, da die Anfrage keine Abfrageparameter enthält. Der `x-amz-content-sha256` kanonische Header kann optional signiert werden, da sein Nutzlast-Hash bereits am Ende der Anfrage bereitgestellt wird. Die letzte Zeile ist der Hash des Hauptteils, der mit dem `x-amz-content-sha256` header Wert übereinstimmen sollte, der in der HTTP-Anfrage an S3 gesendet wurde.

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. Signatur

```
98ad721746da40c64f1a55b78f14c238d841ea1380cd77a1b5971af0ece108bd
```

4. Authorization-Header

Der resultierende Authorization Header lautet wie folgt:

```
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
```

Beispiel: GET Bucket Lifecycle

Die folgende GET-Anforderung ruft die Lebenszykluskonfiguration von `examplebucket` ab. Hinweise zur API-Aktion finden Sie unter [GetBucketLifecycleConfiguration](#).

```
GET ?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Authorization: SignatureToBeCalculated
x-amz-date: 20130524T000000Z
x-amz-content-sha256: e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

Da die Anfrage keinen Textinhalt bereitstellt, ist der `x-amz-content-sha256` Header-Wert der Hash des leeren Anforderungstextes. Die folgenden Schritte zeigen Signaturberechnungen.

1. StringToSign

a. CanonicalRequest

```
GET
/
lifecycle=
host:examplebucket.s3.amazonaws.com
x-amz-content-
sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130524T000000Z

host;x-amz-content-sha256;x-amz-date
```

```
e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
```

In der kanonischen Anfrage ist die letzte Zeile der Hash des leeren Anforderungstextes.

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. Signatur

```
fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

4. Authorization-Header

Der resultierende Authorization Header lautet wie folgt:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/  
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-  
date,Signature=fea454ca298b7da1c68078a5d1bdbfbbe0d65c699e0f91ac7a200a0136783543
```

Beispiel: Bucket abrufen (Objekte auflisten)

Das folgende Beispiel ruft eine Liste von Objekten aus dem examplebucket Bucket ab. Hinweise zur API-Aktion finden Sie unter [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

Da die Anfrage keinen Hauptteil bereitstellt, x-amz-content-sha256 ist der Wert von der Hash des leeren Anforderungstextes. Die folgenden Schritte zeigen Signaturberechnungen.

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
```

In der kanonischen Zeichenfolge ist die letzte Zeile der Hash des leeren Anforderungstextes.

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. Signatur

```
34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

4. Authorization-Header

Der resultierende Authorization Header lautet wie folgt:

```
AWS4-HMAC-SHA256 Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/  
s3/aws4_request,SignedHeaders=host;x-amz-content-sha256;x-amz-  
date,Signature=34b48302e7b5fa45bde8084f4b7868a86f0a534bc59db6670ed5711ef69dc6f7
```

Signaturberechnungen für den Autorisierungs-Header: Übertragung von Nutzdaten in mehreren Blöcken (Chunked Upload) (AWS Signatur (Version 4))

Wie unter beschrieben [Übersicht](#), haben Sie bei der Authentifizierung von Anfragen mithilfe des `Authorization` Headers die Möglichkeit, die Payload in Blöcken hochzuladen. Sie können Daten in Blöcken mit fester oder variabler Größe senden. In diesem Abschnitt wird der Prozess zur Berechnung der Signatur beim Block-Upload beschrieben, wie Sie den Chunk-Body erstellen und wie das verzögerte Signieren funktioniert, wenn Sie den Chunk zuerst hochladen und seine Signatur im nachfolgenden Chunk senden. Der Beispielabschnitt (siehe [Beispiel: PUT-Objekt](#)) zeigt Signaturberechnungen und die daraus resultierenden `Authorization` Header, die Sie als Testsuite zur Überprüfung Ihres Codes verwenden können.

Note

Wenn Sie Daten in einer Reihe von Blöcken übertragen, müssen Sie einen der folgenden Schritte ausführen:

- Geben Sie die Gesamtlänge des Inhalts (Objektlänge in Byte plus Metadaten in jedem Block) mithilfe des `Content-Length` HTTP-Headers explizit an. Dazu müssen Sie die Gesamtlänge der Nutzlast, einschließlich der Metadaten, die Sie in jedem Chunk senden, vorab berechnen, bevor Sie Ihre Anfrage starten.
- Geben Sie den HTTP-Header `anTransfer-Encoding`. Wenn Sie den `Transfer-Encoding` Header einbeziehen und einen anderen Wert als `identity` angeben, müssen Sie den `Content-Length` Header weglassen.

Bei allen Anfragen müssen Sie den `x-amz-decoded-content-length` Header angeben, der die Größe des Objekts in Byte angibt.

Jede Berechnung der Blocksignatur beinhaltet die Signatur des vorherigen Blocks. Zunächst erstellen Sie eine `Startsignatur`, die nur die Header verwendet. Sie verwenden die `Startsignatur` bei der Signaturberechnung des ersten Blocks. Für jeden nachfolgenden Chunk erstellen Sie eine `Chunk-Signatur`, die die Signatur des vorherigen Chunks enthält. Somit sind die `Chunk-Signaturen` miteinander verkettet; das heißt, die Signatur von Chunk n ist eine Funktion $F(\text{Chunk } n, \text{signature}(\text{chunk } n-1))$. Die Verkettung stellt sicher, dass Sie die Chunks in der richtigen Reihenfolge senden.

Gehen Sie wie folgt vor, um einen Upload in Blöcken durchzuführen:

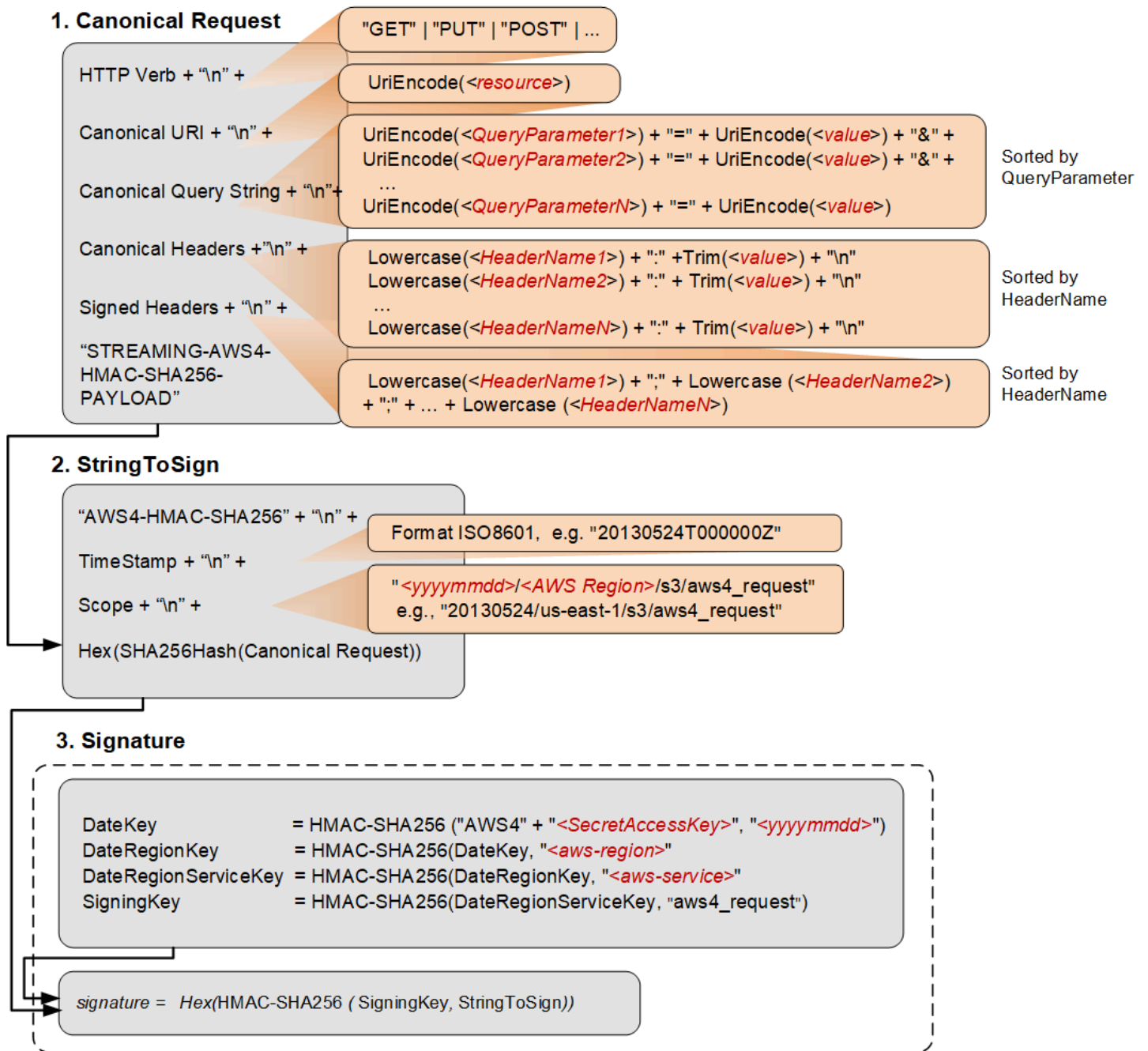
1. Entscheiden Sie sich für die Größe des Payload-Chunks. Sie benötigen dies, wenn Sie den Code schreiben.

Die Chunk-Größe muss mindestens 8 KB betragen. Für eine bessere Leistung empfehlen wir eine Chunk-Größe von mindestens 64 KB. Diese Chunk-Größe gilt für alle Chunks außer dem letzten. Der letzte Chunk, den Sie senden, kann kleiner als 8 KB sein. Wenn Ihre Nutzlast klein ist und in einen Chunk passt, kann sie kleiner als die 8 KB sein.

2. Erstellen Sie die Seed-Signatur für die Aufnahme in den ersten Chunk. Weitere Informationen finden Sie unter [Berechnung der Seed-Signatur](#).
3. Erstellen Sie den ersten Chunk und streamen Sie ihn. Weitere Informationen finden Sie unter [Den Chunk-Körper definieren](#).
4. Berechnen Sie für jeden nachfolgenden Chunk die Chunk-Signatur, die die vorherige Signatur in der von Ihnen signierten Zeichenfolge enthält, erstellen Sie den Chunk und senden Sie ihn. Weitere Informationen finden Sie unter [Den Chunk-Körper definieren](#).
5. Senden Sie den letzten zusätzlichen Block, der mit den anderen Blöcken in der Konstruktion identisch ist, aber keine Datenbytes enthält. Weitere Informationen finden Sie unter [Den Chunk-Körper definieren](#).

Berechnung der Seed-Signatur

Das folgende Diagramm veranschaulicht den Prozess der Berechnung der Startsignatur.



In der folgenden Tabelle werden die im Diagramm dargestellten Funktionen beschrieben. Für diese Funktionen muss Code implementiert werden.

Funktion	Description
Lowercase()	Wandeln Sie die Zeichenfolge in Kleinbuchstaben um.
Hex()	Kodierung in Kleinbuchstaben im Basis-16-Format.

Funktion	Description
SHA256Hash()	Kryptografische Hash-Funktion des Secure Hash Algorithm (SHA).
HMAC-SHA256()	Berechnet HMAC unter Verwendung des SHA256-Algorithmus mit dem bereitgestellten Signaturschlüssel. Dies ist die endgültige Signatur.
Trim()	Entfernen Sie alle führenden oder nachgestellten Leerzeichen.

Funktion	Description
<code>UriEncode()</code>	URI kodieren jedes Byte. <code>UriEncode()</code> muss die folgenden Regeln durchsetzen: • URI kodiert jedes Byte mit Ausnahme der nicht reservierten Zeichen: „A“–„Z“, „a“–„z“, „0“–„9“, „-“, „.“, „_“ und „~“. • Das Leerzeichen ist ein reserviertes Zeichen und muss als „%20“ (und nicht als „+“) codiert werden. • Jedes URI-codierte Byte besteht aus einem „%“ und dem zweistelligen Hexadezimalwert des Bytes. • Buchstaben im Hexadezimalwert müssen in Großbuchstaben geschrieben sein, zum Beispiel „%1A“. • Kodieren Sie den Schrägstrich „/“ überall außer im Objektschlüsselnamen. Wenn der Objektschlüsselname beispielsweise <code>photos/Jan/sample.jpg</code> lautet, wird der Schrägstrich im Schlüsselnamen nicht codiert.  Important Die von Ihrer Entwicklungsplattform bereitgestellten <code>UriEncode</code> Standardfunktionen funktionieren möglicherweise aufgrund von Unterschieden in der Implementierung und den damit verbundenen Mehrdeutigkeiten in den zugrunde liegenden RFCs nicht. Wir empfehlen Ihnen, Ihre eigene benutzerdefinierte <code>UriEncode</code> Funktion zu schreiben, um sicherzustellen, dass Ihre Codierung funktioniert. Das Folgende ist eine Beispielfunktion <code>UriEncode ()</code> in Java. <pre data-bbox="618 1696 1409 1845">public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBu ilder();</pre>

Funktion	Description
	<pre> for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Hinweise zum Signieren finden Sie unter [Signaturberechnungen für den Autorisierungs-Header: Übertragung der Nutzlast in einem einzigen Block \(AWS Signatur \(Version 4\)\)](#). Der Vorgang ist derselbe, mit der Ausnahme, dass sich die Erstellung von wie folgt `CanonicalRequest` unterscheidet:

- Zusätzlich zu den Anforderungsheadern, die Sie hinzufügen möchten, müssen Sie die folgenden Header angeben:

Header	Description
<code>x-amz-content-sha256</code>	Dieser Header ist für alle AWS Signature Version 4-Anfragen erforderlich. Stellen Sie den Wert auf ein, <code>STREAMING-AWS4-HMAC-SHA256-PAYLOAD</code> um anzugeben, dass die Signatur nur Header abdeckt und dass keine Nutzlast vorhanden ist.

Header	Description
Content-Encoding	Legen Sie den Wert auf <code>aws-chunked</code> fest. Amazon S3 unterstützt mehrere Werte für die Inhaltskodierung. Sie können Ihre benutzerdefinierte Inhaltskodierung angeben, wenn Sie die Streaming-API von Signature Version 4 verwenden. Beispiel: Content-Encoding : aws-chunked,gzip Amazon S3 speichert das resultierende Objekt ohne den <code>aws-chunked</code> Wert im <code>content-encoding</code> Header. Wenn dies der einzige Wert <code>aws-chunked</code> ist, den Sie im <code>content-encoding</code> Header übergeben, betrachtet S3 den <code>content-encoding</code> Header als leer und gibt diesen Header nicht zurück, wenn Sie das Objekt abrufen.
x-amz-decoded-content-length	Stellen Sie den Wert auf die Länge der Daten in Byte ein, die aufgeteilt werden sollen, ohne Metadaten zu zählen. Wenn Sie beispielsweise eine 4-GB-Datei hochladen, legen Sie den Wert auf 4294967296 fest. Dies ist die Rohgröße des hochzuladenden Objekts (Daten, die Sie in Amazon S3 speichern möchten).
Content-Length	Stellen Sie den Wert auf die tatsächliche Größe des übertragenen HTTP-Texts ein, der die Länge Ihrer Daten (eingestellter Wert für <code>x-amz-decoded-content-length</code>) sowie die Chunk-Metadaten umfasst. Jeder Chunk hat Metadaten, wie z. B. die Signatur des vorherigen Chunks. Chunk-Berechnungen werden im folgenden Abschnitt behandelt. Wenn Sie den <code>Transfer-Encoding</code> Header einbeziehen und einen anderen Wert als <code>identity</code> angeben, dürfen Sie den <code>Content-Length</code> Header nicht einschließen.

Sie senden den ersten Block mit der Seed-Signatur. Sie müssen den Chunk wie im folgenden Abschnitt beschrieben erstellen.

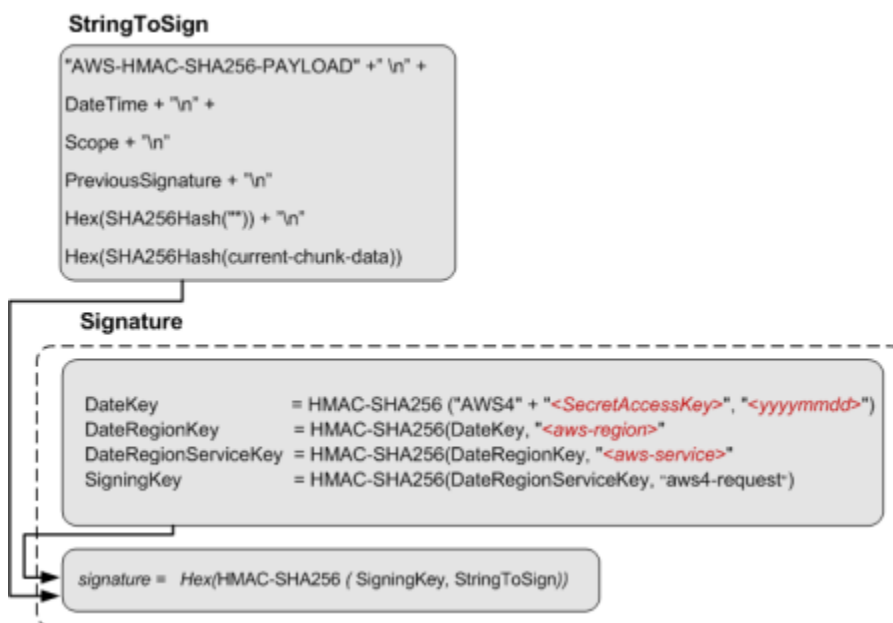
Den Chunk-Körper definieren

Alle Chunks enthalten einige Metadaten. Jeder Chunk muss der folgenden Struktur entsprechen:

```
string(IntHexBase(chunk-size)) + ";chunk-signature=" + signature + \r\n + chunk-data + \r\n
```

Wobei Folgendes gilt:

- `IntHexBase()` ist eine Funktion, die Sie schreiben, um eine Ganzzahl-Blockgröße in eine Hexadezimalgröße umzuwandeln. Wenn Chunk-Size beispielsweise 65536 ist, ist die hexadezimale Zeichenfolge „10000“.
- *chunk-size* ist die Größe der Chunk-Daten in Byte ohne Metadaten. Wenn Sie beispielsweise ein 65-KB-Objekt hochladen und eine Chunk-Größe von 64 KB verwenden, laden Sie die Daten in drei Blöcken hoch: Der erste Block wäre 64 KB groß, der zweite 1 KB und der letzte Block hätte 0 Byte.
- *signature* Für jeden Block berechnen Sie die Signatur mithilfe der folgenden Zeichenfolge zum Signieren. Für den ersten Block verwenden Sie die Seed-Signatur als vorherige Signatur.



Die Größe der letzten Chunk-Daten, die Sie senden, ist 0, obwohl der Chunk-Text immer noch Metadaten enthält, einschließlich der Signatur des vorherigen Chunks.

Beispiel: PUT-Objekt

Sie können die Beispiele in diesem Abschnitt als Referenz verwenden, um die Signaturberechnungen in Ihrem Code zu überprüfen. Bevor Sie sich die Beispiele ansehen, beachten Sie Folgendes:

- Bei den Signaturberechnungen in diesen Beispielen werden die folgenden Beispiel-Sicherheitsanmeldeinformationen verwendet.

Parameter	Wert
AWSSecretAccessKey	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

- Alle Beispiele verwenden den Anforderungszeitstempel 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Alle Beispiele werden als Bucket-Namen verwendet. examplebucket
- Es wird davon ausgegangen, dass sich der Bucket in der Region USA Ost (Nord-Virginia) befindet, und die Anmeldeinformationen Scope und die Signing Key Berechnungen werden us-east-1 als Regionsspezifizierer verwendet. Weitere Informationen finden Sie unter [Regionen und Endpunkte](#) in der Allgemeinen Referenz zu Amazon Web Services.
- Sie können Anfragen im Pfadstil oder im virtuell gehosteten Stil verwenden. In den folgenden Beispielen werden beispielsweise Anfragen im virtuell gehosteten Stil verwendet:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Weitere Informationen finden Sie unter [Virtuelles Hosting von Buckets](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Das folgende Beispiel sendet eine PUT Anfrage zum Hochladen eines Objekts. Bei den Signaturberechnungen wird Folgendes vorausgesetzt:

- Sie laden eine 65-KB-Textdatei hoch, und der Dateiinhalt ist eine einstellige Zeichenfolge, die aus dem Buchstaben 'a' besteht.
- Die Chunk-Größe beträgt 64 KB. Infolgedessen wird die Nutzlast in drei Blöcken hochgeladen: 64 KB, 1 KB und der letzte Block mit 0 Byte an Chunk-Daten.
- Das resultierende Objekt hat den Schlüsselnamen. `chunkObject.txt`
- Sie fordern REDUCED_REDUNDANCY als Speicherklasse an, indem Sie den `x-amz-storage-class` Anforderungsheader hinzufügen.

Hinweise zur API-Aktion finden Sie unter [PutObject](#). Die allgemeine Anforderungssyntax lautet wie folgt:

```
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>
```

Die folgenden Schritte zeigen Signaturberechnungen.

1. Seed-Signatur — Erstellen Sie eine zu signierende Zeichenfolge
 - 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
```

In der kanonischen Anfrage ist die dritte Zeile leer, da die Anfrage keine Abfrageparameter enthält. Die letzte Zeile ist die konstante Zeichenfolge, die als Wert der Hash-Nutzlast bereitgestellt wird. Diese sollte dem Wert von `x-amz-content-sha256` header

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
cee3fed04b70f867d036f722359b0b1f2f0e5dc0efadbc082b76c4c60e316455
```

Note

Informationen zu den einzelnen Zeilen in der zu signierenden Zeichenfolge finden Sie in dem Diagramm, das die Berechnung der Startsignatur erklärt.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Seed-Signatur

```
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
```

4. Authorization-Header

Der resultierende Authorization-Header lautet wie folgt:


```
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. Block 1: (65536 Byte, mit dem Wert 97 für den Buchstaben 'a')

a. Zu signierende Chunk-Zeichenfolge:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
4f232c4386841ef735655705268965c44a0e4690baa4adea153f7db9fa80a0a9
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bec184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

Informationen zu den einzelnen Zeilen der zu signierenden Zeichenfolge finden Sie im vorherigen Diagramm, das die verschiedenen Komponenten der zu signierenden Zeichenfolge zeigt (die letzten drei Zeilen lauten z. B., `previous-signaturehash("")`, `undhash(current-chunk-data)`).

b. Signatur des Blocks:

```
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
```

c. Gesendete Chunk-Daten:

```
10000;chunk-
signature=ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
<65536-bytes>
```

6. Chunk 2: (1024 Byte, mit dem Wert 97 für den Buchstaben 'a')

a. Zu signierende Chunk-Zeichenfolge:

```
AWS4-HMAC-SHA256-PAYLOAD
```

```
20130524T000000Z
20130524/us-east-1/s3/aws4_request
ad80c730a21e5b8d04586a2213dd63b9a0e99e0e2307b0ade35a65485a288648
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Chunk-Signatur:

```
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
```

c. Gesendete Chunk-Daten:

```
400; chunk-
signature=0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
<1024 bytes>
```

7. Block 3: (0-Byte-Daten)

a. Zu signierende Chunk-Zeichenfolge:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
0055627c9e194cb4542bae2aa5492e3c1575bbb81b612b7d234b86a503ef5497
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Chunk-Signatur:

```
b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

c. Gesendete Chunk-Daten:

```
0; chunk-
signature=b6c6ea8a5354eaf15b3cb7646744f4275b71ea724fed81ceb9323e279d449df9
```

Signaturberechnungen für nachfolgende Header (aufgeteilte Uploads) (AWS Signatur (Version 4))

Wenn Sie Anfragen mithilfe des Authorization Headers authentifizieren, können Sie die Nutzlast auch in Blöcken hochladen. Wenn Sie die Daten für das Objekt in Blöcken senden, haben Sie auch die Möglichkeit, nachfolgende Header einzufügen. (Weitere Informationen finden Sie unter [Signaturberechnungen für den Autorisierungs-Header: Übertragung von Nutzdaten in mehreren Blöcken \(Chunked Upload\) \(AWS Signatur \(Version 4\)\)](#).) In diesem Abschnitt werden die Schritte beschrieben, die Sie ausführen müssen, wenn Sie am Ende Ihres Uploads mit mehreren Blöcken einen abschließenden Header einfügen möchten.

Important

Wenn Sie nachfolgende Header hinzufügen, müssen Sie in Ihrem ersten Header Folgendes angeben:

- Sie `x-amz-content-sha256` müssen einen geeigneten Wert angeben, der angibt, dass ein Trailer aufgenommen wird. Die akzeptablen Werte für `x-amz-content-sha256` finden Sie unter [Authentifizieren von Anfragen: Verwenden des Autorisierungs-Headers \(AWS Signatur \(Version 4\)\)](#).
- Sie `x-amz-trailer` müssen festlegen, dass der Inhalt angegeben wird, den Sie in Ihre nachfolgende Kopfzeile aufnehmen.

Nachstehende Header werden erst gesendet, nachdem die Chunks hochgeladen wurden. Frühere Chunks werden wie gewohnt gesendet und signiert, wie in den vorherigen Abschnitten beschrieben, einschließlich des Sendens des letzten Chunks mit einer Nutzlast von 0 Byte. Die nachfolgenden Header werden als eigener Chunk aufgenommen und nach dem letzten Chunk mit einer Nutzlast von 0 Byte gesendet. Wenn Ihre Daten beispielsweise mit einem 100-KB-Block enden, würden Sie Folgendes senden:

- Frühere Datenblöcke
- 100 KB letzter Block des Objekts
- Ein Block von 0 Byte, der das Ende des Objekts angibt
- Block mit nachfolgenden Headern

Beispiele: Überprüfung der Signaturberechnungen

Sie können die Beispiele in diesem Abschnitt als Referenz verwenden, um die Signaturberechnungen in Ihrem Code zu überprüfen. Bevor Sie sich die Beispiele ansehen, beachten Sie Folgendes:

- Bei den Signaturberechnungen in diesen Beispielen werden die folgenden Beispiel-Sicherheitsanmeldeinformationen verwendet.

Parameter	Wert
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

- Alle Beispiele verwenden den Anforderungszeitstempel 20130524T000000Z (). Fri, 24 May 2013 00:00:00 GMT
- Alle Beispiele werden als Bucket-Namen verwendet. `examplebucket`
- Es wird davon ausgegangen, dass sich der Bucket in der Region USA Ost (Nord-Virginia) befindet, und die Anmeldeinformationen Scope und die Signing Key Berechnungen werden `us-east-1` als Regionsspezifizierer verwendet. Weitere Informationen finden Sie unter [Regionen und Endpunkte](#) in der Allgemeinen Amazon Web Services-Referenz.
- Sie können Anfragen im Pfadstil oder im virtuell gehosteten Stil verwenden. In den folgenden Beispielen werden beispielsweise Anfragen im virtuell gehosteten Stil verwendet:

```
https://examplebucket.s3.amazonaws.com/photos/photo1.jpg
```

Weitere Informationen finden Sie unter [Virtuelles Hosting von Buckets](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Das folgende Beispiel sendet eine PUT Anfrage zum Hochladen eines Objekts. Bei den Signaturberechnungen wird Folgendes vorausgesetzt:

- Sie laden eine 65-KB-Textdatei hoch, und der Dateiinhalt ist eine einstellige Zeichenfolge, die aus dem Buchstaben 'a' besteht.

- Die Chunk-Größe beträgt 64 KB. Infolgedessen wird die Nutzlast in drei Blöcken hochgeladen: 64 KB, 1 KB und der letzte Block mit 0 Byte an Chunk-Daten.
- Das resultierende Objekt hat den Schlüsselnamen. `chunkObject.txt`
- Sie fordern REDUCED_REDUNDANCY als Speicherklasse an, indem Sie den `x-amz-storage-class` Anforderungsheader hinzufügen.
- Die Übertragung enthält einen CRC32C-Prüfsummenwert als abschließenden Header.

Hinweise zur API-Aktion finden Sie unter. [PutObject](#) Die allgemeine Anforderungssyntax lautet wie folgt:

```
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>
```

Die folgenden Schritte zeigen Signaturberechnungen.

1. Seed-Signatur — Erstellen Sie eine zu signierende Zeichenfolge
 - 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
```

In der kanonischen Anfrage ist die dritte Zeile leer, da die Anfrage keine Abfrageparameter enthält. Die letzte Zeile ist die konstante Zeichenfolge, die als Wert der Hash-Nutzlast bereitgestellt wird. Diese sollte dem Wert von `x-amz-content-sha256` header

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
44d48b8c2f70eae815a0198cc73d7a546a73a93359c070abbaa5e6c7de112559
```

Note

Informationen zu den einzelnen Zeilen in der zu signierenden Zeichenfolge finden Sie in dem Diagramm, das die Berechnung der Startsignatur erklärt.

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Seed-Signatur

```
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
```

4. Authorization-Header

Der resultierende Authorization-Header lautet wie folgt:

```
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. Block 1: (65536 Byte, mit dem Wert 97 für den Buchstaben 'a')

a. Zu signierende Chunk-Zeichenfolge:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
106e2a8a18243abcf37539882f36619c00e2dfc72633413f02d3b74544bfeb8e
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
bf718b6f653bec184e1479f1935b8da974d701b893afcf49e701f3e2f9f9c5a
```

Note

Informationen zu jeder Zeile in der zu signierenden Zeichenfolge finden Sie im Diagramm im vorherigen Thema ([Berechnung der Seed-Signatur](#)), das verschiedene Komponenten der zu signierenden Zeichenfolge zeigt. Die letzten drei Zeilen bestehen beispielsweise aus den folgenden Elementen:

- *previous-signature*
- `hash("")`
- `hash(current-chunk-data)`

b. Chunk-Signatur:

```
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
```

c. Gesendete Chunk-Daten:

```
10000;chunk-
signature=b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
<65536-bytes>
```

6. Block 2: (1024 Byte, mit dem Wert 97 für den Buchstaben 'a')

a. Zu signierende Chunk-Zeichenfolge:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
b474d8862b1487a5145d686f57f013e54db672cee1c953b3010fb58501ef5aa2
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
2edc986847e209b4016e141a6dc8716d3207350f416969382d431539bf292e4a
```

b. Chunk-Signatur:

```
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
```

c. Gesendete Chunk-Daten:

```
400; chunk-
signature=1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
<1024-bytes>
```

7. Block 3: (0-Byte-Daten)

a. Zu signierende Chunk-Zeichenfolge:

```
AWS4-HMAC-SHA256-PAYLOAD
20130524T000000Z
20130524/us-east-1/s3/aws4_request
1c1344b170168f8e65b41376b44b20fe354e373826ccbbe2c1d40a8cae51e5c7
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
e3b0c44298fc1c149afb4c8996fb92427ae41e4649b934ca495991b7852b855
```

b. Chunk-Signatur:

```
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```

c. Gesendete Chunk-Daten:

```
0; chunk-
signature=2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
```


8. Abschnitt 4: Nachfolgende Header

a. Zu signierende Trailer-Chunk-Zeichenfolge:

```
AWS4-HMAC-SHA256-TRAILER
20130524T000000Z
20130524/us-east-1/s3/aws4_request
2ca2aba2005185cf7159c6277faf83795951dd77a3a99e6e65d5c9f85863f992
1e376db7e1a34a8ef1c4bcee131a2d60a1cb62503747488624e10995f448d774
```

Note

Die letzten beiden Zeilen sind `previous-signature` (die 0-Byte-Datenblock-Signatur), `hash` (*`trailing-checksum-header-name: base64-encoded-trailing-checksum-value`*`\n`).

Der Hash wird wie folgt ohne Leerzeichen berechnet:

- Der Name des Headers der nachfolgenden Prüfsumme
- Ein Doppelpunkt (`:`)
- Der Base64-kodierte abschließende Prüfsummenwert
- Ein Zeichen für den Zeilenumbruch (`\n`).

In diesem Beispiel, in dem Sie den `crc32c` Hash-Algorithmus mit dem Base64-kodierten Prüfsummenwert verwendens `008/Q==`, können Sie die Berechnung wie folgt darstellen: `hash( 'x-amz-checksum-crc32c:s008/Q==\n' )`

b. Blocksignatur:

```
d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

c. Gesendete Chunk-Daten:

```
x-amz-checksum-crc32c:s008/Q==
x-amz-trailer-
signature:d81f82fc3505edab99d459891051a732e8730629a2e4a59689829ca17fe2e435
```

Anfragen authentifizieren: Verwenden von Abfrageparametern (AWS Signatur (Version 4))

Wie in der Authentifizierungsübersicht (siehe [Authentifizierungsmethoden](#)) beschrieben, können Sie Authentifizierungsinformationen mithilfe von Abfragezeichenfolgenparametern angeben. Die Verwendung von Abfrageparametern zur Authentifizierung von Anfragen ist nützlich, wenn Sie eine Anfrage vollständig in einer URL ausdrücken möchten. Diese Methode wird auch als Vorsignierung einer URL bezeichnet.

Ein Anwendungsfall für vorsignierte URLs ist, dass Sie temporären Zugriff auf Ihre Amazon S3-Ressourcen gewähren können. Sie können beispielsweise eine vorsignierte URL in Ihre Website einbetten oder sie alternativ im Befehlszeilenclient (wie Curl) verwenden, um Objekte herunterzuladen.

Note

Sie können das auch verwenden, um AWS CLI vorsignierte URLs zu erstellen. Weitere Informationen finden Sie unter [presign](#) in der Referenz zum AWS CLI -Befehl.

Im Folgenden finden Sie ein Beispiel für eine vorsignierte URL.

```
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>
```

Beachten Sie in der Beispiel-URL Folgendes:

- Die Zeilenumbrüche wurden aus Gründen der Lesbarkeit hinzugefügt.
- Der X-Amz-Credential-Wert in der URL zeigt das Zeichen „/“ nur zur besseren Lesbarkeit an. In der Praxis sollte es als codiert werden. %2F Beispiel:

&X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request

Die folgende Tabelle beschreibt die Abfrageparameter in der URL, die Informationen zur Authentifizierung bereitstellen.

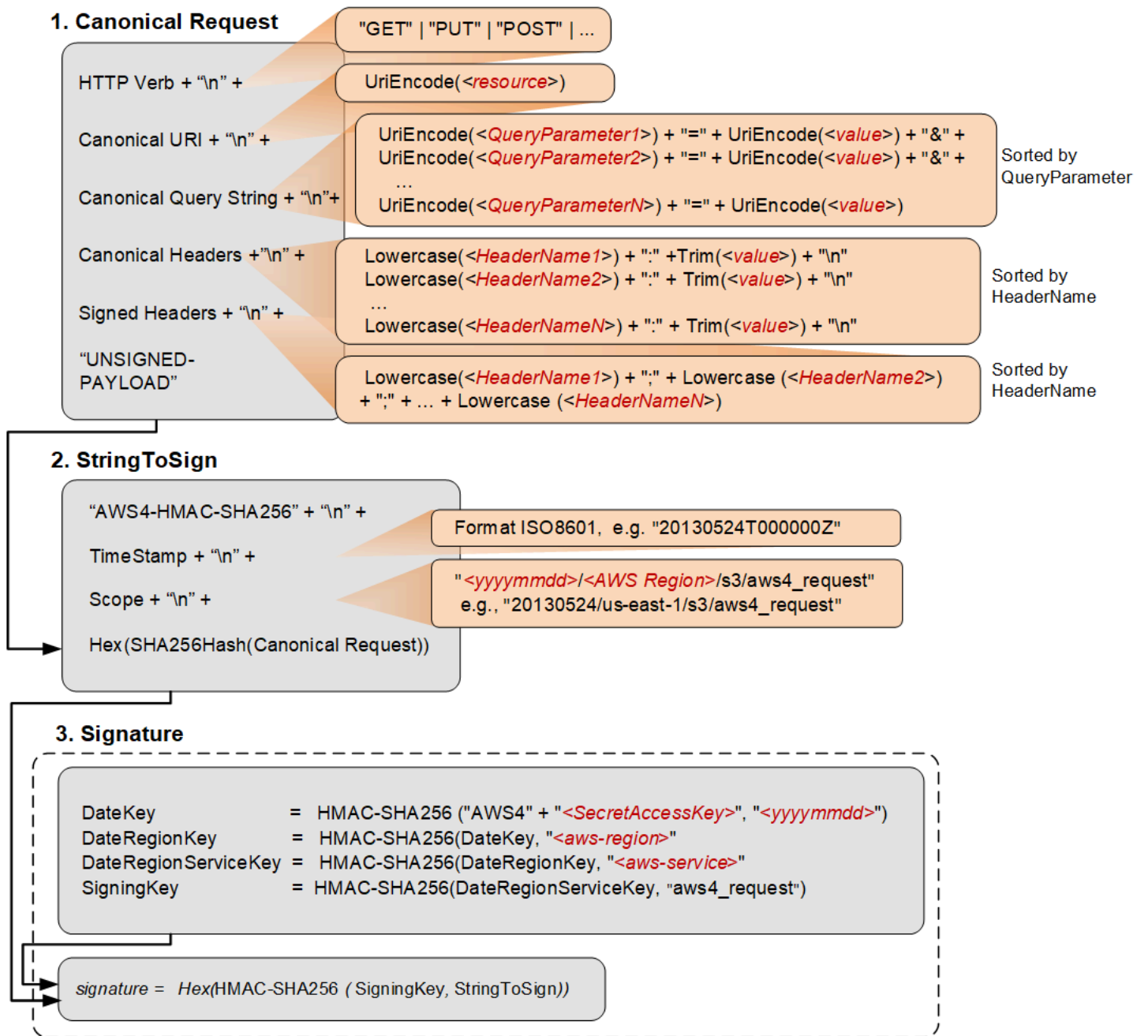
Abfragezeichenfolgen-Parametername	Beispielwert
X-Amz-Algorithm	Identifiziert die Version von AWS Signature und den Algorithmus, den Sie zur Berechnung der Signatur verwendet haben. Für AWS Signature Version 4 setzen Sie diesen Parameterwert auf <code>AWS4-HMAC-SHA256</code> . Diese Zeichenfolge identifiziert AWS Signature Version 4 (AWS 4) und den HMAC-SHA256 Algorithmus (HMAC-SHA256).
X-Amz-Credential	Zusätzlich zu Ihrer Zugriffsschlüssel-ID gibt dieser Parameter auch den Bereich (AWS-Region und den Dienst) an, für den die Signatur gültig ist. Dieser Wert muss mit dem Gültigkeitsbereich übereinstimmen, den Sie für die Signaturberechnungen verwenden, die im folgenden Abschnitt beschrieben werden. Das allgemeine Format für diesen Parameterwert lautet wie folgt: <your-access-key-id> /<date>/<AWS-Region> /<AWS-service> /aws4_request Beispiel: AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request

Abfragezeichenfolgen-Parametername	Beispielwert
	Für Amazon S3 lautet die AWS-service Zeichenfolges3. Eine Liste der AWS-region S3-Zeichenfolgen finden Sie unter Regionen und Endpunkte im Allgemeine AWS-Referenz.
X-Amz-Date	Das Datums- und Uhrzeitformat muss dem ISO-8601-Standard entsprechen und mit dem Format " yyyyMMddT HHmmss Z " formatiert sein. Lauten Datum und Uhrzeit beispielsweise "08/01/2016 15:32:41.982-700", müssen sie zuerst in UTC (Coordinated Universal Time) umgerechnet und dann als „20160801T223241Z“ übermittelt werden.
X-Amz-Expires	Gibt den Zeitraum in Sekunden an, für den die generierte vorsignierte URL gültig ist. 86400Zum Beispiel (24 Stunden). Es handelt sich um einen Ganzzahlwert. Der Mindestwert, den Sie festlegen können, ist 1, und der Höchstwert ist 604800 (sieben Tage). Eine vorsignierte URL kann maximal sieben Tage gültig sein, da der Signaturschlüssel, den Sie für die Signaturberechnung verwenden, bis zu sieben Tage gültig ist.

Abfragezeichenfolgen-Parametername	Beispielwert
X-Amz-SignedHeaders	Listet die Header auf, die Sie zur Berechnung der Signatur verwendet haben. Für die Signaturberechnungen sind folgende Header erforderlich: • Der host HTTP-Header. • Alle x-amz-* Header, die Sie der Anfrage hinzufügen möchten.  Note Für zusätzliche Sicherheit sollten Sie alle Anforderungs-Header signieren, die Sie in Ihre Anfrage aufnehmen möchten.
X-Amz-Signature	Stellt die Signatur zur Authentifizierung Ihrer Anfrage bereit. Diese Signatur muss mit der Signatur übereinstimmen, die Amazon S3 berechnet. Andernfalls lehnt Amazon S3 die Anfrage ab. Beispiel: 733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7 Signaturberechnungen werden im folgenden Abschnitt beschrieben.
X-Amz-Security-Token	Optionaler Parameter für Anmeldeinformationen, wenn Sie Anmeldeinformationen verwenden, die vom STS-Service stammen.

Berechnung einer Signatur

Das folgende Diagramm veranschaulicht den Prozess der Signaturberechnung.



In der folgenden Tabelle werden die im Diagramm dargestellten Funktionen beschrieben. Für diese Funktionen muss Code implementiert werden.

Funktion	Description
Lowercase()	Wandeln Sie die Zeichenfolge in Kleinbuchstaben um.
Hex()	Kodierung in Kleinbuchstaben im Basis-16-Format.

Funktion	Description
SHA256Hash()	Kryptografische Hash-Funktion des Secure Hash Algorithm (SHA).
HMAC-SHA256()	Berechnet HMAC unter Verwendung des SHA256-Algorithmus mit dem bereitgestellten Signaturschlüssel. Dies ist die endgültige Signatur.
Trim()	Entfernen Sie alle führenden oder nachgestellten Leerzeichen.

Funktion	Description
<code>UriEncode()</code>	URI-kodiert jedes Byte. <code>UriEncode()</code> muss die folgenden Regeln durchsetzen: • URI kodiert jedes Byte mit Ausnahme der nicht reservierten Zeichen: „A“–„Z“, „a“–„z“, „0“–„9“, „-“, „.“, „_“ und „~“. • Das Leerzeichen ist ein reserviertes Zeichen und muss als „%20“ (und nicht als „+“) codiert werden. • Jedes URI-codierte Byte besteht aus einem „%“ und dem zweistelligen Hexadezimalwert des Bytes. • Buchstaben im Hexadezimalwert müssen in Großbuchstaben geschrieben sein, zum Beispiel „%1A“. • Kodieren Sie den Schrägstrich „/“ überall außer im Objektschlüsselnamen. Wenn der Objektschlüsselname beispielsweise <code>photos/Jan/sample.jpg</code> lautet, wird der Schrägstrich im Schlüsselnamen nicht codiert.  Important Die von Ihrer Entwicklungsplattform bereitgestellten <code>UriEncode</code> Standardfunktionen funktionieren möglicherweise aufgrund von Unterschieden in der Implementierung und den damit verbundenen Mehrdeutigkeiten in den zugrunde liegenden RFCs nicht. Wir empfehlen Ihnen, Ihre eigene benutzerdefinierte <code>UriEncode</code> Funktion zu schreiben, um sicherzustellen, dass Ihre Codierung funktioniert. Das Folgende ist eine Beispielfunktion <code>UriEncode ()</code> in Java. <pre>public static String UriEncode(CharSequence input, boolean encodeSlash) { StringBuilder result = new StringBui lder();</pre>

Funktion	Description
	<pre> for (int i = 0; i < input.length(); i++) { char ch = input.charAt(i); if ((ch >= 'A' && ch <= 'Z') (ch >= 'a' && ch <= 'z') (ch >= '0' && ch <= '9') ch == '_' ch == '-' ch == '~' ch == '.') { result.append(ch); } else if (ch == '/') { result.append(encodeSlash ? "%2F" : ch); } else { result.append(toHexUTF8(ch)); } } return result.toString(); } </pre>

Weitere Hinweise zum Signaturvorgang (Einzelheiten zur Erstellung einer kanonischen Anfrage, zur zu signierenden Zeichenfolge und zur Berechnung der Signatur) finden Sie unter [Signaturberechnungen für den Autorisierungs-Header: Übertragung der Nutzlast in einem einzigen Block \(AWS Signatur \(Version 4\)\)](#). Der Vorgang ist im Allgemeinen derselbe, mit der Ausnahme, dass sich die Erstellung einer CanonicalRequest vorsignierten URL wie folgt unterscheidet:

- Sie fügen der Canonical Request keinen Nutzlast-Hash hinzu, da Sie beim Erstellen einer vorsignierten URL den Inhalt der Nutzlast nicht kennen, da die URL zum Hochladen einer beliebigen Nutzlast verwendet wird. Stattdessen verwenden Sie eine konstante Zeichenfolge. UNSIGNED-PAYLOAD
- Die kanonische Abfragezeichenfolge muss alle Abfrageparameter aus der vorherigen Tabelle mit Ausnahme von enthalten. X-Amz-Signature
- Für S3 müssen Sie den X-Amz-Security-Token Abfrageparameter in die URL aufnehmen, wenn Sie Anmeldeinformationen verwenden, die vom STS-Dienst stammen.
- Canonical Header müssen den HTTP-Header enthalten. host Wenn Sie beabsichtigen, einen der x-amz-* Header einzubeziehen, müssen diese Header ebenfalls für die Signaturberechnung hinzugefügt werden. Sie können optional alle anderen Header hinzufügen, die Sie in Ihre Anfrage aufnehmen möchten. Für zusätzliche Sicherheit sollten Sie so viele Header wie möglich signieren. Wenn Sie einen signierten Header hinzufügen, der auch ein signierter Abfrageparameter ist,

und sich deren Wert unterscheidet, erhalten Sie eine `InvalidRequest` Fehlermeldung, da die Eingabe widersprüchlich ist.

Ein Beispiel

Angenommen, Sie haben ein Objekt `test.txt` in Ihrem `examplebucket` Eimer. Sie möchten dieses Objekt für einen Zeitraum von 24 Stunden (86400 Sekunden) mit anderen teilen, indem Sie eine vorsignierte URL erstellen.

```
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>
```

Die folgenden Schritte veranschaulichen zunächst die Signaturberechnungen und dann die Konstruktion der vorsignierten URL. Das Beispiel geht von den folgenden zusätzlichen Annahmen aus:

- Der Zeitstempel der Anfrage ist `Fri, 24 May 2013 00:00:00 GMT`.
- Der Bucket befindet sich in der Region USA Ost (Nord-Virginia), und die Anmeldeinformationen `Scope` und die `Signing Key` Berechnungen werden `us-east-1` als Regionsspezifizierer verwendet. Weitere Informationen finden Sie unter [Regionen und Endpunkte](#) in der Allgemeine AWS-Referenz.

Sie können dieses Beispiel als Testfall verwenden, um die Signatur zu überprüfen, die Ihr Code berechnet. Sie müssen jedoch denselben Bucket-Namen, Objektschlüssel, Zeitstempel und die folgenden Beispiel-Anmeldeinformationen verwenden:

Parameter	Wert
<code>AWSSecretAccessKeyId</code>	<code>AKIAIOSFODNN7EXAMPLE</code>
<code>AWSSecretAccessKey</code>	<code>wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY</code>

1. StringToSign

a. CanonicalRequest

```
GET
/test.txt
X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE
%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
host:examplebucket.s3.amazonaws.com

host
UNSIGNED-PAYLOAD
```

b. StringToSign

```
AWS4-HMAC-SHA256
20130524T000000Z
20130524/us-east-1/s3/aws4_request
3bfa292879f6447bbcda7001decf97f4a54dc650c8942174ae0a9121cf58ad04
```

2. SigningKey

```
signing key = HMAC-SHA256(HMAC-SHA256(HMAC-SHA256(HMAC-SHA256("AWS4" +
"<YourSecretAccessKey>", "20130524"), "us-east-1"), "s3"), "aws4_request")
```

3. Signatur

```
aeeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```

Jetzt haben Sie alle Informationen, um eine vorsignierte URL zu erstellen. Die resultierende URL für dieses Beispiel wird wie folgt angezeigt (Sie können dies verwenden, um Ihre vorsignierte URL zu vergleichen):

```
https://examplebucket.s3.amazonaws.com/test.txt?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20130524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host&X-Amz-Signature=aeed9bbccd4d02ee5c0109b86d86835f995330da4c265957d157751f604d404
```

Beispiel 2

Im Folgenden finden Sie ein Beispiel (das nichts mit dem vorherigen Beispiel zu tun hat), das eine vorsignierte URL mit dem Parameter zeigt. X-Amz-Security-Token

```
https://examplebucket.s3.us-east-1.amazonaws.com/test.txt
?X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIAIOSFODNN7EXAMPLE%2F20130524%2Fus-east-1%2Fs3%2Faws4_request
&X-Amz-Date=20200524T000000Z&X-Amz-Expires=86400&X-Amz-SignedHeaders=host
&X-Amz-Security-Token=IQoJb3JpZ2luX2VjEMv%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEaCXVzLWVhc3QtMSJGMEQCIBSubvdj9YGs2g0HKHs0HFdkw0ozjARSKHL987NhhOC8AiBPepRU1obMvIbGU0T
%2BWphFPgK%2Fqpxaf5Snm5M57XFkCqlAgjz%2F%2F%2F%2F%2F%2F%2F%2F%2F2F8BEAAaDDQ3Mjm4NTU0NDY2MCIM83pULBe5%2F
%2BNm1GZBKvkBVslSaJVgwSef7SsoZCJlfJ56weYl3QCWEGr2F4BmCZZyFpmWEYzWnhNK1AnHMj5nKFklKBx30XAT5PZGVr
%2F3HhM0kpdanMXn%2B4PY8lvM8RgnzSu90j0UpGXEOAo
%2F6G80qlMim3%2BZmaQmasn4VYRVeSEd7072QGZ3%2BvDnDVnss0LSYjlV8PP7IujnvHzRNj0Woe0yMe1lL0wTG
%2Fa9usH5hE52w%2FYUJccOn00aZuyROuVsRV4Q70sbWQHuvYUt%2B0tUMKzm8vsFOp4BaNZFqobbjtb36Y92v
%2Bx5ky6i0s8QE886jJtUWMP5ldMziClGx3p0mN5dzsYlM3GyiJ
%2F01mWkPQDwg3mtSp0A9oeeuAMPTA7qmQy9RNuTKBDsx9EW27wvpZBum3SJhEfXv48euadKgrIX3Z79ruQFSQ0c9LUrDjR
%2B4SoWAJqK%2BGX8Q3vPSjsLxhqhEMwd6U4TXcm7ku3gxMbzbqfT8NDg%3D
&X-Amz-Signature=<signature-value>
```

Beispiele: Signaturberechnungen in AWS Signaturversion 4

Themen

- [Beispiele für die Berechnung von Signaturen mit Java \(AWS Signatur \(Version 4\)\)](#)
- [Beispiele für Signaturberechnungen mit C# \(AWS Signatur \(Version 4\)\)](#)

Für authentifizierte Anfragen müssen Sie Code schreiben, um Signaturen zu berechnen, die Authentifizierungsinformationen in Ihren Anfragen bereitstellen, sofern Sie die AWS SDKs nicht verwenden. Die Signaturberechnung in AWS Signature Version 4 (siehe [Anfragen authentifizieren](#)

([AWS Signatur \(Version 4\)](#)) kann ein komplexes Unterfangen sein, und wir empfehlen, dass Sie die AWS SDKs verwenden, wann immer dies möglich ist.

Dieser Abschnitt enthält Beispiele für Signaturberechnungen, die in Java und C# geschrieben wurden. Die Codebeispiele senden die folgenden Anfragen und verwenden den HTTP-Authorization-Header, um Authentifizierungsinformationen bereitzustellen:

- **PUT-Objekt** — Separate Beispiele veranschaulichen sowohl das Hochladen der gesamten Nutzlast auf einmal als auch das Hochladen der Nutzlast in Blöcken. Hinweise zur Verwendung des Authorization-Headers für die Authentifizierung finden Sie unter [Authentifizieren von Anfragen: Verwenden des Autorisierungs-Headers \(AWS Signatur \(Version 4\)\)](#)
- **GET-Objekt** — In diesem Beispiel wird eine vorsignierte URL generiert, um ein Objekt abzurufen. Abfrageparameter stellen die Signatur und andere Authentifizierungsinformationen bereit. Benutzer können eine vorsignierte URL in ihren Browser einfügen, um das Objekt abzurufen, oder Sie können die URL verwenden, um einen anklickbaren Link zu erstellen. Hinweise zur Verwendung von Abfrageparametern für die Authentifizierung finden Sie unter [Anfragen authentifizieren: Verwenden von Abfrageparametern \(AWS Signatur \(Version 4\)\)](#)

Im Rest dieses Abschnitts werden die Beispiele in Java und C# beschrieben. Die Themen enthalten Anweisungen zum Herunterladen und Ausführen der Beispiele.

Beispiele für die Berechnung von Signaturen mit Java (AWS Signatur (Version 4))

Das Java-Beispiel, das die Signaturberechnung zeigt, kann unter heruntergeladen werden <https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AWSS3SigV4JavaSamples.zip>. Beachten Sie, dass diese Beispiele das AWS SDK für Java v1 verwenden, und wir empfehlen, [AWS das SDK für Java v2](#) für neue Anwendungen zu verwenden. In führt die `main()` Funktion Beispielanforderungen aus `RunAllSamples.java`, um ein Objekt zu erstellen, ein Objekt abzurufen und eine vorsignierte URL für das Objekt zu erstellen. Das Beispiel erstellt ein Objekt aus der im Code angegebenen Textzeichenfolge:

```
PutS3ObjectSample.putS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
GetS3ObjectSample.getS3Object(bucketName, regionName, awsAccessKey, awsSecretKey);
PresignedUrlSample.getPresignedUrlToS3Object(bucketName, regionName, awsAccessKey,
    awsSecretKey);
PutS3ObjectChunkedSample.putS3ObjectChunked(bucketName, regionName, awsAccessKey,
    awsSecretKey);
```

Um die Beispiele auf einem Linux-based Computer zu testen

Die folgenden Anweisungen beziehen sich auf das Linux-Betriebssystem.

1. Navigieren Sie in einem Terminal zu dem Verzeichnis, das enthält `AWSS3SigV4JavaSamples.zip`.
2. Entpacken Sie die ZIP-Datei.
3. Öffnen Sie die Datei `./com/amazonaws/services/s3/samples/RunAllSamples.java` in einem Texteditor. Aktualisieren Sie den Code mit den folgenden Informationen:
 - Der Name eines Buckets, in dem das neue Objekt erstellt werden kann.

 Note

Die Beispiele verwenden eine virtuell gehostete Anfrage, um auf den Bucket zuzugreifen. Um mögliche Fehler zu vermeiden, stellen Sie sicher, dass Ihr Bucket-Name den Bucket-Benennungsregeln entspricht, wie unter [Bucket-Einschränkungen und -Beschränkungen](#) im Amazon Simple Storage Service-Benutzerhandbuch beschrieben.

- AWS-Region wo sich der Bucket befindet.

Wenn sich der Bucket in der Region USA Ost (Nord-Virginia) befindet, verwenden Sie `us-east-1`, um die Region anzugeben. Eine Liste der anderen AWS-Regionen finden Sie unter [Amazon Simple Storage Service \(S3\)](#) im Allgemeine AWS-Referenz

4. Kompilieren Sie den Quellcode und speichern Sie die kompilierten Klassen im `bin/` Verzeichnis.

```
javac -d bin -source 6 -verbose com
```

5. Ändern Sie das Verzeichnis in `bin/` und führen Sie es dann aus `RunAllSamples`.

```
java com.amazonaws.services.s3.sample.RunAllSamples
```

Der Code führt alle Methoden in `ausmain()`. Für jede Anfrage zeigt die Ausgabe die kanonische Anfrage, die zu signierende Zeichenfolge und die Signatur an.

Beispiele für Signaturberechnungen mit C# (AWS Signatur (Version 4))

Das C#-Beispiel, das die Signaturberechnung zeigt, kann unter heruntergeladen werden. https://docs.aws.amazon.com/AmazonS3/latest/API/samples/AmazonS3SigV4_Samples_CSharp.zip

Beachten Sie, dass diese Beispiele das AWS SDK für .NET v1 verwenden, und wir empfehlen, [AWS das SDK für.NET v3](#) für neue Anwendungen zu verwenden. In führt die `main()` Funktion Beispielanforderungen aus `Program.cs`, um ein Objekt zu erstellen, ein Objekt abzurufen und eine vorsignierte URL für das Objekt zu erstellen. Der Code für die Signaturberechnung befindet sich im `\Signers` Ordner.

```
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");
```

Um die Beispiele mit Microsoft Visual Studio 2010 oder höher zu testen

1. Entpacken Sie die ZIP-Datei.
2. Starten Sie Visual Studio, und öffnen Sie dann die SLN-Datei.
3. Aktualisieren Sie die App.config Datei mit gültigen Sicherheitsanmeldeinformationen.
4. Aktualisieren Sie den Code wie folgt:
 - Geben Sie `Program.cs` unter den Namen des Buckets und den AWS-Region Speicherort des Buckets ein. Das Beispiel erstellt ein Objekt in diesem Bucket.
5. Führen Sie den Code aus.
6. Um zu überprüfen, ob das Objekt erstellt wurde, kopieren Sie die vorsignierte URL, die das Programm erstellt, und fügen Sie sie dann in ein Browserfenster ein.

Anfragen authentifizieren: Browser-Based Uploads mit POST (AWS Signatur (Version 4))

Amazon S3 unterstützt HTTP-POST-Anfragen, sodass Benutzer Inhalte direkt auf Amazon S3 hochladen können. Die Verwendung von HTTP POST zum Hochladen von Inhalten vereinfacht Uploads und reduziert die Upload-Latenz, wenn Benutzer Daten hochladen, um sie in Amazon S3 zu speichern. In diesem Abschnitt wird beschrieben, wie Sie HTTP-POST-Anfragen authentifizieren. Weitere Informationen zu HTTP-POST-Anfragen, zum Erstellen eines Formulars, zum Erstellen einer POST-Richtlinie und ein Beispiel finden Sie unter [Browser-Based Uploads mit POST \(AWS Signatur \(Version 4\)\)](#).

Gehen Sie wie folgt vor, um eine HTTP-POST-Anfrage zu authentifizieren:

1. Das Formular muss die folgenden Felder enthalten, um die Signatur und relevante Informationen bereitzustellen, anhand derer Amazon S3 die Signatur nach Erhalt der Anfrage erneut berechnen kann:

Elementname	Description
<code>policy</code>	Die Base64-encoded Sicherheitsrichtlinie, die beschreibt, was in der Anfrage zulässig ist. Für die Signaturberechnung ist diese Richtlinie die Zeichenfolge, die Sie signieren. Amazon S3 muss diese Richtlinie erhalten, damit es die Signatur erneut berechnen kann.
<code>x-amz-algorithm</code>	Der verwendete Signaturalgorithmus. Für AWS Signature Version 4 lautet der Wert <code>AWS4-HMAC-SHA256</code> .
<code>x-amz-credential</code>	Zusätzlich zu Ihrer Zugriffsschlüssel-ID enthält dies Informationen zum Umfang, den Sie bei der Berechnung des Signaturschlüssels für die Signaturberechnung verwendet haben.

Elementname	Description
	Es handelt sich um eine Zeichenfolge der folgenden Form: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Beispiel: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .</pre> Für Amazon S3 lautet die AWS-Service-Zeichenfolge. s3 Eine Liste der Amazon aws-region S3-Zeichenfolgen finden Sie unter Regionen und Endpunkte im. Allgemeine AWS-Referenz
x-amz-date	Es ist der Datumswert im ISO8601-Format. Beispiel, 20130728T000000Z . Es ist dasselbe Datum, das Sie bei der Erstellung des Signaturschlüssels verwendet haben. Dies muss auch derselbe Wert sein, den Sie in der Richtlinie (x-amz-date) angeben, die Sie signiert haben.
x-amz-signature	(AWS Signaturversion 4) Der HMAC-SHA256 Hash der Sicherheitsrichtlinie. Weitere Informationen zu den Optionen für die Signatur finden Sie unter Hinzufügen der Signatur zur HTTP-Anfrage im Allgemeine AWS-Referenz.

2. Die POST-Richtlinie muss die folgenden Elemente enthalten:

Elementname	Description
x-amz-algorithm	

Elementname	Description
	Der Signaturalgorithmus, den Sie zur Berechnung der Signatur verwendet haben. Für AWS Signature Version 4 lautet der Wert <code>AWS4-HMAC-SHA256</code> .
<code>x-amz-credential</code>	Zusätzlich zu Ihrer Zugriffsschlüssel-ID enthält dies Informationen zum Umfang, den Sie bei der Berechnung des Signaturschlüssels für die Signaturberechnung verwendet haben. Es handelt sich um eine Zeichenfolge der folgenden Form: <i><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</i> Zum Beispiel <code>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request . .</code>
<code>x-amz-date</code>	Der Datumswert, der in der nach ISO8601 formatierten Zeichenfolge angegeben ist. Zum Beispiel „20130728T000000Z“. Das Datum muss mit dem Datum übereinstimmen, das Sie bei der Erstellung des Signaturschlüssels für die Signaturberechnung verwendet haben.

3. Für die Signaturberechnung ist die POST-Richtlinie die zu signierende Zeichenfolge.

Berechnung einer Signatur

Das folgende Diagramm veranschaulicht den Prozess der Signaturberechnung.



Um eine Signatur zu berechnen

1. Erstellen Sie eine Richtlinie mithilfe der UTF-8 Codierung.
2. Konvertiert die UTF-8-encoded Richtlinie in Base64. Das Ergebnis ist die zu signierende Zeichenfolge.
3. Erstellen Sie die Signatur als HMAC-SHA256 Hash der zu signierenden Zeichenfolge. Sie geben den Signaturschlüssel als Schlüssel für die Hash-Funktion an.
4. Codieren Sie die Signatur mithilfe der Hex-Kodierung.

Weitere Informationen zum Erstellen von HTML-Formularen, Sicherheitsrichtlinien und ein Beispiel finden Sie in den folgenden Unterthemen:

- [Erstellen eines HTML-Formulars \(mit AWS Signatur \(Version 4\)\)](#)
- [POST-Richtlinie](#)
- [Beispiel: Browser-Based Upload mit HTTP POST \(Mit AWS Signatur \(Version 4\)\)](#)

Amazon S3 Signature Version 4: Spezifische Richtlinienschlüssel für die Authentifizierung

Die folgende Tabelle zeigt die Richtlinienschlüssel für die Amazon S3 Signature Version 4-Authentifizierung, die in den Amazon S3-Richtlinien enthalten sein können. In einer Bucket-Richtlinie können Sie diese Bedingungen hinzufügen, um ein bestimmtes Verhalten zu erzwingen, wenn Anforderungen mit der Signature Version 4 authentifiziert werden. Beispiele für Richtlinien finden Sie unter [Beispiele für Bucket-Richtlinien, bei denen Bedingungsschlüssel im Zusammenhang mit Signature Version 4 verwendet werden](#).

Anwendbare Schlüssel	Description
<code>s3:signatureversion</code>	Identifiziert die Version von AWS Signature, die Sie für authentifizierte Anfragen unterstützen möchten. Für authentifizierte Anfragen unterstützt Amazon S3 sowohl Signature Version 4 als auch Signature Version 2. Sie können diese Bedingung zu Ihrer Bucket-Richtlinie hinzufügen, um eine bestimmte Signaturversion zu benötigen. Zulässige Werte: "AWS" identifiziert Signaturversion 2 "AWS4-HMAC-SHA256" identifiziert Signaturversion 4
<code>s3:authType</code>	Amazon S3 unterstützt verschiedene Authentifizierungsmethoden (siehe Anfragen authentifizieren (AWS Signatur (Version 4))). Sie können diesen Bedingungsschlüssel optional verwenden, um eingehende Anfragen auf eine bestimmte Authentifizierungsmethode zu beschränken. Sie können beispielsweise zulassen, dass nur der Authorization

Anwendbare Schlüssel	Description
	HTTP-Header für die Anforderungsauthentifizierung verwendet wird.
	Zulässige Werte:
	REST-HEADER
	REST-QUERY-STRING
	POST

Anwendbare Schlüssel	Description
<code>s3:signatureAge</code>	Die Zeitspanne in Millisekunden, die eine Signatur in einer authentifizierten Anfrage gültig ist. Diese Bedingung funktioniert für: • Vorsignierte URLs — hier gewinnt die restriktivste Bedingung. Weitere Informationen finden Sie unter Arbeiten mit vorsignierten URLs. • Vorsignierter POST — Laden Sie Dateien mithilfe des vorsignierten POSTs direkt auf S3 hoch. Weitere Informationen finden Sie in der Amazon S3-POST-Richtlinie. In Signature Version 2 ist dieser Wert immer auf 0 gesetzt. In Signature Version 4 ist der Signierschlüssel bis zu sieben Tage lang gültig. Daher sind die Signaturen auch bis zu sieben Tage lang gültig. Sie können diese Bedingung verwenden , um das Alter der Unterschrift weiter einzuschränken. Weitere Informationen finden Sie unter Einführung in das Signieren von Anfragen. Beispielwert: 100

Anwendbare Schlüssel	Description
s3:x-amz-content-sha256	Sie können diesen Bedingungsschlüssel verwenden, um nicht signierte Inhalte in Ihrem Bucket zu verbieten. Wenn Sie die Signature Version 4 verwenden, fügen Sie bei Anfragen, die den Authorization Header verwenden, den x-amz-content-sha256 Header in die Signaturrechnung ein und setzen dann seinen Wert auf die Hash-Nutzlast. Beachten Sie, dass dieser Bedingungsschlüssel den x-amz-content-sha256 Header als Abfragezeichenfolgeparameter nicht unterstützt. Sie können diesen Bedingungsschlüssel in Ihrer Bucket-Richtlinie verwenden, um Uploads abzulehnen, bei denen Payloads nicht signiert sind. Sie können beispielsweise Uploads ablehnen, die den Authorization Header zur Authentifizierung von Anfragen verwenden, die Payload jedoch nicht signieren. Weitere Informationen finden Sie unter Signaturberechnungen für den Autorisierungs-Header: Übertragung der Nutzlast in einem einzigen Block (AWS Signatur (Version 4)). Zulässiger Wert: UNSIGNED-PAYLOAD

Beispiele für Bucket-Richtlinien, bei denen Bedingungsschlüssel im Zusammenhang mit Signature Version 4 verwendet werden

Die folgende Bucket-Richtlinie lehnt alle vorsignierten Amazon S3-URL-Anfragen für Objekte ab, examplebucket wenn die Signatur älter als zehn Minuten ist.

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
        }
      }
    }
  ]
}
```

Die folgende Bucket-Richtlinie lässt nur Anfragen zu, die den Authorization-Header für die Anfrageauthentifizierung verwenden. Alle POST- oder vorsignierten URL-Anfragen werden abgelehnt.

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"
        }
      }
    }
  ]
}
```



```

    }
  }
]
}

```

Die folgende Bucket-Richtlinie lehnt Anfragen ab, die vorsignierte URLs für die Anforderungsauthentifizierung verwenden:

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"
        }
      }
    }
  ]
}

```

Die folgende Bucket-Richtlinie lehnt alle Uploads mit unsignierten Nutzlasten ab:

JSON

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Deny uploads with unsigned payloads that use the  
Authorization header.",
      "Effect": "Deny",

```

```
{  
  "Principal": "*",  
  "Action": "s3:*",  
  "Resource": "arn:aws:s3:::examplebucket3/*",  
  "Condition": {  
    "StringEquals": {  
      "s3:x-amz-content-sha256": "UNSIGNED-PAYLOAD"  
    }  
  }  
}
```

Browser-Based Uploads mit POST (AWS Signatur (Version 4))

In diesem Abschnitt wird beschrieben, wie Sie Dateien mithilfe von HTTP-POST-Anfragen über einen Browser direkt auf Amazon S3 hochladen. Er enthält auch Informationen zur Verwendung der AWS JavaScript Amplify-Bibliothek für browserbasierte Datei-Uploads auf Amazon S3.

Themen

- [POST Object](#)
- [POST Object-Wiederherstellung](#)
- [Browser-Based Uploads mit HTTP POST](#)
- [Eine Signatur berechnen](#)
- [Erstellen eines HTML-Formulars \(mit AWS Signatur \(Version 4\)\)](#)
- [POST-Richtlinie](#)
- [Beispiel: Browser-Based Upload mit HTTP POST \(Mit AWS Signatur \(Version 4\)\)](#)
- [Browser-based lädt mit dem auf Amazon S3 hoch AWS Bibliothek erweitern](#)

POST Object

Description

Der POST Vorgang fügt mithilfe von HTML-Formularen ein Objekt zu einem angegebenen Bucket hinzu. POST ist eine alternative Form von PUT, die browserbasierte Uploads ermöglicht, um Objekte in Buckets zu platzieren. Parameter, die PUT über HTTP-Header übergeben werden, werden stattdessen als Formularfelder an den mit multipart/form-data kodierten POST Nachrichtentext übergeben. Um ein Objekt zu einem Bucket hinzuzufügen, müssen Sie WRITE Zugriff auf den Bucket haben. Amazon S3 speichert niemals Teilobjekte. Wenn Sie eine erfolgreiche Antwort erhalten, können Sie sicher sein, dass das gesamte Objekt gespeichert wurde.

Amazon S3 ist ein verteiltes System. Wenn Amazon S3 mehrere Schreib Anforderungen für dasselbe Objekt gleichzeitig empfängt, wird nur die letzte Version des geschriebenen Objekts gespeichert, es sei denn, Sie haben die Versionierung für einen Bucket aktiviert.

Verwenden Sie das Formularfeld, um sicherzustellen, dass Daten beim Durchlaufen des Netzwerks nicht beschädigt werden. Content-MD5 Wenn Sie dieses Formularfeld verwenden, vergleicht Amazon S3 das Objekt mit dem angegebenen MD5-Wert. Wenn sie nicht übereinstimmen, gibt Amazon S3 einen Fehler zurück. Darüber hinaus können Sie den MD5-Wert berechnen, während Sie ein Objekt an Amazon S3 senden, und den zurückgegebenen Wert mit ETag dem berechneten MD5-Wert vergleichen. Das ETag spiegelt nur Änderungen am Inhalt eines Objekts wider, nicht an seinen Metadaten.

Note

Um Ihre Anwendung so zu konfigurieren, dass sie die Anforderungsheader sendet, bevor der Anforderungstext gesendet wird, verwenden Sie den HTTP-Statuscode 100 (Weiter). Bei POST Operationen hilft Ihnen die Verwendung dieses Statuscodes dabei, das Senden des Nachrichtentexts zu vermeiden, wenn die Nachricht aufgrund der Header abgelehnt wird (z. B. aufgrund eines Authentifizierungsfehlers oder einer Umleitung). Weitere Informationen zum HTTP-Statuscode 100 (Weiter) finden Sie in Abschnitt 8.2.3 von. <http://www.ietf.org/rfc/rfc2616.txt>

Amazon S3 verschlüsselt automatisch alle neuen Objekte, die in einen S3-Bucket hochgeladen werden. Die Verschlüsselungseinstellung eines hochgeladenen Objekts hängt von der

Standardverschlüsselungskonfiguration des Ziel-Buckets ab. Standardmäßig verfügen alle Buckets über eine Standardverschlüsselungskonfiguration, die serverseitige Verschlüsselung mit von Amazon S3 verwalteten Schlüsseln () SSE-S3 verwendet.

Wenn der Ziel-Bucket über eine Verschlüsselungskonfiguration verfügt, die serverseitige Verschlüsselung mit einem AWS Key Management Service (AWS KMS) -Schlüssel (SSE-KMS), eine duale serverseitige Verschlüsselung mit einem AWS KMS Schlüssel (DSSE-KMS) oder einen vom Kunden bereitgestellten Verschlüsselungsschlüssel (SSE-C) verwendet, verwendet Amazon S3 den entsprechenden KMS-Schlüssel oder den vom Kunden bereitgestellten Schlüssel, um das hochgeladene Objekt zu verschlüsseln. Wenn Sie beim Hochladen eines Objekts die Verschlüsselungseinstellung des hochgeladenen Objekts ändern möchten, können Sie die Art der serverseitigen Verschlüsselung angeben. Sie können SSE-S3, SSE-KMS DSSE-KMS, oder konfigurieren. SSE-C Weitere Informationen finden Sie unter [Schutz von Daten durch serverseitige Verschlüsselung](#) im Amazon Simple Storage Service User Guide.

 Important

Achten Sie beim Erstellen Ihrer Anfrage darauf, dass das `file` Feld das letzte Feld im Formular ist.

Versionsverwaltung

Wenn Sie die Versionierung für einen Bucket aktivieren, POST wird automatisch eine eindeutige Versions-ID für das hinzugefügte Objekt generiert. Amazon S3 gibt diese ID in der Antwort mithilfe des `x-amz-version-id` Antwort-Headers zurück.

Wenn Sie die Versionierung für einen Bucket aussetzen, verwendet Amazon S3 immer die `null` Versions-ID des in einem Bucket gespeicherten Objekts.

Weitere Informationen zur Rückgabe des Versionsstatus eines Buckets finden Sie unter [GetBucketVersioning](#)

Amazon S3 ist ein verteiltes System. Wenn Sie die Versionierung für einen Bucket aktivieren und Amazon S3 mehrere Schreibanforderungen für dasselbe Objekt gleichzeitig empfängt, werden alle Versionen des Objekts gespeichert.

Beispielanfragen, die Versionierung verwenden, finden Sie unter [Beispielanfrage](#).

Anforderungen

Syntax

```
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--
```

Anforderungsparameter

Bei dieser Implementierung der Operation werden keine Anforderungsparameter verwendet.

Formularfelder

Dieser Vorgang kann die folgenden Formularfelder verwenden.

Name	Description	Erforderlich
AWSSecretAccessKey	Die AWS Zugriffsschlüssel-ID des Besitzers des Buckets, der einem anonymen Benutzer Zugriff auf eine Anfrage gewährt, die die Einschränkungen in der Richtlinie erfüllt.	Bedingt
	Typ: Zeichenfolge	
	Standard: keiner	

Name	Description	Erforderlich
	Einschränkungen: Erforderlich, wenn der Anfrage ein Richtlinienokument beigelegt ist.	
acl	Die angegebene Amazon S3-Zugriffskontrollliste (ACL). Wenn die angegebene ACL nicht gültig ist, wird ein Fehler generiert. Weitere Informationen zu ACLs finden Sie in der Übersicht über die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service-Benutzerhandbuch. Typ: Zeichenfolge Standard: privat Zulässige Werte: private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	Nein
Cache-Control , Content-Type , Content-Disposition , Content-Encoding , Expires	Die REST-specific Header. Weitere Informationen finden Sie unter PutObject. Typ: Zeichenfolge Standard: keiner	Nein
file	Der Datei- oder Textinhalt. Der Datei- oder Textinhalt muss das letzte Feld im Formular sein. Sie können nicht mehr als eine Datei zur gleichen Zeit hochladen. Typ: Datei- oder Textinhalt Standard: keiner	Ja

Name	Description	Erforderlich
key	Der Name des hochgeladenen Schlüssels. Verwenden Sie die <code>\${filename}</code> Variable, um den vom Benutzer angegebenen Dateinamen zu verwenden. Wenn beispielsweise ein Benutzer namens Mary die Datei hochlädt <code>example.jpg</code> und Sie angeben <code>/user/mary/\${filename}</code>, lautet <code>/user/mary/example.jpg</code> der Schlüsselname. Weitere Informationen finden Sie unter Objektschlüssel und Metadaten im Amazon Simple Storage Service-Benutzerhandbuch. Typ: Zeichenfolge Standard: keiner	Ja
policy	Die Sicherheitsrichtlinie, die beschreibt, was in der Anfrage zulässig ist. Anfragen ohne Sicherheitsrichtlinie werden als anonym betrachtet und funktionieren nur in öffentlich beschreibbaren Buckets. Weitere Informationen finden Sie unter HTML-Formulare und Upload-Beispiele im Amazon Simple Storage Service-Benutzerhandbuch. Typ: Zeichenfolge Standard: keiner Einschränkungen: Eine Sicherheitsrichtlinie ist erforderlich, wenn der Bucket nicht öffentlich beschreibbar ist.	Bedingt

Name	Description	Erforderlich
<code>success_action_redirect</code> , <code>redirect</code>	Die URL, zu der der Client nach einem erfolgreichen Upload umgeleitet wird. Wenn <code>success_action_redirect</code> nicht angegeben ist, gibt Amazon S3 den leeren Dokumententyp zurück, der im Feld <code>success_action_status</code> angegeben ist. Wenn Amazon S3 die URL nicht interpretieren kann, verhält es sich so, als ob das Feld nicht vorhanden wäre. Wenn der Upload fehlschlägt, zeigt Amazon S3 einen Fehler an und leitet den Benutzer nicht zu einer URL um. Typ: Zeichenfolge Standard: keiner  Note Der <code>redirect</code> Feldname ist veraltet und die Unterstützung für den <code>redirect</code> Feldnamen wird in Zukunft entfernt.	Nein

Name	Description	Erforderlich
success_action_status	Wenn Sie nichts angebensuccess_action_redirect , wird der Statuscode an den Client zurückgegeben, wenn der Upload erfolgreich ist. Dieses Feld akzeptiert die Werte 200201, oder 204 (die Standardeinstellung). Wenn der Wert auf 200 oder gesetzt ist204, gibt Amazon S3 ein leeres Dokument mit einem Statuscode 200 oder 204 zurück. Wenn der Wert auf gesetzt ist201, gibt Amazon S3 ein XML-Dokument mit einem Statuscode 201 zurück. Wenn der Wert nicht festgelegt ist oder wenn er auf einen ungültigen Wert gesetzt ist, gibt Amazon S3 ein leeres Dokument mit einem 204-Statuscode zurück. Typ: Zeichenfolge Standard: keiner	Nein

Name	Description	Erforderlich
tagging	Der angegebene Satz von Tags, die dem Objekt hinzugefügt werden sollen. Verwenden Sie das folgende Kodierungsschema, um Tags hinzuzufügen. <pre><Tagging> <TagSet> <Tag> <Key>TagName</Key> <Value>TagValue</Value> </Tag> ... </TagSet> </Tagging></pre>	Nein
Weitere Informationen finden Sie unter Objektkenzeichnung im Amazon Simple Storage Service-Benutzerhandbuch. Typ: Zeichenfolge Standard: keiner		

Name	Description	Erforderlich
<code>x-amz-storage-class</code>	Die Speicherklasse, die zum Speichern des Objekts verwendet werden soll. Wenn Sie keine Klasse angeben, verwendet Amazon S3 die Standard-Speicherklasse, STANDARD. Amazon S3 unterstützt andere Speicherklassen. Weitere Informationen finden Sie unter Speicherklassen im Amazon Simple Storage Service-Benutzerhandbuch. Typ: Zeichenfolge Standard: STANDARD Zulässige Werte: REDUCED_REDUNDANCY EXPRESS_ONEZONE DEEP_ARCHIVE GLACIER GLACIER_IR INTELLIGENT_TIERING ONEZONE_IA STANDARD STANDARD_IA	Nein
<code>x-amz-meta-*</code>	Header, die mit diesem Präfix beginnen, sind benutzerdefinierte Metadaten. Jeder wird gespeichert und als Satz von Schlüssel-Wert-Paaren zurückgegeben. Amazon S3 validiert oder interpretiert keine benutzerdefinierten Metadaten. Weitere Informationen finden Sie unter PutObject. Typ: Zeichenfolge Standard: keiner	Nein

Name	Description	Erforderlich
x-amz-security-token	Das DevPay Amazon-Sicherheitstoken. Für jede Anfrage, die Amazon verwendet, DevPay sind zwei x-amz-security-token Formularfelder erforderlich: eines für das Produkt-Token und eines für das Benutzer-Token. Typ: Zeichenfolge Standard: keiner	Nein
x-amz-signature	(AWS Signaturversion 4) Der HMAC-SHA256 Hash der Sicherheitsrichtlinie. Typ: Zeichenfolge Standard: keiner	Bedingt

Name	Description	Erforderlich
<code>x-amz-website-redirect-location</code>	Wenn der Bucket als Website konfiguriert ist, leitet dieses Feld Anfragen für dieses Objekt an ein anderes Objekt im selben Bucket oder an eine externe URL weiter. Amazon S3 speichert den Wert dieses Headers in den Objektmetadaten. Informationen zu Objektmetadaten finden Sie unter Objektschlüssel und Metadaten im Amazon Simple Storage Service-Benutzerhandbuch. Im folgenden Beispiel legt der Anforderungsheader die Weiterleitung auf ein Objekt (<code>anotherPage.html</code>) im selben Bucket fest: <pre>x-amz-website-redirect-location: /anotherPage.html</pre> Im folgenden Beispiel legt der Anforderungsheader die Objektleitung auf eine andere Website fest: <pre>x-amz-website-redirect-location: http://www.example.com/</pre> Weitere Informationen zum Website-Hosting in Amazon S3 finden Sie unter Hosten von Websites auf Amazon S3 und So konfigurieren Sie Webseitenumleitungen im Amazon Simple Storage Service-Benutzerhandbuch. Typ: Zeichenfolge Standard: keiner Einschränkungen: Dem Wert muss <code>/http://</code>, oder <code>https://</code> vorangestellt werden. Die Länge des Werts ist auf 2 KB begrenzt.	Nein

Zusätzliche Felder für das Prüfsummen-Anforderungsformular

Beim Hochladen eines Objekts können Sie verschiedene Prüfsummen angeben, die Sie zur Überprüfung Ihrer Datenintegrität verwenden möchten. Sie können einen zusätzlichen Prüfsummenalgorithmus angeben, den Amazon S3 verwenden soll. Weitere Informationen zu zusätzlichen Prüfsummenwerten finden Sie unter [Prüfen der Objektintegrität](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Name	Description	Erforderlich
x-amz-checksum-algorithm	Gibt den Algorithmus an, der zur Erstellung der Prüfsumme für das Objekt verwendet wurde. Wenn ein Wert angegeben ist, müssen Sie den entsprechenden Prüfsummen-Header angeben. Andernfalls generiert Ihre Anfrage einen 400-Fehler. Mögliche Werte sind CRC32, CRC32C, SHA1 und SHA256.	Nein
x-amz-checksum-crc32	Gibt die Base64-kodierte 32-Bit-CRC32-Prüfsumme des Objekts an. Dieser Parameter ist erforderlich, wenn der Wert von ist. x-amz-checksum-algorithm CRC32	Bedingt
x-amz-checksum-crc32c	Gibt die Base64-kodierte 32-Bit-CRC32C-Prüfsumme des Objekts an. Dieser Parameter ist erforderlich, wenn der Wert von ist. x-amz-checksum-algorithm CRC32C	Bedingt
x-amz-checksum-sha1	Gibt den Base64-kodierten SHA-1 160-Bit-Digest des Objekts an. Dieser Parameter ist erforderlich, wenn der Wert von ist. x-amz-checksum-algorithm SHA1	Bedingt
x-amz-checksum-sha256	Gibt den Base64-kodierten SHA-256 256-Bit-Digest des Objekts an.	Bedingt

Name	Description	Erforderlich
	Dieser Parameter ist erforderlich, wenn der Wert von ist. <code>x-amz-checksum-algorithm</code> <code>SHA256</code>	

Server-Side Verschlüsselungsspezifische Felder im Anforderungsformular

Server-side Verschlüsselung ist Datenverschlüsselung im Ruhezustand. Amazon S3 verschlüsselt Ihre Daten beim Schreiben auf Festplatten in AWS Rechenzentren und entschlüsselt Ihre Daten, wenn Sie darauf zugreifen. Beim Hochladen eines Objekts können Sie die Art der serverseitigen Verschlüsselung angeben, die Amazon S3 für die Verschlüsselung des Objekts verwenden soll.

Es gibt vier Arten der serverseitigen Verschlüsselung:

- Server-side Verschlüsselung mit von Amazon S3 verwalteten Schlüsseln (SSE-S3) — Ab Mai 2022 ist für alle Amazon S3-Buckets standardmäßig eine Verschlüsselung konfiguriert. Die Standardoption für serverseitige Verschlüsselung ist mit. SSE-S3 Jedes Objekt wird mit einem eindeutigen Schlüssel verschlüsselt. SSE-S3 Verschlüsselt als zusätzliche Schutzmaßnahme den Schlüssel selbst mit einem Stammschlüssel, den er regelmäßig wechselt. SSE-S3 verwendet eine der stärksten verfügbaren Blockchiffren, den 256-Bit-Advanced Encryption Standard (AES-256), um Ihre Daten zu verschlüsseln.
- Server-side Verschlüsselung mit AWS KMS Schlüsseln (SSE-KMS) — SSE-KMS wird durch eine Integration des AWS KMS Dienstes mit Amazon S3 bereitgestellt. Mit AWS KMS haben Sie mehr Kontrolle über Ihre Schlüssel. Sie können beispielsweise separate Schlüssel anzeigen, Kontrollrichtlinien bearbeiten und den Schlüsseln in AWS CloudTrail folgen. Darüber hinaus können Sie vom Kunden verwaltete Schlüssel erstellen und verwalten oder von Von AWS verwaltete Schlüssel verwaltete Schlüssel verwenden, die für Sie, Ihren Service und Ihre Region einzigartig sind.
- Dual-layer serverseitige Verschlüsselung mit AWS KMS Schlüsseln (DSSE-KMS) — Die Dual-layer serverseitige Verschlüsselung mit AWS KMS keys (DSSE-KMS) ähnelt der Verschlüsselung auf Objektebene SSE-KMS, wendet jedoch statt einer Ebene zwei einzelne Ebenen an.
- Server-side Verschlüsselung mit vom Kunden bereitgestellten Schlüsseln (SSE-C) — Mit SSE-C verwalten Sie die Verschlüsselungsschlüssel, und Amazon S3 verwaltet die Verschlüsselung beim Schreiben auf Festplatten sowie die Entschlüsselung, wenn Sie auf Ihre Objekte zugreifen.

Note

Wenn Sie die serverseitige Verschlüsselung mit vom Kunden bereitgestellten Schlüsseln (SSE-C) für Ihren Allzweck-Bucket gesperrt haben, erhalten Sie die Fehlermeldung HTTP 403 Access Denied, wenn Sie beim Schreiben neuer Daten in Ihren Bucket die SSE-C Anforderungsheader angeben. Weitere Informationen finden Sie unter [Blockieren oder Entsperren SSE-C für einen Allzweck-Bucket](#).

Weitere Informationen finden Sie unter [Schützen von Daten mithilfe serverseitiger Verschlüsselung](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Je nachdem, welche Art der serverseitigen Verschlüsselung Sie verwenden möchten, geben Sie die folgenden Formularfelder an.

- Verwenden Sie SSE-S3, SSE-KMS, oder DSSE-KMS — Wenn Sie diese Arten der serverseitigen Verschlüsselung verwenden möchten, geben Sie die folgenden Formularfelder in der Anfrage an.

Name	Description	Erforderlich
<code>x-amz-server-side-encryption</code>	Gibt den serverseitigen Verschlüsselungsalgorithmus an, der verwendet werden soll, wenn Amazon S3 ein Objekt erstellt. Um es zu verwenden SSE-S3, geben Sie <code>aws:s3</code> an. Um es zu verwenden SSE-KMS, geben Sie <code>aws:kms</code> an. Um es zu verwenden DSSE-KMS, geben Sie <code>aws:kms:dsse</code> an. Typ: Zeichenfolge Gültiger Wert: <code>aws:s3</code> , <code>aws:kms</code> , <code>aws:kms:dsse</code>	Ja
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	Wenn der <code>x-amz-server-side-encryption</code> Header den gültigen Wert <code>aws:kms</code> oder <code>aws:kms:dsse</code> hat, gibt dieser Header die ID von einem AWS KMS key an, der zum Verschlüsseln des Objekts verwendet wurde. Typ: Zeichenfolge	Ja, wenn der Wert <code>aws:kms</code> oder <code>aws:kms:dsse</code> ist

Name	Description	Erforderlich
		ver-side-encryption ist aws:kms aws:kms:dsse
x-amz-server-side-encryption-context	Wenn der gültige Wert aws:kms oder x-amz-server-side-encryption istaws:kms:dsse , gibt dieser Header den Verschlüsselungskontext für das Objekt an. Der Wert dieses Headers ist eine Base64-kodierte UTF-8 Zeichenfolge, die JSON-formatted Schlüssel-Wert-Paare für den Verschlüsselungskontext enthält. Typ: Zeichenfolge	Nein
x-amz-server-side-encryption-bucket-key-enabled	Wenn der Wert aws:kms oder gültig x-amz-server-side-encryption ist, gibt dieser Header anaws:kms:dsse , ob Amazon S3 einen S3-Bucket-Schlüssel mit oder verwenden soll. SSE-KMS DSSE-KMS Wenn dieser Header auf gesetzt true wird, verwendet Amazon S3 einen S3-Bucket Key für die Objektverschlüsselung mit SSE-KMS oder DSSE-KMS. Typ: Boolescher Wert	Nein

Note

Wenn Sie x-amz-server-side-encryption:aws:kms oder angebenx-amz-server-side-encryption:aws:kms:dsse, aber nicht angebenx-amz-server-side-encryption-aws-kms-key-id, verwendet Amazon S3 das Von AWS verwalteter Schlüssel (aws/S3), um die Daten zu schützen.

- Verwenden SSE-C — Wenn Sie Ihre eigenen Verschlüsselungsschlüssel verwalten möchten, müssen Sie in der Anfrage alle folgenden Formularfelder angeben.

 Note

Wenn Sie diesen Wert verwenden SSE-C, entspricht der ETag Wert, den Amazon S3 in der Antwort zurückgibt, nicht dem MD5-Wert des Objekts.

Name	Description	Erforderlich
<code>x-amz-server-side-encryption-customer-algorithm</code>	Gibt den Algorithmus an, der beim Verschlüsseln des Objekts verwendet werden soll. Typ: Zeichenfolge Standard: keiner Gültiger Wert: AES256 Einschränkungen: Muss von gültigen <code>x-amz-server-side-encryption-customer-key</code> <code>x-amz-server-side-encryption-customer-key-MD5</code> UND-Feldern begleitet werden.	Ja

Name	Description	Erforderlich
<code>x-amz-server-side-encryption-customer-key</code>	Gibt den vom Kunden bereitgestellten Base64-kodierten Verschlüsselungsschlüssel an, den Amazon S3 zum Verschlüsseln von Daten verwenden soll. Dieser Wert wird verwendet, um das Objekt zu speichern, und dann wird es verworfen. Amazon speichert den Verschlüsselungsschlüssel nicht. Der Schlüssel muss für die Verwendung mit dem Algorithmus geeignet sein, der im <code>x-amz-server-side-encryption-customer-algorithm</code>-Header angegeben ist. Typ: Zeichenfolge Standard: keiner Einschränkungen: Muss von gültigen <code>x-amz-server-side-encryption-customer-algorithm</code> und <code>x-amz-server-side-encryption-customer-key-MD5</code>-UND-Feldern begleitet werden.	Ja
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Gibt den Base64-kodierten 128-Bit-MD5-Digest des Verschlüsselungsschlüssels gemäß RFC 1321 an. http://tools.ietf.org/html/rfc1321 Amazon S3 verwendet diesen Header für eine Überprüfung der Nachrichtenintegrität, um sicherzustellen, dass der Verschlüsselungsschlüssel fehlerfrei übertragen wurde. Typ: Zeichenfolge Standard: keiner Einschränkungen: Muss von gültigen <code>x-amz-server-side-encryption-customer-algorithm</code> und <code>x-amz-server-side-encryption-customer-key</code>-UND-Feldern begleitet werden.	Ja

Antworten

Antwort-Header

Diese Implementierung des Vorgangs kann zusätzlich zu den Antwortheadern, die allen Antworten gemeinsam sind, die folgenden Antwortheader enthalten. Weitere Informationen finden Sie unter

[Allgemeine Antwortheader](#).

Name	Description
<code>x-amz-checksum-crc32</code>	Die Base64-kodierte 32-Bit-CRC32-Prüfsumme des Objekts. Typ: Zeichenfolge
<code>x-amz-checksum-crc32c</code>	Die Base64-kodierte 32-Bit-CRC32C-Prüfsumme des Objekts. Typ: Zeichenfolge
<code>x-amz-checksum-sha1</code>	Der Base64-kodierte 160-Bit-Digest des Objekts. SHA-1 Typ: Zeichenfolge
<code>x-amz-checksum-sha256</code>	Der Base64-kodierte 256-Bit-Digest des Objekts. SHA-256 Typ: Zeichenfolge
<code>x-amz-expiration</code>	Wenn im Rahmen der Lifecycle-Konfiguration des Buckets eine <code>Expiration</code> Aktion für das Objekt konfiguriert ist, gibt Amazon S3 diesen Header zurück. Der Header-Wert umfasst eine <code>expiry-date</code> Komponente und eine URL-encoded <code>rule-id</code> Komponente. Bei versionsfähigen Buckets gilt dieser Header nur für aktuelle Versionen. Amazon S3 bietet keinen Header, der angibt, wann eine nicht aktuelle Version dauerhaft gelöscht werden kann. Weitere

Name	Description
	Informationen finden Sie unter PutBucketLifecycle Configuration . Typ: Zeichenfolge
<code>success_action_redirect, redirect</code>	Die URL, zu der der Client bei einem erfolgreichen Upload umgeleitet wird. Typ: Zeichenfolge Vorfahr: PostResponse
<code>x-amz-server-side-encryption</code>	Der serverseitige Verschlüsselungsalgorithmus, der beim Speichern dieses Objekts in Amazon S3 verwendet wurde (z. B., <code>AES256aws:kms,aws:kms:dsse</code>). Typ: Zeichenfolge
<code>x-amz-server-side-encryption-aws-kms-key-id</code>	Wenn der <code>x-amz-server-side-encryption</code> Header einen gültigen Wert von <code>aws:kms</code> , gibt dieser Header die ID des KMS-Schlüssels an, der zum Verschlüsseln des Objekts verwendet wurde. Typ: Zeichenfolge
<code>x-amz-server-side-encryption-bucket-key-enabled</code>	Wenn dieser Header einen gültigen Wert von <code>x-amz-server-side-encryption-aws-kms-key-id</code> hat, gibt dieser Header an, ob das Objekt SSE-KMS mithilfe eines S3-Bucket-Schlüssels verschlüsselt wurde. Wenn dieser Header auf <code>true</code> gesetzt ist, verwendet das Objekt einen S3-Bucket Key mit SSE-KMS. Typ: Boolescher Wert

Name	Description
<code>x-amz-server-side-encryption-customer-algorithm</code>	Wenn eine Anfrage gestellt SSE-C wurde, enthält die Antwort diesen Header, der den verwendeten Verschlüsselungsalgorithmus bestätigt. Typ: Zeichenfolge Zulässige Werte: AES256
<code>x-amz-server-side-encryption-customer-key-MD5</code>	Falls eine Anfrage gestellt SSE-C wurde, enthält die Antwort diesen Header, um die Integrität der Round-Trip-Nachricht des vom Kunden bereitgestellten Verschlüsselungsschlüssels zu überprüfen. Typ: Zeichenfolge
<code>x-amz-version-id</code>	Version des Objekts. Typ: Zeichenfolge

Antwortelemente

Name	Description
Bucket	Der Name des Buckets, in dem das Objekt gespeichert wurde. Typ: Zeichenfolge Vorfahr: PostResponse
ETag	Das Entitäts-Tag (ETag) ist ein MD5-Hash des Objekts, das Sie für bedingte GET Operationen verwenden können, indem Sie das Anforderungs-Tag zusammen mit der <code>If-Modified</code> GET Anforderungsoperation verwenden. ETagspiegelt nur Änderungen am Inhalt eines Objekts wider, nicht an seinen Metadaten. Typ: Zeichenfolge

Name	Description
	Vorfahr: PostResponse
Key	Der Schlüsselname des Objekts. Typ: Zeichenfolge Vorfahr: PostResponse
Location	Die URI des Objekts. Typ: Zeichenfolge Vorfahr: PostResponse

Spezielle Fehler

Diese Implementierung der Operation gibt keine speziellen Fehler zurück. Allgemeine Informationen zu Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlerantworten](#).

Beispiele

Beispielanforderung

```
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

Beispielantwort mit unterbrochener Versionierung

Im Folgenden finden Sie ein Beispiel für eine Antwort, wenn die Bucket-Versionierung ausgesetzt ist:

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
```

```
x-amz-id-2: LriYPLdmOdAiIfgSm/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
```

In dieser Antwort lautet null die Versions-ID.

Beispielantwort mit aktivierter Versionierung

Im Folgenden finden Sie ein Beispiel für eine Antwort, wenn die Bucket-Versionierung aktiviert ist.

```
HTTP/1.1 100 Continue
HTTP/1.1 200 OK
x-amz-id-2: LriYPLdmOdAiIfgSm/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
```

Verwandte Ressourcen

- [CopyObject](#)
- [POST Object](#)
- [GetObject](#)

POST Object-Wiederherstellung

Description

Dieser Vorgang führt die folgenden Arten von Anfragen aus:

- `select`— Führt eine Auswahlabfrage für ein archiviertes Objekt durch
- `restore an archive`— Stellt ein archiviertes Objekt wieder her

Um diesen Vorgang verwenden zu können, müssen Sie über die Berechtigungen zum Ausführen der `s3:GetObject` Aktionen `s3:RestoreObject` und verfügen. Der Bucket-Besitzer hat diese Berechtigung standardmäßig und kann diese Berechtigung anderen gewähren. Weitere Informationen zu Berechtigungen finden Sie unter [Berechtigungen im Zusammenhang mit Bucket-Unterressourcenoperationen](#) und [Verwaltung der Zugriffsberechtigungen für Ihre Amazon S3-Ressourcen](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Archive mit Select Requests abfragen

Sie verwenden einen ausgewählten Anforderungstyp, um SQL-Abfragen an archivierten Objekten durchzuführen. Die archivierten Objekte, die mit der Auswahlanforderung abgefragt werden, müssen als unkomprimierte Dateien mit kommagetrennten Werten (CSV) formatiert werden. Sie können Abfragen und benutzerdefinierte Analysen für Ihre archivierten Daten ausführen, ohne Ihre Daten auf einer höheren Amazon S3-Ebene wiederherstellen zu müssen. Eine Übersicht über ausgewählte Anfragen finden Sie unter [Abfragen archivierter Objekte](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Gehen Sie wie folgt vor, wenn Sie eine Auswahlanfrage stellen:

- Definieren Sie einen Ausgabeort für die Ausgabe der Auswahlabfrage. Dies muss ein Amazon S3-Bucket sein, der dem Bucket entspricht AWS-Region , der das Archivobjekt enthält, das abgefragt wird. Der AWS-Konto , der den Job initiiert, muss über Schreibberechtigungen in den S3-Bucket verfügen. Sie können die Speicherklasse und die Verschlüsselung für die im Bucket gespeicherten Ausgabeobjekte angeben. Weitere Informationen zur Ausgabe finden Sie unter [Abfragen archivierter Objekte](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Weitere Informationen zur S3 Struktur im Anforderungstext finden Sie im Folgenden:

- [PutObject](#)
- [Verwaltung des Zugriffs mit ACLs](#) im Amazon Simple Storage Service-Benutzerhandbuch

- [Schutz von Daten mithilfe von Server-Side Verschlüsselung](#) im Amazon Simple Storage Service-Benutzerhandbuch
- Definieren Sie den SQL-Ausdruck für die SELECT Art der Wiederherstellung Ihrer Abfrage in der `SelectParameters` Struktur des Anforderungstexts. Sie können Ausdrücke wie die folgenden Beispiele verwenden.
- Der folgende Ausdruck gibt alle Datensätze aus dem angegebenen Objekt zurück.

```
SELECT * FROM Object
```

- Unter der Annahme, dass Sie keine Überschriften für die im Objekt gespeicherten Daten verwenden, können Sie Spalten mit positionellen Überschriften angeben.

```
SELECT s._1, s._2 FROM Object s WHERE s._3 > 100
```

- Wenn Sie Header haben und diese `fileHeaderInfo` in der CSV Struktur im Anforderungstext auf `setUse` setzen, können Sie Header in der Abfrage angeben. (Wenn Sie das Feld `fileHeaderInfo` auf `IGNORE` setzen, wird die erste Zeile für die Abfrage übersprungen.) Sie können Ordinalpositionen nicht mit Headerspaltennamen mischen.

```
SELECT s.Id, s.FirstName, s.SSN FROM S3Object s
```

Weitere Informationen zur Verwendung von SQL mit Amazon Glacier Select Restore finden Sie unter [SQL-Referenz für Amazon S3 Select und Amazon Glacier Select](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Wenn Sie eine Auswahlanfrage stellen, können Sie auch Folgendes tun:

- Um Ihre Abfragen zu beschleunigen, geben Sie die `Expedited` Stufe an. Weitere Informationen zu Ebenen finden Sie unter „Archive wiederherstellen“ weiter unten in diesem Thema.
- Geben Sie Details zum Datenserialisierungsformat sowohl des abgefragten Eingabeobjekts als auch der Serialisierung der Abfrageergebnisse an. `CSV-encoded`

Im Folgenden finden Sie weitere wichtige Fakten zur Auswahlfunktion:

- Die Ausgabeergebnisse sind neue Amazon S3-Objekte. Im Gegensatz zu Archivabrufen werden sie so lange gespeichert, bis sie explizit gelöscht werden — manuell oder im Rahmen einer Lebenszyklusrichtlinie.

- Sie können mehr als eine Select-Anfrage für dasselbe Amazon S3-Objekt stellen. Amazon S3 dedupliziert keine Anfragen. Vermeiden Sie es daher, doppelte Anfragen zu stellen.
- Amazon S3 akzeptiert eine Auswahlanfrage, auch wenn das Objekt bereits wiederhergestellt wurde. Eine Auswahlanforderung gibt keine Fehlerantwort zurück⁴⁰⁹.

Archive wiederherstellen

Objekte in den Speicherklassen GLACIER und DEEP_ARCHIVE werden archiviert. Um auf ein archiviertes Objekt zuzugreifen, müssen Sie zunächst eine Wiederherstellungsanforderung initiieren. Dadurch wird eine temporäre Kopie des archivierten Objekts wiederhergestellt. In einer Wiederherstellungsanforderung geben Sie an, wie viele Tage die wiederhergestellte Kopie existieren soll. Nach Ablauf des angegebenen Zeitraums löscht Amazon S3 die temporäre Kopie, das Objekt bleibt jedoch in der Speicherklasse GLACIER oder DEEP_ARCHIVE archiviert, aus der das Objekt wiederhergestellt wurde.

Um eine bestimmte Objektversion wiederherzustellen, können Sie eine Versions-ID angeben. Wenn Sie keine Versions-ID angeben, stellt Amazon S3 die aktuelle Version wieder her.

Die Zeit, die für die Fertigstellung von Wiederherstellungsaufträgen benötigt wird, hängt davon ab, aus welcher Speicherklasse das Objekt wiederhergestellt wird und welche Datenzugriffsebene Sie angeben.

Wenn Sie ein archiviertes Objekt wiederherstellen (oder eine Auswahlanfrage verwenden), können Sie im `Type` Element des Anforderungstextes eine der folgenden Optionen für die Datenzugriffsebene angeben:

- **Expedited** - Mit beschleunigten Abrufen können Sie schnell auf Ihre in der Speicherklasse GLACIER gespeicherten Daten zugreifen, wenn gelegentlich dringende Anfragen für eine Teilmenge von Archiven erforderlich sind. Für alle archivierten Objekte mit Ausnahme der größten (über 250 MB) werden Daten, auf die mit beschleunigten Abrufen zugegriffen wird, in der Regel innerhalb von 1—5 Minuten verfügbar gemacht. Die bereitgestellte Kapazität stellt sicher, dass für Expedited-Abrufe Abrufkapazität verfügbar ist, wenn Sie sie benötigen. Beschleunigte Abrufe und bereitgestellte Kapazität sind für die Speicherklasse DEEP_ARCHIVE nicht verfügbar.
- **Standard** - Standard-Abrufe ermöglichen Ihnen den Zugriff auf Ihre archivierten Objekte innerhalb einiger Stunden. Dies ist die Standardoption für die GLACIER- und DEEP_ARCHIVE-Abrufanforderungen, die die Abrufoption nicht angeben. Standardabrufe werden in der Regel innerhalb von 3 bis 5 Stunden ab der Speicherklasse GLACIER und in der Regel innerhalb von 12 Stunden ab der Speicherklasse DEEP_ARCHIVE abgeschlossen.

- **Bulk**- Massenabrufe sind die kostengünstigste Abruffoption von Amazon Glacier, mit der Sie große Datenmengen, sogar Petabyte, kostengünstig an einem Tag abrufen können. Massenabrufe werden in der Regel innerhalb von 5 bis 12 Stunden ab der Speicherklasse GLACIER und in der Regel innerhalb von 48 Stunden ab der Speicherklasse DEEP_ARCHIVE abgeschlossen.

Weitere Informationen zu den Optionen zum Abrufen von Archiven und zur bereitgestellten Kapazität für den Expedited Datenzugriff finden Sie unter [Wiederherstellung archivierter Objekte](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Sie können das Amazon S3-Wiederherstellungsgeschwindigkeitsupgrade verwenden, um die Wiederherstellungsgeschwindigkeit während der Ausführung auf eine höhere Geschwindigkeit zu ändern. Sie erhöhen die Geschwindigkeit einer laufenden Wiederherstellung, indem Sie eine weitere Wiederherstellungsanforderung an dasselbe Objekt senden und dabei ein neues Tier Anforderungselement festlegen. Bei der Ausgabe einer Anfrage für ein Upgrade der Wiederherstellungsstufe müssen Sie eine Stufe wählen, die schneller als die von der Wiederherstellung in Bearbeitung verwendete Stufe ist. Sie dürfen keine anderen Parameter ändern, beispielsweise das Anforderungselement Days. Weitere Informationen finden Sie unter [Upgrading the Speed of an In-Progress Restore](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Um den Status der Objektwiederherstellung abzurufen, können Sie eine HEAD Anfrage senden. Operationen geben in der Antwort den `x-amz-restore` Header zurück, der Informationen über den Wiederherstellungsstatus enthält. Sie können Amazon S3-Ereignisbenachrichtigungen verwenden, um Sie zu benachrichtigen, wenn eine Wiederherstellung eingeleitet oder abgeschlossen wird. Weitere Informationen finden Sie unter [Konfiguration von Amazon S3-Ereignisbenachrichtigungen](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Nach der Wiederherstellung eines archivierten Objekts können Sie den Wiederherstellungszeitraum aktualisieren, indem Sie die Anfrage mit einem neuen Zeitraum erneut ausstellen. Amazon S3 aktualisiert den Wiederherstellungszeitraum relativ zur aktuellen Uhrzeit und berechnet nur die Anfrage — es fallen keine Gebühren für die Datenübertragung an. Sie können den Wiederherstellungszeitraum nicht aktualisieren, wenn Amazon S3 Ihre aktuelle Wiederherstellungsanforderung für das Objekt aktiv verarbeitet.

Wenn Ihr Bucket über eine Lebenszykluskonfiguration mit einer Regel verfügt, die eine Ablaufaktion beinhaltet, überschreibt der Ablauf des Objekts die Lebensdauer, die Sie in einer Wiederherstellungsanforderung angeben. Wenn Sie beispielsweise eine Objektkopie für 10 Tage wiederherstellen, das Objekt aber voraussichtlich in 3 Tagen abläuft, löscht Amazon S3 das Objekt in 3 Tagen. Weitere Informationen zur Lebenszykluskonfiguration finden Sie unter

[PutBucketLifecycleConfiguration](#) [Object Lifecycle Management](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Anforderungen

Syntax

```
POST /ObjectName?restore&versionId=VersionID HTTP/1.1
Host: BucketName.s3.amazonaws.com
Date: date
Authorization: authorization string (see Anfragen authentifizieren \(AWS Signatur \(Version 4\)\))
Content-MD5: MD5

request body
```

Note

Die Syntax zeigt einige der Anforderungsheader. Eine vollständige Liste finden Sie unter „Anforderungsheader“ weiter unten in diesem Thema.

Anforderungsparameter

Bei dieser Implementierung der Operation werden keine Anforderungsparameter verwendet.

Anforderungsheader

Name	Description	Erforderlich
Content-MD5	Der Base64-kodierte 128-Bit-MD5-Digest der Daten. Sie müssen diesen Header zur Überprüfung der Nachrichtenintegrität verwenden, um sicherzustellen, dass der Anforderungstext bei der Übertragung nicht beschädigt wurde. Weitere Informationen finden Sie unter RFC 1864. Typ: Zeichenfolge	Ja

Name	Description	Erforderlich
	Standard: keiner	

Anfrage-Elemente

Im Folgenden finden Sie ein XML-Beispiel für einen Anforderungstext für die Wiederherstellung eines Archivs.

```
<RestoreRequest>
  <Days>2</Days>
  <GlacierJobParameters>
    <Tier>Bulk</Tier>
  </GlacierJobParameters>
</RestoreRequest>
```

In der folgenden Tabelle wird das XML für die Archivwiederherstellung im Anforderungstext erläutert.

Name	Description	Erforderlich
RestoreRequest	Container für Wiederherstellungsinformationen. Typ: Container	Ja
Days	Lebensdauer der wiederhergestellten (aktiven) Kopie. Die Mindestanzahl an Tagen, für die Sie ein Objekt aus Amazon Glacier wiederherstellen können, ist 1. Nachdem die Objektkopie die angegebene Lebensdauer erreicht hat, entfernt Amazon S3 sie aus dem Bucket. Wenn Sie ein Archiv wiederherstellen, ist dieses Element erforderlich. Verwenden Sie dieses Element nicht mit einer SELECT Art von Anfrage. Typ: Positive Ganzzahl Vorfahren: RestoreRequest	Ja, wenn ein Archiv wiederhergestellt wird

Name	Description	Erforderlich
GlacierJobParameters	Container für Auftragsparameter der S3 Glacier-Speicherkategorie. Verwenden Sie dieses Element nicht mit einer SELECT Art von Anfrage. Typ: Container Vorfahren: RestoreRequest	Nein
Tier	Die Datenzugriffsebene, die bei der Wiederherstellung des Archivs verwendet werden soll. Standard ist die Standardinstellung. Type: Zähler Zulässige Werte: Expedited Standard Bulk Vorfahren: GlacierJobParameters	Nein

Das folgende XML ist der Anforderungstext für eine Auswahlabfrage für ein archiviertes Objekt:

```
<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>
  </SelectParameters>
</RestoreRequest>
```

```

        </CSV>
    </InputSerialization>
    <OutputSerialization>
        <CSV>
            <QuoteFields>ASNEEDED</QuoteFields>
            <RecordDelimiter>\n</RecordDelimiter>
            <FieldDelimiter>,</FieldDelimiter>
            <QuoteCharacter>"</QuoteCharacter>
            <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        </CSV>
    </OutputSerialization>
</SelectParameters>
<OutputLocation>
    <S3>
        <BucketName>Name of bucket</BucketName>
        <Prefix>Key prefix</Prefix>
        <CannedACL>Canned ACL string</CannedACL>
        <AccessControlList>
            <Grantee>
                <Type>Grantee Type</Type>
            <ID>Grantee identifier</ID>
            <URI>Grantee URI</URI>
            <Permission>Granted permission</Permission>
            <DisplayName>Display Name</DisplayName>
            <EmailAddress>email</EmailAddress>
        </Grantee>
    </AccessControlList>
    <Encryption>
        <EncryptionType>Encryption type</EncryptionType>
        <KMSKeyId>KMS Key ID</KMSKeyId>
        <KMSContext>Base64-encoded JSON<KMSContext>
    </Encryption>
    <UserMetadata>
        <MetadataEntry>
            <Name>Key</Name>
            <Value>Value</Value>
        </MetadataEntry>
    </UserMetadata>
    <Tagging>
        <TagSet>
            <Tag>
                <Key>Tag name</Key>
                <Value>Tag value</Value>
            </Tag>

```

```
        </TagSet>
      </Tagging>
      <StorageClass>Storage class</StorageClass>
    </S3>
    </OutputLocation>
  </RestoreRequest>
```

In den folgenden Tabellen wird das XML für einen SELECT Wiederherstellungstyp im Anforderungstext erläutert.

Name	Description	Erforderlich
RestoreRequest	Container für Wiederherstellungsinformationen. Typ: Container	Ja
Tier	Die Datenzugriffsebene, die bei der Wiederherstellung des Archivs verwendet werden soll. Standard ist die Standard Einstellung. Type: Zähler Zulässige Werte: Expedited Standard Bulk Vorfahren: RestoreRequest	Nein
Description	Die optionale Beschreibung der Anfrage. Typ: Zeichenfolge Vorfahren: RestoreRequest	Nein
SelectParameters	Beschreibt die Parameter für die Job-Anfrage zur Auswahl. Typ: Container Vorfahren: RestoreRequest	Ja, wenn der Anfragetyp lautet SELECT

Name	Description	Erforderlich
OutputLocation	Beschreibt den Ort, an dem die Ergebnisse der Auswahlwiederherstellungsanforderung empfangen werden. Typ: Container für Amazon S3 Vorfahren: RestoreRequest	Ja, wenn der Anfragetyp lautet SELECT

Das **SelectParameters** Containerelement enthält die folgenden Elemente.

Name	Description	Erforderlich
Expression	Der SQL-Ausdruck. Beispiel: - Der folgende SQL-Ausdruck ruft die erste Spalte der Daten aus dem im CSV-Format gespeicherten Objekt ab: <pre>SELECT s._1 FROM Object s</pre> - Der folgende SQL-Ausdruck gibt alles aus dem Objekt zurück: <pre>SELECT * FROM Object</pre> Typ: Zeichenfolge Vorfahren: SelectParameters	Ja
ExpressionType	Identifiziert den Ausdruckstyp. Typ: Zeichenfolge Zulässige Werte: SQL Vorfahren: SelectParameters	Ja

Name	Description	Erforderlich
InputSerialization	Beschreibt das Serialisierungsformat des Objekts. Typ: Container für CSV Vorfahren: <code>SelectParameters</code>	Ja
OutputSerialization	Beschreibt, wie die Ergebnisse des Auswahlauftrags serialisiert werden. Typ: Container für CSV Vorfahren: <code>SelectParameters</code>	Ja

Das CSV-Containerelement im **InputSerialization** Element enthält die folgenden Elemente.

Name	Description	Erforderlich
RecordDelimiter	Ein einzelnes Zeichen, das verwendet wird, um einzelne Datensätze in der Eingabe zu trennen. Anstelle des Standardwerts können Sie ein beliebiges Trennzeichen angeben. Typ: Zeichenfolge Standard: <code>\n</code> Vorfahren: CSV	Nein
FieldDelimiter	Ein einzelnes Zeichen, das verwendet wird, um einzelne Felder in einem Datensatz zu trennen. Sie können ein beliebiges Trennzeichen angeben. Typ: Zeichenfolge Standard: <code>,</code>	Nein

Name	Description	Erforderlich
	Vorfahren: CSV	
QuoteCharacter	Ein einzelnes Zeichen, das als Escape-Zeichen verwendet wird, wenn das Feldtrennzeichen Teil des Werts ist. Betrachten Sie dieses Beispiel in einer CSV-Datei: "a, b" Wenn Sie den Wert in Anführungszeichen setzen, wird dieser Wert zu einem einzelnen Feld. Wenn Sie keine Anführungszeichen verwenden, ist das Komma ein Feldtrennzeichen (was es zu zwei getrennten Feldwerten macht, a und b). Typ: Zeichenfolge Standard: " Vorfahren: CSV	Nein
QuoteEscapeCharacter	Ein einzelnes Zeichen, das verwendet wird, um das Anführungszeichen innerhalb eines bereits mit Escape-Zeichen versehenen Werts zu umgehen. Der Wert "" "" a , b "" "" wird beispielsweise als geparst. " a , b " Typ: Zeichenfolge Standard: " Vorfahren: CSV	Nein

Name	Description	Erforderlich
FileHeaderInfo	Beschreibt die erste Zeile in den Eingabedaten. Es ist einer der ENUM-Werte. • NONE: Die erste Zeile ist kein Header. • IGNORE: Die erste Zeile ist eine Überschrift, aber Sie können die Header-Werte nicht verwenden, um die Spalte in einem Ausdruck anzugeben. Sie können die Spaltenposition (z. B. <code>_1</code>, <code>_2</code>,...) verwenden, um die Spalte (<code>SELECT s._1 FROM OBJECT s</code>) anzugeben. • Use: Die erste Zeile ist eine Überschrift, und Sie können den Header-Wert verwenden, um eine Spalte in einem Ausdruck (<code>SELECT "name" FROM OBJECT</code>) zu identifizieren. Type: Zähler Zulässige Werte: NONE USE IGNORE Vorfahren: CSV	Nein
Comments	Ein einzelnes Zeichen, mit dem angegeben wird, dass eine Zeile ignoriert werden soll, wenn das Zeichen am Anfang der Reihe vorhanden ist. Sie können ein beliebiges Zeichen angeben, um eine Kommentarzeile zu kennzeichnen. Typ: Zeichenfolge Vorfahren: CSV	Nein

Das CSV-Containerelement (in den **OutputSerialization** Elementen) enthält die folgenden Elemente.

Name	Description	Erforderlich
QuoteFields	Gibt an, ob Ausgabefelder in Anführungszeichen gesetzt werden sollen. • ALWAYS: Verwenden Sie für Ausgabefelder immer Anführungszeichen. • ASNEEDED: Verwenden Sie bei Bedarf Anführungszeichen für Ausgabefelder. Type: Zähler Zulässige Werte: ALWAYS ASNEEDED Standard: AsNeeded Vorfahren: CSV	Nein
RecordDelimiter	Ein einzelnes Zeichen, das verwendet wird, um einzelne Datensätze in der Ausgabe zu trennen. Anstelle des Standardwerts können Sie ein beliebiges Trennzeichen angeben. Typ: Zeichenfolge Standard: \n Vorfahren: CSV	Nein
FieldDelimiter	Ein einzelnes Zeichen, das verwendet wird, um einzelne Felder in einem Datensatz zu trennen. Sie können ein beliebiges Trennzeichen angeben.	Nein

Name	Description	Erforderlich
	Typ: Zeichenfolge Standard: , Vorfahren: CSV	
QuoteCharacter	Ein einzelnes Zeichen, das als Escape-Zeichen verwendet wird, wenn das Feldtrennzeichen Teil des Werts ist. Wenn der Wert beispielsweise lautet a , b, setzt Amazon S3 diesen Feldwert wie folgt in Anführungszeichen: " a , b " Typ: Zeichenfolge Standard: " Vorfahren: CSV	Nein
QuoteEscapeCharacter	Ein einzelnes Zeichen, das verwendet wird, um das Anführungszeichen innerhalb eines bereits mit Escape-Zeichen versehenen Werts zu umgehen. Wenn der Wert beispielsweise lautet " a , b " , setzt Amazon S3 den Wert wie folgt in Anführungszeichen: "" " a , b "" Typ: Zeichenfolge Vorfahren: CSV	Nein

Das S3-Containerelement (im **OutputLocation** Element) enthält die folgenden Elemente.

Name	Description	Erforderlich
AccessControlList	Eine Liste von Zuweisungen, die den Zugriff auf die gestaffelten Ergebnisse steuern. Typ: Behälter für Grant	Nein

Name	Description	Erforderlich
	Vorfahren: S3	
BucketName	Der Name des S3-Buckets, in dem die ausgewählten Wiederherstellungsergebnisse gespeichert werden. Der Bucket muss sich in derselben AWS-Region wie der Bucket befinden, der das Eingabeobjekt enthält. Typ: Zeichenfolge Vorfahren: S3	Ja
CannedACL	Die vordefinierte Zugriffskontrollliste (ACL), die auf die ausgewählten Wiederherstellungsergebnisse angewendet werden soll. Typ: Zeichenfolge Zulässige Werte: <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> Vorfahren: S3	Nein
Encryption	Enthält Verschlüsselungsinformationen für die gespeicherten Ergebnisse. Typ: Container für Verschlüsselung Vorfahren: S3	Nein

Name	Description	Erforderlich
Prefix	Das Präfix, das den ausgewählten Wiederherstellungsergebnissen vorangestellt wird. Die maximale Länge für das Präfix beträgt 512 Byte. Typ: Zeichenfolge Vorfahren: S3	Ja
StorageClass	Die Speicherklasse, die zum Speichern der Ergebnisse der ausgewählten Anfrage verwendet wird. Typ: Zeichenfolge Zulässige Werte: STANDARD REDUCED_REDUNDANCY STANDARD_IA ONEZONE_IA Vorfahren: S3	Nein
Tagging	Container für Tag-Informationen. Typ: Tag-Struktur Vorfahren: S3	Nein
UserMetadata	Enthält eine Liste von Metadaten, die zusammen mit den ausgewählten Wiederherstellungsergebnissen gespeichert werden sollen. Typ: MetadataEntry Struktur Vorfahren: S3	Nein

Das Grantee-Containerelement (im **AccessControlList** Element) enthält die folgenden Elemente.

Name	Description	Erforderlich
DisplayName	Der Anzeigename des Empfängers Typ: Zeichenfolge Vorfahren: Grantee	Nein
EmailAddress	Die E-Mail-Adresse des Empfängers Typ: Zeichenfolge Vorfahren: Grantee	Nein
ID	Die kanonische Benutzer-ID des Empfängers Typ: Zeichenfolge Vorfahren: Grantee	Nein
Type	Der Typ des Empfängers Typ: Zeichenfolge Vorfahren: Grantee	Nein
URI	Die URI der Empfängergruppe Typ: Zeichenfolge Vorfahren: Grantee	Nein
Permission	Genehmigung erteilt. Typ: Zeichenfolge	Nein

Name	Description	Erforderlich
	Vorfahren: Grantee	

Das Encryption-Containerelement (in S3) enthält die folgenden Elemente.

Name	Description	Erforderlich
EncryptionType	Der serverseitige Verschlüsselungsalgorithmus, der beim Speichern von Auftragsergebnissen verwendet wird. Standardmäßig wird keine Verschlüsselung verwendet. Typ: Zeichenfolge Gültige Werte <code>aws:kms</code> <code>AES256</code> Vorfahren: Encryption	Nein
KMSContext	Optional. Wenn der Verschlüsselungstyp lautet <code>aws:kms</code>, können Sie diesen Wert verwenden, um den Verschlüsselungskontext für die ausgewählten Wiederherstellungsergebnisse anzugeben. Typ: Zeichenfolge Vorfahren: Verschlüsselung	Nein
KMSKeyId	Die AWS Key Management Service (AWS KMS) -Schlüssel-ID, die für die Objektverschlüsselung verwendet werden soll. Typ: Zeichenfolge Vorfahren: Encryption	Nein

Das **TagSet** Containerelement (im Tagging-Element) enthält das folgende Element.

Name	Description	Erforderlich
Tag	Enthält Schlagworte. Typ: Container Vorfahren: TagSet	Nein

Das Tag-Containerelement (im **TagSet** Element) enthält die folgenden Elemente.

Name	Description	Erforderlich
Key	Name des Tags. Typ: Zeichenfolge Vorfahren: Tag	Nein
Value	Wert des Tags. Typ: Zeichenfolge Vorfahren: Tag	Nein

Das **MetadataEntry** Containerelement (im **UserMetadata** Element) enthält die folgenden Schlüssel-Wert-Paar-Elemente, die zusammen mit einem Objekt gespeichert werden sollen.

Name	Description	Erforderlich
MetadataKey	Der Metadatenschlüssel. Typ: Zeichenfolge Vorfahren:	Nein

Name	Description	Erforderlich
MetadataEntry	Der Metadatenwert. Typ: Zeichenfolge Vorfahren:	Nein

Antworten

Ein erfolgreicher Vorgang gibt entweder den 202 Accepted Statuscode 200 OK oder zurück.

- Wenn die Objektkopie zuvor nicht wiederhergestellt wurde, kehrt Amazon S3 202 Accepted in der Antwort zurück.
- Wenn die Objektkopie zuvor wiederhergestellt wurde, kehrt Amazon S3 200 OK in der Antwort zurück.

Antwort-Header

Diese Implementierung des Vorgangs verwendet nur Antwort-Header, die den meisten Antworten gemeinsam sind. Weitere Informationen finden Sie unter [Allgemeine](#) Antwortheader.

Antwortelemente

Dieser Vorgang gibt keine Antwortelemente zurück.

Spezielle Fehler

Fehlercode	Description	HTTP-Statuscode	SOAP-Fehlercode-Präfix
RestoreAlreadyInProgress	Die Objektwiederherstellung ist bereits im Gange. (Dieser Fehler gilt nicht für Anfragen vom Typ SELECT.)	409 Conflict (409 Konflikt)	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
GlacierExpeditedRe trievalNotAvailabl e	Beschleunigte Abrufe der S3 Glacier-Speicherklasse sind derzeit nicht verfügbar. Bitte versuchen Sie es später erneut. (Wird zurückgegeben, wenn die Kapazität nicht ausreicht, um die Anfrage zu verarbeiten. Expedited Dieser Fehler gilt nur für Expedited Abrufe und nicht für Bulk Abrufe.) Standard	503	N/A

Beispiele

Stellen Sie mithilfe der Option „Beschleunigter Abruf“ ein Objekt für zwei Tage wieder her

Mit der folgenden Wiederherstellungsanforderung wird mithilfe der Option „Beschleunigter Abruf“ eine Kopie des photo1.jpg Objekts aus Amazon Glacier für einen Zeitraum von zwei Tagen wiederhergestellt.

```
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>
```


Wenn `examplebucket` keine wiederhergestellte Kopie des Objekts vorhanden ist, gibt Amazon S3 die folgende Antwort zurück. `202 Accepted`

```
HTTP/1.1 202 Accepted
x-amz-id-2: GFihv3y6+kE7KG11GEkQhU7/2/CHR3Yb2fCb2S04nxI423Dqwg2XiQ0B/
UZlZYQvPiBlZNRcovw=
x-amz-request-id: 9F341CD3C4BA79E0
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

Wenn eine Kopie des Objekts bereits wiederhergestellt wurde, gibt Amazon S3 eine `200 OK` Antwort zurück und aktualisiert nur die Ablaufzeit der wiederhergestellten Kopie.

Fragen Sie ein Archiv mit einer `SELECT`-Anfrage ab

Im Folgenden finden Sie ein Beispiel für eine Auswahlwiederherstellungsanforderung.

```
POST /object-one.csv?restore HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Date: Sat, 20 Oct 2012 23:54:05 GMT
Authorization: authorization string
Content-Length: content length

<RestoreRequest xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Type>SELECT</Type>
  <Tier>Expedited</Tier>
  <Description>this is a description</Description>
  <SelectParameters>
    <InputSerialization>
      <CSV>
        <FileHeaderInfo>IGNORE</FileHeaderInfo>
        <Comments>#</Comments>
        <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
        <RecordDelimiter>\n</RecordDelimiter>
        <FieldDelimiter>,</FieldDelimiter>
        <QuoteCharacter>"</QuoteCharacter>
      </CSV>
    </InputSerialization>
    <ExpressionType>SQL</ExpressionType>
    <Expression>select * from object</Expression>
    <OutputSerialization>
```

```

    <CSV>
      <QuoteFields>ALWAYS</QuoteFields>
      <QuoteEscapeCharacter>"</QuoteEscapeCharacter>
      <RecordDelimiter>\n</RecordDelimiter>
      <FieldDelimiter>\t</FieldDelimiter>
      <QuoteCharacter>\'</QuoteCharacter>
    </CSV>
  </OutputSerialization>
</SelectParameters>
<OutputLocation>
  <S3>
    <BucketName>example-output-bucket</BucketName>
    <Prefix>test-s3</Prefix>
    <AccessControlList>
      <Grant>
        <Grantee xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:type="AmazonCustomerByEmail">
          <EmailAddress>jane-doe@example.com</EmailAddress>
        </Grantee>
        <Permission>FULL_CONTROL</Permission>
      </Grant>
    </AccessControlList>
    <UserMetadata>
      <MetadataEntry>
        <Name>test</Name>
        <Value>test-value</Value>
      </MetadataEntry>
      <MetadataEntry>
        <Name>other</Name>
        <Value>something else</Value>
      </MetadataEntry>
    </UserMetadata>
    <StorageClass>STANDARD</StorageClass>
  </S3>
</OutputLocation>
</RestoreRequest>

```

Amazon S3 gibt die folgende 202 Accepted Antwort zurück.

```

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-btz3wbBxfMhPykd3jkl-1vZE7w42/
Date: Sat, 20 Oct 2012 23:54:05 GMT
Content-Length: 0
Server: AmazonS3
```

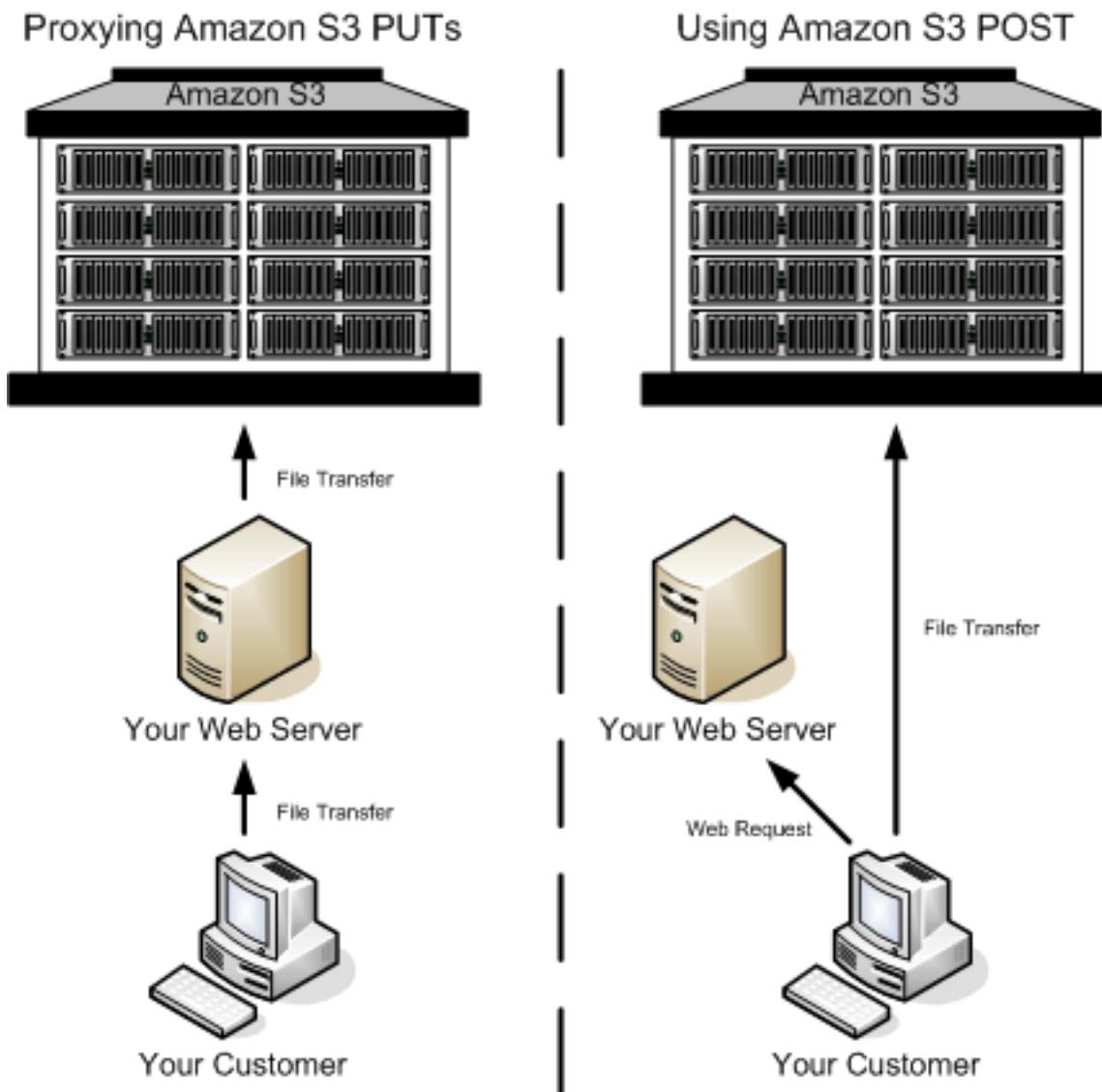
Weitere Infos

- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)
- [SQL-Referenz für Amazon S3 Select und Amazon Glacier Select](#) im Amazon Simple Storage Service-Benutzerhandbuch

Browser-Based Uploads mit HTTP POST

Amazon S3 unterstützt HTTP-POST-Anfragen, sodass Benutzer Inhalte direkt auf Amazon S3 hochladen können. Mithilfe von POST können Endbenutzer Anfragen authentifizieren, ohne Daten über einen sicheren Zwischenknoten leiten zu müssen, der Ihre Anmeldeinformationen schützt. Somit hat HTTP POST das Potenzial, die Latenz zu reduzieren.

Die folgende Abbildung zeigt einen Amazon S3-Upload mithilfe einer POST-Anforderung.



1. Der Benutzer greift über einen Webbrowser auf Ihre Seite zu.
2. Ihre Webseite enthält ein HTML-Formular, das alle Informationen enthält, die der Benutzer benötigt, um Inhalte auf Amazon S3 hochzuladen.
3. Der Benutzer lädt Inhalte über den Webbrowser auf Amazon S3 hoch.

Das Verfahren zum Senden browserbasierter POST-Anfragen ist wie folgt:

1. Erstellen Sie eine Sicherheitsrichtlinie, die Bedingungen festlegt, die einschränken, was Sie in der Anfrage zulassen möchten, z. B. den Bucket-Namen, in den Objekte hochgeladen werden können, und Schlüsselnamenpräfixe, die Sie für das Objekt zulassen möchten, das erstellt wird.

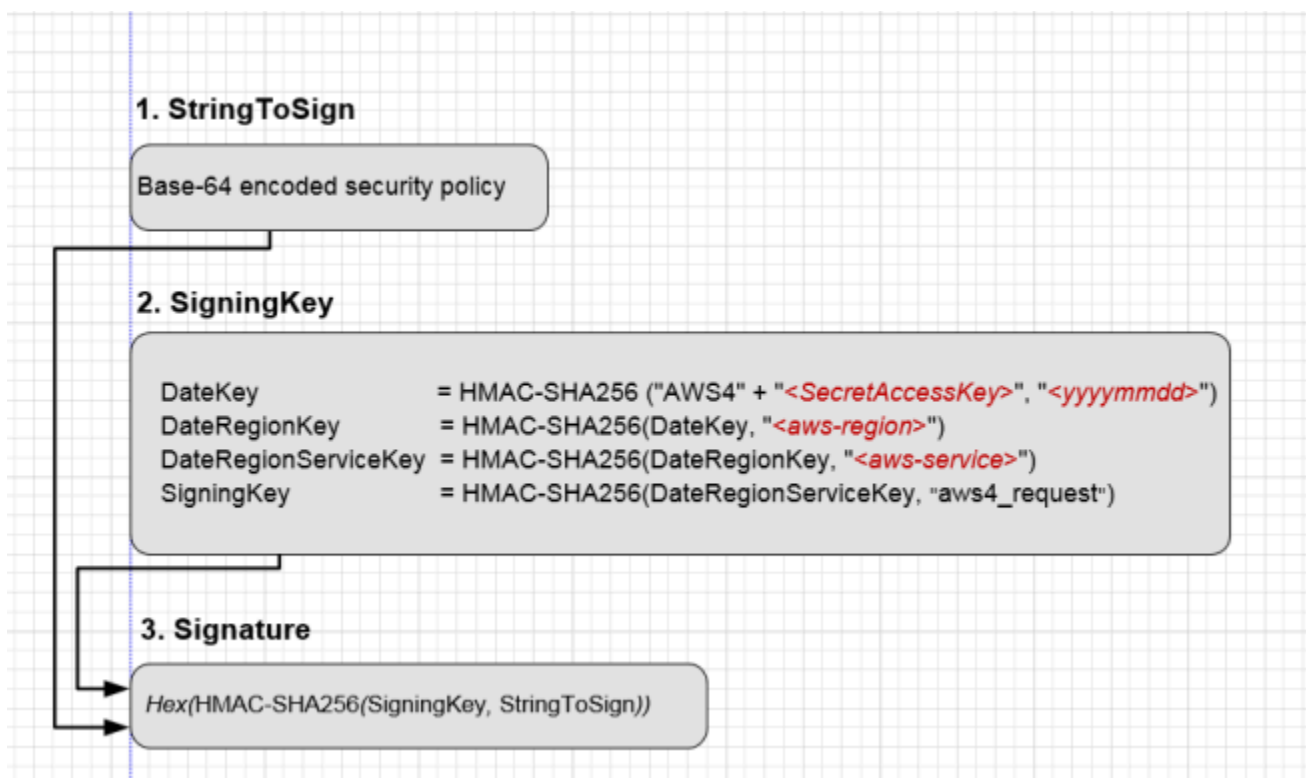
2. Erstellen Sie eine Signatur, die auf der Richtlinie basiert. Für authentifizierte Anfragen muss das Formular eine gültige Signatur und die Richtlinie enthalten.
3. Erstellen Sie ein HTML-Formular, auf das Ihre Benutzer zugreifen können, um Objekte in Ihren Amazon S3-Bucket hochzuladen.

Im folgenden Abschnitt wird beschrieben, wie Sie eine Signatur zur Authentifizierung einer Anfrage erstellen. Informationen zum Erstellen von Formularen und Sicherheitsrichtlinien finden Sie unter [Erstellen eines HTML-Formulars \(mit AWS Signatur \(Version 4\)\)](#).

Eine Signatur berechnen

Für authentifizierte Anfragen muss das HTML-Formular Felder für eine Sicherheitsrichtlinie und eine Signatur enthalten.

- Eine Sicherheitsrichtlinie (siehe [POST-Richtlinie](#)) steuert, was in der Anfrage zulässig ist.
- Die Sicherheitsrichtlinie ist die StringToSign (siehe [Einführung in das Signieren von Anfragen](#)) in Ihrer Signaturberechnung.



Um eine Signatur zu berechnen

1. Erstellen Sie eine Richtlinie mithilfe der UTF-8 Codierung.
2. Konvertiert die UTF-8-encoded Richtlinien-Bytes in Base64. Das Ergebnis ist `derStringToSign`.
3. Erstellen Sie einen Signaturschlüssel.
4. Verwenden Sie den Signaturschlüssel, um den `StringToSign` verwendeten HMAC-SHA256 Signaturalgorithmus zu signieren.

Weitere Informationen zum Erstellen von HTML-Formularen, Sicherheitsrichtlinien und ein Beispiel finden Sie unter den folgenden Links:

- [Erstellen eines HTML-Formulars \(mit AWS Signatur \(Version 4\)\)](#)
- [POST-Richtlinie](#)
- [Beispiel: Browser-Based Upload mit HTTP POST \(Mit AWS Signatur \(Version 4\)\)](#)

Erstellen eines HTML-Formulars (mit AWS Signatur (Version 4))

Themen

- [HTML-Formulardeklaration](#)
- [HTML-Formularfelder](#)

Um Benutzern das Hochladen von Inhalten auf Amazon S3 mithilfe ihrer Browser zu ermöglichen (HTTP-POST-Anfragen), verwenden Sie HTML-Formulare. HTML-Formulare bestehen aus einer Formulardeklaration und Formularfeldern. Die Formulardeklaration enthält allgemeine Informationen zu der Anfrage. Die Formularfelder enthalten detaillierte Anforderungsinformationen.

In diesem Abschnitt wird beschrieben, wie Sie HTML-Formulare erstellen. Ein funktionierendes Beispiel für einen browserbasierten Upload mithilfe von HTTP POST und zugehörigen Signaturberechnungen für die Anforderungsauthentifizierung finden Sie unter [Beispiel: Browser-Based Upload mit HTTP POST \(Mit AWS Signatur \(Version 4\)\)](#).

Das Formular und die Richtlinie müssen UTF-8 codiert sein. Sie können eine UTF-8 Codierung auf das Formular anwenden, indem Sie dies `charset=UTF-8` im `content` Attribut angeben. Das Folgende ist ein Beispiel für die UTF-8 Codierung in der HTML-Überschrift.

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

Es folgt ein Beispiel für die UTF-8 Codierung in einem Anforderungsheader.

```
Content-Type: text/html; charset=UTF-8
```

Note

Die Formulardaten und Grenzen (ohne den Inhalt der Datei) dürfen 20 KB nicht überschreiten.

HTML-Formulardeklaration

Die HTML-Formulardeklaration hat die folgenden drei Attribute:

- **action**— Die URL, die die Anfrage verarbeitet, die auf die URL des Buckets gesetzt werden muss. Wenn der Name Ihres Buckets beispielsweise lautet `examplebucket`, lautet die URL `http://examplebucket.s3.amazonaws.com/`.

Note

Der Schlüsselname wird in einem Formularfeld angegeben.

- **method**— Die Methode muss POST sein.
- **enctype**— Der Gehäusetyp (`enctype`) muss sowohl für Datei-Uploads als auch für Textbereichs-Uploads auf `multipart/form-data` gesetzt sein. Weitere Informationen dazu finden Sie unter RFC `enctype` 1867. <http://www.ietf.org/rfc/rfc1867.txt>

Dies ist eine Formulardeklaration für den Bucket. `examplebucket`

```
<form action="http://examplebucket.s3.amazonaws.com/" method="post"
  enctype="multipart/form-data">
```

HTML-Formularfelder

Die folgende Tabelle beschreibt eine Liste von Feldern, die Sie in einem Formular verwenden können. Neben anderen Feldern gibt es ein Signaturfeld, das Sie zur Authentifizierung von Anfragen verwenden können. Es gibt Felder, in denen Sie den Algorithmus für die Signaturberechnung (`x-amz-algorithm`), den Umfang der Anmeldeinformationen (`x-amz-credential`), den Sie zur Generierung des Signaturschlüssels verwendet haben, und das Datum (`x-amz-date`), das zur Berechnung der Signatur verwendet wurde, angeben können. Amazon S3 verwendet diese Informationen, um die Signatur neu zu erstellen. Wenn die Signaturen übereinstimmen, verarbeitet Amazon S3 die Anfrage.

Note

Die Variable `${filename}` wird automatisch durch den Namen der Datei ersetzt, den der Benutzer bereitstellt, und der von allen Formularfeldern anerkannt wird. Wenn der Browser oder Client einen vollständigen oder teilweisen Pfad zur Datei bereitstellt, wird nur der Text verwendet, der auf den letzten Schrägstrich (/) oder Backslash (\) folgt (`C:\Program Files\directory1\file.txt` wird beispielsweise als interpretiert). `file.txt` Wenn keine Datei bzw. kein Dateiname angegeben ist, wird die Variable durch eine leere Zeichenfolge ersetzt.

Wenn Sie keine Elemente angeben, die für authentifizierte Anfragen erforderlich sind, wie z. B. das `policy` Element, wird davon ausgegangen, dass die Anfrage anonym ist und nur dann erfolgreich ist, wenn Sie den Bucket für öffentliches Lesen und Schreiben konfiguriert haben.

Elementname	Description	Erforderlich
<code>acl</code>	Eine Amazon-S3-Zugriffskontrollliste (ACL). Wenn eine ungültige ACL angegeben ist, lehnt Amazon S3 die Anfrage ab. Weitere Informati	Nein

Elementname	Description	Erforderlich
	onen zu ACLs finden Sie unter Amazon S3-ACLs verwenden. Typ: Zeichenfolge Standard: privat Zulässige Werte: private public-read public-read-write aws-exec-read authenticated-read bucket-owner-read bucket-owner-full-control	
Cache-Control	REST-specific Header. Weitere Informationen finden Sie unter PutObject .	Nein
Content-Type		
Content-Disposition		
Content-Encoding		
Expires		
key	Der Schlüsselname des hochgeladenen Objekts. Um den vom Benutzer angegebenen Dateinamen zu verwenden, verwenden Sie die Variable <code>\${filename}</code>. Wenn Sie beispielsweise eine Datei hochladen <code>photo1.jpg</code> und <code>/user/user1/\${filename}</code> als Schlüsselnamen angeben, wird die Datei unter <code>gespeichert/user/user1/photo1.jpg</code> gespeichert. Weitere Informationen finden Sie unter Objektschlüssel und Metadaten im Amazon Simple Storage Service-Benutzerhandbuch.	Ja

Elementname	Description	Erforderlich
<code>policy</code>	Die Base64-kodierte Sicherheitsrichtlinie, die beschreibt, was in der Anfrage zulässig ist. Für authentifizierte Anfragen ist eine Richtlinie erforderlich. Anfragen ohne Sicherheitsrichtlinie werden als anonym betrachtet und sind nur in einem öffentlich beschreibbaren Bucket erfolgreich.	Erforderlich für authentifizierte Anfragen
<code>success_action_redirect</code>	Die URL, zu der der Client nach einem erfolgreichen Upload umgeleitet wird. Wenn nicht angegeben <code>success_action_redirect</code> ist oder Amazon S3 die URL nicht interpretieren kann, gibt Amazon S3 den leeren Dokumenttyp zurück, der im <code>success_action_status</code> Feld angegeben ist. Schlägt der Upload fehl, gibt Amazon S3 einen Fehler zurück und leitet den Benutzer nicht zu einer anderen URL weiter.	Nein

Elementname	Description	Erforderlich
success_action_status	Der Statuscode, der nach erfolgreichem Upload an den Client zurückgegeben wird, falls <code>success_action_redirect</code> nicht angegeben. Gültige Werte sind 200201, oder 204 (Standard). Wenn der Wert auf 200 oder 204 festgelegt ist, gibt Amazon S3 ein leeres Dokument mit dem angegebenen Statuscode zurück. Wenn der Wert auf 201 gesetzt ist, gibt Amazon S3 ein XML-Dokument mit dem Statuscode 201 aus. Informationen zum Inhalt des XML-Dokuments finden Sie unter POST Object. Wenn der Wert nicht gesetzt oder ungültig ist, gibt Amazon S3 ein leeres Dokument mit einem 204-Statuscode zurück.  Note Einige Versionen von Adobe Flash Player können HTTP-Antworten ohne Text nicht korrekt bearbeiten. Zur Unterstützung von Uploads über Adobe Flash empfehlen wir, <code>success_action_status</code> auf 201 zu setzen.	Nein

Elementname	Description	Erforderlich
x-amz-algorithm	Der Signaturalgorithmus, der zur Authentifizierung der Anfrage verwendet wird. Für AWS Signature Version 4 lautet AWS4-HMAC-SHA256 der Wert. Dies ist ein Pflichtfeld, wenn die Anforderung ein Richtliniendokument beinhaltet.	Erforderlich für authentifizierte Anfragen
x-amz-credential	Zusätzlich zu Ihrer Zugangsschlüssel-ID enthält dieses Feld auch Bereichsinformationen zur Identifizierung der Region und des Dienstes, für die die Signatur gültig ist. Dies sollte derselbe Bereich sein, den Sie bei der Berechnung des Signaturschlüssels für die Signaturberechnung verwendet haben. Es handelt sich um eine Zeichenfolge der folgenden Form: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Beispiel: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Für Amazon S3 lautet die AWS-Service-Zeichenfolge. s3 Eine Liste der Amazon aws-region S3-Zeichenfolgen finden Sie unter Regionen und Endpunkte im. Allgemeine AWS-Referenz Dies ist erforderlich, wenn der Anfrage ein Richtliniendokument beigelegt ist.	Erforderlich für authentifizierte Anfragen

Elementname	Description	Erforderlich
x-amz-date	Es ist der Datumswert im ISO8601-Format. Beispiel, 20130728T000000Z . Es ist dasselbe Datum, das Sie bei der Erstellung des Signaturschlüssels verwendet haben (z. B. 20130728). Dies muss auch derselbe Wert sein, den Sie in der Richtlinie (x-amz-date) angeben, die Sie signiert haben. Dies ist erforderlich, wenn der Anfrage ein Richtliniendokument beigefügt ist.	Erforderlich für authentifizierte Anfragen
x-amz-security-token	Ein von Amazon verwendetes Sicherheitstoken DevPay und Anmeldeinformationen für die Sitzung Wenn die Anfrage Amazon verwendet DevPay, sind zwei x-amz-security-token Formularfelder erforderlich: eines für das Produkt-Token und eines für das Benutzer-Token. Weitere Informationen finden Sie unter Verwenden DevPay im Amazon Simple Storage Service-Benutzerhandbuch. Wenn für die Anfrage Sitzungsanmeldedaten verwendet werden, ist ein x-amz-security-token Formular erforderlich. Weitere Informationen finden Sie unter Anfordern von temporären Sicherheitstoken im IAM-Benutzerhandbuch.	Nein

Elementname	Description	Erforderlich
x-amz-signature	(AWS Signaturversion 4) Der HMAC-SHA256 Hash der Sicherheitsrichtlinie. Dies ist ein Pflichtfeld, wenn die Anforderung ein Richtliniendokument beinhaltet.	Erforderlich für authentifizierte Anfragen
x-amz-meta-*	Feldnamen, die mit diesem Präfix beginnen, sind benutzerdefinierte Metadaten. Jeder wird gespeichert und als Satz von Schlüssel-Wert-Paaren zurückgegeben. Amazon S3 validiert oder interpretiert keine benutzerdefinierten Metadaten. Weitere Informationen finden Sie unter PutObject .	Nein
x-amz-*	Weitere x-amz-* Header finden Sie unter POST-Objekt () POST Object .	Nein
file	Datei oder Textinhalt. Die Datei oder der Inhalt muss das letzte Feld des Formulars sein. Sie können nicht mehr als eine Datei zur gleichen Zeit hochladen.	Ja

Bedingte Elemente sind für authentifizierte Anfragen erforderlich und für anonyme Anfragen optional.

Nachdem Sie nun wissen, wie Formulare erstellt werden, können Sie als Nächstes eine Sicherheitsrichtlinie erstellen, die Sie signieren können. Weitere Informationen finden Sie unter [POST-Richtlinie](#).

POST-Richtlinie

Themen

- [Ablauf](#)
- [Übereinstimmung von Bedingungen](#)
- [Bedingungen](#)
- [Escape-Zeichen](#)

Die Richtlinie, die für authentifizierte Anfragen mithilfe von HTTP POST erforderlich ist, ist ein Base64-kodiertes Dokument, das in JavaScript Object Notation (JSON) geschrieben ist UTF-8 und Bedingungen festlegt, die die Anforderung erfüllen muss. Je nachdem, wie Sie Ihr Richtliniendokument entwerfen, können Sie die Zugriffsgranularität pro Upload, pro Benutzer, für alle Uploads oder nach anderen Designs steuern, die Ihren Anforderungen entsprechen.

In diesem Abschnitt wird die POST-Richtlinie beschrieben. Ein Beispiel für Signaturberechnungen mithilfe der POST-Richtlinie finden Sie unter [Beispiel: Browser-Based Upload mit HTTP POST \(Mit AWS Signatur \(Version 4\)\)](#).

Note

Obwohl das Richtliniendokument optional ist, empfehlen wir dringend, eines zu verwenden, um zu kontrollieren, was in der Anfrage zulässig ist. Wenn Sie den Bucket öffentlich beschreibbar machen, haben Sie keinerlei Kontrolle darüber, welche Benutzer in Ihren Bucket schreiben können.

Das Folgende ist ein Beispiel für ein POST-Richtliniendokument.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"acl": "public-read" },
    {"bucket": "johnsmith" },
    ["starts-with", "$key", "user/eric/"],
  ]
}
```

Die POST-Richtlinie enthält immer die `conditions` Elemente `expiration` und. Die Beispielrichtlinie verwendet zwei Typen für den Abgleich von Bedingungen (exakte Übereinstimmung und beginnt mit Übereinstimmung). In den folgenden Abschnitten werden diese Elemente beschrieben.

Ablauf

Das `expiration` Element gibt das Ablaufdatum und die Uhrzeit der POST-Richtlinie im ISO8601-GMT-Datumsformat an. `2013-08-01T12:00:00.000Z` Gibt beispielsweise an, dass die POST-Richtlinie nach Mitternacht GMT am 1. August 2013 nicht gültig ist.

Übereinstimmung von Bedingungen

In der folgenden Tabelle werden die Typen für den Abgleich von Bedingungen beschrieben, die Sie zur Angabe von POST-Richtlinienbedingungen verwenden können (wie im nächsten Abschnitt beschrieben). Obwohl Sie für jedes Formularfeld, das Sie im Formular angeben, mindestens eine Bedingung angeben müssen, können Sie komplexere Übereinstimmungskriterien erstellen, indem Sie mehrere Bedingungen für ein Formularfeld angeben.

Art der Übereinstimmung mit der Bedingung	Description
Genaue Übereinstimmung	Der Wert des Formularfelds muss mit dem angegebenen Wert übereinstimmen. In diesem Beispiel ist angegeben, dass die ACL auf „public-read“ gesetzt werden muss: <pre>{"acl": "public-read" }</pre> Dieses Beispiel zeigt eine alternative Möglichkeit, um anzuzeigen, dass die ACL auf „public-read“ gesetzt werden muss: <pre>["eq", "\$acl", "public-read"]</pre>
Beginnt mit	

Art der Übereinstimmung mit der Bedingung	Description
	Der Wert muss mit dem angegebenen Wert beginnen. Dieses Beispiel gibt an, dass der Objektschlüssel beginnen muss mit user/user1: <pre data-bbox="375 474 1507 554">["starts-with", "\$key", "user/user1/"]</pre>
Content-Typen In einer Comma-Separated Liste übereinstimmend	Content-Typen Werte für eine starts-with Bedingung, die Kommas enthalten, werden als Listen interpretiert. Jeder Wert in der Liste muss die Bedingung erfüllen, damit die gesamte Bedingung erfüllt ist. Zum Beispiel unter der folgenden Bedingung: <pre data-bbox="375 810 1507 890">["starts-with", "\$Content-Type", "image/"]</pre> Der folgende Wert würde die Bedingung erfüllen: <pre data-bbox="375 999 1507 1079">"image/jpg,image/png,image/gif"</pre> Der folgende Wert würde die Bedingung nicht erfüllen: <pre data-bbox="375 1188 1507 1268">["image/jpg,text/plain"]</pre>  Note Andere Datenelemente als Content-Type werden unabhängig vom Vorhandensein von Kommas als Zeichenfolgen behandelt.

Art der Übereinstimmung mit der Bedingung	Description
Übereinstimmung mit beliebigem Inhalt	Um die POST-Richtlinie so zu konfigurieren, dass jeder Inhalt in einem Formularfeld zulässig ist, verwenden Sie den Befehl <code>starts-with</code> mit einem leeren Wert („“). In diesem Beispiel sind alle Werte zulässig für <code>success_action_redirect</code> : <pre data-bbox="375 617 1507 699">["starts-with", "\$success_action_redirect", ""]</pre>
Angabe von Bereichen	Trennen Sie bei Formularfeldern, die einen Bereich akzeptieren, die obere und untere Grenze durch ein Komma. Dieses Beispiel erlaubt eine Dateigröße von 1 bis 10 MiB: <pre data-bbox="375 932 1507 1014">["content-length-range", 1048576, 10485760]</pre>

Die spezifischen Bedingungen, die in einer POST-Richtlinie unterstützt werden, werden unter [beschrieben](#) [Bedingungen](#).

Bedingungen

`conditions`In einer POST-Richtlinie handelt es sich um ein Array von Objekten, von denen jedes zur Validierung der Anforderung verwendet wird. Sie können diese Bedingungen verwenden, um einzuschränken, was in der Anfrage zulässig ist. Die vorstehenden Richtlinienbedingungen erfordern beispielsweise Folgendes:

- In der Anfrage muss der `johnsmith` Bucket-Name angegeben werden.
- Der Objektschlüsselname muss das `user/eric` Präfix haben.
- Die Objekt-ACL muss auf `gesetzt sein` `public-read`.

Jedes Formularfeld, das Sie in einem Formular angeben (außer `x-amz-signature`, `filepolicy`, und Feldnamen, die ein `x-ignore-` Präfix haben), muss in der Liste der Bedingungen erscheinen.

Note

Alle Variablen im Formular werden vor der Validierung der POST-Richtlinie erweitert. Daher sollten alle Bedingungsübereinstimmungen anhand der erweiterten Formularfelder erfolgen. Angenommen, Sie möchten Ihren Objektschlüsselnamen auf ein bestimmtes Präfix (user/user1) beschränken. In diesem Fall setzen Sie das Schlüsselformularfeld auf `user/user1/${filename}`. Ihre POST-Richtlinie sollte `[ "starts-with", "$key", "user/user1/" ]` lauten (nicht eingeben `[ "starts-with", "$key", "user/user1/${filename}" ]`). Weitere Informationen finden Sie unter [Übereinstimmung von Bedingungen](#).

Die Bedingungen für das Richtliniendokument werden in der folgenden Tabelle beschrieben.

Elementname	Description
acl	Gibt den ACL-Wert an, der beim Senden des Formulars verwendet werden muss. Diese Bedingung unterstützt den exakten Abgleich und den Typ der <code>starts-with</code> Bedingungsübereinstimmung, die im folgenden Abschnitt beschrieben werden.
bucket	Gibt den akzeptablen Bucket-Namen an. Diese Bedingung unterstützt den exakt passenden Zuordnungstyp.
content-length-range	Die zulässige Mindest- und Höchstgröße für den hochgeladenen Inhalt. Diese <code>content-length-range</code> Bedingung unterstützt den Zuordnungstyp.
Cache-Control	REST-specific Überschriften. Weitere Informationen finden Sie unter POST Object .
Content-Type	

Elementname	Description
Content-Disposition	Diese Bedingung unterstützt den exakten Abgleich und den Typ der starts-with Bedingungsübereinstimmung.
Content-Encoding	
Expires	
key	Der zulässige Schlüsselname oder ein Präfix des hochgeladenen Objekts. Diese Bedingung unterstützt den exakten Abgleich und den Typ der starts-with Bedingungsübereinstimmung.
success_action_redirect	Die URL, zu der der Client nach einem erfolgreichen Upload umgeleitet wird.
redirect	
	Diese Bedingung unterstützt die exakte Übereinstimmung und den Typ der starts-with Bedingungsübereinstimmung.
success_action_status	Der Statuscode, der nach erfolgreichem Upload an den Client zurückgegeben success_action_redirect wird, falls nicht angegeben. Diese Bedingung unterstützt den exakten Abgleich.
x-amz-algorithm	Der Signaturalgorithmus, der bei der Signaturberechnung verwendet werden muss. Für AWS Signature Version 4 lautet der Wert <code>AWS4-HMAC-SHA256</code>. Diese Bedingung unterstützt den exakten Abgleich.

Elementname	Description
x-amz-credential	Die Anmeldeinformationen, die Sie zur Berechnung der Signatur verwendet haben. Es enthält Informationen zur Zugriffsschlüssel-ID und zum Geltungsbereich, die die Region und den Dienst identifizieren, für den die Signatur gültig ist. Dabei sollte es sich um denselben Bereich handeln, den Sie bei der Berechnung des Signaturschlüssels für die Signaturberechnung verwendet haben. Es handelt sich um eine Zeichenfolge der folgenden Form: <pre><your-access-key-id> /<date>/<aws-region> /<aws-service> /aws4_request</pre> Beispiel: <pre>AKIAIOSFODNN7EXAMPLE/20130728/us-east-1/s3/aws4_request</pre> Für Amazon S3 lautet die AWS-Service-Zeichenfolge. s3 Eine Liste der Amazon aws-region S3-Zeichenfolgen finden Sie unter Regionen und Endpunkte im. Allgemeine AWS-Referenz Dies ist erforderlich, wenn der Anfrage ein POST-Richtliniendokument beigefügt ist. Diese Bedingung unterstützt den exakten Abgleich.
x-amz-date	Der Datumswert, der in der nach ISO8601 formatierten Zeichenfolge angegeben ist. Beispiel, 20130728T000000Z . Das Datum muss mit dem Datum übereinstimmen, das Sie bei der Erstellung des Signaturschlüssels für die Signaturberechnung verwendet haben. Dies ist erforderlich, wenn der Anfrage ein POST-Richtliniendokument beigefügt ist. Diese Bedingung unterstützt den exakten Abgleich.

Elementname	Description
<code>x-amz-security-token</code>	DevPay Amazon-Sicherheitstoken. Für jede Anfrage, die Amazon verwendet, DevPay sind zwei <code>x-amz-security-token</code> Formularfelder erforderlich: eines für das Produkt-Token und eines für das Benutzer-Token. Daher müssen die Werte durch Kommata voneinander getrennt werden. Wenn beispielsweise das Benutzer-Token <code>eW91dHVizQ==</code> und das Produkt-Token lautet <code>b0hnNVNKNWJlQTA=</code>, setzen Sie den POST-Richtlinieneintrag auf: <code>{ "x-amz-security-token": "eW91dHVizQ==,b0hnNVNKNWJlQTA=" }</code>. Weitere Informationen zu Amazon DevPay finden Sie unter Using DevPay im Amazon Simple Storage Service-Benutzerhandbuch.
<code>x-amz-meta-*</code>	User-specified Metadaten. Diese Bedingung unterstützt den exakten Abgleich und den Typ der <code>starts-with</code> Bedingungsübereinstimmung.
<code>x-amz-*</code>	Weitere <code>x-amz-*</code> Header finden Sie unter POST-Objekt (POST Object). Diese Bedingung unterstützt den exakten Abgleich.

 Note

Wenn Ihr Toolkit weitere Formularfelder hinzufügt (z. B. `Flash-Addsfilename`), müssen Sie diese dem POST-Richtliniendokument hinzufügen. Wenn Sie diese Funktionalität steuern können, setzen Sie `x-ignore-` vor das Feld, sodass Amazon S3 die Funktion ignoriert und künftige Versionen davon unbeeinflusst lässt.

Escape-Zeichen

Zeichen, die in einem POST-Richtliniendokument maskiert werden müssen, werden in der folgenden Tabelle beschrieben.

Escape-Sequenz	Beschreibung
\\	Umgekehrter Schrägstrich
\\$	Dollarsymbol
\b	Backspace
\f	Seitenvorschub
\n	Neue Zeile
\r	Zeilenumschaltung
\t	Horizontaler Tabulator
\V	Vertikaler Tabulator
\ u xxxx	Alle Unicode-Zeichen

Nachdem Sie nun mit Formularen und Richtlinien vertraut sind und wissen, wie das Signieren funktioniert, können Sie ein Beispiel für einen POST-Upload ausprobieren. Sie müssen den Code schreiben, um die Signatur zu berechnen. Das Beispiel enthält ein Beispielformular und eine POST-Richtlinie, mit der Sie Ihre Signaturberechnungen testen können. Weitere Informationen finden Sie unter [Beispiel: Browser-Based Upload mit HTTP POST \(Mit AWS Signatur \(Version 4\)\)](#).

Beispiel: Browser-Based Upload mit HTTP POST (Mit AWS Signatur (Version 4))

Dieser Abschnitt zeigt ein Beispiel für die Verwendung einer HTTP-POST-Anfrage, um Inhalte direkt auf Amazon S3 hochzuladen.

Weitere Informationen zu Signature Version 4 finden Sie unter [Signature Version 4 Signing Process](#).

Hochladen einer Datei auf Amazon S3 mithilfe von HTTP POST

Dieses Beispiel enthält ein Beispiel für eine POST-Richtlinie und ein Formular, das Sie zum Hochladen einer Datei verwenden können. In diesem Thema werden die Beispielrichtlinie und fiktive Anmeldeinformationen verwendet, um Ihnen den Arbeitsablauf und die daraus resultierende Signatur und den Richtlinien-Hash zu zeigen. Sie können diese Daten als Testsuite verwenden, um Ihren Signaturberechnungscode zu verifizieren.

Das Beispiel verwendet die folgenden Beispiel-Anmeldeinformationen für die Signaturberechnungen. Sie können diese Anmeldeinformationen verwenden, um Ihren Signaturberechnungscode zu überprüfen. Sie müssen diese dann jedoch durch Ihre eigenen Anmeldeinformationen ersetzen, wenn Sie Anfragen an senden AWS.

Parameter	Wert
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJa1rXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

Beispiel für eine Richtlinie und ein Formular

Die folgende POST-Richtlinie unterstützt Uploads auf Amazon S3 unter bestimmten Bedingungen.

```
{ "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" }
]
```

Diese POST-Richtlinie legt die folgenden Bedingungen für die Anfrage fest:

- Der Upload muss am 30. Dezember 2015 vor 12:00 Uhr UTC erfolgen.
- Der Inhalt kann nur auf den hochgeladen werdensigv4examplebucket. Der Bucket muss sich in der Region befinden, die Sie im Bereich der Anmeldeinformationen (x-amz-credentialFormularparameter) angegeben haben, da die von Ihnen angegebene Signatur nur innerhalb dieses Bereichs gültig ist.
- Sie können einen beliebigen Schlüsselnamen angeben, der mit user/user1 beginnt. Beispiel, user/user1/MyPhoto.jpg.
- Die ACL muss auf gesetzt seinpublic-read.
- Wenn der Upload erfolgreich ist, wird der Browser des Benutzers zu http://sigv4examplebucket.s3.amazonaws.com/successful_upload.html weitergeleitet.
- Das Objekt muss eine Bilddatei sein.
- Das x-amz-meta-uuid Tag muss auf gesetzt sein14365123651274.
- Die x-amz-meta-tag kann einen beliebigen Wert enthalten.

Das Folgende ist eine Base64-encoded Version dieser POST-Richtlinie. Sie verwenden diesen Wert als Ihre StringToSign Signaturberechnung.

```
eyJhZlZxhwaXJhdGlvbiI6ICIyMDE1LTEyLTMwVDEyOjAwOjAwLjAwMFoiLA0KICAiY29uZGl0aW9ucyI6IFsNCiAgICB7ImJ
```

Wenn Sie copy/paste die vorstehende Richtlinie verwenden, sollte sie Zeilenumbrüche und neue Zeilen enthalten, damit Ihr berechneter Hash diesem Wert entspricht (d. h. ASCII-Text mit CRLF-Zeilenabschlüssen).

Bei Verwendung von Beispiel-Anmeldeinformationen zum Erstellen einer Signatur lautet der Signaturwert wie folgt (bei der Berechnung der Signatur entspricht das Datum dem x-amz-date in der Richtlinie (20151229)):

```
8afdbf4008c03f22c2cd3cdb72e4afbb1f6a588f3255ac628749a66d7f09699e
```

Das folgende Beispielformular gibt die vorhergehende POST-Richtlinie an und unterstützt eine POST-Anforderung an die. `sigv4examplebucket` Copy/paste den Inhalt in einem Texteditor und speichern Sie ihn als `exampleform.html`. Anschließend können Sie mithilfe der Datei `exampleform.html` Bilddateien in den jeweiligen Bucket hochladen. Ihre Anfrage ist erfolgreich, wenn die von Ihnen angegebene Signatur mit der von Amazon S3 berechneten Signatur übereinstimmt.

Note

Sie müssen den Bucket-Namen, die Daten, die Anmeldeinformationen, die Richtlinie und die Signatur mit gültigen Werten aktualisieren, damit dies erfolgreich auf S3 hochgeladen werden kann.

```
<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>
```

Bei den Post-Parametern wird nicht zwischen Groß- und Kleinschreibung unterschieden. Sie können beispielsweise `x-amz-signature` oder `X-Amz-Signature` angeben.

Browser-based lädt mit dem auf Amazon S3 hoch AWS Bibliothek erweitern

In diesem Abschnitt wird beschrieben, wie Sie mithilfe der AWS JavaScript Amplify-Bibliothek Dateien auf Amazon S3 hochladen.

Informationen zur Einrichtung der AWS Amplify-Bibliothek finden Sie unter [AWS Amplify Installation and Configuration](#).

Verwendung der AWS JavaScript Bibliothek erweitern, um Dateien auf Amazon S3 hochzuladen

Das AWS Storage Amplify-Bibliotheksmodul bietet einen einfachen browserbasierten Upload-Mechanismus für die Verwaltung von Benutzerinhalten im öffentlichen oder privaten Amazon S3-Speicher.

Example: AWS Manuelles Setup von Amplify

Das folgende Beispiel zeigt die manuelle Einrichtung für die Verwendung des AWS Amplify Storage-Moduls. Die Standardimplementierung des Speichermoduls verwendet 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: Geben Sie Daten in Amazon S3 ein

Das folgende Beispiel zeigt, wie Daten in Amazon S3 put veröffentlicht werden.

```
Storage.put('test.txt', 'Hello')  
  .then (result => console.log(result))  
  .catch(err => console.log(err));
```

Das folgende Beispiel zeigt, wie Daten in Amazon S3 put privat gespeichert werden.

```
Storage.put('test.txt', 'Private Content', {  
  level: 'private',  
  contentType: 'text/plain'  
})  
  .then (result => console.log(result))  
  .catch(err => console.log(err));
```

Weitere Informationen zur Verwendung des AWS Amplify Storage-Moduls finden Sie unter [AWS Amplify Storage](#).

Weitere Infos

[AWS Amplify Quick Start](#)

Allgemeine Anforderungsheader

In der folgenden Tabelle werden Header beschrieben, die von verschiedenen Typen von Amazon S3-REST-Anfragen verwendet werden können.

Header-Name	Description
Authorization	Die erforderlichen Informationen für die Anforderungsauthentifizierung. Weitere Informationen finden Sie im Amazon Simple Storage Service Developer Guide unter The Authentication Header. Für anonyme Anfragen ist dieser Header nicht erforderlich.
Access-Control-Request-Method	Eine Liste von HTTP-Methoden, die als CORS-Anfrage vor dem Flug gesendet wird. Wenn die CORS-Bewertung vor dem Flug erfolgreich ist, dürfen die angegebenen Methoden in der folgenden CORS-Anfrage verwendet werden.
Content-Length	Länge der Nachricht (ohne Header) gemäß RFC 2616. Dieser Header ist für PUTs und Operationen erforderlich, die XML laden, wie Logging und ACLs.
Content-Type	Der Inhaltstyp der Ressource, falls die Anfrage Inhalt im Hauptteil enthält. Beispiel: text/plain
Content-MD5	Der Base64-kodierte 128-Bit-MD5-Digest der Nachricht (ohne die Header) gemäß RFC 1864. Dieser Header kann zur Überprüfung der Nachrichtenintegrität verwendet werden, um zu überprüfen, ob es sich bei den Daten um dieselben Daten handelt, die ursprünglich gesendet wurden. Obwohl er optional ist, empfehlen wir, den Content-MD5 Mechanismus als durchgängige Integritätsprüfung zu verwenden. Weitere Informationen zur REST-Anforderungsauthentifizierung finden Sie im Amazon Simple Storage Service Developer Guide unter REST-Authentifizierung.

Header-Name	Description
Date	Das Datum, an dem die im Authorization -Header enthaltene Signatur erstellt werden kann. Wenn der Date-Header zum Signieren verwendet werden soll, muss das Format den Grundsätzen von ISO 8601 entsprechen. In diesem Fall wird der x-amz-date -Header nicht benötigt. Wenn x-amz-date vorhanden ist, hat dieser Wert immer Vorrang vor dem Wert des Date-Headers. Wenn der Datum-Header nicht zum Signieren verwendet wird, kann er in einem der vollständigen Datumsformate angegeben werden, die durch RFC 2616, Abschnitt 3.3, festgelegt sind. Das date/time Wed, 01 Mar 2006 12:00:00 GMT ist beispielsweise ein gültiger date/time Header für die Verwendung mit Amazon S3. Wenn der Date-Header zum Signieren verwendet wird, muss er den Grundsätzen von ISO 8601 entsprechen und im Format YYYYMMDD'T'HHMMSS'Z' angegeben werden. Wenn Date angegeben, aber nicht im ISO 8601-Basisformat vorliegt, müssen Sie auch den x-amz-date Header angeben. Wenn Date entsprechend den Grundsätzen von ISO 8601 angegeben ist, reicht diese Angabe zum Signieren von Anforderungen aus und der x-amz-date -Header ist nicht erforderlich. Weitere Informationen finden Sie unter Umgang mit Datumswerten in Signature Version 4 im Amazon Web Services-Glossar.
Expect	Wenn Ihre Anwendung 100-continue verwendet, sendet sie den Anforderungstext erst, wenn sie eine Bestätigung erhält. Wird die Nachricht basierend auf den Header abgewiesen, wird der Rumpf der Meldung nicht gesendet. Dieser Header kann nur verwendet werden, wenn Sie einen Text senden. Gültige Werte: 100-continue

Header-Name	Description
Host	Für Anfragen im Pfadstil lautet der Wert. <code>s3.amazonaws.com</code> Für Anfragen im virtuellen Stil ist der Wert. <code>BucketName.s3.amazonaws.com</code> Weitere Informationen finden Sie im Amazon Simple Storage Service-Benutzerhandbuch unter Virtual Hosting. Dieser Header ist für HTTP 1.1 erforderlich (die meisten Toolkits fügen diesen Header automatisch hinzu); optional für HTTP/1.0 Anfragen.
Origin	Ein Endpunkt, der den Servernamen des ursprünglichen Anforderers angibt.
x-amz-content-sha256	Wenn Sie Signaturversion 4 zur Authentifizierung der Anfrage verwenden, liefert dieser Header einen Hash der Anforderungsnutzlast. Weitere Informationen finden Sie unter Signaturberechnungen für den Autorisierungs-Header: Übertragung der Nutzlast in einem einzigen Block (AWS Signatur (Version 4)). Wenn Sie Objekte in Blöcken hochladen, setzen Sie den Wert auf, <code>STREAMING-AWS4-HMAC-SHA256-PAYLOAD</code> um anzuzeigen, dass die Signatur nur Header abdeckt und keine Nutzlast vorhanden ist. Weitere Informationen finden Sie unter Signaturberechnungen für den Autorisierungs-Header: Übertragung von Nutzdaten in mehreren Blöcken (Chunked Upload) (AWS Signatur (Version 4)).

Header-Name	Description
<code>x-amz-date</code>	Das Datum, an dem die Signatur im Autorisierungs-Header erstellt wurde. Das Datum muss den Grundsätzen von ISO 8601 entsprechen und im Format <code>YYYYMMDD'T'HHMMSS'Z'</code> angegeben werden. Das <code>date/time</code> <code>20170210T120000Z</code> ist beispielsweise für die Verwendung mit Amazon S3 <code>x-amz-date</code> gültig. <code>x-amz-date</code> ist für alle Anfragen optional; es kann verwendet werden, um das Datum zu überschreiben, das für das Signieren von Anfragen verwendet wird. Wenn der Date-Header den Grundsätzen von ISO 8601 entspricht, ist <code>x-amz-date</code> nicht erforderlich. Wenn <code>x-amz-date</code> vorhanden ist, hat dieser Wert immer Vorrang vor dem Wert des Date-Headers. Weitere Informationen finden Sie unter Umgang mit Datumswerten in Signature Version 4 im Amazon Web Services-Glossar.

Header-Name	Description
x-amz-security-token	Dieser Header kann in den folgenden Szenarien verwendet werden: • Um Sicherheitstoken für DevPay Amazon-Operationen bereitzustellen — Jede Anfrage, die Amazon verwendet, DevPay benötigt zwei x-amz-security-token Header: einen für das Produkt-Token und einen für das Benutzer-Token. Wenn Amazon S3 eine authentifizierte Anfrage erhält, vergleicht es die berechnete Signatur mit der bereitgestellten Signatur. Falsch formatierte Header mit mehreren Werten, die zur Berechnung einer Signatur verwendet werden, können zu Authentifizierungsproblemen führen. • Um ein Sicherheitstoken bereitzustellen, wenn Sie temporäre Sicherheitsanmeldeinformationen verwenden — Wenn Sie Anfragen mit temporären Sicherheitsanmeldeinformationen stellen, die Sie von IAM erhalten haben, müssen Sie mithilfe dieses Headers ein Sicherheitstoken bereitstellen. Weitere Informationen zu temporären Sicherheitsanmeldeinformationen finden Sie unter Anfragen https://docs.aws.amazon.com/AmazonS3/latest/userguide/MakingRequests.html stellen. Dieser Header ist für Anfragen erforderlich, die Amazon verwenden, DevPay und für Anfragen, die mit temporären Sicherheitsanmeldeinformationen signiert werden.

Allgemeine Antwortheader

In der folgenden Tabelle werden Antwortheader beschrieben, die den meisten Amazon S3-Antworten gemeinsam sind.

Name	Description
<code>Access-Control-Allow-Credentials</code>	Ein boolescher Wert, der bestimmt, ob der Server zulässt, dass CORS-Anfragen Anmeldeinformationen enthalten. Wenn der <code>Access-Control-Allow-Origin</code> Anforderungsheader auf gesetzt ist, '*' wird der <code>Access-Control-Allow-Credentials</code> Antwortheader weggelassen, andernfalls wird er auf gesetzt, true wenn die CORS-Auswertung erfolgreich ist. Typ: Boolescher Wert Standard: keiner
<code>Access-Control-Allow-Headers</code>	Eine Liste der HTTP-Header, die für Ihre CORS-Anfragen zulässig sind. Der <code>Access-Control-Allow-Headers</code> Antwortheader wird für erfolgreiche CORS-Auswertungen zurückgegeben und gibt explizit alle zulässigen Werte an. <code>Access-Control-Request-Headers</code> Typ: Zeichenfolge Standard: keiner
<code>Access-Control-Allow-Methods</code>	Eine Liste, die angibt, welche HTTP-Methoden zulässig sind. Amazon S3 lässt CORS-Anfragen von erlaubten CORS-Methoden nur zu, wenn die CORS-Bewertung erfolgreich ist. Typ: Zeichenfolge Standard: keiner
<code>Access-Control-Allow-Origin</code>	Der Ort des erlaubten Ursprungs. Amazon S3 sendet den <code>Access-Control-Allow-Origin</code> Antwort-Header nur, wenn die CORS-Bewertung erfolgreich ist. Wenn der Ursprung der Anfrage mit '*' den zulässigen

Name	Description
	Ursprüngen der CORS-Konfiguration übereinstimmt, ' * ' wird der in diesem Antwort-Header anstelle des ursprünglichen Ursprungs zurückgegeben. Typ: Zeichenfolge Standard: keiner
Access-Control-Expose-Headers	Eine Liste, die es einem Server ermöglicht, einen Antwort-Header zu identifizieren, der Anwendungen den Zugriff ermöglicht, wenn die CORS-Auswertung erfolgreich ist. Typ: Zeichenfolge Standard: keiner
Access-Control-Max-Age	Die Zeit in Sekunden, in der Ihr Browser die Antwort auf eine CORS-Pre-Flight-Anfrage zwischenspeichern kann, wie von der Ressource, der HTTP-Methode und dem Ursprung identifiziert. Der Access-Control-Max-Age Antwort-Header wird nur zurückgegeben, wenn die CORS-Auswertung erfolgreich ist. Typ: Ganzzahl Standard: keiner
Vary	Eine Liste, die angibt, von welchen Anforderungsheadern das CORS-Bewertungsergebnis abhängt. Der Vary Antwort-Header wird nur zurückgegeben, wenn die CORS-Auswertung erfolgreich ist. Typ: Zeichenfolge Standard: keiner
Content-Length	Die Länge des Hauptteils der Antwort in Byte. Typ: Zeichenfolge Standard: keiner

Name	Description
Content-Type	Der MIME-Typ des Inhalts. Beispiel, Content-Type: text/html; charset=utf-8 . Typ: Zeichenfolge Standard: keiner
Connection	Ein Wert, der angibt, ob die Verbindung zum Server geöffnet oder geschlossen ist. Type: Zähler Zulässige Werte: open close Standard: keiner
Date	Datum und Uhrzeit der Antwort von Amazon S3, z. B. Mittwoch, 01. März 2006, 12:00:00 GMT. Typ: Zeichenfolge Standard: keiner

Name	Description
ETag	Das Entitäts-Tag (ETag) steht für eine bestimmte Version des Objekts. Das ETag gibt nur Änderungen am Inhalt eines Objekts wieder, nicht an seinen Metadaten. Das ETag kann ein MD5 Digest der Objektdaten sein, muss aber nicht. Ob dies der Fall ist, hängt wie folgt davon ab, wie das Objekt erstellt wurde und wie es verschlüsselt ist: • Objekte, die durch die Copy Operation „AWS-Managementkonsole oder“ oder „PUTObjekt“ erstellt wurden: <code>POST Object</code> • Objekte, die im Klartext vorliegen oder durch serverseitige Verschlüsselung mit von Amazon S3 verwalteten Schlüsseln (SSE-S3) verschlüsselt sind, haben ETags, die eine MD5-Zusammenfassung ihrer Daten sind. • Objekte, die durch serverseitige Verschlüsselung mit vom Kunden bereitgestellten Schlüsseln (SSE-C) oder () -Schlüsseln AWS Key Management Service (AWS KMS) verschlüsselt wurden, haben ETags, die kein MD5-Digest ihrer Objektdaten sind. SSE-KMS • Objekte, die entweder durch den Vorgang Multipart Upload oder Upload Part Copy erstellt wurden, haben ETags, bei denen es sich nicht um MD5-Digests handelt, unabhängig von der Verschlüsselungsmethode. Typ: Zeichenfolge
Server	Der Name des Servers, der die Antwort erstellt hat. Typ: Zeichenfolge Standard: AmazonS3
x-amz-delete-marker	Ein Wert, der angibt, ob das zurückgegebene Objekt eine Löschmarkierung war (<code>true</code><code>false</code>) oder nicht (). Typ: Boolescher Wert Zulässige Werte: <code>true</code> <code>false</code> Standard: <code>false</code>

Name	Description
<code>x-amz-id-2</code>	Ein spezielles Token, das zusammen mit dem <code>x-amz-request-id</code> Header zur AWS Problembehandlung verwendet wird. Informationen zur Support Verwendung dieser Anforderungs-IDs finden Sie unter Problembehandlung bei Amazon S3. Typ: Zeichenfolge Standard: keiner
<code>x-amz-request-id</code>	Ein von Amazon S3 erstellter Wert, der die Anfrage eindeutig identifiziert. Dieser Wert wird zusammen mit dem <code>x-amz-id-2</code> Header verwendet, um Probleme zu AWS beheben. Informationen zur Support Verwendung dieser Anforderungs-IDs finden Sie unter Problembehandlung bei Amazon S3. Typ: Zeichenfolge Standard: keiner
<code>x-amz-server-side-encryption</code>	Der serverseitige Verschlüsselungsalgorithmus, der beim Speichern dieses Objekts in Amazon S3 verwendet wird (z. B. <code>AES256,aws:kms</code>). Zulässige Werte: <code>AES256</code> <code>aws:kms</code>
<code>x-amz-version-id</code>	Die Version des Objekts. Wenn Sie die Versionierung aktivieren, generiert Amazon S3 eine Zufallszahl für Objekte, die einem Bucket hinzugefügt werden. Der Wert ist UTF-8 codiert und für die URL bereit. Wenn Sie PUT ein Objekt in einem Bucket verwenden, in dem die Versionierung unterbrochen wurde, lautet die Versions-ID immer. <code>null</code> Typ: Zeichenfolge Gültige Werte: <code>null</code> eine beliebige URL-ready, UTF-8 kodierte Zeichenfolge Standard: null

Fehlermeldungen

Dieser Abschnitt enthält Referenzinformationen zu Amazon S3-Fehlern.

Note

- Im Allgemeinen werden S3-Bucket-Besitzern Anforderungen mit erfolgreichen Antworten mit Statuscode HTTP 200 OK und Client-Fehlerantworten mit Statuscode HTTP 4XX in Rechnung gestellt. Bucket-Besitzern werden Server-Fehlerantworten mit Statuscode HTTP 5XX, wie etwa Fehler mit Statuscode HTTP 503 Slow Down, nicht in Rechnung gestellt. Weitere Informationen zu S3-Fehlercodes unter HTTP 3XX und zu 4XX Statuscodes, die nicht in Rechnung gestellt werden, finden Sie unter [Abrechnung für Amazon S3-Fehlerantworten](#) im Amazon S3-Benutzerhandbuch. Weitere Informationen zur Abrechnung von Gebühren, wenn Ihr Bucket als Bucket „Requester Pays“ konfiguriert ist, finden Sie im Amazon S3-Benutzerhandbuch [im Amazon S3-Benutzerhandbuch unter](#) So funktionieren die Gebühren des Anforderers.
- Die SOAP-Unterstützung über HTTP ist veraltet, SOAP steht über HTTPS aber noch zur Verfügung. Neue Amazon-S3-Funktionen werden unter SOAP nicht unterstützt. Anstatt SOAP zu verwenden, empfehlen wir, entweder die REST-API oder die AWS SDKs zu verwenden.

Themen

- [REST-Fehlerantworten](#)
- [Liste der Fehlercodes](#)
- [Liste der SELECT-Fehlercodes für den Objekthinhalt](#)
- [Liste der Fehlercodes Replication-related](#)
- [Liste der Tagging-related Fehlercodes](#)
- [Liste der Amazon S3-Fehlercodes im Zusammenhang mit Anmerkungen](#)
- [Liste der Amazon S3-Fehlercodes auf Outposts](#)
- [Liste der Amazon S3 Storage Lens-Fehlercodes](#)
- [Liste der Amazon S3 Object Lambda-Fehlercodes](#)
- [Liste der asynchronen Amazon S3-Fehlercodes](#)

- [Liste der Amazon S3 Access Grants-Fehlercodes](#)
- [Liste der FSx-related Amazon-Fehlercodes](#)
- [Liste der Amazon S3-Tabellen-Fehlercodes](#)
- [Bewährte Methoden für Amazon-S3-Fehler](#)

REST-Fehlerantworten

Wenn ein Fehler auftritt, enthalten die Header-Informationen Folgendes:

- Content-Type: application/xml
- Der entsprechende 3xx-, 4xx- oder 5xx-HTTP-Statuscode

Der Hauptteil der Antwort enthält auch Informationen über den Fehler. Das folgende Beispiel für eine Fehlerantwort zeigt die Struktur der Antwortelemente, die allen REST-Fehlerantworten gemeinsam sind.

```
<?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>
```

In der folgenden Tabelle werden die REST-Fehlerantwortelemente erläutert.

Name	Description
Code	Der Fehlercode ist eine Zeichenfolge, die eine Fehlerbedingung eindeutig identifiziert. Er ist für Programme vorgesehen, die Fehler dem Typ nach erkennen und verarbeiten. Weitere Informationen finden Sie unter Liste der Fehlercodes. Typ: Zeichenfolge Vorfahr: Error
Error	Container für alle Fehlerelemente.

Name	Description
	Typ: Container Vorfahr: Keiner
Message	Die Fehlermeldung enthält eine generische Beschreibung der Fehlerbedingung in englischer Sprache. Sie ist für dafür vorgesehen, von Menschen gelesen zu werden. Einfache Programme zeigen dem Endbenutzer die Meldung direkt an, wenn sie auf eine Fehlerbedingung treffen, die sie nicht kennen, oder die sie nicht verarbeiten können. Komplexere Programme mit einer umfangreichen Fehlerverarbeitung und einer geeigneten Internationalisierung werden die Fehlermeldung eher ignorieren. Typ: Zeichenfolge Vorfahr: Error
RequestId	ID der Anfrage, die mit dem Fehler verknüpft ist. Typ: Zeichenfolge Vorfahr: Error
Resource	Der Bucket oder das Objekt, das an dem Fehler beteiligt ist. Typ: Zeichenfolge Vorfahr: Error

Viele Fehlermeldungen enthalten zusätzliche strukturierte Daten, die von einem Entwickler gelesen und interpretiert werden, der Programmfehler diagnostiziert. Wenn Sie beispielsweise einen Content-MD5 Header mit einer REST-PUT-Anfrage senden, die nicht mit dem auf dem Server berechneten Digest übereinstimmt, erhalten Sie eine `BadDigest` Fehlermeldung. Die Fehlerantwort enthält als Detailelemente auch den Digest, den der Server berechnet hat, und den Digest, den Sie dem Server mitgeteilt haben, zu erwarten. Während der Entwicklung können Sie diese Informationen nutzen, um den Fehler zu diagnostizieren. In der Produktion könnte ein ordnungsgemäß funktionierendes Programm diese Informationen in seinem Fehlerprotokoll enthalten.

Informationen zu allgemeinen Antwortelementen finden Sie unter [Fehlerantworten](#).

Liste der Fehlercodes

In der folgenden Tabelle sind Amazon S3-Fehlercodes aufgeführt.

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ernote-Pr äfix
AccessControlListNotSupported	Der Bucket erlaubt keine ACLs.	400 Bad Request	Client
AccessDenied	Zugriff verweigert	403 Forbidden	Client
AccessPointAlreadyOwnedByYou	In Ihrem Konto ist bereits ein Access Point mit identischem Namen vorhanden.	409 Conflict	Client
AccountProblem	Es liegt ein Problem mit Ihrem vor AWS-Konto , das den erfolgreichen Abschluss des Vorgangs verhindert. Weitere Unterstützung erhalten Sie unter Kontakt .	403 Forbidden	Client
AllAccessDisabled	Jeglicher Zugriff auf diese Amazon S3-Ressource wurde deaktiviert. Weitere Unterstützung erhalten Sie unter Kontakt .	403 Forbidden	Client
AmbiguousGrantByEmailAddress	Die von Ihnen angegebene E-Mail-Adresse ist mit mehr als einem Konto verknüpft.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
AuthorizationHeaderMalformed	Der von Ihnen angegebene Autorisierungs-Header ist ungültig.	400 Bad Request	N/A
AuthorizationQueryParametersError	Die von Ihnen angegebenen Autorisierungs-Abfrageparameter sind nicht gültig.	400 Bad Request	N/A
BadDigest	Der von Ihnen angegebene Wert Content-MD5 oder die Prüfsumme stimmte nicht mit dem überein, was der Server empfangen hat.	400 Bad Request	Client
BucketAlreadyExists	Der angeforderte Bucket-Name ist nicht verfügbar. Der Bucket-Namespace wird von allen Benutzern des Systems gemeinsam genutzt. Geben Sie einen anderen Namen an und versuchen Sie es erneut.	409 Conflict	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
BucketAlreadyOwnedByYou	Der Bucket, den Sie zu erstellen versucht haben, ist bereits vorhanden, und er gehört Ihnen. Amazon S3 gibt diesen Fehler in allen AWS-Regionen außer in der Region USA Ost (Nord-Virginia) (us-east-1) zurück. Aus Gründen der älteren Kompatibilität gibt Amazon S3 die Bucket-Zugriffskontrolllisten (ACLs) zurück 200 OK und setzt sie zurück, wenn Sie einen vorhandenen Bucket, den Sie bereits besitzen, in us-east-1 neu erstellen. Für Amazon S3 on Outposts ist der Bucket, den Sie zu erstellen versucht haben, bereits in Ihrem Outpost vorhanden und Sie besitzen ihn.	409 Conflict allen Regionen außer us-east-1)	Client
BucketHasAccessPointsAttached	An den Bucket, den Sie zu löschen versucht haben, sind Access Points angehängt. Löschen Sie Ihre Access Points, bevor Sie Ihren Bucket löschen.	400 Bad Request	Client
BucketNotEmpty	Der Bucket, den Sie zu löschen versucht haben, ist nicht leer.	409 Conflict	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
<code>ClientTokenConflict</code>	Ihr Multi-Region Access Point-Idempotenz-Token wurde bereits für eine andere Anfrage verwendet.	409 Conflict	Client
<code>ConnectionClosedByRequester</code>	Wird an den ursprünglichen Anrufer zurückgegeben, wenn beim Lesen des Textes ein Fehler auftritt. <code>WriteGetObjectResponse</code>	400 Bad Request	Client
<code>ConditionalRequestConflict</code>	Ein widersprüchlicher Vorgang ist aufgetreten. Bei Verwendung können <code>PutObject</code> Sie die Anfrage erneut versuchen. Wenn Sie den mehrteiligen Upload verwenden, sollten Sie eine weitere <code>CreateMultipartUpload</code> Anfrage stellen und jeden Teil erneut hochladen.	409 Conflict	Client
<code>CredentialsNotSupported</code>	Diese Anfrage unterstützt keine Anmeldeinformationen.	400 Bad Request	Client
<code>CrossLocationLoggingProhibited</code>	Cross-Region Protokollierung ist nicht zulässig. Buckets in einem Bucket AWS-Region können keine Informationen in einem Bucket in einer anderen Region protokollieren.	403 Forbidden	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
DeviceNotActiveError	Das Gerät ist derzeit nicht aktiv.	400 Bad Request	Client
EndpointNotFound	Richten Sie Anforderungen an den richtigen Endpunkt.	400 Bad Request	Client
EntityTooSmall	Ihr empfohlener Upload ist kleiner als die zulässige Mindestobjektgröße.	400 Bad Request	Client
EntityTooLarge	Ihr empfohlener Upload überschreitet die maximal zulässige Objektgröße. Weitere Informationen finden Sie unter Endpunkte und Kontingente von Amazon Simple Storage Service im Allgemeine AWS-Referenz.	400 Bad Request	Client
	Ihr empfohlener Download überschreitet die maximal zulässige Größe.	405 Method Not Allowed	Client
ExpiredToken	Das angegebene Token ist abgelaufen.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
<code>IllegalLocationConstraintException</code>	Dieser Fehler kann aus folgenden Gründen auftreten: • Sie versuchen, auf einen Bucket zuzugreifen, der von einer anderen Region stammt als der, in der sich der Bucket befindet. • Sie versuchen, einen Bucket mit einer Standortbeschränkung zu erstellen, die einer anderen Region als dem regionalen Endpunkt entspricht, an den die Anfrage gesendet wurde.	400 Bad Request	Client
<code>IllegalVersioningConfigurationException</code>	Die in der Anfrage angegebene Versionskonfiguration ist nicht gültig.	400 Bad Request	Client
<code>IncompleteBody</code>	Sie haben nicht die im Content-Length HTTP-Header angegebene Anzahl von Byte angegeben.	400 Bad Request	Client
<code>IncorrectEndpoint</code>	Der angegebene Bucket ist in einer anderen Region vorhanden. Richten Sie Anforderungen an den richtigen Endpunkt.	400 Bad Request	Client
<code>IncorrectNumberOfFilesInPostRequest</code>	POST erfordert genau einen Datei-Upload pro Anfrage.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
InlineDataTooLarge	Die Inline-Daten überschreiten die maximal zulässige Größe.	400 Bad Request	Client
InternalError	Es ist ein interner Fehler aufgetreten. Bitte versuchen Sie es erneut.	500 Internal Server Error	Server
InvalidAccessKeyId	Die von Ihnen angegebene AWS Zugriffsschlüssel-ID ist in unseren Aufzeichnungen nicht vorhanden.	403 Forbidden	Client
InvalidAccessPoint	Der angegebene Zugangspunktname oder das angegebene Konto ist nicht gültig.	400 Bad Request	Client
InvalidAccessPoint AliasError	Der angegebene Aliasname für den Access Point ist nicht gültig.	400 Bad Request	Client
InvalidAddressingHeader	Sie müssen die Rolle „Anonym“ angeben.	N/A	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
InvalidArgument	Dieser Fehler kann aus folgenden Gründen auftreten: • Eine ListBuckets Anforderung wird an einen regionalen Endpunkt gestellt, der sich von der im bucket-region Parameter angegebenen Region unterscheidet. • Das angegebene Argument war nicht gültig. • In der Anforderung fehlte ein erforderlicher Header. • Das angegebene Argument war unvollständig oder hatte das falsche Format. • Das angegebene Argument muss eine Länge von 3 oder mehr haben.	400 Bad Request	Client
InvalidBucketAclWithObjectOwnership	Für Bucket können keine ObjectOwnership ACLs mit der BucketOwnerEnforced Einstellung gesetzt werden.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ernote-Pr äfix
InvalidBucketName	Der angegebene Bucket ist nicht gültig.	400 Bad Request	Client
InvalidBucketOwner AWSAccountID	Der Wert des Parameters für den erwarteten Bucket-Besitzer muss eine AWS-Konto ID sein.	400 Bad Request	Client
InvalidBucketState	Die Anforderung ist für den aktuellen Zustand des Buckets nicht gültig.	409 Conflict	Client
InvalidDigest	Der von Ihnen angegebene Prüfsummenwert Content-MD5 oder ist nicht gültig.	400 Bad Request	Client
InvalidEncryptionAlgorithmError	Die angegebene Verschlüsselungsanforderung ist nicht gültig. Der gültige Wert lautet AES256.	400 Bad Request	Client
InvalidHostHeader	Die Adressierung der in der Anforderung angegebenen Host-Header hatte den falschen Stil.	400 Bad Request	Client
InvalidHttpMethod	Die Anforderung wird mit einer unerwarteten HTTP-Methode gestellt.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
InvalidLocationConstraint	Die angegebene Standorte einschränkung (Region) ist nicht gültig. Weitere Informationen zur Auswahl einer Region für Ihre Buckets finden Sie unter Buckets im Überblick .	400 Bad Request	Client
InvalidObjectState	Der Vorgang ist für den aktuellen Status des Objekts nicht gültig.	403 Forbidde	Client
InvalidPart	Eines oder mehrere der angegebenen Teile konnten nicht gefunden werden. Das Teil wurde möglicherweise nicht hochgeladen, oder das angegebene Entitäts-Tag stimmte möglicherweise nicht mit dem Entitäts-Tag des Teils überein.	400 Bad Request	Client
InvalidPartOrder	Die Liste der Teile war nicht in aufsteigender Reihenfolge. Die Teileliste muss in der Reihenfolge der Artikelnummer angegeben werden.	400 Bad Request	Client
InvalidPayer	Der gesamte Zugriff auf dieses Objekt wurde deaktiviert. Weitere Unterstützung erhalten Sie unter Kontakt .	403 Forbidde	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
InvalidPolicyDocument	Der Inhalt des Formulars erfüllt nicht die im Richtliniendokument angegebenen Bedingungen.	400 Bad Request	Client
InvalidRange	Der angeforderte Bereich kann nicht erfüllt werden.	416 Requested Range Not Satisfiab le (416 Angeforde rter Bereich ist nicht erfüllbar)	Client
InvalidRegion	Sie haben versucht, einen Multi-Region Access Point in einer Region einzurichten, für die Sie sich nicht angemeldet haben.	403 Forbidden	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
InvalidRequest	Dieser Fehler kann aus folgenden Gründen auftreten: • Eine unpaginierte ListBuckets-Anfrage wird von einem Konto aus gestellt, für das ein genehmigtes Allzweck-Bucket-Kontingent von mehr als 10.000 gilt. Sie müssen paginierte Anfragen stellen, um die Buckets in einem Konto mit mehr als 10.000 Buckets aufzulisten. • Die Anforderung verwendet die falsche Signaturversion. Verwenden Sie AWS4-HMAC-SHA256 (Signaturversion 4). • Ein Zugangspunkt kann nur für einen vorhandenen Bucket erstellt werden. • Der Zugangspunkt befindet sich nicht in einem Zustand, in dem er gelöscht werden kann. • Ein Zugangspunkt kann nur für einen vorhandenen Bucket aufgelistet werden. •	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
	Das nächste Token ist nicht gültig. • In einer Lebenszyklusregel muss mindestens eine Aktion angegeben werden. • Es muss mindestens eine Lebenszyklusregel angegeben werden. • Die Anzahl der Lebenszyklusregeln darf den zulässigen Höchstwert von 1.000 Regeln nicht überschreiten. • Der Bereich für den Parameter <code>MaxResults</code> ist nicht gültig. • SOAP-Anforderungen müssen über eine HTTPS-Verbindung gestellt werden. • Amazon S3 Transfer Acceleration wird für Buckets nicht unterstützt, die keinen DNS-konformen Namen haben. • Amazon S3 Transfer Acceleration wird für Buckets nicht unterstützt		

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
	zt, deren Namen Punkte (.) enthalten. • Der Endpunkt von Amazon S3 Transfer Acceleration unterstützt nur virtuelle Anforderungen. • Amazon S3 Transfer Acceleration ist in diesem Bucket nicht konfiguriert. • Amazon S3 Transfer Acceleration ist in diesem Bucket deaktiviert. • Amazon S3 Transfer Acceleration wird in diesem Bucket nicht unterstützt. Wenn Sie Hilfe benötigen, wenden Sie sich an Support. • Amazon S3 Transfer Acceleration kann in diesem Bucket nicht aktiviert werden. Wenn Sie Hilfe benötigen, wenden Sie sich an Support. • Widersprüchliche Werte in HTTP-Headern und Abfrageparametern angegeben. •		

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
	Widersprüchliche Werte in HTTP-Headern und POST-Formularfeldern angegeben. CopyObject Anfrage für Objekte, die größer als 5 GB sind.		
InvalidSessionException	Wird zurückgegeben, wenn die Sitzung nicht mehr existiert, weil das Zeitlimit überschritten wurde oder sie abgelaufen ist.	400 Bad Request	Client
InvalidSignature	Die vom Server berechnete Anforderungssignatur entspricht nicht der angegebenen Signatur. Überprüfen Sie Ihren AWS geheimen Zugangsschlüssel und Ihre Signaturmethode. Weitere Informationen finden Sie unter Signieren und Authentifizieren von REST-Anforderungen .	400 Bad Request	Client
InvalidSecurity	Die angegebenen Sicherheitsanmeldeinformationen sind ungültig.	403 Forbidden	Client
InvalidSOAPRequest	Der Text der SOAP-Anforderung ist nicht gültig.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
InvalidStorageClass	Die angegebene Speicherklasse ist nicht gültig.	400 Bad Request	Client
InvalidTargetBucketForLogging	Der Ziel-Bucket für die Protokollierung ist entweder nicht vorhanden , gehört Ihnen nicht oder er verfügt nicht über die entsprechenden Berechtigungen für die Log-Delivery-Gruppe.	400 Bad Request	Client
InvalidToken	Das angegebene Token ist falsch formatiert oder anderweitig nicht gültig.	400 Bad Request	Client
InvalidURI	Die angegebene URI konnte nicht analysiert werden.	400 Bad Request	Client
KeyTooLongError	Ihr Schlüssel ist zu lang.	400 Bad Request	Client
KMS.DisabledException	Die Anforderung wurde abgelehnt, da der angegebene KMS-Schlüssel nicht aktiviert ist.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
KMS.InvalidKeyUsageException	Die Anforderung wurde aus einem der folgenden Gründen abgelehnt: • Der KeyUsage Wert des KMS-Schlüssels ist nicht mit dem API-Vorgang kompatibel. • Der für den Vorgang angegebene Verschlüsselungsalgorithmus oder Signaturalgorithmus ist nicht mit der Art des Schlüsselmaterials im KMS-Schlüssel (KeySpec) kompatibel. Zum Verschlüsseln, Entschlüsseln, erneuten Verschlüsseln und Generieren von Datenschlüsseln muss der Schlüssel ENCRYPT_DECRYPT lauten. KeyUsage Zum Signieren und Überprüfen von Nachrichten muss der Wert SIGN_VERIFY sein. KeyUsage Zum Generieren und Überprüfen von Nachrichtenauthentifizierungscodes (MACs) muss dieser GENERATE_VERIFY_MAC sein. KeyUsage Um wichtige Vertragsgeheimnisse abzuleiten, muss es KEY_AGREEMENT sein. KeyUsage Verwenden Sie die Operation, um den KeyUsage Wert	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
	eines KMS-Schlüssels zu ermitteln. DescribeKey Verwenden Sie den DescribeKey Vorgang, um die Verschlüsselungs- oder Signaturalgorithmen zu ermitteln, die für einen bestimmten KMS-Schlüssel unterstützt werden.		

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
KMS.KMSInvalidStateException	Die Anforderung wurde abgelehnt , da der Status der angegebenen Ressource für diese Anforderung nicht gültig ist. Diese Ausnahme bedeutet, dass eine der folgenden Aussagen zutrifft: - Der Status des KMS-Schlüssels ist nicht mit der Operation kompatibel. Verwenden Sie die DescribeKey Operation, um den Schlüsselstatus zu ermitteln. Weitere Informationen darüber, welche Schlüsselstatus mit den einzelnen KMS-Operationen kompatibel sind, finden Sie unter Schlüsselstatus von AWS KMS keys im AWS Key Management Service -Entwicklerhandbuch. - Für kryptografische Operationen an KMS-Schlüsseln in benutzerdefinierten Schlüsselspeichern stellt diese Ausnahme einen allgemeinen Fehler mit vielen möglichen Ursachen dar. Identifizieren Sie die Ursache anhand der Fehlermeldung, die zusammen mit der Ausnahme angezeigt wird.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
KMS.NotFoundException	Die Anforderung wurde abgelehnt , da die angegebene Entity oder Ressource nicht gefunden wurde.	400 Bad Request	Client
MalformedACLError	Die angegebene ACL war nicht gut formatiert oder konnte nicht gegen unser veröffentlichtes Schema validiert werden.	400 Bad Request	Client
MalformedPOSTRequest	Der Hauptteil Ihrer POST-Anfrage besteht nicht aus wohlgeformten multipart/form -data.	400 Bad Request	Client
MalformedXML	Der angegebene XML-Code war nicht gut formatiert oder konnte nicht gegen unser veröffentlichtes Schema validiert werden.	400 Bad Request	Client
MaxMessageLengthExceeded	Ihre Anfrage war zu umfangreich.	400 Bad Request	Client
MaxPostPreDataLengthExceededError	Die Felder Ihrer POST-Anforderung vor der Upload-Datei waren zu groß.	400 Bad Request	Client
MetadataTooLarge	Ihre Metadaten-Header überschreiten die maximal zulässige Metadaten­größe.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
MethodNotAllowed	Die angegebene Methode ist für diese Ressource nicht zulässig.	405 Method Not Allowed	Client
MissingAttachment	Es wurde ein SOAP-Anhang erwartet, aber keiner gefunden.	400 Bad Request (400 Ungültige Anfrage)	Client
MissingAuthenticationToken	Die Anforderung war nicht signiert.	403 Forbidden	Client
MissingContentLength	Sie müssen den Content-Length HTTP-Header angeben.	411 Length Required	Client
MissingRequestBodyError	Sie haben ein leeres XML-Dokument als Anforderung gesendet.	400 Bad Request	Client
MissingSecurityElement	In der SOAP 1.1-Anfrage fehlt ein Sicherheitselement.	400 Bad Request	Client
MissingSecurityHeader	In Ihrer Anforderung fehlt ein erforderlicher Header.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
NoLoggingStatusForKey	Es gibt keine Subressource des Protokollierungsstatus für einen Schlüssel.	400 Bad Request	Client
NoSuchAsyncRequest	Die angegebene Anforderung wurde nicht gefunden.	404 Not Found	Client
NoSuchBucket	Der angegebene Bucket ist nicht vorhanden.	404 Not Found	Client
NoSuchBucketPolicy	Für den angegebenen Bucket gibt es keine Bucket-Richtlinie.	404 Not Found	Client
NoSuchCORSConfiguration	Der angegebene Bucket hat keine CORS-Konfiguration.	404 Not Found	Client
NoSuchKey	Der angegebene Schlüssel ist nicht vorhanden.	404 Not Found	Client
NoSuchLifecycleCon figuration	Die angegebene Lebenszyk luskonfiguration ist nicht vorhanden.	404 Not Found	Client
NoSuchMultiRegionA ccessPoint	Der angegebene Multi-Region Access Point ist nicht vorhanden.	404 Not Found	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ernotecode-Pr äfix
NoSuchObjectLockConfiguration	Das angegebene Objekt hat keine ObjectLock Konfiguration.	404 Not Found	Client
NoSuchWebsiteConfiguration	Der angegebene Bucket hat keine Website-Konfiguration.	404 Not Found	Client
NoSuchTagSet	Das angegebene Tag ist nicht vorhanden.	404 Not Found	Client
NoSuchUpload	Der angegebene mehrteilige Upload ist nicht vorhanden. Möglicherweise ist die Upload-ID nicht gültig oder der mehrteilige Upload wurde abgebrochen oder abgeschlossen.	404 Not Found	Client
NoSuchVersion	Die in der Anfrage angegebene Versions-ID stimmt nicht mit einer vorhandenen Version überein.	404 Not Found	Client
NotDeviceOwnerError	Das Gerät, das das Token generiert hat, gehört nicht dem authentifizierten Benutzer.	400 Bad Request	Client
NotImplemented	Ein von Ihnen bereitgestellter Header impliziert Funktionen, die nicht implementiert sind.	501 Not Implemented	Server

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ernotecode-Pr äfix
NotModified	Die Ressource wurde nicht geändert.	304 Not Modified	Server
NoTransformationDefined	Für diesen Zugriffspunkt von Objekt Lambda wurde keine Transformation gefunden.	404 Not Found	Client
NotSignedUp	Das Konto ist nicht für den Amazon-S3-Service registriert. Bevor Sie Amazon S3 verwenden können, müssen Sie sich anmelden. Sie können sich unter der folgenden URL anmelden: https://aws.amazon.com/s3	403 Forbidden	Client
ObjectLockConfigurationNotFoundError	Die Konfiguration von Object Lock ist für diesen Bucket nicht vorhanden.	404 Not Found	Client
OwnershipControlsNotFoundError	Die Kontrollmöglichkeiten für den Bucket wurden nicht gefunden.	404 Not Found	Client
OperationAborted	Für diese Ressource wird derzeit eine bedingte Operation ausgeführt, die einen Konflikt verursacht. Bitte versuchen Sie es erneut.	409 Conflict	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
PermanentRedirect	Der Bucket, auf den Sie zugreifen wollen, muss mit dem angegebenen Endpunkt aufgerufen werden. Senden Sie alle künftigen Aufrufe an diesen Endpunkt.	301 Moved Permanen ly	Client
PermanentRedirectC ontrolError	Die API-Operation, auf die Sie zugreifen wollen, muss mit dem angegebenen Endpunkt aufgerufen werden. Senden Sie alle künftigen Aufrufe an diesen Endpunkt.	301 Moved Permanen ly	Client
PreconditionFailed	Mindestens eine der von Ihnen angegebenen Vorbedingungen war nicht erfüllt.	412 Precondi ion Failed	Client
Redirect	Temporäre Weiterleitung. Sie werden zum Bucket weitergeleitet, während der Domain Name System (DNS)-Server aktualisiert wird.	307 Temporar Redirect	Client
RequestHeaderSecti onTooLarge	Der Anforderungsheader und die Abfrageparameter, mit denen die Anfrage gestellt wurde, überschreiten die maximal zulässige Größe.	400 Bad Request	Client
RequestIsNotMultiP artContent	Eine Bucket-POST-Anfrage muss vom Gehäusetyp -data sein. multipart/form	412 Precondi ion Failed	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
RequestTimeout	Ihre Socket-Verbindung zum Server wurde innerhalb des Timeout-Zeitraums nicht gelesen oder darauf geschrieben.	400 Bad Request	Client
RequestTimeTooSkewed	Der Unterschied zwischen der Anforderungszeit und der Serverzeit ist zu groß.	403 Forbidden	Client
RequestTorrentOfBucketError	Das Anfordern der Torrent-Datei eines Buckets ist nicht zulässig.	400 Bad Request	Client
ResponseInterrupted	Wird an den ursprünglichen Aufrufer zurückgegeben, wenn beim Lesen des Textes ein Fehler auftritt WriteGetObjectResponse .	400 Bad Request	Client
RestoreAlreadyInProgress	Die Objektwiederherstellung ist bereits im Gange.	409 Conflict	Client
ServerSideEncryptionConfigurationNotFound	Die serverseitige Verschlüsselungskonfiguration wurde nicht gefunden.	400 Bad Request	Client
ServiceUnavailable	Der Dienst kann die Anfrage nicht bearbeiten.	503 Service Unavaila le	Server

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
SignatureDoesNotMatch	Die vom Server berechnete Anforderungssignatur entspricht nicht der angegebenen Signatur. Überprüfen Sie Ihren AWS geheimen Zugangsschlüssel und Ihre Signaturmethode. Weitere Informationen finden Sie unter REST-Authentifizierung und SOAP-Authentifizierung .	403 Forbidden	Client
SlowDown	Bitte reduzieren Sie Ihre Anfragerate.	503 Slow Down	Server
503 SlowDown	Verlangsamen Sie	503 Slow Down	Server
TagPolicyException	Die Tag-Richtlinie lässt den angegebenen Wert für den folgenden Tag-Schlüssel nicht zu.	400 Bad Request	Client
TemporaryRedirect	Sie werden zum Bucket weitergeleitet, während der Domain Name System (DNS)-Server aktualisiert wird.	307 Temporary Redirect	Client
TokenCodeInvalidError	Der von Ihnen angegebene and/or Seriennummer-Tokencode ist nicht gültig.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ernote-Pr äfix
TokenRefreshRequired	Das bereitgestellte Token muss aktualisiert werden.	400 Bad Request	Client
TooManyAccessPoints	Sie haben versucht, mehr Access Points zu erstellen, als für ein Konto zulässig sind. Weitere Informationen finden Sie unter Amazon Simple Storage Service-Endpunkte und Kontingente im Allgemeine AWS-Referenz.	400 Bad Request	Client
TooManyBuckets	Sie haben versucht, mehr Buckets zu erstellen, als für ein Konto zulässig sind. Weitere Informationen finden Sie unter Amazon Simple Storage Service-Endpunkte und Kontingente im. Allgemeine AWS-Referenz	400 Bad Request	Client
TooManyMultiRegion AccessPointregionsError	Sie haben versucht, einen Multi-Region Access Point mit mehr Regionen zu erstellen, als für ein Konto zulässig sind. Weitere Informationen finden Sie unter Amazon Simple Storage Service-Endpunkte und Kontingente im Allgemeine AWS-Referenz.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
TooManyMultiRegion AccessPoints	Sie haben versucht, mehr Multi-Region Access Points zu erstellen, als für ein Konto zulässig sind. Weitere Informationen finden Sie unter Amazon Simple Storage Service-Endpunkte und Kontingente im Allgemeine AWS-Referenz.	400 Bad Request	Client
UnauthorizedAccessError	Gilt nur in Regionen in China. Wird zurückgegeben, wenn eine Anforderung an einen Bucket gestellt wird, der keine ICP-Lizenz hat. Weitere Informationen finden Sie unter ICP-Aufzeichnung .	403 Forbidden	Client
UnexpectedContent	Diese Anforderung enthält nicht unterstützte Inhalte.	400 Bad Request	Client
UnexpectedIPError	Gilt nur in Regionen in China. Diese Anforderung wurde abgelehnt, da die IP unerwartet war.	403 Forbidden	Client
UnsupportedArgument	Die Anforderung enthielt ein nicht unterstütztes Argument.	400 Bad Request	Client
UnsupportedSignature	Die bereitgestellte Anforderung ist mit einer nicht unterstützten STS-Token-Version signiert oder die Signaturversion wird nicht unterstützt.	400 Bad Request	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode- Pr äfix
UnresolvableGrantByEmailAddress	Die von Ihnen angegebene E-Mail-Adresse stimmt mit keinem registrierten Konto überein.	400 Bad Request	Client
UserKeyMustBeSpecified	Die Bucket-POST-Anforderung muss den angegebenen Feldnamen enthalten. Falls angegeben, überprüfen Sie die Reihenfolge der Felder.	400 Bad Request	Client
NoSuchAccessPoint	Der angegebene Zugriffspunkt ist nicht vorhanden.	404 Not Found	Client
InvalidTag	Ihre Anforderung enthält eine ungültige Tag-Eingabe. Ihre Anforderung könnte beispielsweise doppelte Schlüssel, zu lange Schlüssel oder Werte oder System-Tags enthalten.	400 Bad Request	Client
MalformedPolicy	Ihre Richtlinie enthält einen Prinzipal, der nicht gültig ist.	400 Bad Request	Client

Liste der SELECT-Fehlercodes für den Objekteinhalt

Important

Amazon S3 Select ist für Neukunden nicht mehr verfügbar. Bestandskunden von Amazon S3 Select können das Feature weiterhin wie gewohnt nutzen. [Weitere Informationen](#)

Die folgende Tabelle enthält spezielle Fehler, die SELECT Object Content möglicherweise auftreten. Allgemeine Informationen zu Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlermeldungen](#).

Fehlercode	Description	HTTP-Statuscode	SOAP-Fehlercode-Präfix
AmbiguousFieldName	Der Feldname entspricht mehreren Feldern in der Datei. Überprüfen Sie den SQL-Ausdruck und die Datei, und versuchen Sie es erneut.	400	Client
Busy	Der Service ist nicht verfügbar. Bitte versuchen Sie es später erneut.	503	Client
CastFailed	Ein Versuch, mithilfe des SQL-Ausdrucks von einem Datentyp in einen anderen zu konvertieren, ist CAST fehlgeschlagen.	400	Client
ColumnTooLong	Die Länge einer Spalte im Ergebnis ist größer als <code>maxCharsPerColumn</code> 1 MB.	400	Client
CSVEscapingRecordDelimiter	In der Datei wurde ein Datensatz trennzeichen in Anführungszeichen gefunden. Um Datensatz	400	Client

Fehlercode	Description	HTTP-Statuscode	SOAP-Fehlercode-Präfix
	trennzeichen in Anführungszeichen zuzulassen, setzen Sie den Wert auf. <code>AllowQuotedRecordDelimiter 'TRUE'</code>		
<code>CSVParsingError</code>	Beim Analysieren der CSV-Datei ist ein Fehler aufgetreten. Überprüfen Sie die Datei und versuchen Sie es erneut.	400	Client
<code>CSVUnescapedQuote</code>	Beim Analysieren der CSV-Datei wurde ein Zitat ohne Escape-Zeichen gefunden. Um Datensatztrennzeichen in Anführungszeichen zuzulassen, setzen Sie den Wert auf. <code>AllowQuotedRecordDelimiter 'TRUE'</code>	400	Client
<code>EmptyRequestBody</code>	Der Anforderungstext darf nicht leer sein.	400	Client
<code>EvaluatorBindingDoesNotExist</code>	Ein bereitgestellter Spaltenname oder Pfad ist im SQL-Ausdruck nicht vorhanden.	400	Client
<code>EvaluatorInvalidArguments</code>	Der Funktionsaufruf im SQL-Ausdruck enthält eine falsche Anzahl von Argumenten.	400	Client
<code>EvaluatorInvalidTimestampFormatPattern</code>	Die Zeichenfolge im Zeitstempelformat im SQL-Ausdruck ist nicht gültig.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
EvaluatorInvalidTimestampFormatPatternSymbol	Das Zeitstempelformatmuster enthält ein Symbol im SQL-Ausdruck, das nicht gültig ist.	400	Client
EvaluatorInvalidTimestampFormatPatternSymbolForParsing	Das Zeitstempelformatmuster enthält ein gültiges Formatsymbol, das nicht auf die Zeitstempelanalyse im SQL-Ausdruck angewendet werden kann.	400	Client
EvaluatorInvalidTimestampFormatPatternToken	Das Zeitstempelformatmuster enthält ein Token im SQL-Ausdruck, das nicht gültig ist.	400	Client
EvaluatorLikePatternInvalidEscapeSequence	Ein dem LIKE Ausdruck übergebenes Argument war nicht gültig.	400	Client
EvaluatorNegativeLimit	LIMIT darf nicht negativ sein.	400	Client
EvaluatorTimestampFormatPatternDuplicateFields	Das Zeitstempel-Formatmuster enthält mehrere Formatbezeichner, die das Zeitstempelfeld im SQL-Ausdruck darstellen.	400	Client
EvaluatorTimestampFormatPatternHourClockAmPmMismatch	Das Zeitstempelformatmuster enthält ein 12-Stunden-Tag-Formatsymbol, aber kein AM/PM Feld, oder es enthält einen 24-stündigen Stunden-Tag-Formatbezeichner und enthält ein AM/PM Feld im SQL-Ausdruck.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
EvaluatorUnterminatedTimestampFormatPatternToken	Das Zeitstempelformatmuster enthält ein Zeichen ohne Terminierung im SQL-Ausdruck.	400	Client
ExpressionTooLong	Der SQL-Ausdruck ist zu lang. Die maximale Bytelänge für einen SQL-Ausdruck beträgt 256 KB.	400	Client
ExternalEvalException	Die Abfrage kann nicht ausgewertet werden. Überprüfen Sie die Datei und versuchen Sie es erneut.	400	Client
IllegalSqlFunctionArgument	In der SQL-Funktion wurde ein unzulässiges Argument verwendet.	400	Client
IncorrectSqlFunctionArgumentType	Bei einem Funktionsaufruf im SQL-Ausdruck wurde ein falscher Argumenttyp angegeben.	400	Client
IntegerOverflow	Im SQL-Ausdruck ist ein Integer-Über- oder -Unterlauf aufgetreten.	400	Client
InternalError	Es ist ein interner Fehler aufgetreten.	500	Client
InvalidCast	Ein Versuch, mithilfe des SQL-Ausdrucks von einem Datentyp in einen anderen zu konvertieren, ist CAST fehlgeschlagen.	400	Client
InvalidColumnIndex	Der Spaltenindex im SQL-Ausdruck ist nicht gültig.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
InvalidCompression Format	Die Datei hat kein unterstütztes Komprimierungsformat. Nur GZIP und BZIP2 werden unterstützt.	400	Client
InvalidDataSource	Der Datenquellentyp ist nicht gültig. Nur CSV, JSON und Parquet werden unterstützt.	400	Client
InvalidDataType	Der SQL-Ausdruck enthält einen Datentyp, der nicht gültig ist.	400	Client
InvalidExpressionT ype	Der ExpressionType Wert ist nicht gültig. Nur SQL-Ausdrücke werden unterstützt.	400	Client
InvalidFileHeaderI nfo	Der FileHeaderInfo Wert ist nicht gültig. Es werden nur NONE, USE und IGNORE unterstützt.	400	Client
InvalidJsonType	Der JsonType Wert ist nicht gültig. Es werden ausschließlich DOCUMENT und LINES unterstützt.	400	Client
InvalidKeyPath	Der Schlüsselpfad im SQL-Ausdr uck ist nicht gültig.	400	Client
InvalidQuoteFields	Der QuoteFields Wert ist nicht gültig. Es werden ausschließlich ALWAYS und ASNEEDED unterstüt zt.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
<code>InvalidRequestParameter</code>	Der Wert eines Parameters im <code>SelectRequest</code> Element ist nicht gültig. Prüfen Sie die Service-API-Dokumentation und versuchen Sie es erneut.	400	Client
<code>InvalidScanRange</code>	Der angegebene Scanbereich ist nicht gültig.	400	Client
<code>InvalidTableAlias</code>	Der SQL-Ausdruck enthält einen Tabellenalias, der nicht gültig ist.	400	Client
<code>InvalidTextEncoding</code>	Der Codierungstyp ist nicht gültig. Nur die UTF-8 Kodierung wird unterstützt.	400	Client
<code>JSONParsingError</code>	Beim Analysieren der JSON-Datei ist ein Fehler aufgetreten. Überprüfen Sie die Datei und versuchen Sie es erneut.	400	Client
<code>LexerInvalidChar</code>	Der SQL-Ausdruck enthält ein ungültiges Zeichen.	400	Client
<code>LexerInvalidIONLiteral</code>	Der SQL-Ausdruck enthält einen Operator, der nicht gültig ist.	400	Client
<code>LexerInvalidLiteral</code>	Der SQL-Ausdruck enthält einen Operator, der nicht gültig ist.	400	Client
<code>LexerInvalidOperator</code>	Der SQL-Ausdruck enthält ein Literal, das nicht gültig ist.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
LikeInvalidInputs	Das Argument, das der LIKE Klausel im SQL-Ausdruck gegeben wurde, ist nicht gültig.	400	Client
MalformedXML	Das bereitgestellte XML war nicht wohlgeformt oder wurde nicht anhand unseres veröffentlichten Schemas validiert. Prüfen Sie die Servicedokumentation und versuchen Sie es erneut.	400	Client
MaxOperatorsExceeded	Fehler beim Analysieren des SQL-Ausdrucks. Versuchen Sie, die Komplexität zu reduzieren. Reduzieren Sie beispielsweise die Anzahl der verwendeten Operatoren.	400	Client
MethodNotAllowed	Die angegebene Methode ist für diese Ressource nicht zulässig.	405 Method Not Allowed	Client
MissingRequiredParameter	Der SelectRequest Entität fehlt ein erforderlicher Parameter. Prüfen Sie die Servicedokumentation und versuchen Sie es erneut.	400	Client
MultipleDataSourceUnsupported	Mehrere Datenquellen werden nicht unterstützt.	400	Client

Fehlercode	Description	HTTP-Statuscode	SOAP-Fehlercode-Präfix
NumberFormatError	Beim Analysieren einer Zahl ist ein Fehler aufgetreten. Dieser Fehler kann durch einen Unter- oder Überlauf von Ganzzahlen verursacht werden.	400	Client
ObjectSerializationConflict	InputSerialization gibt mehr als ein Format an (CSV, JSON oder Parquet) oder OutputSerialization gibt mehr als ein Format an (CSV oder JSON). Für InputSerialization und OutputSerialization können Sie jeweils nur ein Format angeben.	400	Client
OverMaxColumn	Die Anzahl der Spalten im Ergebnis ist größer als die maximal zulässige Anzahl von Spalten.	400	Client
OverMaxParquetBlockSize	Die Parquet-Datei liegt über der maximalen Zeilengruppengröße.	400	Client
OverMaxRecordSize	Die Länge eines Datensatzes in der Eingabe oder im Ergebnis ist größer als das maxCharsPerRecord Limit von 1 MB.	400	Client
ParquetParsingError	Beim Analysieren der Parquet-Datei ist ein Fehler aufgetreten. Überprüfen Sie die Datei und versuchen Sie es erneut.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
ParquetUnsupported CompressionCodec	Der angegebene Parquet-Komprimierungscodec wird nicht unterstützt.	400	Client
ParseAsteriskIsNot AloneInSelectList	Andere Ausdrücke sind in der SELECT Liste nicht zulässig, wenn sie * im SQL-Ausdruck ohne Punktnotation verwendet werden.	400	Client
ParseCannotMixSqbandWildcardInSelect List	Ein [] und derselbe Ausdruck * in einer SELECT Liste im SQL-Ausdruck kann nicht gemischt werden.	400	Client
ParseCastArity	Der SQL-Ausdruck CAST hat eine falsche Arität.	400	Client
ParseEmptySelect	Der SQL-Ausdruck enthält eine leere SELECT Klausel.	400	Client
ParseExpected2TokenTypes	Das erwartete Token im SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedArgumentDelimiter	Das erwartete Argumenttrennzeichen im SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedDatePart	Der Teil mit dem erwarteten Datum im SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedExpression	Der erwartete SQL-Ausdruck wurde nicht gefunden.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
ParseExpectedIdent ForAlias	Der erwartete Bezeichner für den Alias im SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedIdent ForAt	Der erwartete Bezeichner für den AT Namen im SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedIdent ForGroupName	GROUP wird im SQL-Ausdruck nicht unterstützt.	400	Client
ParseExpectedKeywo rd	Das erwartete Schlüsselwort im SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedLeftP arenAfterCast	Die erwartete linke Klammer hinter CAST dem SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedLeftP arenBuiltinFunctionCall	Die erwartete linke Klammer im SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedLeftP arenValueConstructor	Die erwartete linke Klammer im SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedMember	Der SQL-Ausdruck enthält eine nicht unterstützte Verwendung von. MEMBER	400	Client
ParseExpectedNumber	Die erwartete Zahl im SQL-Ausdruck wurde nicht gefunden.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
ParseExpectedRight ParenBuiltinFuncti onCall	Das erwartete Zeichen in der rechten Klammer im SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedToken Type	Das erwartete Token im SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedTypeN ame	Der erwartete Typname im SQL-Ausdruck wurde nicht gefunden.	400	Client
ParseExpectedWhenC lause	Die erwartete WHEN Klausel im SQL-Ausdruck wurde nicht gefunden. CASE wird nicht unterstützt.	400	Client
ParseInvalidContex tForWildcardInSele ctList	Die Verwendung von * in der SELECT Liste im SQL-Ausdruck ist nicht gültig.	400	Client
ParseInvalidPathCo mponent	Der SQL-Ausdruck enthält eine Pfadkomponente, die nicht gültig ist.	400	Client
ParseInvalidTypePa ram	Der SQL-Ausdruck enthält einen Parameterwert, der nicht gültig ist.	400	Client
ParseMalformedJoin	JOIN wird im SQL-Ausdruck nicht unterstützt.	400	Client
ParseMissingIdentA fterAt	Der erwartete Bezeichner nach dem @-Symbol im SQL-Ausdruck wurde nicht gefunden.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
ParseNonUnaryAgreg ateFunctionCall	Für Aggregatfunktionen im SQL-Ausdruck wird nur ein Argument unterstützt.	400	Client
ParseSelectMissing From	Der SQL-Ausdruck enthält ein Zeichen, das FROM hinter der SELECT Liste fehlt.	400	Client
ParseUnExpectedKey word	Der SQL-Ausdruck enthält ein unerwartetes Schlüsselwort.	400	Client
ParseUnexpectedOpe rator	Der SQL-Ausdruck enthält einen unerwarteten Operator.	400	Client
ParseUnexpectedTerm	Der SQL-Ausdruck enthält einen unerwarteten Begriff.	400	Client
ParseUnexpectedTok en	Der SQL-Ausdruck enthält ein unerwartetes Token.	400	Client
ParseUnknownOperat or	Der SQL-Ausdruck enthält einen Operator, der nicht gültig ist.	400	Client
ParseUnsupportedAl ias	Der SQL-Ausdruck enthält eine nicht unterstützte Verwendung vonALIAS.	400	Client
ParseUnsupportedCa llWithStar	Nur COUNT mit (*) als Parameter wird im SQL-Ausdruck unterstützt.	400	Client
ParseUnsupportedCa se	Der SQL-Ausdruck enthält eine nicht unterstützte Verwendung vonCASE.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
ParseUnsupportedCa seClause	Der SQL-Ausdruck enthält eine nicht unterstützte Verwendung von. CASE	400	Client
ParseUnsupportedLi teralsGroupBy	Der SQL-Ausdruck enthält eine nicht unterstützte Verwendung von. GROUP BY	400	Client
ParseUnsupportedSe lect	Der SQL-Ausdruck enthält eine nicht unterstützte Verwendung von. SELECT	400	Client
ParseUnsupportedSy ntax	Der SQL-Ausdruck enthält eine nicht unterstützte Syntax.	400	Client
ParseUnsupportedTo ken	Der SQL-Ausdruck enthält ein nicht unterstütztes Token.	400	Client
TruncatedInput	Die Objektdekomprimierung ist fehlgeschlagen. Überprüfen Sie, ob das Objekt mit dem in der Anforderung angegebenen Format ordnungsgemäß komprimiert wurde.	400	Client
UnauthorizedAccess	Sie sind zur Ausführung dieser Operation nicht autorisiert.	401	Client
UnrecognizedFormat Exception	Der Datensatztyp ist nicht gültig.	400	Client
UnsupportedFunction	S3 hat eine nicht unterstützte SQL-Funktion gefunden.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
UnsupportedParquet Type	Der angegebene Parquet-Typ wird nicht unterstützt.	400	Client
UnsupportedRangeHe ader	Ein Range-Header wird für diesen Vorgang nicht unterstützt.	400	Client
UnsupportedScanRan geInput	Abfragen von Scanbereichen werden für diesen Objekttyp nicht unterstützt.	400	Client
UnsupportedSql0per ation	S3 hat einen nicht unterstützten SQL-Vorgang festgestellt.	400	Client
UnsupportedSqlStru cture	S3 ist auf eine nicht unterstützte SQL-Struktur gestoßen. Überprüfen Sie die SQL-Referenz.	400	Client
UnsupportedStorage Class	Die Speicherklasse wird nicht unterstützt. Nur STANDARD die ONEZONE_IA SpeicherlassenSTANDARD_IA , und werden unterstützt.	400	Client
UnsupportedSyntax	Die Syntax ist nicht gültig.	400	Client
UnsupportedTypeFor Querying	Ihre Abfrage enthält einen Typ, der nicht zum Vergleich unterstützt wird (z. B. um zu überprüfen, ob ein Parquet INT96-Spaltentyp größer als 0 ist).	400	Client
ValueParseFailure	Im SQL-Ausdruck ist ein Fehler beim Analysieren des Zeitstempels aufgetreten.	400	Client

Liste der Fehlercodes Replication-related

Die folgende Tabelle enthält spezielle Fehler, die der Replication Vorgang möglicherweise zurückgibt. Allgemeine Informationen zu Amazon S3-Fehlern und eine Liste der Fehlercodes finden Sie unter [Fehlermeldungen](#).

Fehlercode	Description	HTTP-Statuscode	SOAP-Fehlercode-Präfix
InvalidArgument	Dieser Fehler kann aus folgenden Gründen auftreten: • Das <Account> Element ist leer. Es muss eine gültige Konto-ID enthalten. • Die im <Account> Element AWS-Konto angegebene Angabe muss mit dem Besitzer des Ziel-Buckets übereinstimmen. • ReplicationTime-Status muss einen Wert enthalten. • ReplicationTime-ReplicationTimeValue muss einen Wert enthalten. • Replication-ReplicationTimeValue-Minutes Der Wert muss 15 sein. • ReplicationMetrics muss einen enthaltenStatus.	400	Client

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
	• <code>ReplicationMetrics</code> muss ein <code>EventThreshold</code> enthalten. • <code>EventThreshold-ReplicationTimeValue-Minutes</code> Der Wert muss 15 sein. • <code>Rule ID</code> darf keine Nicht-ASCII-Zeichen enthalten.		

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
InvalidRequest	Dieser Fehler kann aus folgenden Gründen auftreten: • Das <Owner> in <AccessControlTranslation> hat einen Wert, daher muss das <Account> Element angegeben werden. • Das <Account> Element ist leer. Es muss eine gültige Konto-ID enthalten. • Die Replikation destination muss ReplicationTime sowohl Metrics als auch oder keines enthalten. • ReplicationTime und ReplicationMetrics muss den gleichen Status haben. • S3 Replication Time Control (S3 RTC) wird dabei AWS-Region nicht unterstützt.	400	Client
ReplicationConfigurationNotFoundError	Für diesen Bucket gibt es keine Replikationskonfiguration.	404 Not Found (404 Nicht gefunden)	Client

Liste der Tagging-related Fehlercodes

Die folgende Tabelle enthält spezielle Fehler, die bei den `ListTagsForResource` Vorgängen `TagResource` und `UntagResource`, und für Storage Lens-Gruppen möglicherweise zurückgegeben werden. Allgemeine Informationen zu allgemeinen Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlermeldungen](#).

Fehlercode	Description	HTTP-Statuscode	SOAP-Fehlercode-Präfix
<code>InvalidRequest</code>	Der ARN AWS-Region in der Ressource entspricht nicht der Region, die in dieser Anfrage angegeben ist. Das AWS Konto im Ressourcen-ARN stimmt nicht mit der Konto-ID überein, die in dieser Anfrage angegeben ist. Die AWS Partition in der ResourceARN ist ungültig.	400 Bad Request	Nicht unterstützt
<code>InvalidTag</code>	Diese Anforderung enthält einen Tag-Schlüssel oder -Wert, der nicht gültig ist. Zu den gültigen Zeichen gehören die folgenden: <code>[a-zA-Z+-=._:/]</code> . Tag-Schlüssel können bis zu 128 Zeichen enthalten. Tag-Werte können bis zu 256 Zeichen enthalten. Ihre Anfrage enthält doppelte Tag-Schlüssel. User-defined Tag-Schlüssel können nicht mit <code>aws:</code> beginnen.	400 Bad Request	Nicht unterstützt
<code>NoSuchResource</code>	Die angegebene Ressource ist nicht vorhanden.	404 Not Found	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
TagPolicyException	Die Tag-Richtlinie lässt den angegebenen Wert für den folgenden Tag-Schlüssel nicht zu.	400 Bad Request	Nicht unterstützt
TooManyTags	Die Anzahl der Tags überschreitet das Limit von 50 Tags.	400 Bad Request	Nicht unterstützt

Liste der Amazon S3-Fehlercodes im Zusammenhang mit Anmerkungen

Die folgende Tabelle enthält spezielle Fehler, die bei Annotationsoperationen (PutObjectAnnotation,, GetObjectAnnotation

ListObjectAnnotationsDeleteObjectAnnotation,UpdateBucketMetadataAnnotationTableCo und beim Verhalten im Zusammenhang mit Anmerkungen CopyObject auftreten können.

Allgemeine Informationen zu Amazon S3-Fehlern und eine Liste der Fehlercodes finden Sie unter.

[Fehlermeldungen](#)

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
AccessDenied	Zugriff verweigert. Der Aufrufer ist nicht berechtigt, den angeforderten Kommentarvorgang auszuführen, oder das Objekt ist im COMPLIANCE-Modus durch Objektsperre geschützt.	403 Forbidden	Nicht unterstützt
AnnotationLimitExceeded	Die Anzahl der Objektanmerkungen überschreitet den Grenzwert (1000).	400 Bad Request	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
AnnotationNameTooLong	Ihr Name für die Anmerkung ist zu lang. Namen von Anmerkungen können bis zu 512 Byte lang sein.	400 Bad Request	Nicht unterstützt
AnnotationRequestConflict	Für diese Anmerkung wird derzeit ein widersprüchlicher Vorgang ausgeführt. Dies kann auftreten , wenn eine Quellenmerkung während einer regionsübergreifen den Kopie gelöscht wird.	409 Conflict	Nicht unterstützt
EntityTooSmall	Ihr empfohlener Upload ist kleiner als die zulässige Mindestgröße. Die Nutzlast für Anmerkungen muss mindestens 1 Byte groß sein.	400 Bad Request	Nicht unterstützt
InvalidAnnotationName	Ihr Annotationsname enthält ungültige Zeichen, oder der Name beginnt mit einem reservierten Präfix (awsoder unterscheidet nicht zwischen Groß- s3 und Kleinschreibung). Annotationsnamen können nur Unicode-Buchstaben, Zahlen, Unterstrich (_), Punkt (.) und Bindestrich (-) enthalten. -	400 Bad Request	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
InvalidArgument	Die Annotationsdirektive ist nicht gültig. Gültige Werte sind COPY und EXCLUDE. Für ListObjectAnnotations, max-annotation-results muss zwischen 1 und 1000 liegen.	400 Bad Request	Nicht unterstützt
InvalidPrefix	Das von Ihnen angegebene Annotationspräfix ist nicht gültig.	400 Bad Request	Nicht unterstützt
InvalidRequest	Dieser Fehler kann aus den folgenden Gründen auftreten: Das Objekt ist verschlüsselt mit SSE-C, was keine Anmerkungen unterstützt. Das Quellobjekt enthält Anmerkungen, die nicht in das Ziel kopiert werden können, da sich das Ziel in einer Region befindet, die nicht unterstützt wird. Die Konfigurationsanforderung für die Annotationstabelle ist nicht gültig (z. B. das Deaktivieren der Tabelle beim Angeben der Verschlüsselung, das Aktivieren einer bereits aktivierten Tabelle, das Aktualisieren einer Tabelle im Status CREATING oder das Senden einer leeren Anforderung).	400 Bad Request	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
MethodNotAllowed	Die angegebene Methode ist für diese Ressource nicht zulässig. Dieser Fehler tritt beispielsweise auf, wenn die Anforderung auf eine Löschmarkierung abzielt.	405 Method Not Allowed	Nicht unterstützt
NoSuchAnnotation	Die angegebene Anmerkung ist nicht vorhanden.	404 Not Found	Nicht unterstützt
NoSuchKey	Der angegebene Schlüssel ist nicht vorhanden.	404 Not Found	Nicht unterstützt
NoSuchVersion	Die in der Anfrage angegebene Versions-ID stimmt nicht mit einer vorhandenen Version überein.	404 Not Found	Nicht unterstützt
NotImplemented	Anmerkungen werden für Amazon S3-Verzeichnis-Buckets nicht unterstützt.	501 Not Implement ed	Nicht unterstützt
OperationAborted	Für diese Ressource wird derzeit eine bedingte Operation ausgeführt, die einen Konflikt verursacht. Dies kann bei gleichzeitigen Änderungen auftreten oder wenn sich die Anzahl der Quellenanmerkungen oder ETags während einer regionsübergreifenden Kopie ändert. Bitte versuchen Sie es erneut.	409 Conflict	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
PreconditionFailed	Mindestens eine der von Ihnen angegebenen Vorbedingungen war nicht erfüllt. Das x-amz-object-if-match ETag entspricht nicht dem aktuellen Objekt-ETag.	412 Precondit ion Failed	Nicht unterstützt
UnsupportedMediaTy pe	Die Nutzlast der Anforderung enthält einen Medientyp, der nicht unterstützt wird. Bei der Nutzlast für Anmerkungen muss es sich um gültigen UTF-8 codierten Text handeln.	415 Unsupport ed Media Type	Nicht unterstützt
UserAnnotationName MustBeSpecified	Sie müssen einen Namen für die Benutzeranmerkung angeben.	400 Bad Request	Nicht unterstützt

Liste der Amazon S3-Fehlercodes auf Outposts

Die folgende Tabelle enthält spezielle Fehler, die bei einem Vorgang von Amazon S3 on Outposts möglicherweise zurückgegeben werden. Allgemeine Informationen zu Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlermeldungen](#).

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
BadRequest	Der Bucket befindet sich aufgrund eines früheren Löschversuchs in einem Übergangszustand. Bitte versuchen Sie es später erneut.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
InvalidRequest	Dieser Fehler kann aus folgenden Gründen auftreten: • Eine Amazon VPC-Konfiguration ist erforderlich. • Öffentlicher Zugriff ist auf S3 auf Outposts-Zugangspunkten nicht erlaubt.	400 Bad Request (400 Ungültige Anfrage)	Client
InvalidOutpostState	Die Anfrage ist für den aktuellen Status des Outposts nicht gültig.	409 Conflict (409 Konflikt)	Nicht unterstützt
InvalidRequest	Der Zugangspunkt befindet sich nicht in einem Zustand, in dem er gelöscht werden kann.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
NoSuchOutpost	Der angegebene Außenposten existiert nicht.	404 Not Found (404 Nicht gefunden)	Nicht unterstützt
UnsupportedOperation	Die angegebene Aktion wurde nicht unterstützt.	404 Not Found (404 Nicht gefunden)	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
InsufficientCapaci ty	Ungenügende Kapazität.	507 Ungenügen der Speicherp latz	Nicht unterstützt

Liste der Amazon S3 Storage Lens-Fehlercodes

Die folgende Tabelle enthält spezielle Fehler, die bei Amazon S3 Storage Lens-Vorgängen möglicherweise zurückgegeben werden. Allgemeine Informationen zu allgemeinen Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlermeldungen](#).

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
AccessDenied	Diese Region wird nicht als Heimatregion für S3 Storage Lens unterstützt.	403 Forbidden	Nicht unterstützt
AccountNotAuthoriz ed	Dieses Konto ist nicht zur Verwendung autorisiert AWS Organizations. Verwenden Sie Ihr Verwaltungskonto oder Ihr delegiertes Administratorkonto.	403 Forbidden	Nicht unterstützt
ActivityMetricsMus tEnabled	Aktivitätsmetriken müssen aktiviert sein.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
<code>AWSOrganizationsNotInUseException</code>	Dieses Konto ist nicht Teil Ihrer Organisation.	403 Forbidden	Nicht unterstützt
<code>DefaultConfigurationDeleteForbidden</code>	Die Standardkonfiguration kann nicht gelöscht werden.	403 Forbidden	Nicht unterstützt
<code>DuplicateStorageLensGroupARN</code>	In dieser Konfiguration gibt es zwei oder mehr Einträge desselben Storage Lens-Gruppen-ARN.	400 Bad Request	Nicht unterstützt
<code>EmptyExcludeContainer</code>	Dieser Fehler tritt aus den folgenden Gründen auf: • Der Ausschlusscontainer darf nicht leer sein. • Der Ausschlusscontainer darf keine Null-Buckets enthalten. • Der Ausschlusscontainer darf keine Regionen enthalten.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
<code>EmptyExcludeElement</code>	Sie müssen eine Storage Lens-Gruppe mit Ihrem Exclude-Element angeben.	400 Bad Request	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
EmptyIncludeContai ner	Dieser Fehler tritt aus den folgenden Gründen auf: • Der Include-Container darf nicht leer sein. • Der Include-Container darf keine Null-Buckets enthalten. • Der Include-Container darf keine Regionen enthalten.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidAWSOrgArn	Die Konfiguration enthält einen falsch formatierten AWS Organizations ARN.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
EmptyIncludeElement	Sie müssen eine Storage Lens-Gruppe mit Ihrem Include-Element angeben.	400 Bad Request	Nicht unterstützt
InvalidBucketFilter	Organization-level Konfigurationen unterstützen keine Bucket-Filter.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidConfigId	Die Konfigurations-ID ist nicht gültig.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
InvalidDestination	Der S3-Bucket-ARN ist falsch formatiert.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidEncryptionMethod	Es kann nur eine Verschlüsselungsmethode angegeben werden.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidFilterForDefaultConfiguration	Die Standardkonfiguration darf keine Filter enthalten.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidIncludeExcludeContainers	Sie können in einer Konfiguration entweder einen Include-Container oder einen Exclude-Container angeben. Sie können nicht beide in einer Konfiguration angeben.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidIncludeExcludeElements	Nur ein Include- oder Exclude-Element ist zulässig. Mindestens ein Include- oder Exclude-Element muss vorhanden sein.	400 Bad Request	Nicht unterstützt
InvalidKMSEncryptionKeyId	Die KMS-Schlüssel-ID ARN ist nicht gültig.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
InvalidMaximumPref ixDepth	MaxDepthmuss innerhalb des Bereichs [1,10] liegen.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidMinimumStor ageBytesPercentage	MinStorageBytesPer centage muss innerhalb des Bereichs [1.00,100,00] liegen.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidOrganizatio nARN	Der AWS Organizations ARN in der Konfiguration ist nicht gültig.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidOrganizatio nForDefaultConfigu ration	Die Standardkonfiguration unterstützt keine Metriken auf Organisationsebene.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidRegionForDe faultConfiguration	Die angegebene Region wird für die Standardkonfiguration nicht unterstützt.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
InvalidRegionName	Der Name der Region ist nicht gültig.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidStorageLensArn	Der S3 Storage Lens ARN ist bei der Eingabe nicht erforderlich.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
InvalidStorageLensGroupARN	Dieser ARN für die Storage Lens-Gruppe ist nicht gültig oder es sind nur Storage Lens-Gruppen in Ihrem Konto zulässig. Darüber hinaus müssen Sie der ARN-Struktur der Storage Lens-Gruppe folgen: <code>arn::s3:::storage-lens-group/</code> und das Limit von 64 Zeichen einhalten. Die Gruppennamen von Storage Lens dürfen auch nur die folgenden Zeichen enthalten: a-z, 0-9 A-Z, Bindestriche (-) und Unterstriche (_).	400 Bad Request	Nicht unterstützt
MissingAccountLevelActivityMetrics	Aktivitätsmetriken müssen auf Kontoebene aktiviert werden, wenn Aktivitätsmetriken auf Bucket-Ebene aktiviert werden.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
MissingBucketLevel ActivityMetrics	Aktivitätsmetriken müssen auf Bucket-Ebene aktiviert werden, wenn Aktivitätsmetriken auf Kontoebene aktiviert werden.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
MissingEncryptionM ethod	Die Verschlüsselungsmethode darf nicht leer sein. Geben Sie entweder SSE-KMS oder SSE-S3 an.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
MissingPrefixLevel StorageMetrics	Speichermetriken auf Präfixebene sind obligatorisch, wenn die Präfixebene aktiviert ist.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
OrganizationAccess Denied	Dieses Konto ist nicht zum Hinzufügen autorisiert AWS Organizations.	403 Forbidden	Nicht unterstützt
OrgConfigurationNo tSupported	Die angegebene Region wird AWS Organizations in der Konfiguration nicht unterstützt.	403 Forbidden	Nicht unterstützt
ServiceNotEnabledF orOrg	Die serviceverknüpfte Rolle S3 Storage Lens ist für das Unternehmen nicht aktiviert.	403 Forbidden	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
StorageMetricsMust Enabled	Prefix-level Speichermetriken müssen aktiviert sein.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
TooManyBuckets	Der Buckets-Container kann nicht mehr als 50 Buckets enthalten.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
TooManyRegions	Der Regions-Container kann nicht mehr als 50 Regionen enthalten.	400 Bad Request (400 Ungültige Anfrage)	Nicht unterstützt
TooManyStorageLens Groups	Sie können Ihrem Storage Lens- Dashboard nicht mehr als 50 Storage Lens-Gruppen hinzufüge n.	400 Bad Request	Nicht unterstützt

Die folgende Tabelle enthält spezielle Fehler, die bei Vorgängen von S3 Storage Lens-Gruppen möglicherweise zurückgegeben werden. Allgemeine Informationen zu allgemeinen Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlermeldungen](#).

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
AccessDenied	Sie sind nicht berechtigt, Storage Lens-Gruppenaktionen durchzuführen. Diese Region wird nicht als Heimatregion für S3 Storage Lens-Gruppen unterstützt.	403 Forbidden	Nicht unterstützt
ConfigurationAlreadyExists	Die angegebene Konfiguration ist bereits vorhanden.	409 Conflict	Nicht unterstützt
DuplicateElement	Schlagworte müssen eindeutig sein. Der And logische Operator enthält doppelte Tag-Schlüssel. Der Or logische Operator enthält doppelte Tags. Der logische Operator enthält doppelte Präfixe oder Suffixe.	400 Bad Request	Nicht unterstützt
InvalidAge	DaysLessThan und DaysGreaterThan müssen positive Zahlen sein.	400 Bad Request	Nicht unterstützt
InvalidFilter	Ein Filter muss eines der folgenden Elemente enthalten :And, OrMatchAnyTag ,MatchAnyPrefix ,MatchAnySuffix ,MatchObjectAge ,MatchObjectSize .	400 Bad Request	Nicht unterstützt
InvalidLogicalOperator	In den logischen Operatoren And oder müssen mindestens zwei Unterelemente vorhanden seinOr.	400 Bad Request	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
InvalidMatchAnyPre fix	Der MatchAnyPrefix Parameter darf nicht leer sein.	400 Bad Request	Nicht unterstützt
InvalidMatchAnySuf fix	Der MatchAnySuffix Parameter darf nicht leer sein.	400 Bad Request	Nicht unterstützt
InvalidMatchAnyTag	Der MatchAnyTag Parameter darf nicht leer sein.	400 Bad Request	Nicht unterstützt
InvalidMatchObject Age	Der MatchObjectAge Parameter darf nicht leer sein.	400 Bad Request	Nicht unterstützt
InvalidMatchObject Size	Der MatchObjectSize Parameter darf nicht leer sein.	400 Bad Request	Nicht unterstützt
InvalidName	Der Parameter Name der Storage Lens-Gruppe muss zwischen 1 und 64 Zeichen lang sein. Der Storage Name Lens-Grup penparameter muss das <code>^[a-zA-Z0-9\-_]+\$</code> Muster verwenden.	400 Bad Request	Nicht unterstützt
InvalidNumericComb ination	Diese Kombination aus Objektalter oder Objektgröße ist nicht gültig.	400 Bad Request	Nicht unterstützt
InvalidPrefix	Die maximale Länge eines Präfixes beträgt 1.024 Zeichen. Die Präfixzeichenfolge darf nicht leer sein.	400 Bad Request	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
InvalidSize	BytesLessThan und BytesGreaterThan müssen positive Zahlen sein. Die maximale Objektgröße darf 50 TB nicht überschreiten. Die minimale Objektgröße darf nicht größer oder gleich 50 TB sein.	400 Bad Request	Nicht unterstützt
InvalidSuffix	Die maximale Länge eines Suffixes beträgt 1.024 Zeichen. Die Suffixzeichenfolge darf nicht leer sein.	400 Bad Request	Nicht unterstützt
InvalidTag	Der Objekt-Tag-Schlüssel darf 128 Zeichen nicht überschreiten. Die Schlüsselzeichenfolge für das Objekt-Tag darf weder Null noch leer sein. Die maximale Länge eines Tag-Wertes beträgt 256 Zeichen. Der Objekt-Tag-Schlüssel enthält ungültige Zeichen. Der Objekt-Tag-Schlüssel darf nur a-z, 0-9 A-Z, Leerzeichen und die folgenden Zeichen enthalten: ^(_ . : / = + \ - @] *) \$	400 Bad Request	Nicht unterstützt
MismatchedName	Der in der Anfrage angegebene Name stimmt nicht mit dem Storage Lens-Gruppennamen überein.	400 Bad Request	Nicht unterstützt

Fehlercode	Description	HTTP-Stat uscode	SOAP-Fehl ercode-Pr äfix
TooManyConfigurati ons	Sie haben versucht, mehr Storage Lens-Gruppenkonfigurationen als die 50 zulässigen zu erstellen.	400 Bad Request	Nicht unterstützt
TooManyElements	Der Element überschreitet die maximal zulässige Anzahl von Elementen innerhalb eines logischen Operators. Nur 10 Präfixe, Suffixe oder Tags sind zulässig.	400 Bad Request	Nicht unterstützt

Liste der Amazon S3 Object Lambda-Fehlercodes

Die folgende Tabelle enthält spezielle Fehler, die S3 Object Lambda möglicherweise zurückgibt. Informationen zu allgemeinen Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlermeldungen](#).

Fehlerantworten, die von den unterstützenden Access Points bei GetObject Nichtanfragen eingehen, werden unverändert an den Anrufer gesendet.

Fehlercode	Description	HTTP-Stat uscode	
LambdaInvalidResponse	Wird an den ursprünglichen Anrufer zurückgegeben, wenn mit an WriteGetObjectResponse geantwortet wird. Validatio nError AWS Lambda Weitere Informationen finden Sie in der ValidationError Nachricht . Nicht alle Fälle Validatio nError führen zu einem	400 Bad Request	

Fehlercode	Description	HTTP-Statuscode	
	LambdaInvalidResponse Fehler.		
LambdaInvocationFailed	Der Aufruf der Lambda-Funktion ist fehlgeschlagen. Aufrufer erhalten möglicherweise den folgenden Fehler, wenn S3 Object Lambda die konfigurierte Lambda-Funktion nicht erfolgreich aufrufen kann. Die Fehlermeldung kann Details zu einem eventuellen Fehler enthalten , der vom AWS Lambda Dienst beim Aufrufen der Funktion zurückgegeben wurde (z. B. Statuscode, Fehlercode, Fehlermeldung und Anforderungs-ID).	400 Bad Request	

Fehlercode	Description	HTTP-Statuscode	
LambdaNotFound	Die AWS Lambda Funktion wurde nicht gefunden. Die konfigurierte Lambda-Funktion, die Version oder der Alias wurden beim Versuch, sie aufzurufen, nicht gefunden. Stellen Sie sicher, dass die Konfiguration der Zugriffspunkte von S3 Objekt Lambda auf den korrekten ARN der Lambda-Funktion verweist. Die Fehlermeldung kann Details zu einem eventuellen Fehler enthalten , der vom AWS Lambda Dienst beim Aufrufen der Funktion zurückgegeben wurde (z. B. Statuscode, Fehlercode, Fehlermeldung und Anforderungs-ID).	404 Not Found	

Fehlercode	Description	HTTP-Statuscode	
LambdaPermissionError	Der Aufrufer ist nicht berechtigt, die Lambda-Funktion aufzurufen. Der Aufrufer benötigt die Berechtigung zum Aufrufen der Lambda-Funktion. Überprüfen Sie die Richtlinien, die zu dem Aufrufer gehören, und stellen Sie sicher, dass er <code>lambda:Invoke</code> für die konfigurierte Funktion verwenden darf. Die Fehlermeldung kann Details zu einem eventuellen Fehler enthalten, der vom AWS Lambda Dienst beim Aufrufen der Funktion zurückgegeben wurde (z. B. Statuscode, Fehlercode, Fehlermeldung und Anforderungs-ID).	403 Forbidden	
LambdaResponseNotReceived	Die Lambda-Funktion wurde ohne erfolgreichen Aufruf beendet. <code>WriteGetObjectResponse</code> <code>GetObject</code> Antwortdaten werden von der Lambda-Funktion durch Aufrufen der <code>WriteGetObjectResponse</code> API-Operation bereitgestellt. Die CloudWatch Amazon-Protokolle für die Funktion geben möglicherweise mehr Aufschluss darüber, warum die Funktion diesen API-Vorgang nicht erfolgreich aufgerufen hat, obwohl er normal beendet wurde.	500 Internal Service Error	

Fehlercode	Description	HTTP-Statuscode	
LambdaRuntimeError	Die Lambda-Funktion ist während der Ausführung fehlgeschlagen. Ein expliziter Fehler wurde von der Lambda-Funktion empfangen . Einzelheiten zum Fehler finden Sie in den AWS CloudFormation Protokollen.	500 Internal Service Error	
LambdaTimeout	Die Lambda-Funktion hat nicht in der zulässigen Zeit reagiert. Die Lambda-Funktion konnte ihren Aufruf nicht WriteGetObjectResponse innerhalb von 60 Sekunden abschließen.	500 Internal Service Error	

Fehlercode	Description	HTTP-Statuscode	
SlowDown	Reduzieren Sie Ihre Anforderungsrate für Operationen, die Folgendes beinhalten AWS Lambda. Der Funktionsaufruf wurde gedrosselt AWS Lambda, vielleicht weil die konfigurierte Parallelitätsbeschränkung erreicht wurde. Weitere Informationen finden Sie im Entwicklerhandbuch unter Verwaltung der Parallelität für eine Lambda-Funktion. AWS Lambda Die Fehlermeldung kann Details zu einem eventuellen Fehler enthalten, der vom AWS Lambda Dienst beim Aufrufen der Funktion zurückgegeben wurde (z. B. Statuscode, Fehlercode, Fehlermeldung und Anforderungs-ID).	503 Slow Down	
ValidationError	Validierungsfehler können vom WriteGetObjectResponse API-Vorgang zurückgegeben werden und können aus zahlreichen Gründen auftreten. Weitere Details finden Sie in der Fehlermeldung.	400 Bad Request	

Liste der asynchronen Amazon S3-Fehlercodes

Die folgende Tabelle enthält spezielle Fehler, die bei asynchronen Anfragen möglicherweise zurückgegeben werden. Allgemeine Informationen zu Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlermeldungen](#)

Diese Fehler werden zurückgegeben, wenn Sie den Status einer asynchronen Anfrage abfragen, z. B. mithilfe von `DescribeMultiRegionAccessPointOperation`. Da diese Anfragen asynchron sind, haben alle diese Fehler den Statuscode `200 OK`.

Fehlercode	Description	HTTP-Statuscode	
<code>AccessDenied</code>	Zugriff verweigert.	<code>200 OK</code>	
<code>InternalErrors</code>	Es ist ein interner Serverfehler aufgetreten.	<code>200 OK</code>	
<code>MalformedPolicy</code>	Die angegebene Richtlinienyntax ist nicht gültig.	<code>200 OK</code>	
<code>MultiRegionAccessPointAlreadyOwnedByYou</code>	Sie haben bereits einen Multi-Region Access Point mit demselben Namen.	<code>200 OK</code>	
<code>MultiRegionAccessPointModifiedByAnotherRequest</code>	Die Aktion ist fehlgeschlagen, weil eine andere Anforderung die angegebene Ressource ändert. Versuchen Sie, Ihre Anfrage erneut einzureichen, nachdem die vorherige Anfrage abgeschlossen wurde.	<code>200 OK</code>	
<code>MultiRegionAccessPointNotReady</code>	Der angegebene Multi-Region Access Point ist nicht bereit, aktualisiert zu werden.	<code>200 OK</code>	
<code>MultiRegionAccessPointSameBucketRegion</code>	Die Buckets, die zum Erstellen eines Multi-Region Access Points verwendet werden, dürfen sich nicht in derselben Region befinden.	<code>200 OK</code>	
<code>MultiRegionAccessPointUnsupportedRegion</code>	Einer der Buckets, die zur Erstellung des Multi-Region Access Points bereitgestellt wurden, befindet sich	<code>200 OK</code>	

Fehlercode	Description	HTTP-Stat uscode	
	in einer Region, die nicht unterstützt wird.		
NoSuchBucket	Der angegebene Bucket ist nicht vorhanden.	200 OK	
NoSuchMultiRegionAccessPoint	Der angegebene Multi-Region Access Point existiert nicht.	200 OK	

Liste der Amazon S3 Access Grants-Fehlercodes

Die folgende Tabelle enthält spezielle Fehler, die bei S3 Access Grants-Anfragen möglicherweise zurückgegeben werden. Allgemeine Informationen zu Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlermeldungen](#).

Fehlercode	Description	HTTP-Stat uscode	
AccessGrantAlreadyExists	Die angegebene Zugriffsgewährung ist bereits vorhanden	409	
AccessGrantsInstanceAlreadyExists	Die Access Grants-Instanz ist bereits vorhanden	409	
AccessGrantsInstanceNotEmptyError	Bitte bereinigen Sie die Standorte, bevor Sie die Access Grants-Instanz löschen	400	
AccessGrantsInstanceDoesNotExistError	Die Access Grants-Instanz ist nicht vorhanden	404	
AccessGrantsInstanceResourcePolicyDoesNotExist	Die Ressourcenrichtlinie für die Access Grants Instance ist nicht vorhanden	404	

Fehlercode	Description	HTTP-Statuscode	
AccessGrantsLocationAlreadyExistsError	Der angegebene Speicherort für Zugriffsberechtigungen ist bereits vorhanden	409	
AccessGrantsLocationNotEmptyError	Bitte bereinigen Sie die Zugriffsberechtigungen, bevor Sie den Speicherort für Zugriffsberechtigungen löschen	400	
AccessGrantsLocationsQuotaExceededError	Das Standortkontingent für Zugangsberechtigungen wurde überschritten. Quote für Standorte von Access Grants: <i><value></i> . Bitte wenden Sie sich an S3, wenn eine Erhöhung erforderlich ist.	409	
AccessGrantsQuotaExceededError	Das Kontingent für Zugangsberechtigungen wurde überschritten. Kontingent für Zugriffsberechtigungen: <i><value></i> . Bitte wenden Sie sich an S3, wenn eine Erhöhung erforderlich ist.	409	
InvalidTag	Ihre Anfrage enthält doppelte Tag-Schlüssel. Entfernen Sie die doppelten Tag-Schlüssel und versuchen Sie es erneut.	400	
InvalidAccessGrant	Die angegebene Zugriffsgewährung ist ungültig	400	
InvalidAccessGrantLocation	Der angegebene Standort für Zugriffsberechtigungen ist ungültig	400	

Fehlercode	Description	HTTP-Stat uscode	
InvalidIamRole	Die angegebene IAM-Rolle ist ungültig	400	
InvalidIdentityCenterInstance	Die angegebene Identity Center-Instanz ist ungültig	400	
InvalidResourcePolicy	Die angegebene Ressourcenrichtlinie ist ungültig	400	
InvalidResourcePolicy	Die angegebene Ressourcenrichtlinie ist ungültig	400	
InvalidTag	Diese Anforderung enthält einen Tag-Schlüssel oder -Wert, der nicht gültig ist. Zu den gültigen Zeichen gehören die folgenden: [a-z A-Z +-=._:/]. Tag-Schlüssel können bis zu 128 Zeichen enthalten. Tag-Werte können bis zu 256 Zeichen enthalten.	400	
NoSuchAccessGrantError	Die angegebene Zugriffsgewährung ist nicht vorhanden	404	
NoSuchAccessGrantsLocationError	Der angegebene Standort für Zugriffsberechtigungen ist nicht vorhanden	404	
AccessDenied	Sie haben keine <i><requested permission></i> Berechtigungen für das angeforderte S3-Präfix: <i><requested target></i>	403 Forbidden	

Fehlercode	Description	HTTP-Statuscode	
StsNotAuthorizedError	Beim Aufrufen der GetDataAccess Operation: User: access-grants.s3.amazonaws.com is not authorized to perform: sts:AssumeRole on resource: StsNotAuthorizedError <i><IAM Role ARN></i>	403	
StsPackedPolicyTooLargeError	Beim Aufrufen der GetDataAccess Operation ist ein Fehler aufgetreten (StsPackedPolicyTooLargeError): Das serialisierte Token ist zu groß für die Sitzung	400	
StsValidationError	Die Fehlermeldung variiert je nach Validierungsfehler.	400	
InvalidTags	Tag-Schlüssel können nicht mit einem AWS reservierten Präfix für System-Tags beginnen.“	400	
TooManyTags	Die Anzahl der Tags überschreitet das Limit von 50 Tags. Entfernen Sie einige Tags und versuchen Sie es erneut.	400	

Liste der FSx-related Amazon-Fehlercodes

Die folgende Tabelle enthält spezielle Fehler, die bei Anfragen, die über einen Amazon S3-Zugriffspunkt an Amazon FSx gestellt werden, möglicherweise zurückgeben.

Allgemeine Informationen zu Amazon S3-Fehlern und eine Liste der Fehlercodes finden Sie unter [Fehlermeldungen](#).

Fehlercode	Description	HTTP-Stat uscode	
AccessDenied	Dieser Fehler kann aus folgenden Gründen auftreten: • Der Anrufer ist nicht berechtigt, den angegebenen S3-Vorgang auszuführen. • Der Anrufer ist nicht berechtigt, auf die Amazon FSx-Ressource für den angegebenen Schlüssel zuzugreifen.	403 Forbidden	
InsufficientCapacity	Die maximale Speicherkapazität des Dateisystems wurde erreicht.	507 Insuffici ent Capacity	
InvalidKey	Der angegebene Schlüssel ist nicht gültig.	400 Bad Request	
SlowDown	Bitte reduzieren Sie Ihre Anforderu ngsrate.	503 Slow Down	

Liste der Amazon S3-Tabellen-Fehlercodes

Die folgende Tabelle enthält spezielle Fehler, die Amazon S3 Tables-Operationen möglicherweise zurückgeben. Allgemeine Informationen zu Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlermeldungen](#).

Fehlercode	Description	HTTP-Statuscode
<code>AccessDeniedException</code>	Die Aktion kann nicht ausgeführt werden, da Sie nicht über die erforderliche Berechtigung verfügen.	403 Forbidden
<code>BadRequestException</code>	Die Anfrage ist ungültig oder falsch formatiert.	400 Bad Request
<code>ConflictException</code>	Die Anfrage ist fehlgeschlagen, da ein Konflikt mit einem vorherigen Schreibvorgang besteht. Sie können die Anfrage wiederholen.	409 Conflict
<code>ForbiddenException</code>	Der Anrufer ist nicht berechtigt, die Anfrage zu stellen.	403 Forbidden
<code>InternalServerErrorException</code>	Die Anfrage ist aufgrund eines internen Serverfehlers fehlgeschlagen.	500 Internal Server Error
<code>NotFoundException</code>	Die Anfrage wurde abgelehnt, da die angegebene Ressource nicht gefunden werden konnte.	404 Not Found
<code>TooManyRequestsException</code>	Das Limit für die Anzahl der Anfragen pro Sekunde wurde überschritten.	429 Too Many Requests

Bewährte Methoden für Amazon-S3-Fehler

Viele Fehlermeldungen enthalten zusätzliche strukturierte Daten, die von einem Entwickler gelesen und interpretiert werden, der Programmfehler diagnostiziert. Wenn Sie beispielsweise einen Content-MD5 Header mit einer REST-PUT-Anfrage senden, die nicht mit dem auf dem Server berechneten Digest übereinstimmt, erhalten Sie eine `BadDigest` Fehlermeldung. Die Fehlerantwort enthält als

Detailelemente auch den berechneten Digest und den Digest, den Sie zu erwarten bereitgestellt haben. Während der Entwicklung können Sie diese Informationen nutzen, um den Fehler zu diagnostizieren. In der Produktion könnte ein ordnungsgemäß funktionierendes Programm diese Informationen in seinem Fehlerprotokoll enthalten.

Wenn Sie eine Anwendung für Amazon S3 entwerfen, sollten Sie darauf achten, dass Amazon-S3-Fehler auf geeignete Weise verarbeitet werden. Dieser Abschnitt beschreibt Aspekte, die Sie beim Entwurf Ihrer Anwendung berücksichtigen sollten.

Versuchen Sie es erneut InternalErrors

Interne Fehler sind Fehler, die innerhalb der Amazon-S3-Umgebung auftreten.

Anfragen, die eine InternalError Antwort erhalten, wurden möglicherweise nicht bearbeitet. Wenn beispielsweise eine PUT-Anforderung zurückkehrt InternalError, kann ein nachfolgendes GET den alten oder den aktualisierten Wert abrufen.

Wenn Amazon S3 eine InternalError Antwort zurückgibt, versuchen Sie es erneut mit der Anfrage.

Optimieren Sie die Anwendung für wiederholte Fehler SlowDown

Wie jedes verteilte System verfügt S3 über Schutzmechanismen, die einen vorsätzlichen oder unbeabsichtigten Ressourcenüberverbrauch erkennen und entsprechend reagieren. SlowDown Fehler können auftreten, wenn eine hohe Anforderungsrate einen dieser Mechanismen auslöst. Eine Reduzierung Ihrer Anfragerate verringert oder eliminiert Fehler dieses Typs. Im Allgemeinen treten diese Fehler bei den meisten Benutzern nicht regelmäßig auf. Wenn Sie jedoch weitere Informationen wünschen oder häufig oder unerwartete SlowDown Fehler auftreten, posten Sie dies bitte in unserem [Amazon S3-Entwicklerforum](https://aws.amazon.com/s3-developer-forum/) oder registrieren Sie sich für Support <https://aws.amazon.com/premiumsupport/>.

Isolieren von Fehlern

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Amazon S3 unterstützt verschiedene Fehlercodes, die sowohl von der SOAP, als auch von der REST-API verwendet werden. Die SOAP API gibt Amazon-S3-Standardfehlercodes zurück. Die REST-API ist darauf ausgelegt, wie ein HTTP-Standardserver auszusehen und mit vorhandenen HTTP-Clients zu interagieren (z. B. Browsern, HTTP-Clientbibliotheken, Proxies, Caches usw.). Um sicherzustellen, dass die HTTP-Clients Fehler ordnungsgemäß behandeln, ordnet Amazon S3 jeden Amazon S3-Fehler einem HTTP-Statuscode zu.

HTTP-Statuscodes sind weniger ausdrucksstark als Amazon-S3-Fehlercodes und enthalten weniger Informationen über den Fehler. Beispielsweise werden die Amazon-S3-Fehler `NoSuchKey` und `NoSuchBucket` auf den Statuscode `HTTP 404 Not Found` abgebildet.

Obwohl HTTP-Statuscodes weniger Informationen über den Fehler enthalten, können Clients, die HTTP verstehen, aber nicht die Amazon-S3-API, den Fehler normalerweise korrekt verarbeiten.

Wenn Sie also Fehler verarbeiten oder Amazon-S3-Fehler an Endbenutzer melden, verwenden Sie den Amazon-S3-Fehlercode statt des HTTP-Statuscodes, weil dieser die meisten Informationen über den Fehler enthält. Beim Debuggen Ihrer Anwendung sollten Sie außerdem das lesbare Element `<Details>` der XML-Fehlerantwort heranziehen.

AWS Glossar

Die neueste AWS Terminologie finden Sie im [AWS Glossar](#) in der AWS-Glossar Referenz.

Amazon S3-Ressourcen

Die folgende Tabelle enthält verwandte Ressourcen, die für die Arbeit mit diesem Service nützlich sind.

Ressource	Description
Amazon Simple Storage Service — Benutzerhandbuch	Das Handbuch für die ersten Schritte bietet eine kurze Anleitung zum Service anhand eines einfachen Anwendungsfalls.
Amazon Simple Storage Service — Benutzerhandbuch	Das Entwicklerhandbuch beschreibt, wie Aufgaben mithilfe von Amazon S3-Vorgängen ausgeführt werden.
Häufig gestellte technische Fragen zu Amazon S3	Die häufig gestellten Fragen decken die 20 wichtigsten Fragen ab, die Entwickler zu diesem Produkt gestellt haben.
Amazon S3 — Versionshinweise	Die Versionshinweise geben einen allgemeinen Überblick über die aktuelle Version. Im Einzelnen werden neue Funktionen, Korrekturen und bekannte Probleme vorgestellt.
Tools für Amazon Web Services	Ein zentraler Ausgangspunkt für die Suche nach Dokumentation, Codebeispielen, Versionshinweisen und anderen Informationen, die Ihnen bei der Entwicklung innovativer Anwendungen mit AWS SDKs und Tools helfen.
AWS-Managementkonsole	Mit der Konsole können Sie die meisten Funktionen von Amazon S3 ohne Programmierung ausführen.
Diskussionsforen	Ein Community-Forum für Entwickler, um technische Fragen im Zusammenhang mit Amazon Web Services zu erörtern.
Support Zentrum	Die Startseite für den AWS technischen Support, einschließlich Zugriff auf unsere Entwicklerforen, technische FAQs,

Ressource	Description
	die Seite mit dem Servicestatus und den Premium-Support.
Support	Die wichtigste Webseite mit Informationen über einen persönlichen Support, schnell reagierenden Supportkanal, der Sie bei der Entwicklung und Ausführung von Anwendungen auf AWS Infrastructure Services unterstützt.
Amazon S3-Produktinformationen	Hauptwebsite für Informationen zu Amazon S3
Kontakt	Eine zentrale Anlaufstelle für Anfragen zu AWS Abrechnung, Konto, Veranstaltungen, Missbrauch usw.
Nutzungsbedingungen	Ausführliche Informationen zum Urheberrecht und zur Markennutzung unter Amazon.com und zu weiteren Themen.

Anhang

Themen

- [Anhang: Antwort SelectObjectContent](#)
- [Anhang: OPTIONS-Objekt](#)
- [Anhang: SOAP-API](#)
- [Anhang: Anfragen authentifizieren \(AWS Signatur \(Version 2\)\)](#)
- [Anhang: APIs für die Lifecycle-Konfiguration \(veraltet\)](#)

Anhang: Antwort SelectObjectContent

Description

Die Amazon S3 Select-Operation filtert den Inhalt eines Amazon S3-Objekts auf der Grundlage einer einfachen SQL-Anweisung (Structured Query Language). Da die Antwortgröße dieses Vorgangs unbekannt ist, streamt Amazon S3 Select die Antwort als eine Reihe von Nachrichten und schließt einen Transfer-Encoding Header ein, dessen Wert in der Antwort enthalten ist. **chunked**

Weitere Informationen zu Amazon S3 Select finden Sie unter Inhalte aus Objekten [auswählen](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Weitere Informationen zur Verwendung von SQL mit Amazon S3 Select finden Sie unter [SQL-Referenz für Amazon S3 Select und Amazon Glacier Select](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Antworten

Ein erfolgreicher Amazon S3 Select-Vorgang gibt 200 OK den Statuscode zurück.

Antwort-Header

Bei dieser Implementierung des Vorgangs werden nur Antwort-Header verwendet, die den meisten Antworten gemeinsam sind. Weitere Informationen finden Sie unter [Allgemeine](#) Antwortheader.

Antworttext

Da die Amazon S3 Select-Antwortgröße unbekannt ist, streamt Amazon S3 die Antwort als eine Reihe von Nachrichten und nimmt einen Transfer-Encoding Header mit **chunked** als Wert in die Antwort auf. Das folgende Beispiel zeigt das Antwortformat auf der obersten Ebene:

```
<Message 1>
<Message 2>
<Message 3>
.....
<Message n>
```

Jede Nachricht besteht aus zwei Abschnitten: der Einleitung und den Daten. Der Vorspielabschnitt besteht aus 1) der gesamten Bytelänge der Nachricht und 2) der kombinierten Bytelänge aller Header. Der Datenbereich besteht aus 1) den Headern und 2) einer Nutzlast.

Jeder Abschnitt endet mit einer 4-Byte-Big-Endian-Integer-Prüfsumme (CRC). Amazon S3 Select verwendet CRC32 (oft als GZIP CRC32 bezeichnet), um beide CRCs zu berechnen. Weitere Informationen zu CRC32 finden Sie unter [GZIP-Dateiformatspezifikation Version 4.3](#).

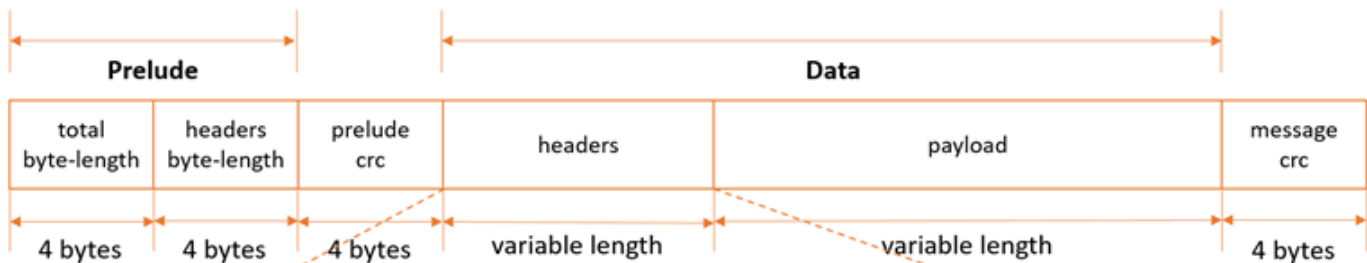
Der gesamte Nachrichten-Overhead einschließlich des Vorspiels und beider Prüfsummen beträgt 16 Byte.

Note

Alle integer Werte in Nachrichten sind in der Netzwerk-Byte-Reihenfolge oder in der Big-Endian-Reihenfolge angegeben.

Das folgende Diagramm zeigt die Komponenten, aus denen eine Nachricht und ein Header bestehen. Beachten Sie, dass es mehrere Header pro Nachricht gibt.

Message:



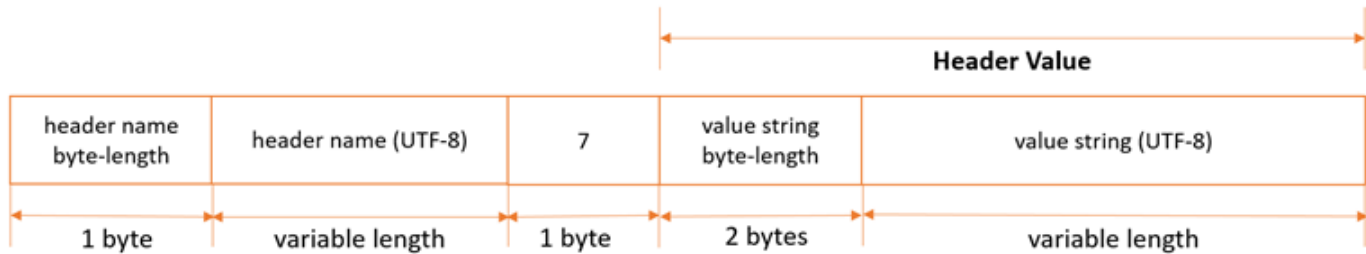
Headers (multiple headers per message):



Note

Für Amazon S3 Select ist der Header-Werttyp immer 7 (type=STRING). Bei diesem Typ besteht der Header-Wert aus zwei Komponenten, einer 2-Byte-Big-Endian-Ganzzahl und einer UTF-8 Zeichenfolge, die diese Byte-Länge hat. Das folgende Diagramm zeigt die Komponenten, aus denen die Amazon S3 Select-Header bestehen.

Amazon S3 Select Headers (type=String):



Berechnungen der Nutzlast in der Bytelänge (diese beiden Berechnungen sind gleichwertig):

- $\text{payload\_length} = \text{total\_length} - \text{header\_length} - \text{sizeof}(\text{total\_length}) - \text{sizeof}(\text{header\_length}) - \text{sizeof}(\text{prelude\_crc}) - \text{sizeof}(\text{message\_crc})$
- Nutzlastlänge = Gesamtlänge — Kopfzeilenlänge — 16

Jede Nachricht enthält die folgenden Komponenten:

- Präludium: Immer feste Größe von 8 Byte (zwei Felder mit jeweils 4 Byte):
 - Die ersten vier Byte: Gesamtbytelänge: Big-endian Ganzzahl-Bytelänge der gesamten Nachricht (einschließlich des Felds mit der Gesamtlänge von 4 Byte selbst).
 - Die zweiten vier Byte: Bytelänge der Kopfzeile: Big-endian Ganzzahl-Bytelänge des Header-Teils der Nachricht (mit Ausnahme des Felds für die Header-Länge selbst).
- Prelude CRC: 4-Byte-Big-Endian-Integer-Prüfsumme (CRC) für den Auftaktteil der Nachricht (ohne den CRC selbst). Das Präludium hat einen vom Nachrichten-CRC getrennten CRC (siehe unten), um sicherzustellen, dass beschädigte Informationen mit Bytelänge sofort erkannt werden können, ohne dass es zu pathologischem Pufferverhalten kommt.
- Header: Eine Reihe von Metadaten, die die Nachricht mit Anmerkungen versehen, z. B. der Nachrichtentyp, das Nutzdatenformat usw. Nachrichten können mehrere Header haben, sodass dieser Teil der Nachricht je nach Nachrichtentyp unterschiedliche Bytelängen haben kann. Header sind Schlüssel-Wert-Paare, wobei sowohl der Schlüssel als auch der Wert Zeichenketten sind. UTF-8 Header können in beliebiger Reihenfolge innerhalb des Header-Bereichs der Nachricht erscheinen, und jeder angegebene Header-Typ kann nur einmal vorkommen.

Für Amazon S3 Select finden Sie im Folgenden eine Liste mit Header-Namen und den Satz gültiger Werte, abhängig vom Nachrichtentyp.

- MessageType Kopfzeile:
 - HeaderName => „:Nachrichtentyp“
 - Gültig HeaderValues => „Fehler“, „Ereignis“
- EventType Kopfzeile:
 - HeaderName => „:Ereignistyp“
 - Valid HeaderValues => „Datensätze“, „Cont“, „Progress“, „Stats“, „End“
- ErrorCode Kopfzeile:
 - HeaderName => „:Fehlercode“
 - Gültig HeaderValues => Fehlercode aus der Tabelle im Abschnitt. [Liste der SELECT-Fehlercodes für den Objekteinhalt](#)
- ErrorMessage Kopfzeile:
 - HeaderName => „:Fehlermeldung“
 - Gültig HeaderValues => Vom Service zurückgegebene Fehlermeldung, um Fehler auf Anforderungsebene zu diagnostizieren.
- Payload: Kann alles sein.
- Nachrichten-CRC: 4-Byte-Big-Endian-Integer-Prüfsumme (CRC) vom Anfang der Nachricht bis zum Anfang der Prüfsumme (d. h. alles in der Nachricht, außer dem Nachrichten-CRC selbst).

Jeder Header enthält die folgenden Komponenten. Es kann mehrere Header pro Nachricht geben.

- Header-Name Byte-Length: Byte-length des Header-Namens.
- Header-Name: Name der Kopfzeile, der den Header-Typ angibt. Gültige Werte: „:message-type“ „:event-type“ „:error-code“ „:error-message“
- Header-Werttyp: Enum, das den Header-Werttyp angibt. Für Amazon S3 Select ist dies immer 7.
- Wertzeichenfolge Byte-Length: (Für Amazon S3 Select) Byte-length der Header-Wertzeichenfolge.
- Zeichenfolge mit Header-Wert: (Für Amazon S3 Select) Wert der Header-Zeichenfolge. Die gültigen Werte für dieses Feld variieren je nach Typ des Headers. In den folgenden Abschnitten finden Sie gültige Werte für jeden Headertyp und Nachrichtentyp.

Bei Amazon S3 Select können Antworten Nachrichten der folgenden Typen sein:

- **Nachricht mit Datensätzen:** Kann einen einzelnen Datensatz, Teildatensätze oder mehrere Datensätze enthalten. Je nach Größe des Ergebnisses kann eine Antwort eine oder mehrere dieser Meldungen enthalten.
- **Fortsetzungsnachricht:** Amazon S3 sendet diese Nachricht regelmäßig, um die TCP-Verbindung aufrechtzuerhalten. Diese Nachrichten erscheinen in den Antworten nach dem Zufallsprinzip. Der Client muss den Nachrichtentyp erkennen und entsprechend verarbeiten.
- **Fortschrittsmeldung:** Amazon S3 sendet diese Meldung regelmäßig, falls gewünscht. Sie enthält Informationen über den Fortschritt einer Abfrage, die gestartet, aber noch nicht abgeschlossen wurde.
- **Statistikmeldung:** Amazon S3 sendet diese Nachricht am Ende der Anfrage. Sie enthält Statistiken über die Abfrage.
- **Endnachricht:** Zeigt an, dass die Anfrage abgeschlossen ist und keine weiteren Nachrichten gesendet werden. Sie sollten nicht davon ausgehen, dass die Anfrage abgeschlossen ist, bis der Client eine End Nachricht erhält.
- **RequestLevelError Nachricht:** Amazon S3 sendet diese Nachricht, wenn die Anfrage aus irgendeinem Grund fehlgeschlagen ist. Sie enthält den Fehlercode und die Fehlermeldung für den Fehler. Wenn Amazon S3 eine RequestLevelError Nachricht sendet, wird keine End Nachricht gesendet.

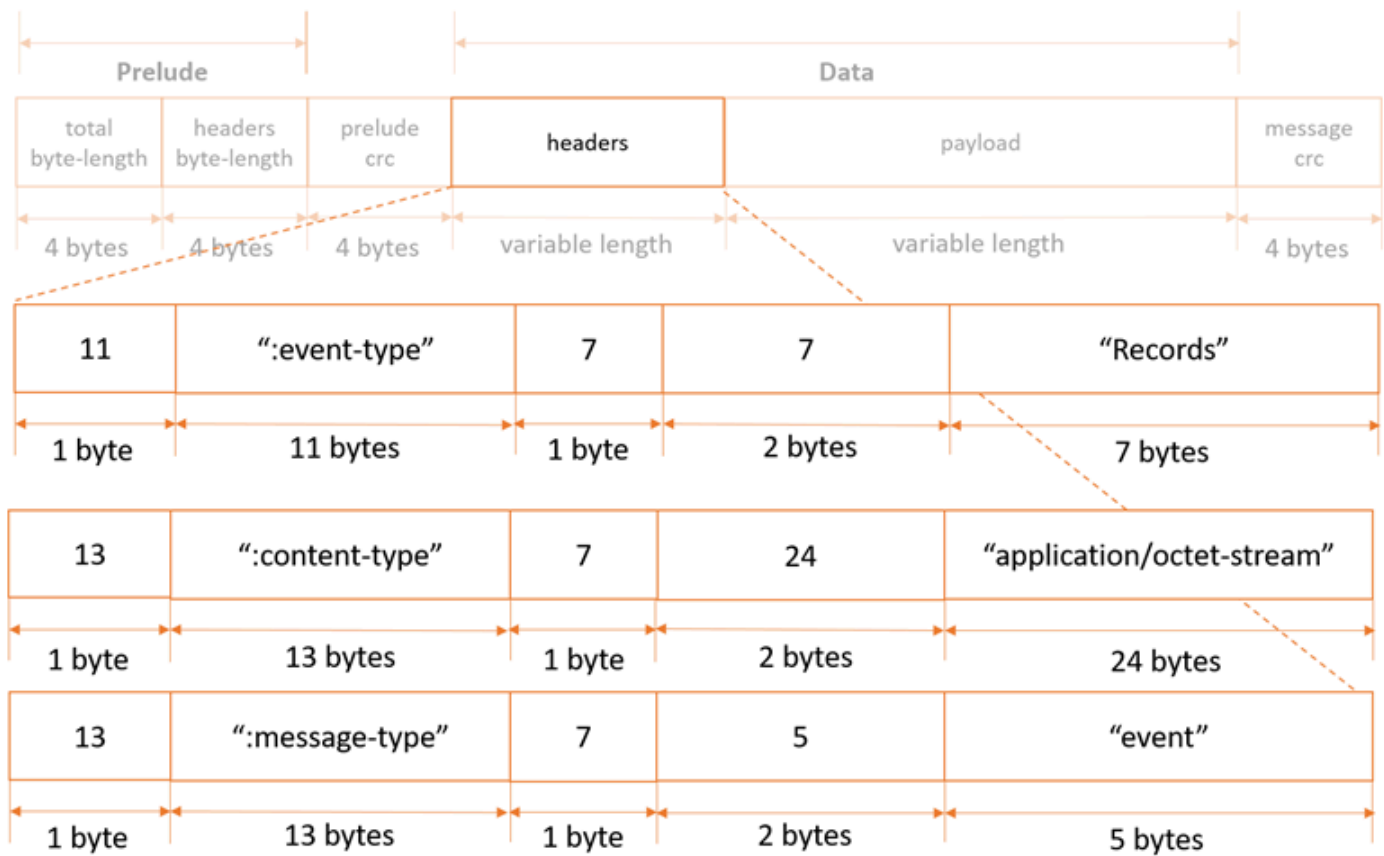
In den folgenden Abschnitten wird die Struktur der einzelnen Nachrichtentypen ausführlicher erläutert.

Beispielcode und Komponententests, die dieses Protokoll verwenden, finden Sie [auf der GitHub Website unter](#) AWS C Event Stream.

Zeichnet die Nachricht auf

Header-Spezifikation

Datensatznachrichten enthalten drei Header, wie folgt:



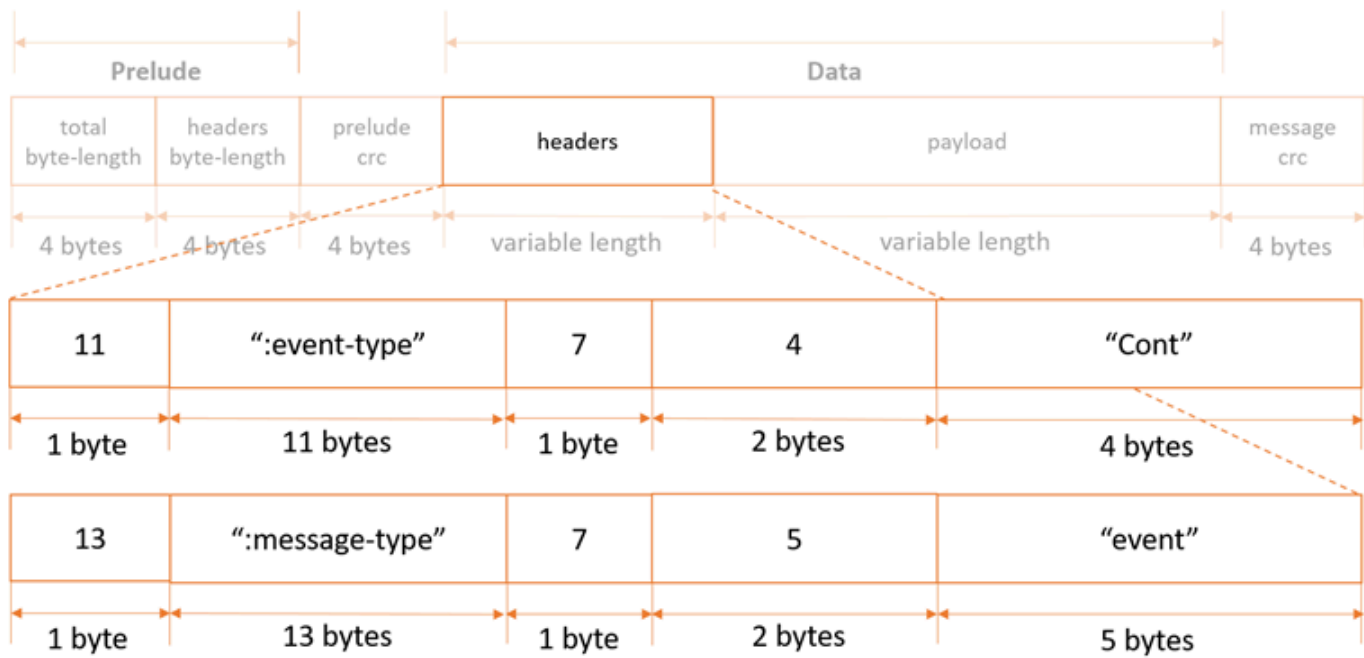
Spezifikation der Nutzlast

Payloads für Datensatznachrichten können einen einzelnen Datensatz, Teildatensätze oder mehrere Datensätze enthalten.

Meldung zur Fortsetzung

Header-Spezifikation

Fortsetzungsnachrichten enthalten zwei Header, wie folgt:



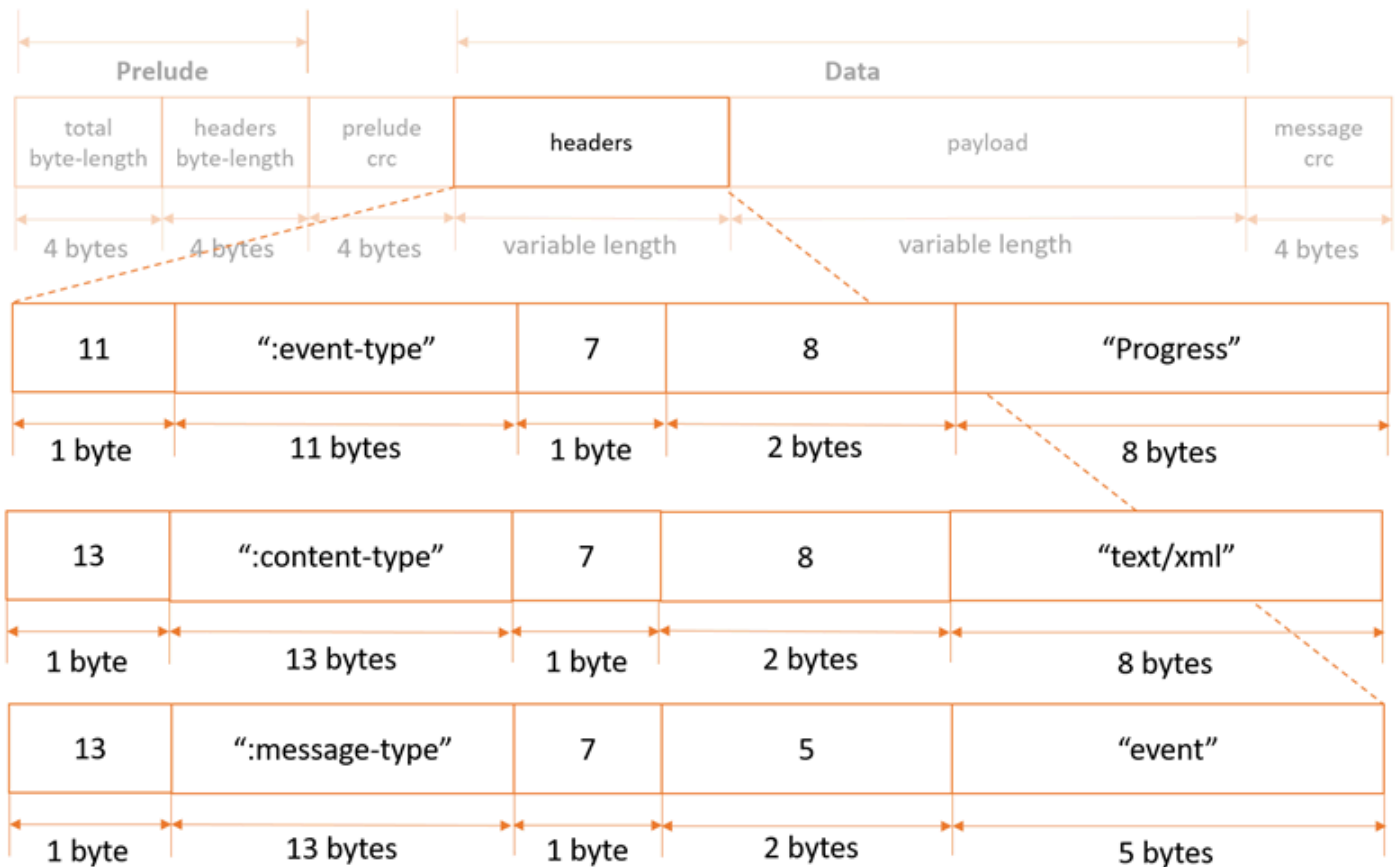
Spezifikation der Nutzlast

Fortsetzungsnachrichten haben keine Nutzlast.

Fortschrittsmeldung

Header-Spezifikation

Fortschrittsmeldungen enthalten drei Header, wie folgt:



Spezifikation der Nutzlast

Die Nutzlast für Fortschrittsnachrichten ist ein XML-Dokument, das Informationen über den Fortschritt einer Anfrage enthält.

- **BytesScanned**=> Anzahl der Byte, die vor der Dekomprimierung verarbeitet wurden (falls die Datei komprimiert ist).
- **BytesProcessed**=> Anzahl der Bytes, die nach dem Entpacken verarbeitet wurden (falls die Datei komprimiert ist).
- **BytesReturned**=> Aktuelle Anzahl der von Amazon S3 zurückgegebenen Nutzlastdaten in Byte für Datensätze.

Für unkomprimierte Dateien **BytesProcessed** sind **BytesScanned** und gleich.

Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
```

```

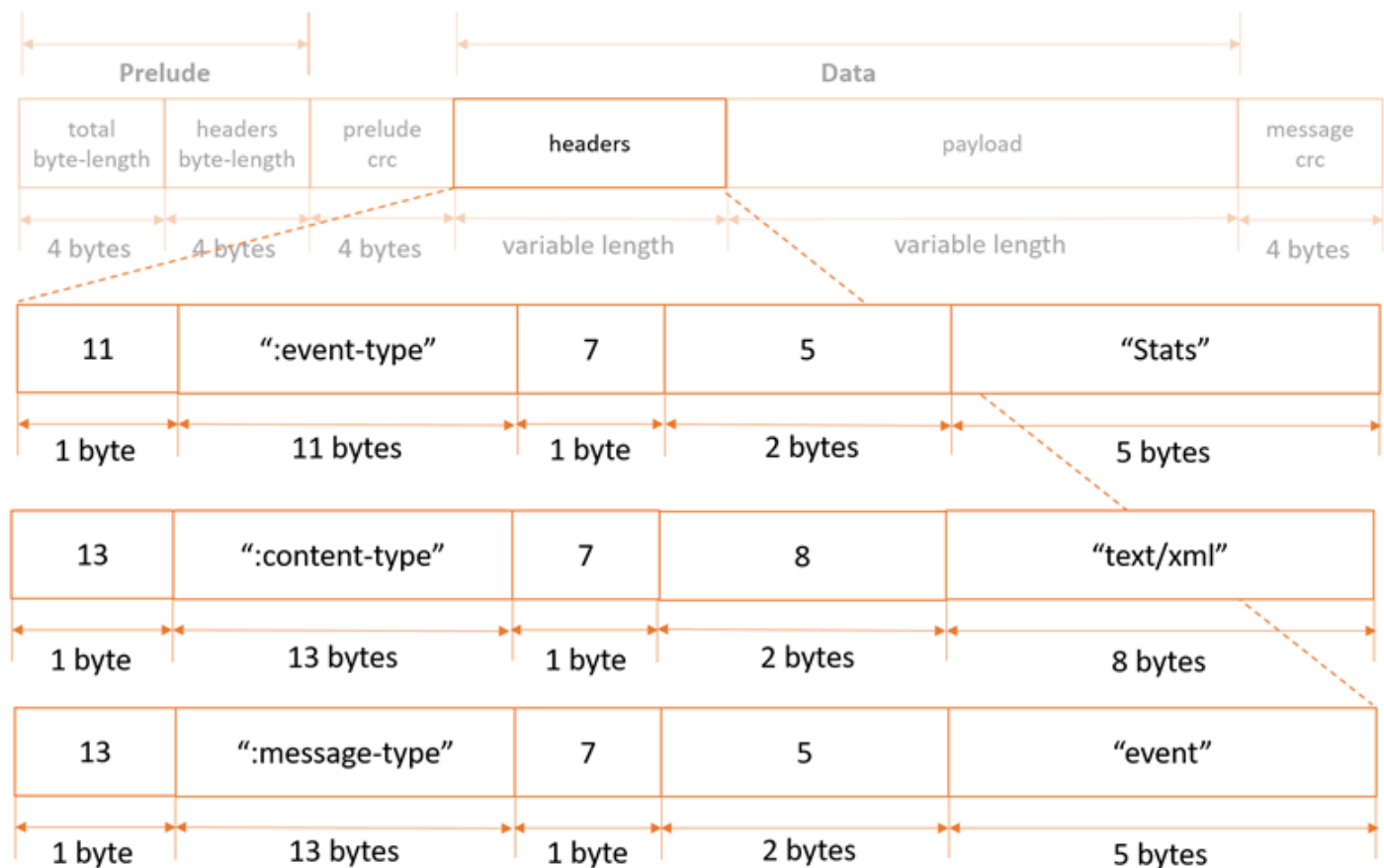
<Progress>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Progress>

```

Statistische Meldung

Header-Spezifikation

Statistiknachrichten enthalten drei Header, wie folgt:



Spezifikation der Nutzlast

Die Nutzlast der Statistiknachricht ist ein XML-Dokument, das Informationen über die Statistiken einer Anfrage enthält, wenn die Verarbeitung abgeschlossen ist.

- BytesScanned=> Anzahl der Bytes, die vor der Dekomprimierung verarbeitet wurden (falls die Datei komprimiert ist).

- BytesProcessed=> Anzahl der Bytes, die nach dem Entpacken verarbeitet wurden (falls die Datei komprimiert ist).
- BytesReturned=> Gesamtzahl der Byte der von Amazon S3 zurückgegebenen Nutzdaten für Datensätze.

Für unkomprimierte Dateien BytesProcessed sind BytesScanned und gleich.

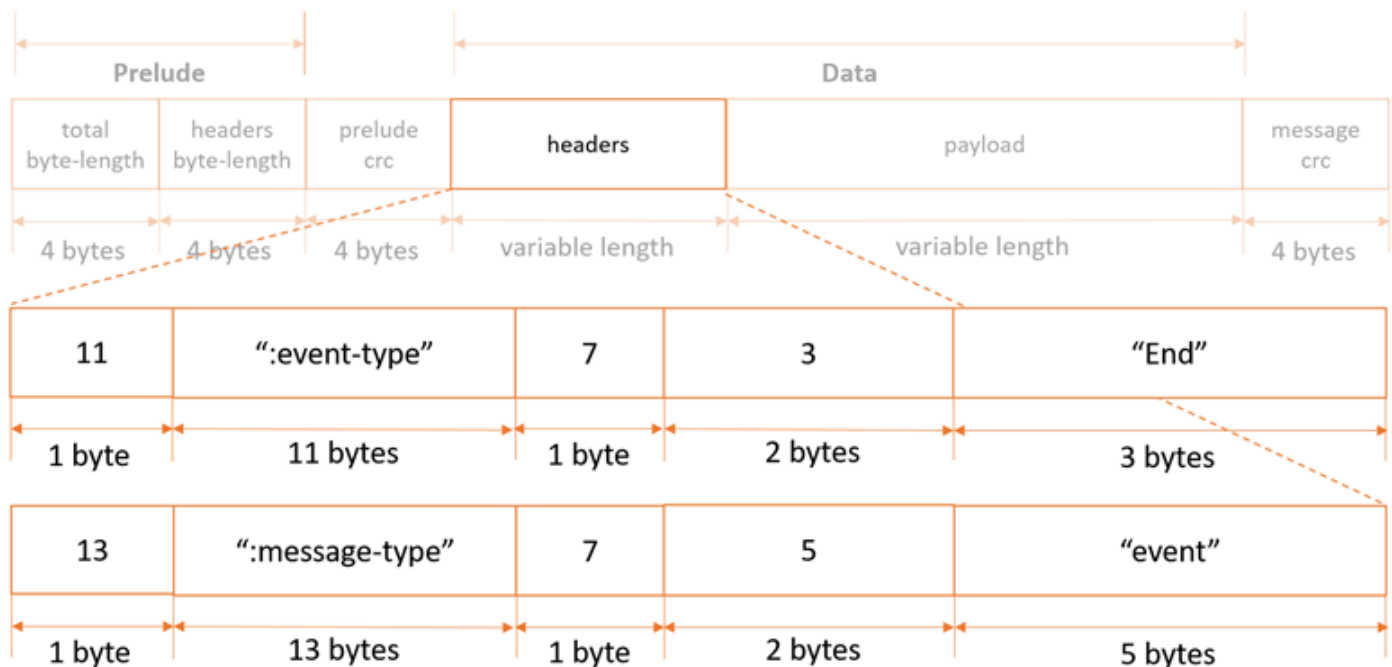
Beispiel:

```
<?xml version="1.0" encoding="UTF-8"?>
<Stats>
  <BytesScanned>512</BytesScanned>
  <BytesProcessed>1024</BytesProcessed>
  <BytesReturned>1024</BytesReturned>
</Stats>
```

Nachricht beenden

Header-Spezifikation

Endnachrichten enthalten zwei Header, wie folgt:



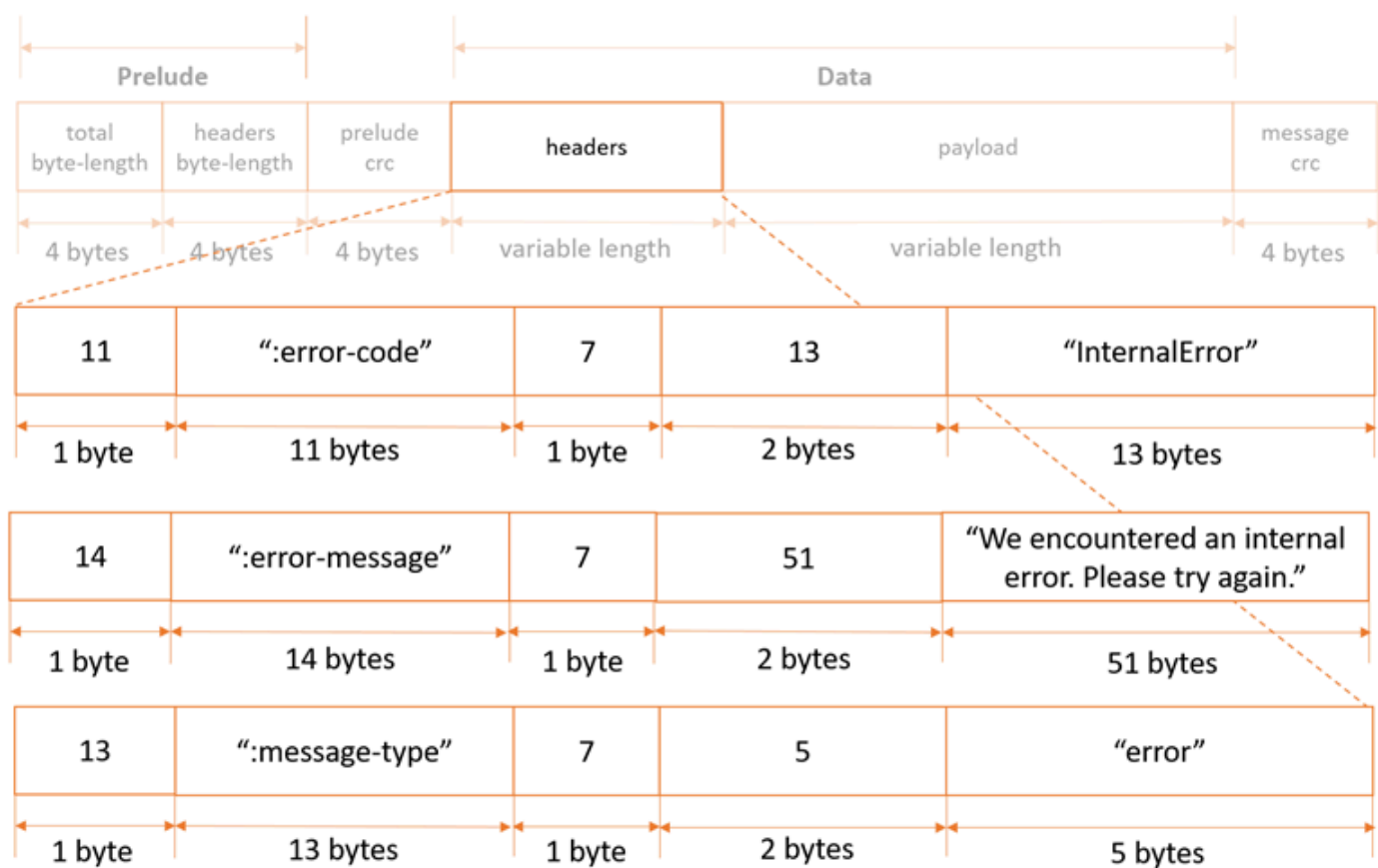
Spezifikation der Nutzlast

Endnachrichten haben keine Nutzlast.

Fehlermeldung auf Anforderungsebene

Header-Spezifikation

Request-level Fehlermeldungen enthalten drei Header, wie folgt:



Eine Liste möglicher Fehlercodes und Fehlermeldungen finden Sie im [Liste der SELECT-Fehlercodes für den Objekteinhalt](#).

Spezifikation der Nutzlast

Request-level Fehlermeldungen haben keine Nutzlast.

Verwandte Ressourcen

- [SelectObjectContent](#)

- [GetObject](#)
- [GetBucketLifecycleConfiguration](#)
- [PutBucketLifecycleConfiguration](#)

Anhang: OPTIONS-Objekt

Description

Ein Browser kann diese Preflight-Anfrage an Amazon S3 senden, um festzustellen, ob eine tatsächliche Anfrage mit dem spezifischen Ursprung, der HTTP-Methode und den Headern gesendet werden kann.

Amazon S3 unterstützt Cross-Origin Resource Sharing (CORS), indem es Ihnen ermöglicht, eine cors Unterressource zu einem Bucket hinzuzufügen. Wenn ein Browser diese Preflight-Anfrage sendet, reagiert Amazon S3, indem es die in der Konfiguration definierten Regeln auswertet. cors

Wenn im Bucket nicht aktiviert cors ist, gibt Amazon S3 eine 403 Forbidden Antwort zurück.

Weitere Informationen zu CORS finden Sie [im Amazon Simple Storage Service-Benutzerhandbuch unter](#) Enabling Cross-Origin Resource Sharing.

Anforderungen

Syntax

```
OPTIONS /ObjectName HTTP/1.1
Host: BucketName.s3.amazonaws.com
Origin: Origin
Access-Control-Request-Method: HTTPMethod
Access-Control-Request-Headers: RequestHeader
```

Anforderungsparameter

Bei diesem Vorgang werden keine spezifischen Anforderungsparameter eingeführt, er kann jedoch alle Anforderungsparameter enthalten, die für die eigentliche Anfrage erforderlich sind.

Anforderungsheader

Name	Description	Erforderlich
Origin	Identifiziert den Ursprung der ursprungsübergreifenden Anfrage an Amazon S3. Beispiel: http://www.example.com.	Ja

Name	Description	Erforderlich
	Typ: Zeichenfolge Standard: keiner	
Access-Control-Request-Method	Identifiziert, welche HTTP-Methode in der tatsächlichen Anfrage verwendet wird. Typ: Zeichenfolge Standard: keiner	Ja
Access-Control-Request-Headers	Eine kommasetrennte Liste von HTTP-Headern, die in der eigentlichen Anfrage gesendet werden. Um beispielsweise ein Objekt mit serverseitiger Verschlüsselung zu versehen, bestimmt diese Preflight-Anfrage, ob der Header in die Anfrage aufgenommen werden kann. <code>x-amz-server-side-encryption</code> Typ: Zeichenfolge Standard: keiner	Nein

Anfrage-Elemente

Diese Implementierung des Vorgangs verwendet keine Anforderungselemente.

Antworten

Antwort-Header

Header	Description
Access-Control-Allow-Origin	

Header	Description
	Der Ursprung, den Sie in Ihrer Anfrage gesendet haben. Wenn der Ursprung Ihrer Anfrage nicht zulässig ist, nimmt Amazon S3 diesen Header nicht in die Antwort auf. Typ: Zeichenfolge
Access-Control-Max-Age	Gibt an, wie lange (in Sekunden) die Ergebnisse der Preflight-Anfrage zwischengespeichert werden können. Typ: Zeichenfolge
Access-Control-Allow-Methods	Die HTTP-Methode, die in der ursprünglichen Anfrage gesendet wurde. Wenn die Methode in der Anfrage nicht zulässig ist, nimmt Amazon S3 diesen Header nicht in die Antwort auf. Typ: Zeichenfolge
Access-Control-Allow-Headers	Eine durch Kommas getrennte Liste von HTTP-Headern, die der Browser in der eigentlichen Anfrage senden kann. Wenn einer der angeforderten Header nicht zulässig ist, nimmt Amazon S3 diesen Header nicht in die Antwort auf, und die Antwort enthält auch keinen der Header mit dem Präfix. <code>Access-Control</code> Typ: Zeichenfolge
Access-Control-Expose-Headers	Eine durch Kommas getrennte Liste von HTTP-Headern. Dieser Header bietet dem JavaScript Client Zugriff auf diese Header in der Antwort auf die eigentliche Anfrage. Typ: Zeichenfolge

Antwortelemente

Diese Implementierung der Operation gibt keine Antwortelemente zurück.

Beispiele

Beispiel: Senden Sie eine Preflight-OPTIONS-Anfrage an einen **Cors-fähigen Bucket**

Ein Browser kann diese Preflight-Anfrage an Amazon S3 senden, um festzustellen, ob er die tatsächliche PUT-Anfrage vom `http://www.example.com` Ursprung an den angegebenen Amazon S3-Bucket senden kann. `examplebucket`

Beispielanforderung

```
OPTIONS /exampleobject HTTP/1.1
Host: examplebucket.s3.amazonaws.com
Origin: http://www.example.com
Access-Control-Request-Method: PUT
```

Beispielantwort

```
HTTP/1.1 200 OK
x-amz-id-2: 6SvaESv3VULYPLik5LLl7lSPPtSnBvDdGmnk1X1HfU17uS2m1DF6td6KWKNjYMXZ
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
```

Verwandte Ressourcen

- [GetBucketCors](#)
- [DeleteBucketCors](#)
- [PutBucketCors](#)

Anhang: SOAP-API

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

In diesem Abschnitt wird die SOAP-API in Bezug auf Service-, Bucket- und Objektoperationen beschrieben. Beachten Sie, dass sowohl authentifizierte als auch anonyme SOAP-Anfragen mithilfe von SSL an Amazon S3 gesendet werden müssen. Amazon S3 gibt einen Fehler zurück, wenn Sie eine SOAP-Anfrage über HTTP senden.

Die neueste Amazon S3-WSDL ist unter docs.aws.amazon verfügbar. [com/200601/AmazonS3-03-wsdl](https://docs.aws.amazon.com/200601/AmazonS3-03-wsdl).

Themen

- [Operationen im Dienst \(SOAP-API\)](#)
- [Operationen mit Buckets \(SOAP-API\)](#)
- [Operationen an Objekten \(SOAP-API\)](#)
- [Authentifizieren von SOAP-Anforderungen](#)
- [Festlegen der Zugriffsrichtlinie mit SOAP](#)
- [Allgemeine Elemente](#)
- [Antworten auf SOAP-Fehler](#)

Operationen im Dienst (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, dass Sie entweder die REST-API oder die AWS SDKs verwenden.

In diesem Abschnitt werden Vorgänge beschrieben, die Sie mit dem Amazon S3-Service ausführen können.

Themen

- [ListAllMyBuckets \(SOAP-API\)](#)

ListAllMyBuckets (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Der ListAllMyBuckets Vorgang gibt eine Liste aller Buckets zurück, die dem Absender der Anfrage gehören.

Example

Sample Request

```
<ListAllMyBuckets xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListAllMyBuckets>
```

Sample Response

```
<ListAllMyBucketsResult xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <Owner>
    <ID>bcaf1ffd86f41161ca5fb16fd081034f</ID>
    <DisplayName>webfile</DisplayName>
  </Owner>
  <Buckets>
    <Bucket>
      <Name>quotes</Name>
      <CreationDate>2006-02-03T16:45:09.000Z</CreationDate>
```

```
</Bucket>
<Bucket>
  <Name>samples</Name>
  <CreationDate>2006-02-03T16:41:58.000Z</CreationDate>
</Bucket>
</Buckets>
</ListAllMyBucketsResult>
```

Antworttext

- **Owner :**

Dies liefert Informationen, die Amazon S3 verwendet, um Ihre Identität für Zwecke der Authentifizierung und Zugriffskontrolle darzustellen. Die ID ist eine eindeutige und permanente Kennung für den Entwickler, der die Anfrage gestellt hat. DisplayName ist ein für Menschen lesbarer Name, der den Entwickler darstellt, der die Anfrage gestellt hat. Er ist nicht eindeutig und kann sich im Laufe der Zeit ändern. Wir empfehlen Ihnen, Ihren Namen mit Ihrem DisplayName Forennamen abzugleichen.

- **Name :**

Der Name eines Buckets. Beachten Sie, dass, wenn einer Ihrer Buckets kürzlich gelöscht wurde, der Name des gelöschten Buckets möglicherweise noch eine Zeit lang in dieser Liste enthalten ist.

- **CreationDate :**

Die Uhrzeit, zu der der Bucket erstellt wurde.

Zugriffskontrolle

Sie müssen sich mit einer gültigen AWS Zugriffsschlüssel-ID authentifizieren. Anonyme Anfragen dürfen niemals Buckets auflisten, und Sie können nur Buckets auflisten, deren Besitzer Sie sind.

Operationen mit Buckets (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

In diesem Abschnitt werden Vorgänge beschrieben, die Sie mit Amazon S3-Buckets ausführen können.

Themen

- [CreateBucket \(SOAP-API\)](#)
- [DeleteBucket \(SOAP-API\)](#)
- [ListBucket \(SOAP-API\)](#)
- [GetBucketAccessControlPolicy \(SOAP-API\)](#)
- [SetBucketAccessControlPolicy \(SOAP-API\)](#)
- [GetBucketLoggingStatus \(SOAP-API\)](#)
- [SetBucketLoggingStatus \(SOAP-API\)](#)

CreateBucket (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Der `CreateBucket` Vorgang erstellt einen Bucket. Nicht jede Zeichenfolge ist ein akzeptabler Bucket-Name. Informationen zu den Einschränkungen bei der Benennung von Buckets finden Sie unter [Arbeiten mit Amazon S3-Buckets](#).

Note

Um festzustellen, ob ein Bucket-Name existiert, verwenden Sie `ListBucket` und setzen Sie ihn `MaxKeys` auf 0. Eine `NoSuchBucket` Antwort gibt an, dass der Bucket verfügbar ist, eine `AccessDenied` Antwort gibt an, dass der Bucket einer anderen Person gehört, und eine Erfolgs-Antwort gibt an, dass Sie Eigentümer des Buckets sind oder die Berechtigung haben, darauf zuzugreifen.

Example Erstellen Sie einen Bucket mit dem Namen „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>
```

Elemente

- **Bucket**: Der Name des Buckets, den Sie erstellen möchten.
- **AccessControlList**: Die Zugriffskontrollliste für den neuen Bucket. Das Element ist optional. Falls nicht angegeben, wird der Bucket mit einer Zugriffsrichtlinie erstellt, die dem Anforderer FULL_CONTROL-Zugriff gewährt.

Zugriffskontrolle

Sie müssen sich mit einer gültigen AWS Zugriffsschlüssel-ID authentifizieren. Anonyme Anfragen dürfen niemals Buckets erstellen.

Verwandte Ressourcen

- [ListBucket \(SOAP-API\)](#)

DeleteBucket (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Der DeleteBucket Vorgang löscht einen Bucket. Alle Objekte im Bucket müssen gelöscht werden, bevor der Bucket selbst gelöscht werden kann.

Example

In diesem Beispiel wird der Bucket „Quotes“ gelöscht.

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>
```

Elemente

- **Bucket**: Der Name des Buckets, den Sie löschen möchten.

Zugriffskontrolle

Nur der Besitzer eines Buckets darf ihn löschen, unabhängig von der Zugriffskontrollrichtlinie für den Bucket.

ListBucket (SOAP-API)

 Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Der ListBucket Vorgang gibt Informationen zu einigen Elementen im Bucket zurück.

Eine allgemeine Einführung in den Listenvorgang finden Sie unter Objektschlüssel [auflisten](#).

Anforderungen

In diesem Beispiel werden bis zu 1000 Schlüssel im Bucket „Anführungszeichen“ aufgeführt, die das Präfix „notes“ haben.

Syntax

```
<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>
```

Parameter

Name	Description	Erforderlich
prefix	Beschränkt die Antwort auf Schlüssel, die mit dem angegebenen Präfix beginnen. Sie können Präfixe verwenden, um einen	Nein

Name	Description	Erforderlich
	Bucket in verschiedene Schlüsselsätze zu unterteilen, ähnlich wie ein Dateisystem Ordner verwendet. Typ: Zeichenfolge Standard: keiner	
<code>marker</code>	Gibt an, an welcher Stelle im Bucket mit der Auflistung begonnen werden soll. Die Liste enthält nur Schlüssel, die lexikografisch hinter der Markierung vorkommen. Das ist praktisch für die Seitennummerierung: Um zur nächsten Ergebnisseite zu gelangen, verwenden Sie den letzten Schlüssel der aktuellen Seite als Markierung. Typ: Zeichenfolge Standard: keiner	Nein
<code>max-keys</code>	Die maximale Anzahl von Schlüsseln, die Sie im Antworttext sehen möchten. Der Server gibt möglicherweise weniger als so viele Schlüssel zurück, aber nicht mehr. Typ: Zeichenfolge Standard: keiner	Nein
<code>delimiter</code>	Bewirkt, dass Schlüssel, die dieselbe Zeichenfolge zwischen dem Präfix und dem ersten Vorkommen des Trennzeichens enthalten, zu einem einzigen Ergebniselement in der CommonPrefixes Auflistung zusammengefasst werden. Diese zusammengefassten Schlüssel werden an keiner anderen Stelle in der Antwort zurückgegeben. Typ: Zeichenfolge Standard: keiner	Nein

Antwort: Erfolg

Bei dieser Antwort wird davon ausgegangen, dass der Bucket die folgenden Schlüssel enthält:

```
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
```

Syntax

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Name>backups</Name>
  <Prefix>notes/</Prefix>
  <MaxKeys>1000</MaxKeys>
  <Delimiter></Delimiter>
  <IsTruncated>>false</IsTruncated>
  <Contents>
    <Key>notes/todos.txt</Key>
    <LastModified>2006-01-01T12:00:00.000Z</LastModified>
    <ETag>"828ef3fdfa96f00ad9f27c383fc9ac7f"</ETag>
    <Size>5126</Size>
    <StorageClass>STANDARD</StorageClass>
    <Owner>
      <ID>75aa57f09aa0c8caeab4f8c24e99d10f8e7faeebf76c078efc7c6caea54ba06a</ID>
      <DisplayName>webfile</DisplayName>
    </Owner>
    <StorageClass>STANDARD</StorageClass>
  </Contents>
  <CommonPrefixes>
    <Prefix>notes/2005-05-23/</Prefix>
  </CommonPrefixes>
  <CommonPrefixes>
    <Prefix>notes/2005-05-28/</Prefix>
  </CommonPrefixes>
</ListBucketResult>
```

Wie Sie sehen können, geben viele Felder in der Antwort die Anforderungsparameter wieder. `IsTruncated`, `Contents`, und `CommonPrefixes` sind die einzigen Antwortelemente, die neue Informationen enthalten können.

Antwortelemente

Name	Description
Contents	Metadaten zu jedem zurückgegebenen Objekt. Typ: XML-Metadaten Vorfahr: ListBucketResult
CommonPrefixes	Eine Antwort kann <code>CommonPrefixes</code> nur enthalten, wenn Sie eine <code>delimiter</code> angeben. Wenn Sie dies tun, <code>CommonPrefixes</code> enthält sie alle Schlüssel (falls es welche gibt) zwischen <code>Prefix</code> und dem nächsten Vorkommen der Zeichenfolge, die durch angegeben ist <code>delimiter</code> . <code>CommonPrefixes</code> listet tatsächlich Schlüssel auf, die sich wie Unterverzeichnisse in dem von <code>Prefix</code> angegebenen Verzeichnis verhalten. Wenn beispielsweise ein Schrägstrich (/) <code>prefix delimiter</code> ist <code>notes/</code> und ein Schrägstrich (/) ist, ist in <code>notes/summer/july</code> das allgemeine Präfix. <code>notes/summer/</code> Typ: Zeichenfolge Vorfahr: ListBucketResult
Delimiter	Bewirkt, dass Schlüssel, die dieselbe Zeichenfolge zwischen dem Präfix und dem ersten Vorkommen des Trennzeichens enthalten, zu einem einzigen Ergebniselement in der Sammlung zusammengefasst werden. <code>CommonPrefixes</code> Diese zusammengefassten Schlüssel werden an keiner anderen Stelle in der Antwort zurückgegeben. Typ: Zeichenfolge Vorfahr: ListBucketResult
IsTruncated	Gibt an, ob (wahr) oder nicht (falsch) alle Ergebnisse zurückgegeben wurden. Es werden möglicherweise nicht alle Ergebnisse zurückgegeben, wenn die Anzahl der Ergebnisse die angegebene Anzahl von überschreitet <code>MaxKeys</code>. Typ: Zeichenfolge

Name	Description
	Vorfahr: boolesch
Marker	Gibt an, an welcher Stelle im Bucket mit dem Listing begonnen werden soll. Typ: Zeichenfolge Vorfahr: ListBucketResult
MaxKeys	Die maximale Anzahl von Schlüsseln, die im Antworttext zurückgegeben werden. Typ: Zeichenfolge Vorfahr: ListBucketResult
Name	Name des Eimers. Typ: Zeichenfolge Vorfahr: ListBucketResult
Prefix	Schlüssel, die mit dem angegebenen Präfix beginnen. Typ: Zeichenfolge Vorfahr: ListBucketResult

Antworttext

Informationen zur Listenantwort finden Sie unter [Listing Keys Response](#).

Zugriffskontrolle

Um die Schlüssel eines Buckets aufzulisten, müssen Sie READ Zugriff auf den Bucket erhalten haben.

GetBucketAccessControlPolicy (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Der `GetBucketAccessControlPolicy` Vorgang ruft die Zugriffskontrollrichtlinie für einen Bucket ab.

Example

In diesem Beispiel wird die Zugriffskontrollrichtlinie für den Bucket „Quotes“ abgerufen.

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>
```

Antworttext

Die Antwort enthält die Zugriffskontrollrichtlinie für den Bucket. Eine Erläuterung dieser Antwort finden Sie unter [SOAP-Zugriffsrichtlinie](#).

Zugriffskontrolle

Sie müssen über READ_ACP Rechte für den Bucket verfügen, um die Zugriffskontrollrichtlinie für einen Bucket abrufen zu können.

SetBucketAccessControlPolicy (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Der SetBucketAccessControlPolicy Vorgang legt die Zugriffskontrollrichtlinie für einen vorhandenen Bucket fest. Bei Erfolg wird die vorherige Zugriffskontrollrichtlinie für den Bucket vollständig durch die angegebene Zugriffskontrollrichtlinie ersetzt.

Example

Gewähren Sie dem angegebenen Benutzer (in der Regel dem Eigentümer) FULL_CONTROL Zugriff auf den 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>

```

Zugriffskontrolle

Sie müssen über WRITE_ACP Rechte für den Bucket verfügen, um die Zugriffskontrollrichtlinie für einen Bucket festlegen zu können.

GetBucketLoggingStatus (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Der GetBucketLoggingStatus ruft den Protokollierungsstatus für einen vorhandenen Bucket ab.

Eine allgemeine Einführung in diese Funktion finden Sie unter [Serverprotokolle](#).

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>
```

Zugriffskontrolle

Nur der Besitzer eines Buckets darf diesen Vorgang aufrufen.

SetBucketLoggingStatus (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Der SetBucketLoggingStatus Vorgang aktualisiert den Protokollierungsstatus für einen vorhandenen Bucket.

Eine allgemeine Einführung in diese Funktion finden Sie unter [Serverprotokolle](#).

Example

Diese Beispielanfrage aktiviert die Protokollierung des Serverzugriffs für den Bucket 'mybucket' und konfiguriert die Logs so, dass sie unter dem Präfix 'access_log-' an 'mylogs' übermittelt werden

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>
```

Zugriffskontrolle

Nur der Besitzer eines Buckets darf diesen Vorgang aufrufen.

Operationen an Objekten (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

In diesem Abschnitt werden Operationen beschrieben, die Sie mit Amazon S3-Objekten ausführen können.

Themen

- [PutObjectInline \(SOAP-API\)](#)
- [PutObject \(SOAP-API\)](#)
- [CopyObject \(SOAP-API\)](#)
- [GetObject \(SOAP-API\)](#)
- [GetObjectExtended \(SOAP-API\)](#)

- [DeleteObject \(SOAP-API\)](#)
- [GetObjectAccessControlPolicy \(SOAP-API\)](#)
- [SetObjectAccessControlPolicy \(SOAP-API\)](#)

PutObjectInline (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Die `PutObjectInline` Operation fügt einem Bucket ein Objekt hinzu. Die Daten für das Objekt werden im Hauptteil der SOAP-Nachricht bereitgestellt.

Wenn ein Objekt bereits in einem Bucket existiert, überschreibt es das neue Objekt, da Amazon S3 die letzte Schreibanforderung speichert. Amazon S3 ist jedoch ein verteiltes System. Wenn Amazon S3 fast gleichzeitig mehrere Schreibanforderungen für dasselbe Objekt erhält, werden möglicherweise alle Objekte gespeichert, auch wenn am Ende nur eines gewinnt. Amazon S3 bietet keine Objektsperren. Wenn Sie diese benötigen, stellen Sie sicher, dass Sie sie in Ihre Anwendungsebene integrieren.

Um sicherzustellen, dass ein Objekt im Netzwerk nicht beschädigt wird, können Sie den MD5-Wert eines Objekts berechnen, es in Amazon S3 speichern und das zurückgegebene Etag mit dem berechneten MD5-Wert vergleichen.

`PutObjectInline` ist nicht für die Verwendung mit großen Objekten geeignet. Das System beschränkt diesen Vorgang auf die Arbeit mit Objekten von 1 MB oder weniger. `PutObjectInlineschlägt` mit dem `InlineDataTooLargeError` Statuscode fehl, wenn der Datenparameter ein Objekt kodiert, das größer als 1 MB ist. Um große Objekte hochzuladen, sollten Sie erwägen, stattdessen die `PutObject` Nicht-Inline-API oder die REST-API zu verwenden.

Example

In diesem Beispiel werden Text und Metadaten in das „Nelson“-Objekt im Bucket „quotes“ geschrieben, einem Benutzer (normalerweise dem Eigentümer) `FULL_CONTROL` Zugriff auf das Objekt gewährt und das Objekt für anonyme Parteien lesbar gemacht.

Sample Request

```
<PutObjectInline xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <Metadata>
    <Name>Content-Type</Name>
    <Value>text/plain</Value>
  </Metadata>
  <Metadata>
    <Name>family</Name>
    <Value>Muntz</Value>
  </Metadata>
  <Data>aGEtaGE=</Data>
  <ContentLength>5</ContentLength>
  <AccessControlList>
    <Grant>
      <Grantee xsi:type="CanonicalUser">
        <ID>a9a7b886d6fde241bf9b1c61be666e9</ID>
        <DisplayName>chriscustomer</DisplayName>
      </Grantee>
      <Permission>FULL_CONTROL</Permission>
    </Grant>
    <Grant>
      <Grantee xsi:type="Group">
        <URI>http://acs.amazonaws.com/groups/global/AllUsers</URI>
      </Grantee>
      <Permission>READ</Permission>
    </Grant>
  </AccessControlList>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</PutObjectInline>
```

Sample Response

```
<PutObjectInlineResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <PutObjectInlineResponse>
    <ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <LastModified>2006-01-01T12:00:00.000Z</lastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>
```


Elemente

- **Bucket**: Der Bucket, dem das Objekt hinzugefügt werden soll.
- **Key**: Der Schlüssel, der dem Objekt zugewiesen werden soll.
- **Metadata**: Sie können Name-Wert-Metadatenpaare im Metadatenelement angeben. Diese werden zusammen mit dem Objekt gespeichert.
- **Data**: Die Base-64-kodierte Form der Daten.
- **ContentLength**: Die Länge der Daten in Byte.
- **AccessControlList**: Eine Zugriffskontrollliste für die Ressource. Das Element ist optional. Wenn nicht angegeben, erhält der Anforderer FULL_CONTROL Zugriff auf das Objekt. Wenn das Objekt bereits existiert, wird die bereits bestehende Zugriffskontrollrichtlinie ersetzt.

Antworten

- **ETag**: Das Entitäts-Tag ist ein MD5-Hash des Objekts, mit dem Sie bedingte Abrufe des Objekts durchführen können. `GetObjectExtended` Das ETag spiegelt nur Änderungen am Inhalt eines Objekts wider, nicht an seinen Metadaten.
- **LastModified**: Der Amazon S3-Zeitstempel für das gespeicherte Objekt.

Zugriffskontrolle

Sie müssen WRITE Zugriff auf den Bucket haben, um Objekte in den Bucket legen zu können.

Verwandte Ressourcen

- [PutObject \(SOAP-API\)](#)
- [CopyObject \(SOAP-API\)](#)

PutObject (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Die `PutObject` Operation fügt einem Bucket ein Objekt hinzu. Die Daten für das Objekt werden als DIME-Anlage angehängt.

Um sicherzustellen, dass ein Objekt im Netzwerk nicht beschädigt wird, können Sie den MD5-Wert eines Objekts berechnen, es an Amazon S3 senden und das zurückgegebene Etag mit dem berechneten MD5-Wert vergleichen.

Wenn ein Objekt bereits in einem Bucket existiert, überschreibt das neue Objekt es, da Amazon S3 die letzte Schreibanforderung speichert. Amazon S3 ist jedoch ein verteiltes System. Wenn Amazon S3 fast gleichzeitig mehrere Schreibanforderungen für dasselbe Objekt erhält, werden möglicherweise alle Objekte gespeichert, auch wenn am Ende nur eines gewinnt. Amazon S3 bietet keine Objektsperren. Wenn Sie diese benötigen, stellen Sie sicher, dass Sie sie in Ihre Anwendungsebene integrieren.

Example

In diesem Beispiel werden einige Daten und Metadaten in das „Nelson“-Objekt des Buckets „Quotes“ eingefügt, einem Benutzer (in der Regel dem Eigentümer) `FULL_CONTROL` Zugriff auf das Objekt gewährt und das Objekt für anonyme Parteien lesbar gemacht. In diesem Beispiel wird der eigentliche Anhang nicht angezeigt.

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>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <LastModified>2006-03-01T12:00:00.183Z</LastModified>
  </PutObjectResponse>
</PutObjectResponse>

```

Elemente

- **Bucket**: Der Bucket, dem das Objekt hinzugefügt werden soll.
- **Key**: Der Schlüssel, der dem Objekt zugewiesen werden soll.
- **Metadata**: Sie können Name-Wert-Metadatenpaare im Metadatenelement angeben. Diese werden zusammen mit dem Objekt gespeichert.
- **ContentLength**: Die Länge der Daten in Byte.

- **AccessControlList**: Eine Zugriffskontrollliste für die Ressource. Das Element ist optional. Wenn nicht angegeben, erhält der Anforderer `FULL_CONTROL` Zugriff auf das Objekt. Wenn das Objekt bereits existiert, wird die bereits bestehende Zugriffskontrollrichtlinie ersetzt.

Antworten

- **ETag**: Das Entitäts-Tag ist ein MD5-Hash des Objekts, mit dem Sie bedingte Abrufe des Objekts durchführen können. `GetObjectExtended` Das ETag spiegelt nur Änderungen am Inhalt eines Objekts wider, nicht an seinen Metadaten.
- **LastModified**: Der Amazon S3-Zeitstempel für das gespeicherte Objekt.

Zugriffskontrolle

Um Objekte in einen Bucket zu legen, müssen Sie `WRITE` Zugriff auf den Bucket haben.

Verwandte Ressourcen

- [CopyObject \(SOAP-API\)](#)

CopyObject (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Description

Der `CopyObject` Vorgang erstellt eine Kopie eines Objekts, wenn Sie den Schlüssel und den Bucket eines Quellobjekts sowie den Schlüssel und den Bucket eines Zielziels angeben.

Beim Kopieren eines Objekts können Sie alle Metadaten beibehalten (Standard) oder neue Metadaten angeben. Die ACL wird jedoch nicht beibehalten und ist `private` für den Benutzer, der die Anfrage stellt, auf gesetzt. Um die ACL-StandardEinstellung außer Kraft zu setzen, geben Sie beim Generieren einer Kopieranforderung eine neue ACL an. Weitere Informationen finden Sie unter [Verwendung von ACLs](#).

Alle Kopieranfragen müssen authentifiziert werden. Darüber hinaus müssen Sie Lesezugriff auf das Quellobjekt und Schreibzugriff auf den Ziel-Bucket haben. Weitere Informationen finden Sie unter [Verwenden des Authentifizierungszugriffs](#).

Um ein Objekt nur unter bestimmten Bedingungen zu kopieren, z. B. ob das Etag übereinstimmt oder ob das Objekt vor oder nach einem bestimmten Datum geändert wurde, verwenden Sie die Anforderungsparameter `CopySourceIfUnmodifiedSince`, `CopyIfUnmodifiedSince`, `CopySourceIfMatch`, oder `CopySourceIfNoneMatch`.

Note

Möglicherweise müssen Sie das SOAP-Stack-Socket-Timeout für das Kopieren großer Objekte konfigurieren.

Anforderungssyntax

```
<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>

```

Anforderungsparameter

Name	Description	Erforderlich
SourceBucket	Der Name des Quell-Buckets. Typ: Zeichenfolge Standard: keiner Einschränkungen: Ein gültiger Quell-Bucket.	Ja
SourceKey	Der Schlüsselname des Quellobjekts. Typ: Zeichenfolge Standard: keiner Einschränkungen: Der Schlüssel für ein gültiges Quellobjekt, auf das Sie LESE-Zugriff haben.	Ja
DestinationBucket	Der Name des -Ziel-Buckets. Typ: Zeichenfolge Standard: keiner Einschränkungen: Sie müssen WRITE-Zugriff auf den Ziel-Bucket haben.	Ja
DestinationKey	Der Schlüssel des Zielobjekts. Typ: Zeichenfolge	Ja

Name	Description	Erforderlich
	Standard: keiner Einschränkungen: Sie müssen WRITE-Zugriff auf den Ziel-Bucket haben.	
MetadataDirective	Gibt an, ob die Metadaten aus dem Quellobjekt kopiert oder durch die in der Anforderung bereitgestellten Metadaten ersetzt werden. Typ: Zeichenfolge Standard: COPY Gültige Werte: COPY REPLACE Einschränkungen: Andere Werte als COPY oder REPLACE führen zu einem sofortigen Fehler. Sie können ein Objekt nur in sich selbst kopieren, wenn der MetadataDirective Header angegeben und sein Wert auf gesetzt ist REPLACE.	Nein
Metadata	Gibt Metadaten-Name-Wert-Paare an, die für das Objekt festgelegt werden sollen. Wenn auf gesetzt MetadataDirective ist COPY, werden alle Metadaten ignoriert. Typ: Zeichenfolge Standard: keiner Einschränkungen: Keine.	Nein

Name	Description	Erforderlich
AccessControlList	Gewährt Benutzern Zugriff über E-Mail-Adressen oder kanonische Benutzer-IDs. Typ: Zeichenfolge Standard: keiner Einschränkungen: Keine	Nein
CopySourceIfMatch	Kopiert das Objekt, wenn sein Entitäts-Tag (ETag) mit dem angegebenen Tag übereinstimmt; gibt andernfalls <code>PreconditionFailed</code> zurück. Typ: Zeichenfolge Standard: keiner Einschränkungen: Keine. Wenn das Etag nicht übereinstimmt, wird das Objekt nicht kopiert.	Nein
CopySourceIfNoneMatch	Kopiert das Objekt, wenn sein Entitäts-Tag (ETag) vom angegebenen Etag abweicht; andernfalls wird ein Fehler zurückgegeben. Typ: Zeichenfolge Standard: keiner Einschränkungen: Keine.	Nein

Name	Description	Erforderlich
CopySourceIfUnmodifiedSince	Kopiert das Objekt, wenn es seit der angegebenen Zeit nicht geändert wurde; andernfalls wird ein zurückgegeben PreconditionFailed. Typ: DateTime Standard: keiner	Nein
CopySourceIfModifiedSince	Kopiert das Objekt, wenn es seit dem angegebenen Zeitpunkt geändert wurde; andernfalls wird ein Fehler zurückgegeben. Typ: DateTime Standard: keiner	Nein

Antwortsyntax

```
<CopyObjectResponse xmlns="http://bucket_name.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"etag"</ETag>
    <LastModified>timestamp</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Antwortelemente

Im Folgenden finden Sie eine Liste der Antwortelemente.

Note

Die SOAP-API gibt kein zusätzliches Leerzeichen zurück. Zusätzliche Leerzeichen werden nur von der REST-API zurückgegeben.

Name	Description
Etag	Gibt das Etag des neuen Objekts zurück. Das ETag spiegelt nur Änderungen am Inhalt eines Objekts wider, nicht an seinen Metadaten. Typ: Zeichenfolge Vorfahr: CopyObjectResult
LastModified	Gibt das Datum zurück, an dem das Objekt zuletzt geändert wurde. Typ: Zeichenfolge Vorfahr: CopyObjectResult

Informationen zu allgemeinen Antwortelementen finden Sie unter [Verwenden von](#) REST-Fehlerantwort-Headern.

Spezielle Fehler

Für diesen Vorgang gibt es keine speziellen Fehler. Informationen zu allgemeinen Amazon S3-Fehlern finden Sie unter [Liste der Fehlercodes](#).

Beispiele

Dieses Beispiel kopiert das flotsam-Objekt aus dem pacific-Bucket in das jetsam-Objekt des atlantic-Buckets, wobei seine Metadaten beibehalten werden.

Beispielanforderung

```
<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>
```

Beispielantwort

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

In diesem Beispiel wird das Objekt „tweedledee“ aus dem Wonderland-Bucket in das „Tweedledum“-Objekt des Wonderland-Buckets kopiert und die zugehörigen Metadaten ersetzt.

Beispielanforderung

```
<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>
```

Beispielantwort

```
<CopyObjectResponse xmlns="http://doc.s3.amazonaws.com/2006-03-01">
  <CopyObjectResponse>
    <ETag>"828ef3fd96f00ad9f27c383fc9ac7f"</ETag>
    <LastModified>2008-02-18T13:54:10.183Z</LastModified>
  </CopyObjectResponse>
</CopyObjectResponse>
```

Verwandte Ressourcen

- [PutObject \(SOAP-API\)](#)
- [PutObjectInline \(SOAP-API\)](#)

GetObject (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Die `GetObject` Operation gibt die aktuelle Version eines Objekts zurück. Wenn Sie versuchen, `GetObject` ein Objekt zu verwenden, dessen aktuelle Version eine Löschmarkierung enthält, gibt S3 einen 404-Fehler zurück. Sie können die SOAP-API nicht verwenden, um eine bestimmte Version eines Objekts abzurufen. Verwenden Sie dazu die REST-API. Weitere Informationen finden Sie unter [Versionierung](#). Verwenden Sie die [GetObjectExtended \(SOAP-API\)](#) Operation, um weitere Optionen zu erhalten.

Note

Objektschlüsselnamen mit dem Wert „soap“ werden für [Anfragen im virtuellen Hosting-Format](#) nicht unterstützt. Für Werte von Objektschlüsselnamen, bei denen „soap“ verwendet wird, muss stattdessen eine [URL im Pfadformat](#) verwendet werden.

Example

In diesem Beispiel wird das „Nelson“-Objekt aus dem Bucket „quotes“ abgerufen.

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>

```

Elemente

- **Bucket**: Der Bucket, aus dem das Objekt abgerufen werden soll.
- **Key**: Der Schlüssel, der das Objekt identifiziert.
- **GetMetadata**: Die Metadaten werden zusammen mit dem Objekt zurückgegeben, wenn dies zutrifft.
- **GetData**: Die Objektdaten werden zurückgegeben, wenn dies zutrifft.
- **InlineData**: Wenn dies zutrifft, werden die Daten Base-64-kodiert als Teil des SOAP-Hauptteils der Antwort zurückgegeben. Wenn der Wert falsch ist, werden die Daten als SOAP-Anlage zurückgegeben. Die Inlinedata Option ist nicht für die Verwendung mit großen Objekten geeignet. Das System beschränkt diesen Vorgang auf die Arbeit mit 1 MB an Daten oder weniger. Eine GetObject Anfrage mit gesetztem Inlinedata Flag schlägt mit dem InlineDataTooLargeError

Statuscode fehlt, wenn der resultierende Datenparameter mehr als 1 MB codiert hätte. Um große Objekte herunterzuladen, sollten Sie erwägen, aufzurufen, `GetObject` ohne das `InlineData` Flag zu setzen, oder verwenden Sie stattdessen die REST-API.

Zurückgegebene Elemente

- **Metadata**: Die mit dem Objekt gespeicherten Metadaten aus Name und Wert.
- **Data**: Wenn in der Anfrage „true“ angegeben `InlineData` wurde, enthält dies die Basis-64-kodierten Objektdaten.
- **LastModified**: Die Uhrzeit, zu der das Objekt in Amazon S3 gespeichert wurde.
- **ETag**: Das Entitäts-Tag des Objekts. Dies ist ein Hash des Objekts, der für bedingte Abrufe verwendet werden kann. Das ETag spiegelt nur Änderungen am Inhalt eines Objekts wider, nicht an seinen Metadaten.

Zugriffskontrolle

Sie können ein Objekt nur lesen, wenn Ihnen `READ` Zugriff auf das Objekt gewährt wurde.

Aufgeteilte und wiederaufladbare SOAP-Downloads

Um GET Flexibilität zu gewährleisten, unterstützt Amazon S3 gestaffelte und wiederaufrufbare Downloads.

Wählen Sie aus den folgenden Optionen:

- Bei großen Objektdownloads sollten Sie diese möglicherweise in kleinere Abschnitte aufteilen. Weitere Informationen finden Sie unter [Bereich GETs](#).
- Bei fehlgeschlagenen GET Vorgängen können Sie Ihre Anwendung so gestalten, dass der Rest und nicht die gesamte Datei heruntergeladen wird. Weitere Informationen finden Sie unter [REST GET-Fehlerbehebung](#).

Bereich GETs

Bei einigen Clients möchten Sie möglicherweise große Downloads in kleinere Downloads aufteilen. Um ein GET in kleinere Einheiten aufzuteilen, verwenden Sie `Range`.

Bevor Sie ein GET in kleinere Einheiten aufteilen können, müssen Sie seine Größe bestimmen. Mit der folgenden Anforderung wird beispielsweise die Größe des Bigfile-Objekts abgerufen.

```
<ListBucket xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Prefix>bigfile</Prefix>
  <MaxKeys>1</MaxKeys>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
</ListBucket>
```

Amazon S3 gibt die folgende Antwort zurück.

```
<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>
```

Es folgt eine Anfrage, die das erste Megabyte aus dem Bigfile-Objekt herunterlädt.

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>0</ByteRangeStart>
  <ByteRangeEnd>1048576</ByteRangeEnd>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
```

```
</GetObject>
```

Amazon S3 gibt das erste Megabyte der Datei und das Etag der Datei zurück.

```
<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>
```

Um sicherzustellen, dass sich die Datei seit dem Herunterladen des vorherigen Abschnitts nicht geändert hat, geben Sie das IfMatch Element an. Obwohl das IfMatch Element nicht erforderlich ist, wird es für Inhalte empfohlen, die sich wahrscheinlich ändern werden.

Die folgende Anforderung ruft mithilfe des IfMatch Anforderungsheaders den Rest der Datei ab.

```
<GetObject xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>bigbucket</Bucket>
  <Key>bigfile</Key>
  <GetMetadata>true</GetMetadata>
  <GetData>true</GetData>
  <InlineData>true</InlineData>
  <ByteRangeStart>10485761</ByteRangeStart>
  <ByteRangeEnd>2023276</ByteRangeEnd>
  <IfMatch>"828ef3fdfa96f00ad9f27c383fc9ac7f"</IfMatch>
  <AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
  <Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
  <Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE=</Signature>
```



```
</GetObject>
```

Amazon S3 gibt die folgende Antwort und den Rest der Datei zurück.

```
<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>
```

Versioniert GetObject

Die folgende Anforderung gibt die angegebene Version des Objekts im Bucket zurück.

```
<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>
```

Beispielantwort

```
<GetObjectResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
```

```
<GetObjectResponse>
<Status>
<Code>200</Code>
<Description>OK</Description>
</Status>
<Metadata>
<Name>Content-Type</Name>
<Value>text/plain</Value>
</Metadata>
<Metadata>
<Name>family</Name>
<Value>Muntz</Value>
</Metadata>
<Data>aGEtaGE=</Data>
<LastModified>2006-01-01T12:00:00.000Z</LastModified>
<ETag>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
</GetObjectResponse>
</GetObjectResponse>
```

REST GET-Fehlerbehebung

Wenn ein GET-Objekt fehlschlägt, können Sie den Rest der Datei abrufen, indem Sie den Bereich angeben, der heruntergeladen werden soll. Dazu müssen Sie die Größe des Objekts mithilfe ListBucket eines Bereichs für GET den Rest der Datei ermitteln. Weitere Informationen finden Sie unter [GetObjectExtended \(SOAP-API\)](#).

Verwandte Ressourcen

[Operationen an Objekten \(SOAP-API\)](#)

GetObjectExtended (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

GetObjectExtended ist genau wie [GetObject \(SOAP-API\)](#), außer dass es die folgenden zusätzlichen Elemente unterstützt, die verwendet werden können, um einen Großteil der Funktionen

zu erreichen, die von HTTP-GET-Headern bereitgestellt werden (siehe). <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html>

GetObjectExtended unterstützt die folgenden Elemente zusätzlich zu denen, die von GetObject unterstützt werden:

- **ByteRangeStart, ByteRangeEnd**: Diese Elemente geben an, dass nur ein Teil der Objektdaten abgerufen werden soll. Sie folgen dem Verhalten der HTTP-Bytebereiche (siehe <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.35>).
- **IfModifiedSince**: Gibt das Objekt nur zurück, wenn der Zeitstempel des Objekts nach dem angegebenen Zeitstempel liegt. (<http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.25>)
- **IfUnmodifiedSince**: Gibt das Objekt nur zurück, wenn der Zeitstempel des Objekts vor dem angegebenen Zeitstempel liegt oder diesem entspricht. (gehe zu <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.28>)
- **IfMatch**: Gibt das Objekt nur zurück, wenn sein ETag mit dem/den angegebenen Tag (s) übereinstimmt. (gehe zu <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.24>)
- **IfNoneMatch**: Gibt das Objekt nur zurück, wenn sein ETag nicht mit dem/den angegebenen Tag (s) übereinstimmt. (gehe zu <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.26>)
- **ReturnCompleteObjectOnConditionFailure**: Falls wahr, wird das gesamte Objekt und nicht ein Fehler zurückgegeben, wenn die Anforderung ein Range-Element und eines oder beide IfUnmodifiedSince/IfMatch Elemente enthält und die Bedingung fehlschlägt. Dadurch wird die If-Range Funktionalität aktiviert (siehe <http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.27>).

DeleteObject (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Der DeleteObject Vorgang entfernt das angegebene Objekt aus Amazon S3. Nach dem Löschen gibt es keine Methode zum Wiederherstellen oder Wiederherstellen eines Objekts.

 Note

Wenn Sie ein Objekt löschen, das nicht existiert, gibt Amazon S3 eine Erfolgsmeldung zurück (keine Fehlermeldung).

Example

In diesem Beispiel wird das Objekt „Nelson“ aus dem Bucket „quotes“ gelöscht.

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>
```

Elemente

- **Bucket** : Der Bucket, der das Objekt enthält.
- **Key** : Der Schlüssel, der das Objekt identifiziert.

Zugriffskontrolle

Sie können ein Objekt nur löschen, wenn Sie **WRITE** Zugriff auf den Bucket haben, unabhängig davon, wem das Objekt gehört oder welche Rechte ihm gewährt wurden.

GetObjectAccessControlPolicy (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Der `GetObjectAccessControlPolicy` Vorgang ruft die Zugriffskontrollrichtlinie für ein Objekt ab.

Example

In diesem Beispiel wird die Zugriffskontrollrichtlinie für das „Nelson“-Objekt aus dem Bucket „quotes“ abgerufen.

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>
```

Antworttext

Die Antwort enthält die Zugriffskontrollrichtlinie für den Bucket. Eine Erläuterung dieser Antwort finden Sie unter [SOAP Access Policy](#).

Zugriffskontrolle

Sie müssen über READ_ACP Rechte für das Objekt verfügen, um die Zugriffskontrollrichtlinie für ein Objekt abrufen zu können.

SetObjectAccessControlPolicy (SOAP-API)

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Der SetObjectAccessControlPolicy Vorgang legt die Zugriffskontrollrichtlinie für ein vorhandenes Objekt fest. Bei Erfolg wird die vorherige Zugriffskontrollrichtlinie für das Objekt vollständig durch die angegebene Zugriffsrichtlinie ersetzt.

Example

In diesem Beispiel erhält der angegebene Benutzer (in der Regel der Besitzer) FULL_CONTROL Zugriff auf das „Nelson“-Objekt aus dem Bucket „quotes“.

Sample Request

```
<SetObjectAccessControlPolicy xmlns="https://doc.s3.amazonaws.com/2006-03-01">
  <Bucket>quotes</Bucket>
  <Key>Nelson</Key>
  <AccessControlList>
```

```
<Grant>
  <Grantee xsi:type="CanonicalUser">
    <ID>a9a7b886d6fd24a52fe8ca5bef65f89a64e0193f23000e241bf9b1c61be666e9</ID>
    <DisplayName>chriscustomer</DisplayName>
  </Grantee>
  <Permission>FULL_CONTROL</Permission>
</Grant>
</AccessControlList>
<AWSAccessKeyId>AKIAIOSFODNN7EXAMPLE</AWSAccessKeyId>
<Timestamp>2006-03-01T12:00:00.183Z</Timestamp>
<Signature>Iuyz3d3P0aTou39dzbqaEXAMPLE= </Signature>
</SetObjectAccessControlPolicy>
```

Sample Response

```
<SetObjectAccessControlPolicyResponse xmlns="https://s3.amazonaws.com/doc/2006-03-01">
  <SetObjectAccessControlPolicyResponse>
    <Code>200</Code>
    <Description>OK</Description>
  </SetObjectAccessControlPolicyResponse>
</SetObjectAccessControlPolicyResponse>
```

Key (Schlüssel)

Zugriffskontrolle

Sie müssen über WRITE_ACP Rechte für das Objekt verfügen, um die Zugriffskontrollrichtlinie für einen Bucket festlegen zu können.

Authentifizieren von SOAP-Anforderungen

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Jede nicht anonyme Anfrage muss Authentifizierungsinformationen enthalten, um die Identität des Prinzipals einzurichten, der die SOAP-Anfrage stellt. In SOAP wird die Authentifizierungsinformation in den folgenden Elemente der SOAP-Anfrage abgelegt:

- Ihre AWS Zugangsschlüssel-ID

Note

Bei authentifizierten SOAP-Anfragen werden keine temporären Sicherheitsanmeldeinformationen unterstützt. Weitere Informationen den verschiedenen Anmeldeinformationen finden Sie unter [Senden von Anforderungen](#).

- **Timestamp**: Dies muss ein DateTime-Wert (gehe zu <http://www.w3.org/TR/xmlschema-2/#dateTime>) in der Zeitzone der koordinierten Weltzeit (Greenwich Mean Time) sein, z. B. 2009-01-01T12:00:00.000Z Die Autorisierung schlägt fehl, wenn dieser Zeitstempel um mehr als 15 Minuten von der Uhr auf Amazon-S3-Servern abweicht.
- **Signature**: Der RFC HMAC-SHA1 2104-Digest (siehe <http://www.ietf.org/rfc/rfc2104.txt>) der Verkettung von „AmazonS3“ + OPERATION + Timestamp, wobei Ihr Secret Access Key als Schlüssel verwendet wird. AWS In der folgenden CreateBucket Beispielanforderung würde das Signature-Element beispielsweise den Digest des Werts „,enthalten: HMAC-SHA1 AmazonS3CreateBucket2009-01-01T12:00:00.000Z

In der folgenden CreateBucket Beispielanforderung würde das Signaturelement beispielsweise den HMAC-SHA1 Digest des Werts „,AmazonS3CreateBucket2009-01-01T12:00:00.000Zenthalten:

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

SOAP-Anfragen, und zwar sowohl authentifizierte als auch anonyme, müssen mit SSL an Amazon S3 gesendet werden. Amazon S3 gibt einen Fehler zurück, wenn Sie eine SOAP-Anfrage über HTTP senden.

⚠ Important

Aufgrund von unterschiedlichen Interpretationen dahingehend, wie eine zusätzliche Zeitgenauigkeit wegfallen kann, sollten .NET-Benutzer darauf achten, Amazon S3 keine übermäßig spezifischen Zeitstempel zu senden. Dies kann bewerkstelligt werden, indem manuell DateTime-Objekte mit einer Millisekunde Genauigkeit erstellt werden.

Festlegen der Zugriffsrichtlinie mit SOAP

ℹ Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Die Zugriffskontrolle kann zu dem Zeitpunkt festgelegt werden, zu dem ein Bucket oder Objekt geschrieben wird, indem das Element `AccessControlList` in die Anfrage an `CreateBucketPutObjectInline`, oder `PutObject` aufgenommen wird. Das `AccessControlList` Element wird in [Identitäts- und Zugriffsmanagement für Amazon S3 beschrieben](#). Wenn bei diesen Vorgängen keine Zugriffskontrollliste angegeben wird, wird die Ressource mit einer Standardzugriffsrichtlinie erstellt, die dem Anforderer `FULL_CONTROL`-Zugriff gewährt (dies ist auch dann der Fall, wenn es sich bei der Anfrage um eine `PutObjectInline` oder `PutObject` Anfrage für ein Objekt handelt, das bereits existiert).

Nachfolgend sehen Sie eine Anfrage, die Daten in ein Objekt schreibt, das Objekt lesbar für anonyme Prinzipale macht, und dem angegebenen Benutzer `FULL_CONTROL`-Rechte für den Bucket erteilt (die meisten Entwickler wollen sich selbst `FULL_CONTROL`-Zugriff auf ihren eigenen Bucket geben).

Example

Die nachfolgende Anfrage schreibt Daten in ein Objekt und macht das Objekt lesbar für anonyme Prinzipale.

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>&quot;828ef3fdfa96f00ad9f27c383fc9ac7f&quot;</ETag>
    <LastModified>2009-01-01T12:00:00.000Z</LastModified>
  </PutObjectInlineResponse>
</PutObjectInlineResponse>

```

Die Zugriffskontrollrichtlinie kann für einen vorhandenen Bucket oder ein vorhandenes Objekt mit den Methoden `GetBucketAccessControlPolicy`, `GetObjectAccessControlPolicy`, `SetBucketAccessControlPolicy` und `SetObjectAccessControlPolicy` gelesen oder eingerichtet werden. Weitere Informationen finden Sie in der detaillierten Erklärung dieser Methoden.

Allgemeine Elemente

Sie können die folgenden autorisierungsbezogenen Elemente in jede SOAP-Anfrage aufnehmen:

- **AWSSignatureVersion**: Die AWS Zugriffsschlüssel-ID des Anforderers
- **Timestamp**: Die aktuelle Systemzeit
- **Signature**: Die Signatur für die Anforderung

Antworten auf SOAP-Fehler

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die SDKs zu verwenden. AWS

In SOAP wird eine Fehlermeldung als SOAP-Fehler an den Client zurückgegeben, mit dem HTTP-Antwortcode 500. Wenn Sie keinen SOAP-Fehler erhalten, war Ihre Anfrage erfolgreich. Der Amazon S3-SOAP-Fehlercode besteht aus einem Standard-SOAP 1.1-Fehlercode (entweder „Server“ oder „Client“), der mit dem Amazon-Fehlercode verkettet ist. S3-specific Zum Beispiel: `Client.NoSuchBucket` Das SOAP-Fehlerzeichenfolgeelement enthält eine generische, vom Menschen lesbare Fehlermeldung in englischer Sprache. Das SOAP-Fehlerdetail-element schließlich enthält verschiedene für den Fehler relevante Informationen.

Wenn Sie beispielsweise versuchen, das Objekt „Fred“ zu löschen, das nicht existiert, enthält der Hauptteil der SOAP-Antwort einen `NoSuchKey` SOAP-Fehler.

Das folgende Beispiel zeigt ein Beispiel für eine SOAP-Fehlerantwort.

```
<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>
```

In der folgenden Tabelle werden die SOAP-Fehlerantwortelemente erläutert

Name	Description
Detail	Container für den am Fehler beteiligten Schlüssel Typ: Container Vorfahr: Body.Fault
Fault	Container für Fehlerinformationen. Typ: Container Vorfahr: Körper
Faultcode	Der Fehlercode ist eine Zeichenfolge, die eine Fehlerbedingung eindeutig identifiziert. Er ist für Programme vorgesehen, die Fehler dem Typ nach erkennen und verarbeiten. Weitere Informationen finden Sie unter Liste der Fehlercodes. Typ: Zeichenfolge Vorfahr: Body.Fault
Faultstring	Die Fehlerzeichenfolge enthält eine allgemeine Beschreibung des Fehlerzustands in Englisch. Sie ist für dafür vorgesehen, von Menschen gelesen zu werden. Einfache Programme zeigen dem Endbenutzer die Meldung direkt an, wenn sie auf eine Fehlerbedingung treffen, die sie nicht kennen, oder die sie nicht verarbeiten können. Raffinierte Programme mit umfassenderer Fehlerbehandlung und korrekter Internationalisierung ignorieren die Fehlerzeichenfolge mit größerer Wahrscheinlichkeit. Typ: Zeichenfolge Vorfahr: Body.Fault
Key	Identifiziert den Schlüssel, der an dem Fehler beteiligt ist

Name	Description
	Typ: Zeichenfolge
	Vorfahr: Body.Fault

Anhang: Anfragen authentifizieren (AWS Signatur (Version 2))

Important

In diesem Abschnitt wird beschrieben, wie Anfragen mithilfe von AWS Signature Version 2 authentifiziert werden. Signature Version 2 wird deaktiviert (veraltet), Amazon S3 akzeptiert nur API-Anforderungen, die mit Signature Version 4 signiert wurden. Weitere Informationen finden Sie unter [AWS Signature Version 2 ist für Amazon S3 deaktiviert \(veraltet\)](#). Signature Version 4 wird in allen AWS-Regionen unterstützt und ist die einzige Version, die für neue Regionen unterstützt wird. Weitere Informationen finden Sie unter [Authentifizieren von Anfragen \(AWS Signature Version 4\)](#) in der Amazon Simple Storage Service API-Referenz.

Amazon S3 ermöglicht das Ermitteln der zum Signieren einer Anforderung verwendeten API Signature Version. Um Beeinträchtigungen der betrieblichen Prozesse zu vermeiden, muss festgestellt werden, ob Workflows mit Signature Version 2 signieren. Diese Workflows müssen dann auf die Verwendung von Signature Version 4 umgestellt werden.

- Wenn Sie CloudTrail Ereignisprotokolle verwenden (empfohlene Option), finden Sie unter [Identifizieren von Amazon S3 Signature Version 2-Anfragen mithilfe von CloudTrail](#) Informationen zur Abfrage und Identifizierung solcher Anfragen.
- Wenn Sie die Amazon S3-Serverzugriffsprotokolle verwenden, finden Sie weitere Informationen unter [Verwenden von Amazon S3-Serverzugriffsprotokollen zur Identifizierung von Anfragen](#)

Themen

- [Authentifizieren von Anfragen mithilfe der REST-API \(AWS Signatur \(Version 2\)\)](#)
- [Signieren und Authentifizieren von REST-Anfragen \(AWS Signatur \(Version 2\)\)](#)
- [Browser-based lädt mit POST \(\) hoch AWS Signatur \(Version 2\)](#)

Authentifizieren von Anfragen mithilfe der REST-API (AWS Signatur (Version 2))

Beim Zugriff auf Amazon S3 mit REST müssen Sie in Ihrer Anfrage die folgenden Elemente angeben, damit diese authentifiziert werden kann.:

Anfordern von Elementen

- **AWS Zugriffsschlüssel-ID** — Jede Anfrage muss die Zugriffsschlüssel-ID der Identität enthalten, die Sie zum Senden Ihrer Anfrage verwenden.
- **Signatur** – Jede Anforderung muss eine gültige Anforderungssignatur enthalten, oder die Anforderung wird abgelehnt.

Eine Anforderungssignatur wird unter Verwendung Ihres geheimen Zugriffsschlüssels berechnet, das ist ein gemeinsames Geheimnis, das nur Sie und AWS kennen.

- **Zeitstempel** – Jede Anfrage muss Datum und Uhrzeit des Zeitpunktes enthalten, zu dem die Anfrage erstellt wurde, dargestellt als Zeichenfolge in UTC.
- **Datum** – Jede Anfrage muss den Zeitstempel der Anfrage enthalten.

Abhängig von der verwendeten API-Aktion können Sie ein Ablaufdatum und eine Ablaufzeit für die Anfrage angeben, statt dem Zeitstempel oder zusätzlich zu diesem. Um festzustellen, was notwendig ist, lesen Sie im Authentifizierungsthema für die jeweiligen Aktion nach.

Nachfolgend finden Sie die allgemeinen Schritte für die Authentifizierung von Anfragen für Amazon S3. Es wird vorausgesetzt, dass Sie über die erforderlichen Sicherheitsanmeldeinformationen, die Zugriffsschlüssel-ID und den geheimen Zugriffsschlüssel verfügen.

You**1 Create a request:****Request**

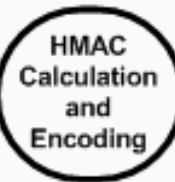
AccessKeyId = ...
 Action = ...
 Timestamp = ...
 ParameterA = ...

2 Create an HMAC-SHA1 signature:

String based on
request contents

+

Your Secret Access Key



Your Signature

3 Send the request and signature to AWS:**Request**

AccessKeyId = ...
 Action = ...
 Timestamp = ...
 ParameterA = ...

Your Signature

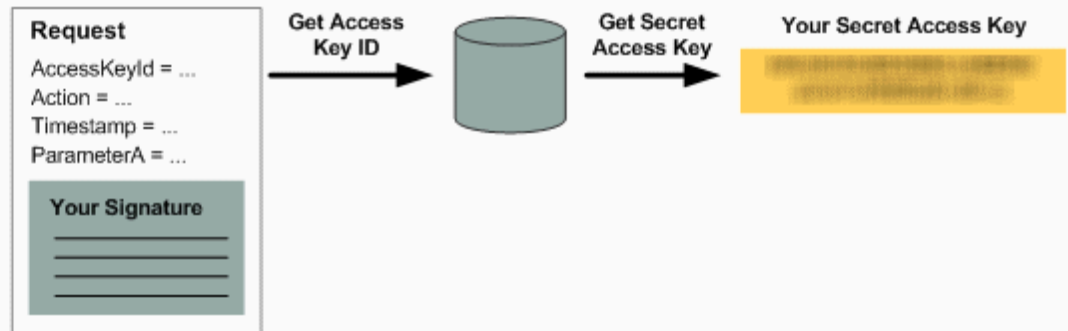


AWS

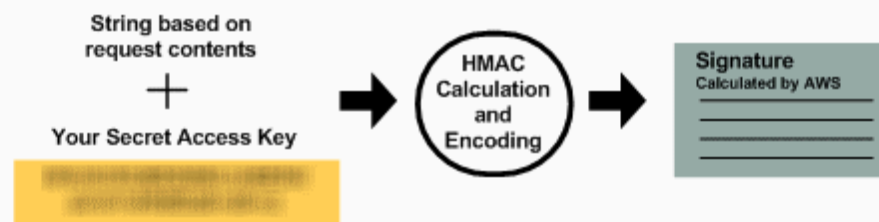
- 1 Erstellen Sie eine Anfrage an AWS
- 2 Berechnung der Signatur unter Verwendung Ihres geheimen Zugriffsschlüssels.
- 3 Senden Sie die Anfrage an Amazon S3. Nehmen Sie Ihre Zugriffsschlüssel-ID und die Signatur in Ihre Anforderung auf. Amazon S3 führt die nächsten drei Schritte aus.

AWS

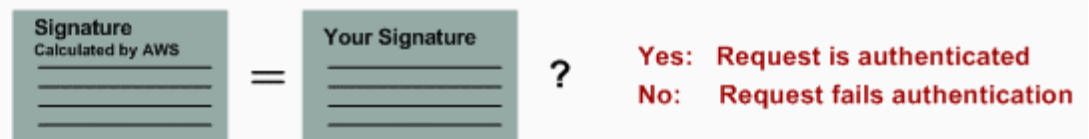
4 Retrieve your Secret Access Key:



5 Create an HMAC-SHA1 signature:



6 Compare the two signatures:



4 Amazon S3 verwendet die Zugriffsschlüssel-ID, um Ihren geheimen Zugriffsschlüssel abzurufen.

5 Amazon S3 berechnet eine Signatur aus den Anforderungsdaten und dem geheimen Zugriffsschlüssel mit demselben Algorithmus, mit dem die in der Anforderung gesendete Signatur berechnet wurde.

6 Wenn die von Amazon S3 generierte Signatur mit der in der Anforderung gesendete n Signatur übereinstimmt, wird die Anforderung als authentisch eingestuft. Falls der Vergleich fehlschlägt, wird die Anfrage verworfen, und Amazon S3 gibt eine Fehlerantwort zurück.

Detaillierte Authentifizierungsdaten

Weitere Informationen zur REST-Authentifizierung finden Sie unter [Signieren und Authentifizieren von REST-Anfragen \(AWS Signatur \(Version 2\)\)](#).

Signieren und Authentifizieren von REST-Anfragen (AWS Signatur (Version 2))

Themen

- [Verwenden von temporären Sicherheitsanmeldeinformationen](#)
- [Der Authentifizierungsheader](#)
- [Anfordern einer Kanonisierung für die Signatur](#)
- [CanonicalizedResource Konstruieren des Elements](#)
- [Das Element konstruieren CanonicalizedAmzHeaders](#)
- [Positionelle und benannte HTTP-Header-Elemente StringToSign](#)
- [Zeitstempel-Anforderung](#)
- [Authentifizierungsbeispiele](#)
- [Probleme mit dem Signieren von REST-Anforderungen](#)
- [Alternative zur Authentifizierung der Abfragezeichenfolge](#)

Note

Dieses Thema erklärt Authentifizierungsanfragen unter Verwendung von Signature Version 2. Amazon S3 unterstützt jetzt die neueste Signature Version 4. Diese neueste Signatur-Version wird in allen Regionen unterstützt. Alle neuen Regionen unterstützen nach dem 30. Januar 2014 nur Signature Version 4. Weitere Informationen finden Sie unter [Authentifizieren von Anfragen \(AWS Signature Version 4\)](#) in der Amazon Simple Storage Service API-Referenz.

Die Authentifizierung ist der Prozess, Ihre Identität gegenüber dem System nachzuweisen. Die Identität ist ein wichtiger Faktor in den Zugriffssteuerungsentscheidungen von Amazon S3. Anforderungen werden zum Teil basierend auf der Identität des Auftraggebers zugelassen oder abgewiesen. Das Recht, Buckets zu erstellen, ist beispielsweise für registrierte Entwickler reserviert (standardmäßig), und das Recht, Objekte in einem Bucket zu erstellen, ist für den Eigentümer

des betreffenden Buckets reserviert. Als Entwickler machen Sie Anforderungen, für die diese Berechtigungen gelten müssen, deshalb müssen Sie Ihre Identität gegenüber dem System belegen, indem Sie Ihre Anforderungen authentifizieren. In diesem Abschnitt erfahren Sie mehr darüber.

 Note

Der Inhalt dieses Abschnitts gilt nicht für HTTP POST. Weitere Informationen finden Sie unter [Browser-based lädt mit POST \(\) hoch AWS Signatur \(Version 2\)](#).

Die Amazon-S3-REST-API verwendet ein allgemeines HTTP-Schema basierend auf einem verschlüsselten HMAC (Hash Message Authentication Code) für die Authentifizierung. Um eine Anforderung zu authentifizieren, verknüpfen Sie zuerst ausgewählte Elemente der Anforderung, um eine Zeichenfolge zu erstellen. Anschließend verwenden Sie Ihren AWS geheimen Zugriffsschlüssel, um den HMAC dieser Zeichenfolge zu berechnen. Informell wird dieser Vorgang als „Signieren der Anfrage“ bezeichnet, und die Ausgabe des HMAC-Algorithmus ist die Signatur, da sie die Sicherheitseigenschaften einer echten Signatur simuliert. Schließlich fügen Sie diese Signatur als Parameter der Anforderung hinzu. Dazu verwenden Sie die in diesem Abschnitt beschriebene Syntax.

Wenn das System eine authentifizierte Anfrage erhält, ruft es den AWS geheimen Zugriffsschlüssel ab, den Sie angeblich besitzen, und verwendet ihn auf die gleiche Weise, um eine Signatur für die empfangene Nachricht zu berechnen. Anschließend vergleicht es die berechnete Signatur mit der Signatur, die der Anforderer vorgezeigt hat. Wenn die beiden Signaturen übereinstimmen, kommt das System zu dem Schluss, dass der Anforderer Zugriff auf den AWS geheimen Zugriffsschlüssel haben muss, und handelt daher mit der Autorität des Prinzipals, an den der Schlüssel ausgestellt wurde. Stimmen die beiden Signaturen nicht überein, wird die Anforderung verworfen und das System antwortet mit einer Fehlermeldung.

Example Authentifizierte Amazon S3 REST-Anfrage

```
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=
```

Verwenden von temporären Sicherheitsanmeldeinformationen

Wenn Sie Ihre Anfrage unter Verwendung temporärer Sicherheitsanmeldeinformationen signieren (siehe [Senden von Anforderungen](#)), müssen Sie das entsprechende Sicherheitstoken in Ihre Anfrage aufnehmen, indem Sie den `x-amz-security-token`-Header hinzufügen.

Wenn Sie unter Verwendung der AWS -Security-Token-Service API temporäre Sicherheitsanmeldeinformationen erhalten haben, beinhaltet die Antwort temporäre Sicherheitsanmeldeinformationen und ein Sitzungstoken. Sie geben den Wert des Sitzungstokens im `x-amz-security-token`-Header an, wenn Sie Anfragen an Amazon S3 senden. Informationen zur von IAM bereitgestellten AWS -Security-Token-Service API finden Sie [im AWS -Security-Token-Service API-Referenzhandbuch unter](#) Action.

Der Authentifizierungsheader

Die Amazon-S3-REST-API verwendet den standardmäßigen HTTP-Authorization-Header, um Authentifizierungs-Informationen weiterzugeben. (Der Name des Standardheaders ist unglücklich gewählt, weil er eine Authentifizierung weitergibt, keine Autorisierung.) Unter dem Amazon-S3-Authentifizierungsschema hat der Autorisierungsheader die folgende Form:

```
Authorization: AWS AWSAccessKeyId:Signature
```

Entwickler erhalten bei der Registrierung eine AWS Zugriffsschlüssel-ID und einen AWS geheimen Zugriffsschlüssel. Für die Anforderungs-Authentifizierung identifiziert das `AWSAccessKeyId`-Element die Zugriffsschlüssel-ID, die für die Berechnung der Signatur verwendet wurde, und indirekt für den Entwickler steht, der die Anforderung gestellt hat.

Das `Signature` Element ist der RFC 2104 HMAC-SHA1 der ausgewählten Elemente aus der Anfrage, weshalb der `Signature` Teil des Authorization-Headers von Anfrage zu Anfrage unterschiedlich ist. Wenn die vom System berechnete Anforderungssignatur mit der in der Anfrage `Signature` enthaltenen übereinstimmt, hat der Anforderer nachgewiesen, dass er im Besitz des AWS geheimen Zugangsschlüssels ist. die Anforderung wird unter der Identität verarbeitet, und mit der Genehmigung des Entwicklers, dem der Schlüssel ausgestellt wurde.

Die folgende Pseudogrammatik verdeutlicht den Aufbau des `Authorization`-Anforderungs-Headers. (In dem Beispiel steht `\n` für den Unicode-Code U+000A, häufig auch als Newline bezeichnet.)

```
Authorization = "AWS" + " " + AWSAccessKeyId + ":" + Signature;
```

```
Signature = Base64( HMAC-SHA1( UTF-8-Encoding-Of(YourSecretAccessKey), UTF-8-Encoding-Of( StringToSign ) ) );
```

```
StringToSign = HTTP-Verb + "\n" +
Content-MD5 + "\n" +
Content-Type + "\n" +
Date + "\n" +
CanonicalizedAmzHeaders +
CanonicalizedResource;
```

```
CanonicalizedResource = [ "/" + Bucket ] +
<HTTP-Request-URI, from the protocol name up to the query string> +
[ subresource, if present. For example "?acl", "?location", or "?logging"];
```

```
CanonicalizedAmzHeaders = <described below>
```

HMAC-SHA1 ist ein in [RFC 2104 definierter Algorithmus Keyed-Hashing für die Nachrichtenauthentifizierung](#). Der Algorithmus nimmt zwei Byte-Zeichenfolgen als Eingabe entgegen, einen Schlüssel und eine Nachricht. Verwenden Sie für die Amazon S3-Anforderungsauthentifizierung Ihren AWS geheimen Zugriffsschlüssel (YourSecretAccessKey) als Schlüssel und die UTF-8 Kodierung von StringToSign als Nachricht. Die Ausgabe von HMAC-SHA1 ist ebenfalls eine Bytezeichenfolge, die als Digest bezeichnet wird. Der Signature-Anforderungsparameter wird durch eine Base64-Codierung dieses Digest erstellt.

Anfordern einer Kanonisierung für die Signatur

Wenn das System eine authentifizierte Anforderung erhält, vergleicht es die berechnete Anforderungssignatur mit der in der Anforderung im bereitgestellten Signatur, wie bereits beschrieben StringToSign. Aus diesem Grund müssen Sie die Signatur nach derselben Methode berechnen, die auch Amazon S3 verwendet. Der Vorgang, bei dem eine Anfrage zur Unterzeichnung der Kanonisierung in einer vereinbarten Form gestellt wird.

CanonicalizedResource Konstruieren des Elements

CanonicalizedResource stellt die für die Anfrage relevante Amazon-S3-Ressource dar. Für eine REST-Anforderung erstellen Sie sie wie folgt:

1. Beginnen Sie mit einer leeren Zeichenfolge ("").
2. Wenn die Anforderung unter Verwendung des HTTP Host-Headers (virtueller gehosteter Stils) einen Bucket angibt, fügen Sie den Bucket-Namen mit vorgestelltem "/" an (z. B. „/bucketname“).

Für Anforderungen im Pfad-Stil und Anforderungen, die an keinen spezifischen Bucket gerichtet sind, machen Sie nichts. Weitere Informationen zu Anfragen im virtuellen Hosting-Stil finden Sie unter [Virtuelles Hosten von Buckets](#).

Für eine virtuelle Hosting-Anfrage "" ist das `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg „/awsexamplebucket1". CanonicalizedResource`

Für die Anfrage im Pfadstil "" ist das „`. https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg CanonicalizedResource`

3. Hängen Sie den Pfadteil des nicht dekodierten HTTP an, bis zur Request-URI Abfragezeichenfolge, aber nicht einschließlich.

Für eine virtuelle Hosting-Anfrage "" ist das `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/photos/puppy.jpg CanonicalizedResource „//puppy.jpg". awsexamplebucket1/photos`

Für eine Anfrage im Pfadstil "" CanonicalizedResource ist das `https://s3.us-west-1.amazonaws.com/awsexamplebucket1/photos/puppy.jpg „//puppy.jpg". awsexamplebucket1/photos` An dieser Stelle ist der CanonicalizedResource für Anforderungen mit virtuell gehostetem Stil und Pfad-Stil gleich.

Im Fall einer Anforderung, die sich nicht an einen Bucket richtet, wie [GET Service](#), fügen Sie „/" an.

4. Wenn sich die Anforderung an eine Subressource richtet, wie beispielsweise `?versioning`, `?location`, `?acl`, `?lifecycle`, oder `?versionid`, fügen Sie die Subressource an, gegebenenfalls ihren Wert und das Fragezeichen. Beachten Sie, dass bei mehreren Unterressourcen die Unterressourcen lexikografisch nach dem Namen der Unterressource sortiert und durch `'&'` getrennt werden müssen, z. B. `? ACL&VersionId=. value`

Die Unterressourcen, die bei der Erstellung des CanonicalizedResource Elements berücksichtigt werden müssen, sind `acl`, `lifecycle`, `location`, `logging`, `notification`, `partNumber`, `policy`, `requestPayment`, `UploadID`, `uploads`, `versionId`, `versioning`, `versions` und `website`.

Wenn die Anfrage Abfragezeichenfolgen-Parameter angibt, die die Antwort-Header-Werte überschreiben (siehe [Get Object](#)), fügen Sie die Abfragezeichenfolgen-Parameter und ihre Werte an. Bei einer Signatur codieren Sie diese Werte nicht. Bei einer Anforderung dagegen müssen Sie diese Parameterwerte codieren. Die Abfrageparameter in einer GET-Anforderung sind unter anderem `response-content-type`, `response-content-language`, `response-expires`,

`response-cache-control`, `response-content-disposition` und `response-content-encoding`.

Der `delete` Abfragezeichenfolgenparameter muss enthalten sein, wenn Sie die Löschanforderung für mehrere Objekte erstellen. `CanonicalizedResource`

Elemente von `CanonicalizedResource`, die aus dem HTTP stammen, Request-URI sollten wörtlich so signiert werden, wie sie in der HTTP-Anfrage vorkommen, einschließlich URL-Encoding Metazeichen.

Das `CanonicalizedResource` könnte anders sein als das HTTP Request-URI. Insbesondere, wenn Ihre Anfrage den Host HTTP-Header verwendet, um einen Bucket anzugeben, erscheint der Bucket nicht im HTTP Request-URI. Die `CanonicalizedResource` beinhaltet den Bucket jedoch weiterhin. Abfragezeichenfolgenparameter können auch in der vorkommen Request-URI, sind aber nicht in `CanonicalizedResource`. Weitere Informationen finden Sie unter [Virtuelles Hosten von Buckets](#).

Das Element konstruieren `CanonicalizedAmzHeaders`

Um den `CanonicalizedAmzHeaders` Teil von `StringToSign` zu konstruieren, wählen Sie alle HTTP-Anforderungsheader aus, die mit 'x-amz-' beginnen (bei einem Vergleich ohne Berücksichtigung der Groß- und Kleinschreibung), und gehen Sie wie folgt vor.

1. Wandeln Sie jeden HTTP-Headernamen in Kleinbuchstaben um. Beispielsweise wird 'X-Amz-Date' zu 'x-amz-date'.
2. Sortieren Sie die Header alphabetisch nach dem Headernamen.
3. Kombinieren Sie Header-Felder mit demselben Namen zu einem „header-name:comma-separated-value-list“-Paar, wie von RFC 2616 Abschnitt 4.2 vorgegeben, ohne Leerzeichen zwischen den Werten. Die beiden Metadaten-Header 'x-amz-meta-username: fred' und 'x-amz-meta-username: barney' würden beispielsweise zu einem einzigen Header 'x-amz-meta-username: fred,barney' zusammengefasst.
4. „Falten“ Sie lange Header „auf“, die sich über mehrere Zeilen erstrecken (wie durch RFC 2616 Abschnitt 4.2 erlaubt), indem Sie die „faltenden“ Leerzeichen (einschließlich Neue-Zeile-Zeichen) durch ein einzelnes Leerzeichen ersetzen.
5. Löschen Sie alle Leerzeichen um den Doppelpunkt in der Kopfzeile. Zum Beispiel würde der Header " zu 'x-amz-meta-username: fred,barney' werden. x-amz-meta-username:fred,barney

6. Schließlich fügen Sie jedem kanonisierten Header in der Ergebnisliste ein Neue-Zeile-Zeichen (U+000A) hinzu. Konstruieren Sie das CanonicalizedResource Element, indem Sie alle Header in dieser Liste zu einer einzigen Zeichenfolge verketteten.

Positionelle und benannte HTTP-Header-Elemente StringToSign

Die ersten Header-Elemente von `StringToSign` (Content-Type, Date und Content-MD5) sind positioneller Natur. `StringToSign` enthält nicht die Namen dieser Header, sondern nur ihre Werte aus der Anfrage. Im Gegensatz dazu sind die 'x-amz-'-Elemente benannt. Sowohl die Header-Namen als auch die Header-Werte erscheinen in `StringToSign`.

Wenn ein positionaler Header, der in der Definition von `StringToSign` vorgegeben ist, in Ihrer Anforderung nicht vorhanden ist (z. B. sind Content-Type oder Content-MD5 optional für PUT-Anforderungen und sinnlos für GET-Anforderungen), setzen Sie in dieser Position die leere Zeichenfolge ("") ein.

Zeitstempel-Anforderung

Ein gültiger Zeitstempel (unter Verwendung des HTTP-Headers Date oder einer x-amz-date-Alternative) ist zwingend erforderlich für authentifizierte Anforderungen. Darüber hinaus muss der Client-Zeitstempel in einer authentifzierten Anfrage innerhalb eines Zeitraums von 15 Minuten zur Amazon-S3-Systemzeit liegen, wenn die Anfrage empfangen wird. Wenn dies nicht der Fall ist, schlägt die Anforderung mit RequestTimeTooSkewed-Fehlercode fehl. Diese Einschränkungen sollen verhindern, dass abgefangene Anforderungen böswillig wiederholt werden. Für einen strengeren Schutz gegen ein Abhören transportieren Sie authentifizierte Anforderung mit HTTPS.

Note

Die Auswertungsbeschränkung im Hinblick auf das Anforderungsdatum gilt nur für authentifizierte Anforderungen, die keine Abfrage-String-Authentifizierung verwenden. Weitere Informationen finden Sie unter [Alternative zur Authentifizierung der Abfragezeichenfolge](#).

Einige HTTP-Client-Bibliotheken unterstützen die Möglichkeit nicht, den Date-Header für eine Anforderung einzurichten. Wenn Sie Probleme damit haben, den Wert des 'Date'-Headers in kanonisierte Header aufzunehmen, können Sie den Zeitstempel für die Anforderung stattdessen auch mit einem 'x-amz-date'-Header einrichten. Der Wert des x-amz-date-Headers muss in einem der

von RFC 2616 vorgegebenen Formate vorliegen (<http://www.ietf.org/rfc/rfc2616.txt>). Wenn in einer Anforderung ein x-amz-date-Header vorhanden ist, ignoriert das System bei der Berechnung der Anforderungssignatur jeden Date-Header. Wenn Sie also den x-amz-date-Header aufnehmen, verwenden Sie die leere Zeichenfolge für Date, wenn Sie den StringToSign erstellen. Ein Beispiel finden Sie im nächsten Abschnitt.

Authentifizierungsbeispiele

Die Beispiele in diesem Abschnitt verwenden die (nicht funktionierenden) Anmeldeinformationen aus der folgenden Tabelle.

Parameter	Wert
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE
AWSSecretAccessKey	wJalrXUtnFEMI/K7MDENG/bPxrFiCYEXAMPLEKEY

In den StringToSigns des Beispiels ist die Formatierung nicht relevant, und \n steht für den Unicode-Punkt U+000A, allgemein als Neue-Zeile-Zeichen bezeichnet. Außerdem verwenden die Beispiele "+0000", um die Zeitzone anzugeben. Sie können die Zeitzone stattdessen mit "GMT" angeben, aber in den Beispielen werden andere Signaturen verwendet.

Object GET

In diesem Beispiel wird ein Objekt aus dem Bucket „awsexamplebucket1“ abgerufen.

Anforderung	StringToSign
<pre>GET /photos/puppy.jpg HTTP/1.1 Host: awsexamplebucket1.us- west-1.s3.amazonaws.com Date: Tue, 27 Mar 2007 19:36:42 +0000 Authorization: AWS AKIAIOSFO DNN7EXAMPLE: qgk2+6Sv9/oM7G3qLEjTH1a1l1g=</pre>	<pre>GET\n \n \n Tue, 27 Mar 2007 19:36:42 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Beachten Sie, dass der Bucket-Namen CanonicalizedResource enthält, HTTP jedoch Request-URI nicht. (Der Bucket wird vom Host-Header spezifiziert.)

Note

Das folgende Python-Skript berechnet die vorhergehende Signatur unter Verwendung der bereitgestellten Parameter. Sie können dieses Skript verwenden, um Ihre eigenen Signaturen zu erstellen und gegebenenfalls die Schlüssel StringToSign zu ersetzen.

```
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/\nphotos/puppy.jpg'.encode("UTF-8")
signature = base64.b64encode(
    hmac.new(
        secret_key, string_to_sign, sha1
    ).digest()
).strip()

print(f"AWS {access_key.decode()}:{signature.decode()}")
```

Object PUT

In diesem Beispiel wird ein Objekt in den Bucket „awsexamplebucket1“ eingefügt.

Anforderung	StringToSign
<pre>PUT /photos/puppy.jpg HTTP/1.1 Content-Type: image/jpeg Content-Length: 94328 Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:15:45 +0000</pre>	<pre>PUT\n \n image/jpeg\n Tue, 27 Mar 2007 21:15:45 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Anforderung	StringToSign
<pre>Authorization: AWS AKIAIOSFODNN7EXAMPLE LE: iqRzw+ileNPu1fhspnRs8n0jjIA=</pre>	

Notieren Sie sich den Content-Type Header in der Anfrage und in der StringToSign. Beachten Sie auch, dass das in der leer gelassen Content-MD5 wird StringToSign, da es in der Anfrage nicht vorhanden ist.

Auflisten

In diesem Beispiel wird der Inhalt des Buckets „awsexamplebucket1“ aufgelistet.

Anforderung	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>

Beachten Sie den abschließenden Schrägstrich bei CanonicalizedResource und das Fehlen von Abfragezeichenfolgenparametern.

Fetch

In diesem Beispiel wird die Zugriffskontrollrichtlinien-Subressource für den Bucket „awsexamplebucket1“ abgerufen.

Anforderung	StringToSign
<pre>GET /?acl HTTP/1.1</pre>	<pre>GET\n</pre>

Anforderung	StringToSign
<pre>Host: awsexamplebucket1.s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 19:44:46 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: 82ZHiFIjc+WbcwFKGUVeQspPn+0=</pre>	<pre>\n \n Tue, 27 Mar 2007 19:44:46 +0000\n /awsexamplebucket1/?acl</pre>

Beachten Sie, dass der Parameter für die Subresource-Abfragezeichenfolge in der enthalten ist. CanonicalizedResource

Delete

In diesem Beispiel wird ein Objekt über die path-style- und Date-Alternative aus dem Bucket „awsexamplebucket1“ entfernt.

Anforderung	StringToSign
<pre>DELETE /awsexamplebucket1/photos/puppy.jpg HTTP/1.1 User-Agent: dotnet Host: s3.us-west-1.amazonaws.com Date: Tue, 27 Mar 2007 21:20:27 +0000 x-amz-date: Tue, 27 Mar 2007 21:20:26 +0000 Authorization: AWS AKIAIOSFODNN7EXAMPLE: LE:XbyTlbQdu9Xw5o8P4iMwPktxQd8=</pre>	<pre>DELETE\n \n \n Tue, 27 Mar 2007 21:20:26 +0000\n /awsexamplebucket1/photos/puppy.jpg</pre>

Beachten Sie die alternative Methode „x-amz-date“ zur Angabe des Datums (weil unsere Client-Bibliothek uns beispielsweise daran gehindert hat, das Datum festzulegen). In diesem Fall hat x-amz-date Vorrang vor dem Date-Header. Aus diesem Grund muss der Datumseintrag in der Signatur den Wert des x-amz-date-Headers enthalten.

Hochladen

Dieses Beispiel lädt ein Objekt in einen virtuell gehosteten Bucket mit CNAME-Stil mit Metadaten.

Anforderung	StringToSign
<pre> PUT /db-backup.dat.gz HTTP/1.1 User-Agent: curl/7.15.5 Host: static.example.com:8080 Date: Tue, 27 Mar 2007 21:06:08 +0000 x-amz-acl: public-read content-type: application/x-download Content-MD5: 4gJE4saaMU4BqNR0kLY+lw== X-Amz-Meta-ReviewedBy: joe@example.com X-Amz-Meta-ReviewedBy: jane@exam ple.com X-Amz-Meta-FileChecksum: 0x02661779 X-Amz-Meta-ChecksumAlgorithm: crc32 Content-Disposition: attachment; filename=database.dat Content-Encoding: gzip Content-Length: 5913339 Authorization: AWS AKIAIOSFODNN7EXAMP LE: jtBQa0Aq+DkULFI8qrpwIjGEx0E= </pre>	<pre> PUT\n 4gJE4saaMU4BqNR0kLY+lw==\n application/x-download\n Tue, 27 Mar 2007 21:06:08 +0000\n x-amz-acl:public-read\n x-amz-meta-checksumalgorithm: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>

Beachten Sie, wie die „x-amz“-Header sortiert, von Leerzeichen befreit und in Kleinbuchstaben umgewandelt wurden. Beachten Sie außerdem, dass mehrere Header mit demselben Namen unter Verwendung von Kommas zum Trennen der Werte verknüpft wurden.

Beachten Sie, wie nur die HTTP-Header Content-Type und Content-MD5 in StringToSign erscheinen. Die anderen Content-* -Header erscheinen nicht.

Beachten Sie auch hier, dass der den Bucket-Namen CanonicalizedResource enthält, HTTP Request-URI jedoch nicht. (Der Bucket wird vom Host-Header spezifiziert.)

Auflisten aller meiner Buckets

Anforderung	StringToSign
<pre>GET / HTTP/1.1</pre>	<pre>GET\n</pre>

Anforderung	StringToSign
Host: s3.us-west-1.amazonaws.com Date: Wed, 28 Mar 2007 01:29:59 +0000 <i>Authorization: AWS AKIAIOSFODNN7EXAMPLE:qGdzdE RIC03wnaRNKh60qZehG9s=</i>	\n \n Wed, 28 Mar 2007 01:29:59 +0000\n /

Unicode-Schlüssel

Anforderung	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

Die ElementeStringToSign, die aus dem abgeleitet wurden, Request-URI werden wörtlich genommen, einschließlich URL-Encoding und Großschreibung.

Probleme mit dem Signieren von REST-Anforderungen

Wenn die Authentifizierung einer REST-Anforderung fehlschlägt, reagiert das System mit einem XML-Fehlerdokument auf die Anforderung. Die in diesem Fehlerdokument enthaltene Information soll Entwicklern helfen, das Problem zu diagnostizieren. Insbesondere erkennen Sie an dem StringToSign-Element des SignatureDoesNotMatch-Fehlerdokuments genau, welche Anforderungs-Kanonisierung das System verwendet.

Einige Toolkits fügen stillschweigend Header ein, die Sie zuvor nicht kennen, wie beispielsweise den Header Content-Type bei einem PUT. In den meisten dieser Fälle bleibt der Wert des eingefügten Headers konstant, sodass Sie fehlende Header unter Verwendung von Tools wie Ethereal oder tcpmon erkennen können.

Alternative zur Authentifizierung der Abfragezeichenfolge

Einige Anforderungstypen können Sie authentifizieren, indem Sie die angeforderten Informationen als Abfragezeichenfolgenparameter übergeben, statt den HTTP-Header `Authorization` zu verwenden. Dies ist praktisch, um direkten Zugriff durch einen Browser von Dritten auf Ihre privaten Amazon-S3-Daten zu ermöglichen, ohne dass die Anfrage über einen Proxy gesendet wird. Die Idee besteht in der Konstruierung einer „vorsignierten“ Anforderung und ihrer Codierung als einer URL, die vom Browser eines Endbenutzers geladen werden kann. Darüber hinaus können Sie eine vorsignierte Anforderung durch die Angabe einer Ablaufzeit begrenzen.

Weitere Informationen zur Verwendung von Abfrageparametern zur Authentifizierung von Anfragen finden Sie unter [Authentifizieren von Anfragen: Verwenden von Abfrageparametern \(AWS Signature Version 4\)](#) in der Amazon Simple Storage Service API-Referenz. Beispiele für die Verwendung der AWS SDKs zum Generieren vorsignierter URLs finden Sie unter [Objekte mit vorsignierten URLs teilen](#).

Erstellen einer Signatur

Das folgende Beispiel zeigt eine mit Abfragezeichenfolge authentifizierte Amazon-S3-REST-Anfrage.

```
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
```

Die Authentifizierungsmethode für die Abfragezeichenkette benötigt keine spezifischen HTTP-Header. Stattdessen werden die erforderlichen Authentifizierungselemente als Abfragezeichenfolgenparameter angegeben:

Abfragezeichenfolgen-Parametername	Beispielwert	Description
AWSAccessKeyId	AKIAIOSFODNN7EXAMPLE	Ihre AWS Zugangsschlüssel-ID. Gibt den AWS geheimen Zugriffsschlüssel an, der zum Signieren der Anfrage verwendet wurde, und indirekt die

Abfragezeichenfolgen-Parametername	Beispielwert	Description
		Identität des Entwicklers, der die Anfrage stellt.
Expires	1141889120	Die Zeit, wann die Signatur abläuft, angegeben als die Anzahl der Sekunden, die seit dem 1. Januar 1970 00:00:00 UTC verstrichen sind. Eine Anforderung, die nach dieser Zeit empfangen wird (abhängig vom Server), wird abgelehnt.
Signature	vjbyPxybdZaNmGa%2ByT272YEAiv4%3D	Die URL-Kodierung der Base64-Kodierung von HMAC-SHA1 von StringToSign.

Die Methode zur Authentifizierung der Abfragezeichenfolge unterscheidet sich leicht von der üblichen Methode, aber nur im Format des Signature-Anforderungsparameters und des StringToSign-Elements. Die folgende Pseudogrammatik verdeutlicht die Authentifizierungsmethode für die Abfragezeichenfolge.

```
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 ist die AWS geheime Zugriffsschlüssel-ID, die Amazon Ihnen zuweist, wenn Sie sich als Amazon Web Service-Entwickler registrieren. Beachten Sie URL-Encoded, wie Signature das für die Platzierung in der Abfragezeichenfolge geeignet ist. Beachten Sie

auch, dass in `StringToSign` das HTTP-Positionselement `Date` durch `Expires` ersetzt wurde. `CanonicalizedAmzHeaders` und `CanonicalizedResource` sind gleich.

Note

In der Methode für die Abfrage-String-Authentifizierung verwenden Sie den Header `Date` oder `x-amz-date request` nicht, wenn Sie die zu signierende Zeichenfolge berechnen.

Authentifizierung von Abfragezeichenfolgen-Anforderungen

Anforderung	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>

In diesem Beispiel wird davon ausgegangen, dass ein Browser, wenn er die GET-Anfrage stellt, weder einen Content-MD5 oder Content-Type -Header bereitstellt, noch irgendwelche x-amz-Header setzt, sodass diese Teile von leer `StringToSign` bleiben.

Verwendung der Base64-Codierung

Signaturen von HMAC-Anforderungen müssen mit Base64 codiert werden. Die Base64-Codierung wandelt die Signatur in einen einfachen ASCII-String um, der der Anforderung hinzugefügt werden kann. Zeichen, die in der Signatur erscheinen können, wie beispielsweise Plus (+), Schrägstrich (/) oder Gleichheitszeichen (=), müssen wie in einer URI codiert werden. Enthält beispielsweise der Authentifizierungscode ein Plussymbol (+), codieren Sie es in der Anforderung als `&2B`. Einen Schrägstrich codieren Sie als `&2F`, ein Gleichheitszeichen als `%3D`.

Beispiele für die Base64-Codierung finden Sie in Amazon S3 [Authentifizierungsbeispiele](#).

Browser-based lädt mit POST () hoch AWS Signatur (Version 2)

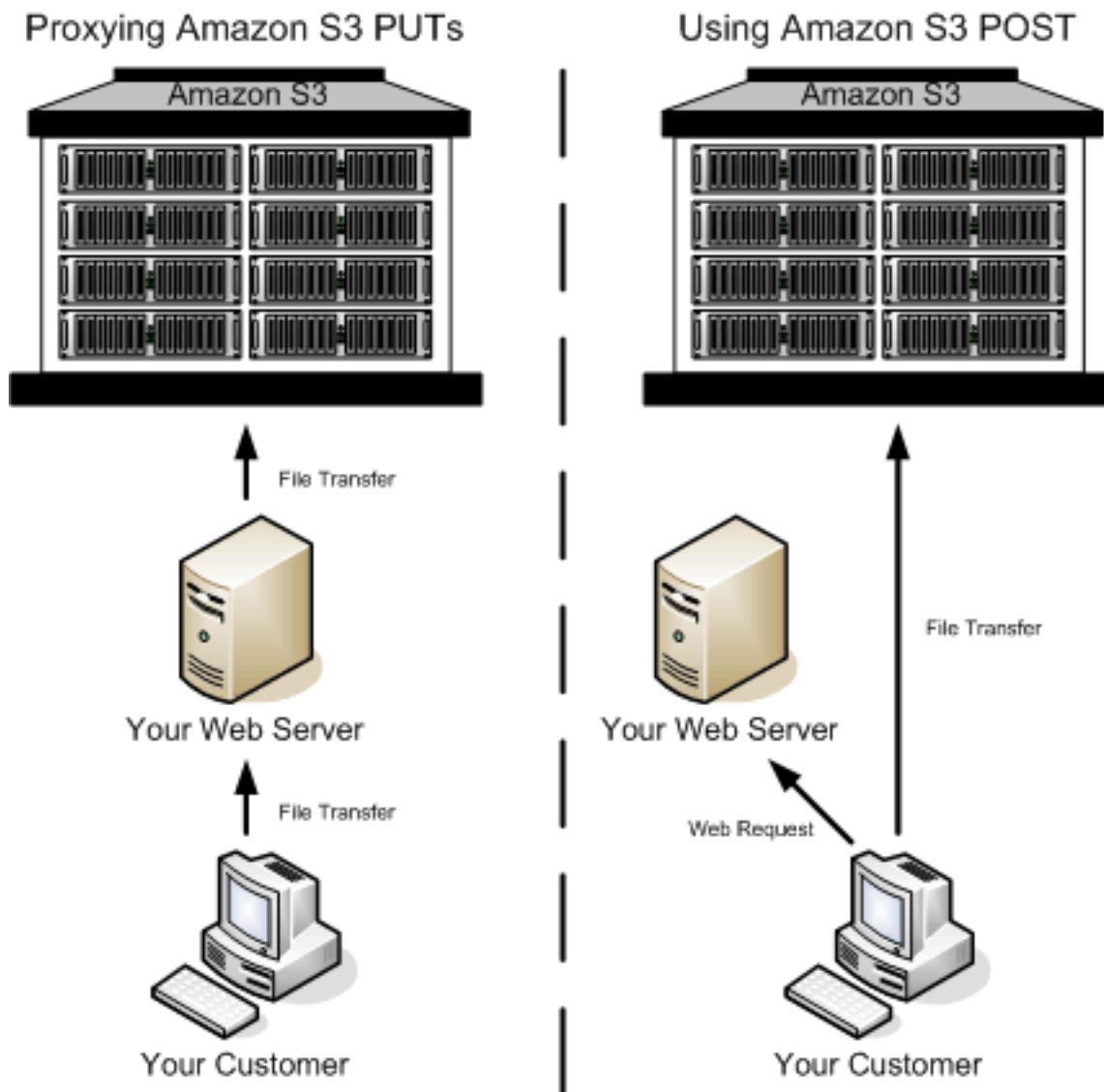
Amazon S3 unterstützt POST; damit können Ihre Benutzer Inhalte direkt zu Amazon S3 hochladen. POST dient dazu, Upload-Vorgänge zu vereinfachen, die Latenz bei Uploads zu reduzieren und Ihnen bei Anwendungen Geld zu sparen, bei denen Benutzer Daten zur Speicherung auf Amazon S3 hochladen.

Note

Die in diesem Abschnitt beschriebene Anforderungsauthentifizierung basiert auf AWS Signature Version 2, einem Protokoll zur Authentifizierung eingehender API-Anfragen an Dienste. AWS

Amazon S3 unterstützt jetzt insgesamt Signature Version 4, ein Protokoll zur Authentifizierung eingehender API-Anfragen an AWS Dienste. AWS-Regionen Derzeit wird das vor dem 30. Januar 2014 AWS-Regionen erstellte Protokoll, Signature Version 2, weiterhin unterstützt. Alle nach dem 30. Januar 2014 neu erstellten Regionen unterstützen nur Signature Version 4, weshalb alle Anforderung an diese Regionen Signature Version 4 verwenden müssen. Weitere Informationen finden Sie unter [Authentifizieren von Anfragen in Browser-Based Uploads mit POST \(AWS Signature Version 4\)](#) in der Amazon Simple Storage Service API-Referenz.

Die folgende Abbildung zeigt einen Upload mit Amazon S3 POST.



Upload mit POST

- 1 Der Benutzer öffnet einen Webbrowser, und greift auf Ihre Webseite zu.
- 2 Ihre Webseite enthält ein HTTP-Formular mit allen Informationen, die der Benutzer benötigt, um Inhalte zu Amazon S3 hochladen zu können.
- 3 Der Benutzer lädt die Inhalte direkt zu Amazon S3 hoch.

Note

Die Abfrage-String-Authentifizierung wird für POST nicht unterstützt.

HTML-Formulare (AWS Signatur (Version 2))

Themen

- [HTML-Formular-Kodierung](#)
- [HTML-Formulardeklaration](#)
- [HTML-Formularfelder](#)
- [Richtlinienerstellung](#)
- [Erstellen einer Signatur](#)
- [Umleitung](#)

Wenn Sie mit Amazon S3 kommunizieren, verwenden Sie normalerweise die REST- oder SOAP-API zum Laden, Abrufen, Löschen und für andere Operationen. Mit POST laden die Benutzer die Daten direkt zu Amazon S3 über ihre Browser hoch, die nicht in der Lage sind, die SOAP-API zu verarbeiten oder eine REST-PUT-Anfrage zu erstellen.

Note

SOAP APIs für Amazon S3 sind für Neukunden nicht verfügbar und nähern sich am 31. August 2025 dem Ende des Lebenszyklus (EOL). Wir empfehlen, entweder die REST-API oder die AWS SDKs zu verwenden.

Um Benutzern zu erlauben, Inhalte über ihre Browser zu Amazon S3 hochzuladen, verwenden Sie HTML-Formulare. HTML-Formulare bestehen aus einer Formulardeklaration und Formularfeldern. Die Formulardeklaration enthält allgemeine Informationen zu der Anfrage. Die Formularfelder enthalten detaillierte Informationen zu der Anfragen sowie die Richtlinie, die verwendet wird, um die Anfrage zu authentifizieren und um sicherzustellen, dass sie den von Ihnen angegebenen Bedingungen entspricht.

Note

Daten und Grenzen des Formulars (ausschließlich der Inhalte der Datei) dürfen 20 KB nicht überschreiten.

Dieser Abschnitt erläutert die Verwendung von HTML-Formularen.

HTML-Formular-Kodierung

Das Formular und die Richtlinie müssen UTF-8 codiert sein. Sie können das Formular UTF-8 codieren, indem Sie es in der HTML-Überschrift oder als Anforderungsheader angeben.

Note

Die HTML-Formulardeklaration akzeptiert keine Abfrage-String-Authentifizierungsparameter.

Das Folgende ist ein Beispiel für die UTF-8 Codierung in der HTML-Überschrift:

```
<html>
  <head>
    ...
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    ...
  </head>
  <body>
```

Das Folgende ist ein Beispiel für die UTF-8 Kodierung in einem Anforderungsheader:

```
Content-Type: text/html; charset=UTF-8
```

HTML-Formulardeklaration

Die Formulardeklaration besteht aus drei Komponenten: der Aktion, der Methode und dem Umschließungstyp. Wenn einer dieser Werte nicht korrekt eingestellt ist, schlägt die Anforderung fehl.

Die Aktion gibt die URL an, die die Anforderung verarbeitet; dies muss die URL des Buckets sein. Wenn der Name Ihres Buckets beispielsweise `awsexamplebucket1` lautet und die Region USA West (Nordkalifornien) ist, lautet die URL `https://awsexamplebucket1.s3.us-west-1.amazonaws.com/`.

Note

Der Schlüsselname wird in einem Formularfeld angegeben.

Die Methode muss POST sein.

Der Gehäusetyp (enctype) muss angegeben und sowohl für Datei-Uploads als auch für Textbereichs-Uploads auf multipart/form -data gesetzt werden. Weitere Informationen finden Sie unter [RFC 1867](#).

Example

Das folgende Beispiel zeigt eine Formulardeklaration für den Bucket „awsexamplebucket1“.

```
<form action="https://awsexamplebucket1.s3.us-west-1.amazonaws.com/" method="post"
enctype="multipart/form-data">
```

HTML-Formularfelder

Die folgende Tabelle beschreibt die Felder, die in einem HTML-Formular verwendet werden können.

 Note

Die Variable `${filename}` wird automatisch durch den Namen der Datei ersetzt, den der Benutzer bereitstellt, und der von allen Formularfeldern anerkannt wird. Wenn der Browser oder Client einen vollständigen oder teilweisen Pfad zu der Datei bereitstellt, wird nur der Text nach dem letzten Schrägstrich (/) oder umgekehrten Schrägstrich (\) verwendet. Beispielsweise wird „C:\Program Files\directory1\file.txt“ als „file.txt“ interpretiert. Wenn keine Datei bzw. kein Dateiname angegeben ist, wird die Variable durch eine leere Zeichenfolge ersetzt.

Feldname	Beschreibung	Erforderlich
AWSAccessKeyId	Die AWS Zugriffsschlüssel-ID des Besitzers des Buckets, der einem anonymen Benutzer Zugriff auf eine Anfrage gewährt, die die Einschränkungen der Richtlinie erfüllt. Dies ist ein Pflichtfeld, wenn die Anforderung ein Richtliniendokument beinhaltet.	Bedingt
acl		Nein

Feldname	Beschreibung	Erforderlich
	Eine Amazon-S3-Zugriffskontrollliste (ACL). Wenn eine ungültige Zugriffskontrollliste angegeben ist, wird ein Fehler generiert. Typ: Zeichenfolge Standard: privat Zulässige Werte: <code>private</code> <code>public-read</code> <code>public-read-write</code> <code>aws-exec-read</code> <code>authenticated-read</code> <code>bucket-owner-read</code> <code>bucket-owner-full-control</code>	
Cache-Control, Content-Type, Content-Disposition, Content-Encoding, Expires	REST-specific Überschriften. Weitere Informationen finden Sie unter PUT Object .	Nein
key	Der Name des hochgeladenen Schlüssels. verwenden Sie die Variable <code>\$(filename)</code>, um den von dem Benutzer angegebenen Dateinamen zu verwenden. Wenn beispielsweise der Benutzer Betty die Datei <code>lolcatz.jpg</code> hochlädt und Sie <code>user/betty/\${filename}</code> angeben, wird die Datei als <code>//lolcatz.jpg</code> gespeichert. <code>user/betty</code> Weitere Informationen finden Sie unter Arbeiten mit Objektmetadaten.	Ja

Feldname	Beschreibung	Erforderlich
policy	Die Sicherheitsrichtlinie, die angibt, was in der Anforderung zulässig ist. Anforderungen ohne Sicherheitsrichtlinie gelten als anonym und sind nur auf öffentlich zugänglichen Buckets erfolgreich.	Nein
success_action_redirect, redirect	Die URL, zu der der Client nach einem erfolgreichen Upload umgeleitet wird. Amazon S3 hängt die Bucket-, Schlüssel- und Etag-Werte als Abfragezeichenfolgenparameter an die URL an. Wenn success_action_redirect nicht angegeben ist, gibt Amazon S3 den leeren Dokumententyp zurück, der im Feld success_action_status angegeben ist. Wenn Amazon S3 die URL nicht interpretieren kann, wird das Feld ignoriert. Wenn der Upload fehlschlägt, zeigt Amazon S3 einen Fehler an und leitet den Benutzer nicht zu einer URL um. Weitere Informationen finden Sie unter Umleitung.  Note Der Name des Umleitungsfeldes ist veraltet, und das Feld wird künftig nicht mehr unterstützt.	Nein

Feldname	Beschreibung	Erforderlich
success_action_status	Der an den Client bei einem erfolgreichen Upload ausgegebene Statuscode, wenn success_action_redirect nicht angegeben ist. Gültige Werte sind 200, 201 und 204 (Standard). Wenn der Wert auf 200 oder 204 gesetzt ist, gibt Amazon S3 ein leeres Dokument mit dem Statuscode 200 oder 204 aus. Wenn der Wert auf 201 gesetzt ist, gibt Amazon S3 ein XML-Dokument mit dem Statuscode 201 aus. Informationen zum Inhalt des XML-Dokuments finden Sie unter POST Object. Wenn der Wert nicht oder auf einen falschen Wert gesetzt ist, gibt Amazon S3 ein leeres Dokument mit dem Statuscode 204 aus.	Nein

 **Note**

Einige Versionen von Adobe Flash Player können HTTP-Antworten ohne Text nicht korrekt bearbeiten. Zur Unterstützung von Uploads über Adobe Flash empfehlen wir, success_action_status auf 201 zu setzen.

Feldname	Beschreibung	Erforderlich
signature	Die HMAC-Signatur, die mithilfe des geheimen Zugriffsschlüssels erstellt wurde, der dem angegebenen entspricht. <code>AWSSessionToken</code> Dies ist ein Pflichtfeld, wenn die Anforderung ein Richtliniendokument beinhaltet. Weitere Informationen finden Sie unter Identitäts- und Zugriffsmanagement für Amazon S3.	Bedingt
x-amz-security-token	Ein Sicherheitstoken, das von Sitzungsanmeldeinformationen verwendet wird Wenn die Anfrage Amazon DevPay verwendet, sind zwei x-amz-security-token Formularfelder erforderlich: eines für das Produkt-Token und eines für das Benutzer-Token. Wenn die Anforderung Sitzungsanmeldeinformationen verwendet, ist ein x-amz-security-token -Formular erforderlich. Weitere Informationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen im IAM-Benutzerhandbuch.	Nein
Andere Feldnamen mit dem Präfix x-amz-meta-	User-specified Metadaten. Amazon S3 validiert oder verwendet diese Daten nicht. Weitere Informationen finden Sie unter PUT Object.	Nein

Feldname	Beschreibung	Erforderlich
file	Datei oder Textinhalt. Die Datei oder der Inhalt muss das letzte Feld des Formulars sein. Alle Felder darunter werden ignoriert. Sie können nicht mehr als eine Datei zur gleichen Zeit hochladen.	Ja

Richtlinienerstellung

Themen

- [Ablauf](#)
- [Bedingungen](#)
- [Übereinstimmung von Bedingungen](#)
- [Escape-Zeichen](#)

Die Richtlinie ist ein UTF-8 Base64-encoded JSON-Dokument, das die Bedingungen festlegt, die die Anforderung erfüllen muss, und das zur Authentifizierung des Inhalts verwendet wird. Je nach dem, wie Sie Ihre Richtliniendokumente gestalten, können Sie sie pro Upload, pro Benutzer, für alle Uploads oder nach anderen Kriterien, die Ihren Anforderungen entsprechen, verwenden.

Note

Obwohl das Richtliniendokument optional ist, empfehlen nachdrücklich, es zu verwenden, anstatt einen Bucket öffentlich beschreibbar zu machen.

Nachstehend finden Sie ein Beispiel für ein Richtliniendokument:

```
{ "expiration": "2007-12-01T12:00:00.000Z",  
  
  "conditions": [
```

```
{ "acl": "public-read" },  
  
  { "bucket": "awsexamplebucket1" },  
  
  [ "starts-with", "$key", "user/eric/" ],  
  
]  
  
}
```

Das Richtliniendokument enthält den Ablauf und die Bedingungen.

Ablauf

Das Ablaufelement gibt das Ablaufdatum der Richtlinie im Datumsformat nach ISO 8601 UTC an. Beispielsweise gibt „2007-12-01T12:00:00.000Z“ an, dass die Richtlinie am 01.12.2007 nach Mitternacht UTC nicht gültig ist. Die Angabe des Ablaufs ist in einer Richtlinie erforderlich.

Bedingungen

Die Bedingungen in dem Richtliniendokument validieren den Inhalt des hochgeladenen Objekts. Jedes Formularfeld, das Sie im Formular angeben (mit Ausnahme von Signatur-AWSAccessKeyId, Datei-, Richtlinien- und Feldnamen mit dem Präfix x-ignore-), muss in der Liste der Bedingungen enthalten sein.

Note

Wenn Sie mehrere Felder mit dem gleichen Namen haben, müssen die Werte durch Kommata abgeteilt sein. Zum Beispiel: Wenn Sie zwei Felder mit der Bezeichnung „x-amz-meta-tag“ haben und das erste den Wert „Ninja“ und das zweite den Wert „Stallman“ hat, setzen Sie das Richtliniendokument auf `Ninja,Stallman`.

Alle Variablen in dem Formular werden vor der Validierung der Richtlinie erweitert. Daher müssen alle Bedingungsabgleiche anhand der erweiterten Felder vorgenommen werden. Zum Beispiel: Wenn Sie das Schlüsselfeld auf `user/betty/${filename}` setzen, ist Ihre Richtlinie möglicherweise `[ "starts-with", "$key", "user/betty/" ]`. Geben Sie nicht `[ "starts-with", "$key", "user/betty/${filename}" ]`. Weitere Informationen finden Sie unter [Übereinstimmung von Bedingungen](#).

Die folgende Tabelle beschreibt die Bedingungen für Richtliniendokumente.

Elementname	Beschreibung
acl	Gibt die Bedingungen an, die die ACL erfüllen muss. Unterstützt exakte Übereinstimmung und <code>starts-with</code> .
content-length-range	Gibt die erlaubte Mindest- und Höchstgröße des hochgeladenen Inhalts an. Unterstützt den Bereichsabgleich.
Cache-Control,, Content-Type, Content-DispositionContent-Encoding, Läuft ab	REST-specific Header. Unterstützt exakte Übereinstimmung und <code>starts-with</code> .
Schlüssel	Der Name des hochgeladenen Schlüssels. Unterstützt exakte Übereinstimmung und <code>starts-with</code> .
success_action_redirect, redirect	Die URL, zu der der Client nach einem erfolgreichen Upload umgeleitet wird. Unterstützt exakte Übereinstimmung und <code>starts-with</code> .
success_action_status	Der an den Client bei einem erfolgreichen Upload ausgegebene Statuscode, wenn <code>success_action_redirect</code> nicht angegeben ist. Unterstützt exakte Übereinstimmung.
x-amz-security-token	DevPay Amazon-Sicherheitstoken. Für jede Anfrage, die Amazon verwendet, DevPay sind zwei <code>x-amz-security-token</code> Formularfelder erforderlich: eines für das Produkt-Token und eines für das Benutzer-Token. Daher müssen die Werte durch Kommata voneinander

Elementname	Beschreibung
	er getrennt werden. Zum Beispiel: Wenn der Benutzer-Token <code>eW91dHViZQ==</code> und der Produkt-Token <code>b0hnNVNKWVJIQTA=</code> ist, setzen Sie den Richtlinieneintrag auf: <pre>{ "x-amz-security-token": "eW91dHViZQ==,b0hnNVNKWVJIQTA=" }</pre>
Andere Feldnamen mit dem Präfix <code>x-amz-meta-</code>	User-specified Metadaten. Unterstützt exakte Übereinstimmung und <code>starts-with</code> .

Note

Wenn Ihr Toolkit weitere Felder hinzufügt (beispielsweise fügt Flash den Dateinamen hinzu), müssen Sie diese dem Richtliniendokument hinzufügen. Wenn Sie diese Funktionalität steuern können, setzen Sie `x-ignore-` vor das Feld, sodass Amazon S3 die Funktion ignoriert und künftige Versionen davon unbeeinflusst lässt.

Übereinstimmung von Bedingungen

Die folgende Tabelle beschreibt die Übereinstimmungstypen von Bedingungen. Obwohl Sie für jedes Formularfeld, das Sie in dem Formular angeben, eine Bedingung angeben müssen, können Sie auch komplexere Übereinstimmungskriterien erstellen, indem Sie für ein Formularfeld mehrere Bedingungen angeben.

Bedingung	Beschreibung
Genaue Übereinstimmung	Hierbei müssen die Felder spezifische Werte enthalten. In diesem Beispiel ist angegeben, dass die ACL auf „public-read“ gesetzt werden muss: <pre>{"acl": "public-read" }</pre> Dieses Beispiel zeigt eine alternative Möglichkeit, um anzuzeigen, dass die ACL auf „public-read“ gesetzt werden muss:

Bedingung	Beschreibung
	<pre>["eq", "\$acl", "public-read"]</pre>
Beginnt mit	Verwenden Sie diese Bedingung, wenn der Wert mit einem bestimmten Wert beginnen muss. Dieses Beispiel gibt an, dass der Schlüssel beginnen muss mit user/betty: <pre>["starts-with", "\$key", "user/betty/"]</pre>
Übereinstimmung mit beliebigem Inhalt	Verwenden Sie „starts-with“ mit einem leeren Wert, um die Richtlinie so zu konfigurieren, dass in einem Feld beliebiger Inhalt zulässig ist. Dieses Beispiel lässt jeden Wert für success_action_redirect zu: <pre>["starts-with", "\$success_action_redirect", ""]</pre>
Angabe von Bereichen	Wenn Felder Bereiche akzeptieren, trennen Sie die Ober- und die Untergrenze durch ein Komma voneinander ab. In diesem Beispiel sind Dateigrößen von 1 bis 10 Megabyte erlaubt: <pre>["content-length-range", 1048579, 10485760]</pre>

Escape-Zeichen

Die folgende Tabelle beschreibt Zeichen, für die in einem Richtliniendokument ein Escape-Zeichen verwendet werden muss.

Escape-Sequenz	Beschreibung
\\	Umgekehrter Schrägstrich

Escape-Sequenz	Beschreibung
<code>\\$</code>	Dollarzeichen
<code>\b</code>	Backspace
<code>\f</code>	Seitenvorschub
<code>\n</code>	Neue Zeile
<code>\r</code>	Zeilenumschaltung
<code>\t</code>	Horizontaler Tabulator
<code>\V</code>	Vertikaler Tabulator
<code>\ u <i>xxxx</i></code>	Alle Unicode-Zeichen

Erstellen einer Signatur

Schritt	Beschreibung
1	Codieren Sie die Richtlinie mit UTF-8.
2	Codieren Sie diese UTF-8 Bytes mithilfe von Base64.
3	Signieren Sie die Richtlinie mit Ihrem geheimen Zugriffsschlüssel, indem Sie HMAC verwenden. SHA-1
4	Codieren Sie die SHA-1 Signatur mithilfe von Base64.

Allgemeine Informationen zur Authentifizierung finden Sie unter [Identitäts- und Zugriffsmanagement für Amazon S3](#).

Umleitung

In diesem Abschnitt wird beschrieben, wie Sie mit Umleitungen umgehen.

Allgemeine Umleitung

Beim Abschluss der POST-Anforderung wird der Benutzer zu dem Ort umgeleitet, den Sie im Feld `success_action_redirect` angeben. Wenn Amazon S3 die URL nicht interpretieren kann, wird das Feld `success_action_redirect` ignoriert.

Wenn `success_action_redirect` nicht angegeben ist, gibt Amazon S3 den leeren Dokumententyp zurück, der im Feld `success_action_status` angegeben ist.

Wenn die POST-Anfrage fehlschlägt, zeigt Amazon S3 einen Fehler an und führt keine Umleitung durch.

Pre-upload Umleitung

Wenn Ihr Bucket mit `< CreateBucketConfiguration >` erstellt wurde, benötigen Ihre Endbenutzer möglicherweise eine Weiterleitung. Wenn dies der Fall ist, kann es sein, dass einige Browser mit der Umleitung nicht korrekt umgehen. Dies kommt relativ selten vor, meistens dann, wenn der Bucket gerade eben erstellt wurde.

Laden Sie Beispiele hoch (AWS Signatur (Version 2))

Themen

- [Datei-Upload](#)
- [Textbereich-Upload](#)

Note

Die in diesem Abschnitt beschriebene Anforderungsauthentifizierung basiert auf AWS Signature Version 2, einem Protokoll zur Authentifizierung eingehender API-Anfragen an AWS Dienste.

Amazon S3 unterstützt jetzt insgesamt Signature Version 4, ein Protokoll zur Authentifizierung eingehender API-Anfragen an AWS Dienste. AWS-Regionen Derzeit

wird das vor dem 30. Januar 2014 AWS-Regionen erstellte Protokoll, Signature Version 2, weiterhin unterstützt. Alle nach dem 30. Januar 2014 neu erstellten Regionen unterstützen nur Signature Version 4, weshalb alle Anforderung an diese Regionen Signature Version 4 verwenden müssen. Weitere Informationen finden Sie unter [Beispiele: Browser-Based Upload mit HTTP POST \(unter Verwendung von AWS Signature Version 4\)](#) in der Amazon Simple Storage Service API-Referenz.

Datei-Upload

Dieses Beispiel zeigt den vollständigen Vorgang der Erstellung einer Richtlinie und eines Formulars für den Upload eines Dateianhangs.

Erstellung von Richtlinie und Formular

Die folgende Richtlinie unterstützt Uploads zu Amazon S3 für den Bucket „awsexamplebucket1“.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html"},
    ["starts-with", "$Content-Type", "image/"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

Für diese Richtlinie ist Folgendes erforderlich:

- Der Upload muss vor 12:00 Uhr UTC am 1. Dezember 2007 erfolgen.
- Der Inhalt muss zum Bucket „awsexamplebucket1“ hochgeladen werden.
- Der Schlüssel muss mit "user/eric/" beginnen.
- Die ACL ist auf „public-read“ gesetzt.
- Der success_action_redirect ist gesetzt auf https://awsexamplebucket1.s3.us-west-1.amazonaws.com/successful_upload.html.
- Bei dem Objekt handelt es sich um eine Bilddatei.

- Der Tag x-amz-meta-uuid muss auf 14365123651274 gesetzt sein.
- Der Tag x-amz-meta-tag kann einen beliebigen Wert enthalten.

Das Folgende ist eine Base64-encoded Version dieser Richtlinie.

```
eyAiZXhwaXJhdGlvbiI6IClyMDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgeyJidWN
```

Erstellen Sie unter Verwendung Ihrer Anmeldedaten eine Signatur, zum Beispiel ist 0RavWzkygo6QX9caELEqKi9kDbU= die Signatur für das vorangehende Richtliniendokument.

Das folgende Formular unterstützt eine POST-Anfrage an den amzn-s3-demo-bucket Bucket, der diese Richtlinie verwendet.

```
<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>
```

Beispielanforderung

Diese Anforderung geht davon aus, dass das hochgeladene Bild 117.108 Bytes groß ist, die Bilddaten nicht eingeschlossen.

```
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"

eyJhZXBwaXJhdGlvbiI6ICIyMDA3LTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgcyJidWN
--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--
```

Beispielantwort

```
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
```

Textbereich-Upload

Themen

- [Erstellung von Richtlinie und Formular](#)
- [Beispielanforderung](#)
- [Beispielantwort](#)

Das folgende Beispiel zeigt den vollständigen Prozess der Erstellung einer Richtlinie und eines Formulars für den Upload eines Textbereiches. Der Upload eines Textbereiches ist nützlich für die Übermittlung von Benutzern erstellter Inhalte, etwa von Blog-Postings.

Erstellung von Richtlinie und Formular

Die folgende Richtlinie unterstützt das Hochladen von Textbereichen zu Amazon S3 für den Bucket „awsexamplebucket1“.

```
{ "expiration": "2007-12-01T12:00:00.000Z",
  "conditions": [
    {"bucket": "awsexamplebucket1"},
    ["starts-with", "$key", "user/eric/"],
    {"acl": "public-read"},
    {"success_action_redirect": "https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html"},
    ["eq", "$Content-Type", "text/html"],
    {"x-amz-meta-uuid": "14365123651274"},
    ["starts-with", "$x-amz-meta-tag", ""]
  ]
}
```

Für diese Richtlinie ist Folgendes erforderlich:

- Der Upload muss vor 12:00 Uhr GMT am 1. Dezember 2007 erfolgen.
- Der Inhalt muss zum Bucket „awsexamplebucket1“ hochgeladen werden.
- Der Schlüssel muss mit "user/eric/" beginnen.
- Die ACL ist auf „public-read“ gesetzt.
- Der success_action_redirect ist gesetzt auf https://awsexamplebucket1.s3.us-west-1.amazonaws.com/new_post.html.
- Bei dem Objekt handelt es sich um HTML-Text.
- Der Tag x-amz-meta-uuid muss auf 14365123651274 gesetzt sein.
- Der Tag x-amz-meta-tag kann einen beliebigen Wert enthalten.

Im Folgenden finden Sie eine Base64-encoded Version dieser Richtlinie.

```
eyJhZlZxhwaXJhdGlvbiI6IClYMDA3LTExLTExVDEyMDAwMDAwLjAwMFoiLAogICJjb25kaXR
```

```
pb25zIjogWwogICAgeyJidWNrZXQiOiAiam9obnNtaXR0In0sCiAgICBbInN0YXJ0cy13aXR0IiwgIiRrZXkiLCaidXNlci
LAogICAgeyJhY2wiOiAicHVibGljLXJlYWQifSwKICAgIHsic3VjY2Vzcl9hY3Rpb25fcmVkaXJlY3QiOiAiaHR0cDovL2p
C5zMy5hbWV6b25hd3MuY29tL25ld19wb3N0Lmh0bWwifSwKICAgIFsiZXEiLCAiJENvbnRlbnQtVHlwZSI6ICJ0ZXh0L2h0
CAgIHsieC1hbXotbWV0YS11dWlkIjogIjE0MzY1MTIzNjUxMjc0In0sCiAgICBbInN0YXJ0cy13aXR0IiwgIiR4LWFtei1t
IsICIiXQogIF0KfQo=
```

Erstellen Sie unter Verwendung Ihrer Anmeldedaten eine Signatur. Beispielsweise ist qA7FWXKq6VvU68lI9KdveT1cWgF= die Signatur für das vorherige Richtliniendokument.

Das folgende Formular unterstützt eine POST-Anfrage an den amzn-s3-demo-bucket Bucket, der diese Richtlinie verwendet.

```
<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>
```

Beispielanforderung

Diese Anforderung geht davon aus, dass das hochgeladene Bild 117.108 Bytes groß ist, die Bilddaten nicht eingeschlossen.

```
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"
eyJhZlZxhwaXJhdGlvbiI6ICIyMDA3LTExLTExVDEyOjAwOjAwLjAwMFoiLAogICJjb25kaXRpb25zIjogWwogICAgcyJidWw
--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--
```

Beispielantwort

```
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 mit Adobe Flash (AWS Signatur (Version 2))

In diesem Abschnitt wird die Verwendung von POST mit Adobe Flash beschrieben.

Adobe Flash Player-Sicherheit

Standardmäßig verbietet das Sicherheitsmodell von Adobe Flash, dass Adobe Flash Players Netzwerkverbindungen zu Servern außerhalb der Domäne der SWF-Datei herstellen.

Um diesen Standard zu übergehen, müssen Sie eine öffentlich lesbare `crossdomain.xml`-Datei zu dem Bucket hochladen, der die POST-Uploads annehmen soll. Nachfolgend sehen Sie ein Beispiel für eine solche `crossdomain.xml`-Datei.

```
<?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

Weitere Informationen zum Sicherheitsmodell von Adobe Flash finden Sie auf der Adobe-Website.

Durch die Hinzufügung der `crossdomain.xml`-Datei kann jeder Adobe Flash Player eine Verbindung zu der `crossdomain.xml`-Datei in Ihrem Bucket herstellen; diese Datei gewährt jedoch keinen Zugriff auf den Amazon-S3-Bucket selbst.

Überlegungen zu Adobe Flash

Die FileReference API in Adobe Flash fügt das `Filename` Formularfeld zur POST-Anfrage hinzu. Wenn Sie Adobe Flash-Anwendungen erstellen, die mithilfe der FileReference API-Aktion auf Amazon S3 hochgeladen werden, nehmen Sie die folgende Bedingung in Ihre Richtlinie auf:

```
['starts-with', '$Filename', '']
```

Einige Versionen von Adobe Flash Player können HTTP-Antworten ohne Text nicht korrekt bearbeiten. Um POST so zu konfigurieren, dass es eine Antwort ausgibt, die nicht leer ist, setzen Sie `success_action_status` auf 201. Amazon S3 gibt dann ein XML-Dokument mit dem Statuscode 201 zurück. Informationen zum Inhalt des XML-Dokuments finden Sie unter [POST Object](#). Informationen zu den Formularfeldern finden Sie unter [HTML-Formularfelder](#).

Anhang: APIs für die Lifecycle-Konfiguration (veraltet)

Die Bucket-Lifecycle-Konfiguration wurde aktualisiert, um Filter zu unterstützen, die auf Objekt-Tags basieren. Das heißt, Sie können jetzt eine Regel angeben, die ein Schlüsselnamenpräfix, ein oder

mehrere Objekt-Tags oder beides angibt, um eine Teilmenge von Objekten auszuwählen, für die die Regel gilt. Die APIs wurden entsprechend aktualisiert. In den folgenden Themen wird die vorherige Version der PUT- und GET-Bucket-Lebenszyklusoperationen aus Gründen der Abwärtskompatibilität beschrieben.

Themen

- [PUT-Bucket-Lebenszyklus \(veraltet\)](#)
- [BUCKET-Lebenszyklus ABRUFEN \(veraltet\)](#)

PUT-Bucket-Lebenszyklus (veraltet)

Description

Important

Eine aktualisierte Version dieser API finden Sie unter [PutBucketLifecycleConfiguration](#). Diese Version ist veraltet. Bestehende Lifecycle-Konfigurationen werden funktionieren. Verwenden Sie für neue Lebenszykluskonfigurationen die aktualisierte API.

Erstellt eine Lebenszykluskonfiguration für den Bucket oder ersetzt eine vorhandene Lebenszykluskonfiguration. Informationen zur Lebenszykluskonfiguration finden Sie unter [Object Lifecycle Management](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Berechtigungen

Standardmäßig sind alle Amazon S3-Ressourcen, einschließlich Buckets, Objekte und zugehörige Unterressourcen (z. B. Lifecycle-Konfiguration und Website-Konfiguration), privat. Nur der Ressourcenbesitzer, der AWS-Konto die Ressource erstellt hat, kann darauf zugreifen. Der Ressourcenbesitzer kann optional anderen Zugriffsberechtigungen gewähren, indem er eine Zugriffsrichtlinie schreibt. Für diesen Vorgang müssen Benutzer die `s3:PutLifecycleConfiguration` Erlaubnis erhalten.

Sie können Berechtigungen auch explizit verweigern. Die ausdrückliche Verweigerung hat auch Vorrang vor allen anderen Berechtigungen. Wenn Sie verhindern möchten, dass Benutzer oder Konten Objekte aus Ihrem Bucket entfernen oder löschen, müssen Sie ihnen die Berechtigungen für die folgenden Aktionen verweigern:

- `s3:DeleteObject`
- `s3:DeleteObjectVersion`
- `s3:PutLifecycleConfiguration`

Weitere Informationen zu Berechtigungen finden Sie unter [Verwalten von Zugriffsberechtigungen für Ihre Amazon S3-Ressourcen](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Anforderungen

Syntax

```
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

Einzelheiten zu Autorisierungszeichenfolgen finden Sie unter [Anfragen authentifizieren \(AWS Signatur \(Version 4\)\)](#).

Anforderungsparameter

Bei dieser Implementierung der Operation werden keine Anforderungsparameter verwendet.

Anforderungsheader

Name	Description	Erforderlich
Content-MD5	Der Base64-kodierte 128-Bit-MD5-Digest der Daten. Sie müssen diesen Header zur Überprüfung der Nachrichtenintegrität verwenden, um sicherzustellen, dass der Anforderungstext bei der Übertragung nicht beschädigt wurde. Weitere Informationen finden Sie unter RFC 1864. Typ: Zeichenfolge Standard: keiner	Ja

Anforderungstext

In der Anfrage geben Sie die Lifecycle-Konfiguration im Anforderungstext an. Die Lifecycle-Konfiguration ist als XML angegeben. Das Folgende ist ein Beispiel für eine grundlegende

Lebenszykluskonfiguration. Sie spezifiziert eine Regel. Das Prefix in der Regel identifiziert Objekte, für die die Regel gilt. Die Regel spezifiziert auch zwei Aktionen (Transition und Expiration). Jede Aktion gibt einen Zeitplan an, zu dem Amazon S3 die Aktion ausführen soll. Das Status gibt an, ob die Regel aktiviert oder deaktiviert ist.

```
<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>
```

Wenn der Status Ihres Buckets die Versionierung aktiviert oder die Versionierung gesperrt ist, können Sie mehrere Versionen desselben Objekts haben: eine aktuelle Version und keine oder mehrere nicht aktuelle Versionen. Die folgende Lifecycle-Konfiguration spezifiziert die Aktionen (NoncurrentVersionTransition, NoncurrentVersionExpiration), die für nicht aktuelle Objektversionen spezifisch sind.

```
<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>
```

Sie können die API für mehrteilige Uploads verwenden, um große Objekte in Teilen hochzuladen. Weitere Informationen zu mehrteiligen Uploads finden Sie unter Überblick über [mehrtelilige Uploads](#) im Amazon Simple Storage Service-Benutzerhandbuch. Mit der Lebenszykluskonfiguration können Sie Amazon S3 anweisen, unvollständige mehrteilige Uploads, die durch das in der Regel angegebene Schlüsselnamenpräfix gekennzeichnet sind, zu stornieren, wenn sie nicht innerhalb einer bestimmten Anzahl von Tagen abgeschlossen werden. Wenn Amazon S3 einen mehrteiligen Upload storniert, werden alle mit dem Upload verknüpften Teile gelöscht. Dadurch wird sichergestellt, dass Sie keine unvollständigen mehrteiligen Uploads haben, bei denen noch Teile in Amazon S3 gespeichert sind, sodass Sie keine Lagerkosten dafür zahlen müssen. Das folgende Beispiel zeigt eine Lebenszykluskonfiguration, die eine Regel mit der Aktion `AbortIncompleteMultipartUpload` spezifiziert. Durch diese Aktion wird Amazon S3 angewiesen, unvollständige mehrteilige Uploads sieben Tage nach der Initiierung zu stornieren.

```
<LifecycleConfiguration>
  <Rule>
    <ID>sample-rule</ID>
    <Prefix>SomeKeyPrefix</Prefix>
    <Status>rule-status</Status>
    <AbortIncompleteMultipartUpload>
      <DaysAfterInitiation>7</DaysAfterInitiation>
    </AbortIncompleteMultipartUpload>
  </Rule>
</LifecycleConfiguration>
```

In der folgenden Tabelle werden die XML-Elemente in der Lifecycle-Konfiguration beschrieben.

Name	Description	Erforderlich
AbortIncompleteMultipartUpload	Container zur Angabe, wann ein unvollständiger mehrteiliger Upload für einen Abbruchvorgang in Frage kommt. Untergeordnet: DaysAfterInitiation Typ: Container Vorfahr: Rule	Ja, wenn für die Regel keine andere Aktion angegeben ist

Name	Description	Erforderlich
Date	Datum, an dem Amazon S3 die Aktion ausführen soll. Weitere Informationen finden Sie unter Lebenszyklusregeln: Basierend auf einem bestimmten Datum im Amazon Simple Storage Service-Benutzerhandbuch. Der Datumswert muss dem ISO 8601-Format entsprechen. Die Uhrzeit ist stets Mitternacht UTC. Typ: Zeichenfolge Vorfahr: oder Expiration Transition	Ja, wenn Days und ExpiredObjectDeleteMarker sind abwesend
Days	Gibt die Anzahl der Tage nach der Objekterstellung an, an denen die spezifische Regelaktion wirksam wird. Typ: Nicht negative Ganzzahl bei Verwendung mit Transition , Positive Ganzzahl bei Verwendung mit Expiration Vorfahr:, Expiration Transition	Ja, wenn Date und ExpiredObjectDeleteMarker sind abwesend

Name	Description	Erforderlich
<code>DaysAfterInitiation</code>	Gibt an, wie viele Tage nach dem Initiieren eines mehrteiligen Uploads der mehrteilige Upload abgeschlossen sein muss. Wenn der Vorgang nicht innerhalb der angegebenen Anzahl von Tagen abgeschlossen ist, kann ein Vorgang abgebrochen werden, und Amazon S3 storniert den unvollständigen mehrteiligen Upload. Typ: Positive Ganzzahl Vorfahr: <code>AbortIncompleteMultipartUpload</code>	Ja, wenn ein übergeordnetes Tag angegeben ist

Name	Description	Erforderlich
Expiration	Diese Aktion gibt einen Zeitraum in der Lebensdauer eines Objekts an, in dem Amazon S3 die entsprechende Ablaufaktion ergreifen sollte. Die Aktion, die Amazon S3 ergreift, hängt davon ab, ob der Bucket für die Versionierung aktiviert ist. • Wenn die Versionierung für den Bucket noch nie aktiviert wurde, löscht Amazon S3 die einzige Kopie des Objekts dauerhaft. • Wenn für den Bucket die Versionierung aktiviert ist (oder die Versionierung unterbrochen ist), gilt die Aktion nur für die aktuelle Version des Objekts. Ein Bucket mit Versionskontrolle kann viele Versionen desselben Objekts enthalten: eine aktuelle Version und keine oder mehrere nicht aktuelle Versionen. Anstatt die aktuelle Version zu löschen, macht Amazon S3 sie zu einer nicht aktuellen Version, indem eine Löschmarkierung als neue aktuelle Version hinzugefügt wird.  Important Wenn der Status eines Buckets „Version-Suspended“ lautet, erstellt Amazon S3 eine Löschmarkierung mit der Versions-ID. null Wenn Sie eine Version mit Versions-ID	Ja, wenn keine andere Aktion in der Rule vorhanden ist.

Name	Description	Erforderlich
	habennull1, überschreibt Amazon S3 diese Version.  Note Verwenden Sie die Aktion, um das Ablaufdatum für nicht aktuelle Objekte festzulegen. NoncurrentVersionExpiration Typ: Container Kinder: Tage oder Datum Vorfahr: Regel	
ID	Ein eindeutiger Bezeichner für die Regel. Der Wert darf nicht länger als 255 Zeichen sein. Typ: Zeichenfolge Vorfahr: Regel	Nein

Name	Description	Erforderlich
LifecycleConfiguration	Container für Lebenszyklusregeln. Sie können bis zu 1000 Regeln hinzufügen. Typ: Container Kinder: Regel Vorfahr: Keiner	Ja
ExpiredObjectDeleteMarker	In einem versionierten Bucket (einem Bucket mit Versionierung oder einem Bucket mit Versionssuspendierung) können Sie dieses Element zur Lebenszykluskonfiguration hinzufügen, um Amazon S3 anzuweisen, Markierungen zum Löschen abgelaufener Objekte zu löschen. Ein Beispiel finden Sie unter Beispiel 8: Entfernen von Markierungen zum Löschen abgelaufener Objekte im Amazon Simple Storage Service-Benutzerhandbuch. Fügen Sie es nicht zu einem Bucket ohne Versionierung hinzu, da dieser Bucket-Typ keine Löschmarkierungen enthalten kann. Typ: Zeichenfolge Gültige Werte: true false (der Wert false ist zulässig, aber No-Op, was bedeutet, dass Amazon S3 keine Maßnahmen ergreift) Vorfahr: Expiration	Ja, wenn Date und Days sind abwesend

Name	Description	Erforderlich
NoncurrentDays	Gibt die Anzahl der Tage an, die ein Objekt nicht-aktuell ist, bevor Amazon S3 die zugehörige Aktion ausführen kann. Informationen zu den Berechnungen nicht aktueller Tage finden Sie im Amazon Simple Storage Service-Benutzerhandbuch unter So berechnet Amazon S3, wenn ein Objekt nicht mehr aktuell ist. Typ: Nicht negative Ganzzahl bei Verwendung in KombinationNoncurrentVersionTransition , positive Ganzzahl bei Verwendung mit NoncurrentVersionExpiration Vorfahr: oder NoncurrentVersionExpiration NoncurrentVersionTransition	Ja

Name	Description	Erforderlich
<code>NoncurrentVersionExpiration</code>	Gibt an, wann nicht-aktuelle Objektversionen ablaufen. Nach Ablauf löscht Amazon S3 permanent die nicht-aktuellen Objektversionen. Stellen Sie diese Lebenszyklus-Konfigurationsaktion für einen Bucket ein, für den die Versionierung aktiviert (oder ausgesetzt) ist, um Amazon S3 anzuweisen, nicht aktuelle Objektversionen zu einem bestimmten Zeitraum in der Lebensdauer des Objekts zu löschen. Typ: Container Kinder: <code>NoncurrentDays</code> Vorfahr: Regel	Ja, wenn keine andere Aktion in der Rule
<code>NoncurrentVersionTransition</code>	Container für die Übergangsregel, die beschreibt, wann nicht aktuelle Objekte in die Speicherklasse <code>STANDARD_IA</code>, <code>ONEZONE_IA</code>, <code>IA</code>, oder übergehen. Wenn Ihr Bucket für die Versionierung aktiviert ist (oder wenn die Versionierung unterbrochen ist), können Sie diese Aktion so einrichten, dass Amazon S3 angewiesen wird, nicht aktuelle Objektversionen zu einem bestimmten Zeitpunkt in der Lebensdauer des Objekts zu migrieren. Typ: Container Kinder: <code>und NoncurrentDays StorageClass</code> Vorfahr: Regel	Ja, wenn keine andere Aktion in der Rule

Name	Description	Erforderlich
Prefix	Objektschlüsselpräfix, das ein oder mehrere Objekte identifiziert, für die die Regel gilt. Typ: Zeichenfolge Vorfahr: Regel	Ja
Rule	Container für eine Lebenszyklusregel. Eine Lebenszykluskonfiguration kann bis zu 1000 Regeln enthalten. Typ: Container Ancestor:LifecycleConfiguration	Ja
Status	Wenn diese Option aktiviert ist, führt Amazon S3 die Regel wie geplant aus. Wenn sie deaktiviert ist, ignoriert Amazon S3 die Regel. Typ: Zeichenfolge Ancestor: Regel Gültige Werte: Aktiviert, Deaktiviert	Ja

Name	Description	Erforderlich
StorageClass	Gibt die Amazon S3-Speicherklasse an, in die das Objekt übergehen soll. Typ: Zeichenfolge Ancestor: Übergang und NoncurrentVersionTransition Gültige Werte: STANDARD_IA ONEZONE_IA GLACIER	Ja Dieses Element ist nur erforderlich, wenn Sie einen oder beide seiner Vorfahren angeben.

Name	Description	Erforderlich
Transition	Diese Aktion gibt einen Zeitraum in der Lebensdauer der Objekte an, in dem Amazon S3 sie in die Speicherklasse <code>STANDARD_IA</code>, <code>ONEZONE_IA</code>, oder umstellen soll. Wenn diese Aktion in Kraft ist, hängt das, was Amazon S3 tut, davon ab, ob der Bucket versionierungsfähig ist. • Wenn die Versionierung für den Bucket noch nie aktiviert wurde, überträgt Amazon S3 die einzige Kopie des Objekts in die angegebene Speicherklasse. • Wenn für Ihren Bucket die Versionierung aktiviert ist (oder die Versionierung ausgesetzt ist), überträgt Amazon S3 nur die aktuellen Versionen der in der Regel identifizierten Objekte.  Note Ein Bucket mit Versionskontrolle kann viele Versionen eines Objekts enthalten. Diese Aktion hat keine Auswirkung auf nicht aktuelle Objektversionen. Um nicht aktuelle Objekte zu übertragen, müssen Sie die <code>NoncurrentVersionTransition</code> Aktion verwenden.	Ja, wenn keine andere Aktion in der Rule

Name	Description	Erforderlich
	Typ: Container	
	Kinder: Tage oder Datum und StorageClass	
	Vorfahr: Regel	

Antworten

Antwort-Header

Bei dieser Implementierung des Vorgangs werden nur Antwortheader verwendet, die den meisten Antworten gemeinsam sind. Weitere Informationen finden Sie unter [Allgemeine](#) Antwortheader.

Antwortelemente

Diese Implementierung der Operation gibt keine Antwortelemente zurück.

Spezielle Fehler

Diese Implementierung der Operation gibt keine speziellen Fehler zurück. Allgemeine Informationen zu Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlerantworten](#).

Beispiele

Beispiel 1: Fügen Sie einem Bucket eine Lifecycle-Konfiguration hinzu, die es nicht ist Versioning-enabled

Die folgende Lebenszykluskonfiguration spezifiziert zwei Regeln mit jeweils einer Aktion.

- Die Transition -Aktion weist Amazon S3 an, Objekte mit dem Präfix „documents/“ 30 Tage nach der Erstellung in die Speicherklasse zu übertragen.
- Die Aktion „Ablauf“ weist Amazon S3 an, Objekte mit dem Präfix „logs/“ 365 Tage nach der Erstellung zu löschen.

```
<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>

```

Im Folgenden finden Sie eine PUT `/?lifecycle` Beispielanforderung, mit der die vorherige Lebenszykluskonfiguration zum Bucket hinzugefügt wird `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>
```

Im Folgenden wird eine Beispielantwort dargestellt:

```
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
```

Beispiel 2: Lifecycle-Konfiguration zu einem Versioning-enabled Bucket hinzufügen

Die folgende Lebenszykluskonfiguration spezifiziert zwei Regeln mit jeweils einer Aktion, die Amazon S3 ausführen muss. Sie geben diese Aktionen an, wenn Ihr Bucket für die Versionierung aktiviert ist oder die Versionierung unterbrochen ist:

- Die `NoncurrentVersionExpiration` Aktion weist Amazon S3 an, nicht aktuelle Versionen von Objekten mit dem Präfix „logs/“ 100 Tage, nachdem die Objekte nicht mehr aktuell sind, ablaufen zu lassen.
- Die `NoncurrentVersionTransition` Aktion weist Amazon S3 an, nicht aktuelle Versionen von Objekten mit dem Präfix „documents/“ 30 Tage, nachdem sie nicht mehr aktuell sind, auf die Speicherklasse umzustellen.

```
<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>

```

Im Folgenden finden Sie eine PUT `/?lifecycle` Beispielanforderung, mit der die vorherige Lifecycle-Konfiguration zum Bucket hinzugefügt wird. `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>

```

Im Folgenden wird eine Beispielantwort dargestellt:

```

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
```

Weitere Beispiele

Weitere Beispiele für die Umstellung von Objekten auf Speicherklassen wie STANDARD_IA oder ONEZONE_IA finden Sie unter Beispiele für die Lifecycle-Konfiguration. <https://docs.aws.amazon.com/AmazonS3/latest/userguide/intro-lifecycle-rules.html#lifecycle-configuration-examples>

Verwandte Ressourcen

- [GetBucketLifecycleConfiguration](#)
- [POST Object-Wiederherstellung](#)
- Standardmäßig kann ein Ressourcenbesitzer — in diesem Fall ein Bucket-Besitzer, der den Bucket erstellt hat — alle Operationen ausführen AWS-Konto . Ein Ressourcenbesitzer kann auch anderen Personen die Erlaubnis erteilen, den Vorgang auszuführen. Weitere Informationen finden Sie in den folgenden Themen im Benutzerhandbuch zum Amazon Simple Storage Service:
 - [Angaben von Berechtigungen in einer Richtlinie](#)
 - [Verwaltung der Zugriffsberechtigungen für Ihre Amazon-S3-Ressourcen](#)

BUCKET-Lebenszyklus ABRUFEN (veraltet)

Description

Important

Eine aktualisierte Version dieser API finden Sie unter [GetBucketLifecycleConfiguration](#). Wenn Sie mithilfe des <filter>Elements einen Bucket-Lebenszyklus konfiguriert haben, sollten Sie eine aktualisierte Version dieses Themas sehen. Dieses Thema dient der Abwärtskompatibilität.

Gibt die für den Bucket festgelegten `lifecycle` Konfigurationsinformationen zurück. Informationen zur Lifecycle-Konfiguration finden Sie [im Amazon Simple Storage Service-Benutzerhandbuch unter Object Lifecycle Management](#).

Um diesen Vorgang verwenden zu können, benötigen Sie die Berechtigung zum Ausführen der `s3:GetLifecycleConfiguration` Aktion. Der Bucket-Besitzer hat diese Berechtigung standardmäßig. Der Bucket-Besitzer kann diese Erlaubnis anderen gewähren. Weitere Informationen zu Berechtigungen finden Sie unter [Verwalten von Zugriffsberechtigungen für Ihre Amazon S3-Ressourcen](#) im Amazon Simple Storage Service-Benutzerhandbuch.

Anforderungen

Syntax

```
GET /?lifecycle HTTP/1.1
Host: bucketname.s3.amazonaws.com
Date: date
Authorization: authorization string (see Anfragen authentifizieren \(AWS Signatur \(Version 4\)\))
```

Anforderungsparameter

Bei dieser Implementierung der Operation werden keine Anforderungsparameter verwendet.

Anforderungsheader

Diese Implementierung des Vorgangs verwendet nur Anforderungsheader, die allen Vorgängen gemeinsam sind. Weitere Informationen finden Sie unter [Allgemeine](#) Anforderungsheader.

Anfrage-Elemente

Diese Implementierung des Vorgangs verwendet keine Anforderungselemente.

Antworten

Antwort-Header

Diese Implementierung des Vorgangs verwendet nur Antwortheader, die den meisten Antworten gemeinsam sind. Weitere Informationen finden Sie unter [Allgemeine](#) Antwortheader.

Antwortelemente

Diese Implementierung von GET gibt die folgenden Antwortelemente zurück.

Name	Description	Erforderlich
AbortIncompleteMultipartUpload	Container zur Angabe, wann ein unvollständiger mehrteiliger Upload für einen Abbruchvorgang in Frage kommt. Untergeordnet: DaysAfterInitiation Typ: Container Vorfahr: Rule	Ja, wenn für die Regel keine andere Aktion angegeben ist
Date	Datum, an dem Amazon S3 die Aktion ausführen soll. Weitere Informationen finden Sie unter Lebenszyklusregeln: Basierend auf einem bestimmten Datum im Amazon Simple Storage Service-Benutzerhandbuch. Der Datumswert muss konform zum Format ISO 8601 angegeben werden. Die Uhrzeit ist stets Mitternacht UTC. Typ: Zeichenfolge Vorfahr: Expiration oder Transition	Ja, wenn Days und ExpiredObjectDeleteMarker sind abwesend

Name	Description	Erforderlich
Days	Gibt die Anzahl der Tage nach der Objekterstellung an, an denen die spezifische Regelaktion wirksam wird. Die Zulassungszeit des Objekts wird als Erstellungszeit plus Anzahl der Tage berechnet, wobei die resultierende Zeit auf Mitternacht UTC des nächsten Tages gerundet wird. Typ: Non-negative Ganzzahl bei Verwendung mit <code>Transition</code> , positive Ganzzahl bei Verwendung mit <code>Expiration</code> . Vorfahr: <code>Transition</code> oder <code>Expiration</code>	Ja, wenn <code>Date</code> und <code>ExpiredObjectDeleteMarker</code> sind abwesend
DaysAfterInitiation	Gibt an, wie viele Tage nach dem Initiieren eines mehrteiligen Uploads der mehrteilige Upload abgeschlossen sein muss. Wenn der Vorgang nicht innerhalb der angegebenen Anzahl von Tagen abgeschlossen ist, kann ein Vorgang abgebrochen werden, und Amazon S3 storniert den unvollständigen mehrteiligen Upload. Typ: Positive Ganzzahl Vorfahr: <code>AbortIncompleteMultipartUpload</code>	Ja, wenn es abwesend ist

Name	Description	Erforderlich
Expiration	Diese Aktion gibt einen Zeitraum in der Lebensdauer des Objekts an, in dem Amazon S3 die entsprechende Ablaufmaßnahme ergreifen soll. Die Ablaufaktion wird nur für Objekte ausgeführt, die gemäß dem im untergeordneten Days Element Date oder Element angegebenen Zeitraum in Frage kommen. Die Aktion, die Amazon S3 ergreift, hängt davon ab, ob für den Bucket die Versionierung aktiviert ist. • Wenn die Versionierung für den Bucket noch nie aktiviert wurde, löscht Amazon S3 die einzige Kopie des Objekts dauerhaft. • Andernfalls gilt die Aktion nur für die aktuelle Version des Objekts, wenn für Ihren Bucket die Versionierung aktiviert ist (oder die Versionierung ausgesetzt ist). Buckets, für die die Versionierung aktiviert oder für die die Versionierung gesperrt ist, können viele Versionen desselben Objekts enthalten: eine aktuelle Version und keine oder mehrere nicht aktuelle Versionen. Anstatt die aktuelle Version zu löschen, macht Amazon S3 sie zu einer nicht aktuellen Version, indem eine Löschmarkierung als neue aktuelle Version hinzugefügt wird.	Ja, wenn das übergeordnete Tag angegeben ist

Name	Description	Erforderlich
	 Important Wenn der Status eines Buckets „Version-Suspended“ lautet, erstellt Amazon S3 eine Löschmarkierung mit der Versions-ID. null Wenn Sie eine Version mit Versions-ID habennull, überschreibt Amazon S3 diese Version.  Note Um das Ablaufdatum für nicht aktuelle Objekte festzulegen, müssen Sie die NoncurrentVersionExpiration Aktion verwenden. Typ: Container Kinder: Tage oder Datum Vorfahr: Regel	

Name	Description	Erforderlich
ID	Ein eindeutiger Bezeichner für die Regel. Der Wert darf nicht länger als 255 Zeichen sein. Typ: Zeichenfolge Vorfahr: Regel	Nein
LifecycleConfiguration	Container für Lebenszyklusregeln. Sie können bis zu 1000 Regeln hinzufügen. Typ: Container Kinder: Regel Vorfahr: Keiner	Ja
ExpiredObjectDeleteMarker	In einem versionierten Bucket (Bucket mit Versionierung oder Bucket mit Versionierung suspendiert) gibt dieses Element an, ob Amazon S3 alle Markierungen zum Löschen abgelaufener Objekte im Bucket löscht. Ein Beispiel finden Sie im Amazon Simple Storage Service-Benutzerhandbuch unter Beispiel 8: Ablaufaktion angeben, um Markierungen zum Löschen abgelaufener Objekte zu entfernen. Typ: Zeichenfolge Gültige Werte: true false (der Wert false ist zulässig, aber es handelt sich um einen No-Op, Amazon S3 ergreift keine Maßnahmen, wenn der Wert falsch ist) Vorfahr: Expiration	Ja, wenn Date und Days sind abwesend

Name	Description	Erforderlich
NoncurrentDays	Gibt die Anzahl der Tage an, an denen ein Objekt nicht aktuell ist, bevor Amazon S3 die zugehörige Aktion ausführen kann. Informationen zur Berechnung nicht aktueller Tage finden Sie unter Lebenszyklusregeln auf der Grundlage der Anzahl der Tage im Amazon Simple Storage Service-Benutzerhandbuch. Typ: Nicht negative Ganzzahl bei Verwendung in KombinationNoncurrentVersionTransition , positive Ganzzahl bei Verwendung mit NoncurrentVersionExpiration Vorfahr: oder NoncurrentVersionExpiration NoncurrentVersionTransition	Ja, nur wenn der Vorfahr anwesend ist
NoncurrentVersionExpiration	Gibt an, wann nicht-aktuelle Objektversionen ablaufen. Nach Ablauf löscht Amazon S3 permanent die nicht-aktuellen Objektversionen. Stellen Sie diese Lebenszyklus-Konfigurationsaktion für einen Bucket ein, für den die Versionierung aktiviert (oder ausgesetzt) ist, um Amazon S3 aufzufordern, nicht aktuelle Objektversionen zu einem bestimmten Zeitraum in der Lebensdauer des Objekts zu löschen. Typ: Container Kinder: NoncurrentDays Vorfahr: Regel	Ja, wenn keine andere Aktion in der Rule

Name	Description	Erforderlich
NoncurrentVersionTransition	Container für die Übergangsregel, die beschreibt, wann nicht aktuelle Objekte in die STANDARD_IA ONEZONE_IA , oder die Speicherklasse übergehen. Wenn Ihr Bucket für die Versionierung aktiviert ist (oder die Versionierung unterbrochen ist), können Sie diese Aktion so einrichten, dass Amazon S3 aufgefordert wird, nicht aktuelle Objektversionen zu einem bestimmten Zeitpunkt in der Lebensdauer des Objekts auf die Speicherklasse umzustellen. Typ: Container Kinder: und NoncurrentDays StorageClass Vorfahr: Regel	Ja, wenn keine andere Aktion in der Rule
Prefix	Objektschlüsselpräfix, das ein oder mehrere Objekte identifiziert, für die die Regel gilt. Typ: Zeichenfolge Vorfahr: Regel	Ja
Rule	Container für eine Lebenszyklusregel. Typ: Container Vorfahr: LifecycleConfiguration	Ja

Name	Description	Erforderlich
Status	Wenn aktiviert, führt Amazon S3 die Regel wie geplant aus. Wenn deaktiviert, ignoriert Amazon S3 die Regel. Typ: Zeichenfolge Ancestor: Regel Gültige Werte: Aktiviert oder Deaktiviert	Ja
StorageClass	Gibt die Amazon S3-Speicherklasse an, zu der Sie das Objekt übertragen möchten. Typ: Zeichenfolge Ancestor: Übergang und NoncurrentVersionTransition Gültige Werte: STANDARD_IA ONEZONE_IA	Ja

Name	Description	Erforderlich
Transition	Diese Aktion gibt einen Zeitraum in der Lebensdauer der Objekte an, in dem Amazon S3 sie in die Speicherklasse <code>STANDARD_IA</code>, <code>ONEZONE_IA</code>, oder umstellen soll. Wenn diese Aktion in Kraft ist, hängt das, was Amazon S3 tut, davon ab, ob der Bucket versionierungsfähig ist. • Wenn die Versionierung für den Bucket noch nie aktiviert wurde, überträgt Amazon S3 die einzige Kopie des Objekts in die angegebene Speicherklasse. • Wenn für Ihren Bucket die Versionierung aktiviert ist (oder die Versionierung ausgesetzt ist), überträgt Amazon S3 nur die aktuellen Versionen der in der Regel identifizierten Objekte.  Note Ein Bucket mit aktivierter oder gesperrter Versionierung kann viele Versionen eines Objekts enthalten. Diese Aktion hat keine Auswirkung auf die nicht aktuellen Objektversionen. Um nicht aktuelle Objekte zu übertragen, müssen Sie die <code>NoncurrentVersionTransition</code> Aktion verwenden.	Ja, wenn keine andere Aktion in der Rule

Name	Description	Erforderlich
	Typ: Container Kinder: Tage oder Datum und StorageClass Vorfahr: Regel	

Spezielle Fehler

Fehlercode	Description	HTTP-Statuscode	SOAP-Fehlercode-Präfix
NoSuchLifecycleConfiguration	Die Lebenszykluskonfiguration ist nicht vorhanden.	404 Not Found (404 Nicht gefunden)	Client

Allgemeine Informationen zu Amazon S3-Fehlern und eine Liste von Fehlercodes finden Sie unter [Fehlermeldungen](#).

Beispiele

Beispiel 1: Rufen Sie eine Lifecycle-Unterressource ab

Dieses Beispiel ist eine GET-Anforderung zum Abrufen der `lifecycle` Unterressource aus dem angegebenen Bucket und eine Beispielantwort mit der zurückgegebenen Lebenszykluskonfiguration.

Beispielanforderung

```
GET /?lifecycle HTTP/1.1
Host: examplebucket.s3.amazonaws.com
x-amz-date: Thu, 15 Nov 2012 00:17:21 GMT
Authorization: signatureValue
```


Beispielantwort

```
HTTP/1.1 200 OK
x-amz-id-2: ITnGT1y4RyTmXa3rPi4hklTXouTf0hccUjo0iCPjz6FnfIutBj3M7fPGlW02SEWp
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>
```

Verwandte Ressourcen

- [PutBucketLifecycleConfiguration](#)
- [DeleteBucketLifecycle](#)

Dokumentverlauf

In der folgenden Tabelle werden die wichtigen Änderungen am Amazon S3-Entwicklerhandbuch beschrieben. Um Benachrichtigungen über Aktualisierungen dieser Dokumentation zu erhalten, können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Neues Entwicklerhandbuch	Alle Inhalte, die nicht zur API gehören, wie z. B. das Entwickeln mit S3, Codebeispiele, Fehlerantworten und mehr, wurden aus dem Amazon S3-API-Referenzhandbuch in dieses neue Amazon S3-Entwicklerhandbuch verschoben. Weitere Informationen finden Sie in der Amazon S3-API-Referenz .	15. Juni 2026

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.